RX64M Group

TCP/IP Protocol Stack Based Network Solution for Industrial Applications

RX Driver Package Application

R01AN2153EJ0100
Rev.1.00
Sep 1, 2014

Introduction

This application note describes a network solution for industrial applications that uses the M3S-T4-Tiny TCP/IP protocol stack. This application note includes sample code for a main program that performs web server and module initialization and drive processing and, when used in combination with the RX64M Group RX Driver Package, allows the construction of web server systems. A sample application that operates combined with the RX Driver Package is referred to as an RX Driver Package Application.

A web server is an application program that operates using TCP/IP. In general, a web server is accessed from web browsers and provides functions for using TCP/IP to transmit content stored on the web server to those browsers.

This application note describes the procedure for main program and web server evaluation by combining the USB driver (host mass storage), FAT file system (M3S-TFAT-Tiny), Ethernet driver, and TCP/IP protocol stack (M3S-T4-Tiny) included in the RX64M Group RX Driver Package.

Target Device

RX64M Group (Renesas Starter Kit+ RX64M)

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

Contents

1. Overview 2

2. Acquiring a Development Environment 7

3. Environment Preparation 10

4. Building a Project 12

5. Verify Operation 24

6. Web Server Specifications 35

7. Main Program Specifications 51

8. User-Defined Functions 61

9. When CubeSuite+ is Used 62

10. Supplement 65

1. Overview

1.1 This Application Note

This application note describes a network solution for industrial applications that uses the M3S-T4-Tiny TCP/IP protocol stack. This application note includes sample code for a main program that performs web server and module initialization and drive processing and, when used in combination with the RX64M Group RX Driver Package, allows the construction of web server systems. A sample application that operates combined with the RX Driver Package is referred to as an RX Driver Package Application.

A web server is an application program that operates using TCP/IP. In general, a web server is accessed from web browsers and provides functions for using TCP/IP to transmit content stored on the web server to those browsers.

This application note describes the procedure for main program and web server evaluation by combining the USB driver (host mass storage), FAT file system (M3S-TFAT-Tiny), Ethernet driver, and TCP/IP protocol stack (M3S-T4-Tiny) included in the RX64M Group RX Driver Package.

This application note operates on the Renesas Starter Kit+ for RX64M (referred to as “RSK” in the remainder of this document).

1.2 Operating Environment

This application note operates in the following environment.

Table 1.2.1 Operating Environment

Microcontroller	RX64M Group
Evaluation board	Renesas Starter Kit+ RX64M http://japan.renesas.com/products/tools/introductory_tools/renesas_starter_kits/rsk_plus_rx64m/index.jsp
Integrated development environment (IDE)	e^2 studio, V3.0.1.09 or later Or: CubeSuite+ V2.02.00 or later
Cross tools	RX Family C/C++ Compiler Package V2.02.00 or later
Emulator	E1 (included in the Renesas Starter Kit+ for RX64M), E20
RX Driver Package	RX64M Group RX Driver Package Ver1.00 (R01AN2144EJ0100)*

Note: * Operation of this application note has been verified when the modules in the RX Driver Package mentioned above are incorporated. If any of the modules used in this application note are replaced with a different module, the user must verify the operation.

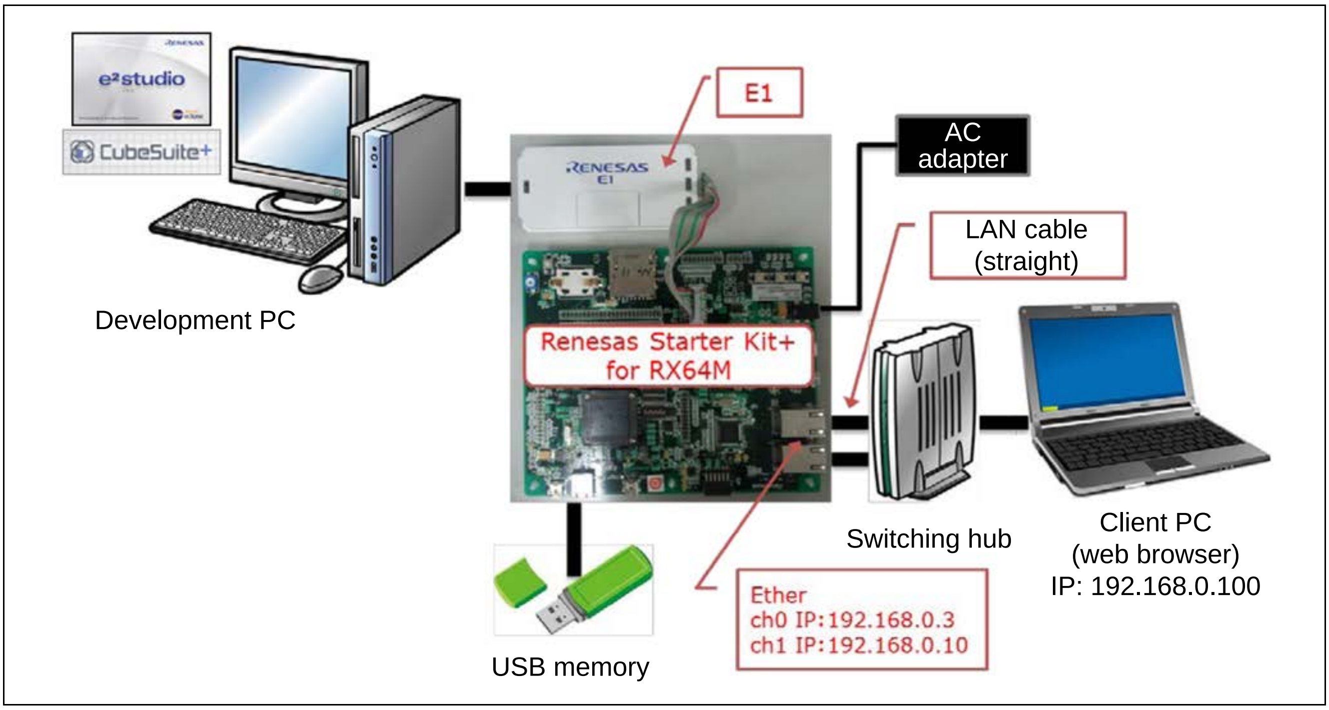

e²studio
CubeSuite+
Development PC
E1
AC adapter
LAN cable (straight)
Renesas Starter Kit+ for RX64M
Switching hub
Client PC (web browser) IP: 192.168.0.100
Ether ch0 IP:192.168.0.3 ch1 IP:192.168.0.10
USB memory

Figure 1.2.1 Sample Operating Environment

1.3 Module Structure

This section shows the structure of the modules used by this application note and a list of those modules.

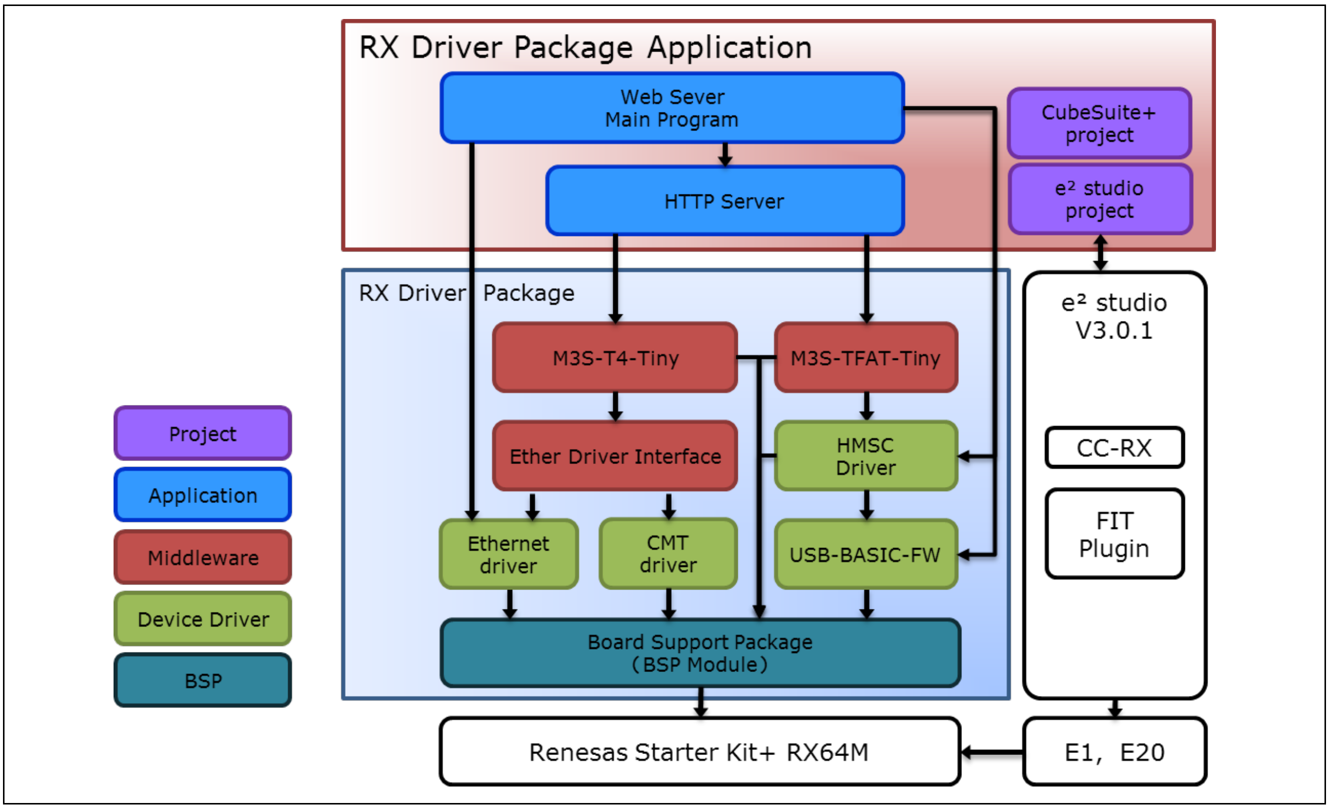


Figure 1.3.1 Module Structure

Table 1.3.1 Modules

Type	Module	FIT Module Name	Version
Board Support Package	Board support package (BSP module)	r_bsp	2.60
Device Driver	Compare match timer (CMT)	r_cmt_rx	2.30
Device Driver	Ethernet controller (ETHERC)	r_ether_rx	1.00
Middleware	M3S-T4-Tiny interface conversion module	r_t4_driver_rx64m	1.00
Middleware	TCP/IP protocol stack (M3S-T4-Tiny)	r_t4_rx	2.00
Middleware	FAT file system (M3S-TFAT-Tiny)	r_tfat_rx	3.00
Device Driver	USB basic firmware	r_usb_basic	1.00
Device Driver	USB host mass storage class	r_usb_hmsc	1.00
Application	HTTP server	r_t4_http_server_rx	1.03
Application	Web server system main program	r_httpd_main_rx64m	1.00

1.4 File Structure

This section describes the file structure used in this application note.

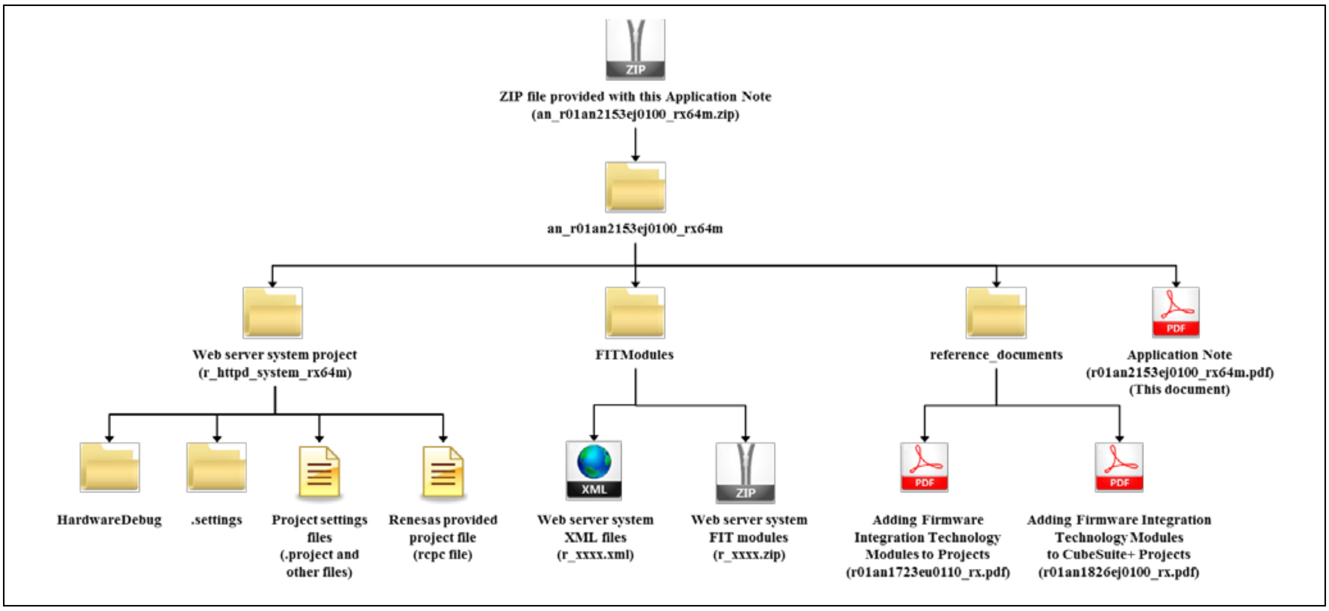

ZIP
ZIP file provided with this Application Note
(an_r01an2153ej0100_rx64m.zip)
an_r01an2153ej0100_rx64m
Web server system project
(r_httpd_system_rx64m)
FITModules
reference_documents
PDF
Application Note
(r01an2153ej0100_rx64m.pdf)
(This document)
HardwareDebug
.settings
Project settings files
(.project and other files)
Renesas provided project file
(rcpc file)
XML
Web server system XML files
(r_xxxx.xml)
ZIP
Web server system FIT modules
(r_xxxx.zip)
PDF
Adding Firmware Integration Technology Modules to Projects
(r01an1723eu0110_rx.pdf)
PDF
Adding Firmware Integration Technology Modules to CubeSuite+ Projects
(r01an1826ej0100_rx.pdf)

Figure 1.4.1 File Structure

When the ZIP file provided with this application note is decompressed, a folder with the same name is created and the various folders and files are created within that folder.

The project is s special-purpose project for building a web server. It is used by inputting it to an e^2 studio workspace. Also, Renesas provided project files, which are used to read the project with CubeSuite+, are also included.

The Web server FIT modules are included in the FITModules folder.

Documents that describe using the FIT modules in various development environments are included in the reference_documents folder. The document “Adding Firmware Integration Technology Modules to Projects” (r01an1723eu0110_rx.pdf) describes the method for including the FIT modules, as a FIT plugin, in an e^2 studio project. The document “Adding Firmware Integration Technology Modules to CubeSuite+ Projects” (r01an1826ej0100_rx.pdf) describes the method for including the FIT modules in a CubeSuite+ project.

The file “Application Note” (r01an2153ej0100_rx64m.pdf) is this document.

1.5 Projects

This application note includes an e^2 studio and a CubeSuite+ project for building and evaluating a web server system. These projects register both a build structure (build mode in CubeSuite+) that stores the build settings and a debug structure (debug tool in CubeSuite+) that stores debug settings.

The table below lists the build structure and debug structure registered in these projects.

Table 1.5.1 Project Settings

	Structure	Description
Build structure (referred to as build mode in CubeSuite+)	HardwareDebug (Debug on hardware)	This structure is used to generate a load module with debugging information included. Main settings • Debug information present • No optimization (-optimize=0)
Debug structure (referred to as debug tool in CubeSuite+)	HardwareDebug (E1) (This is RX E1 (JTAG) in CubeSuite+)	Used for hardware debugging over an E1 emulator using a load module generated by HardwareDebug (Debug on hardware).

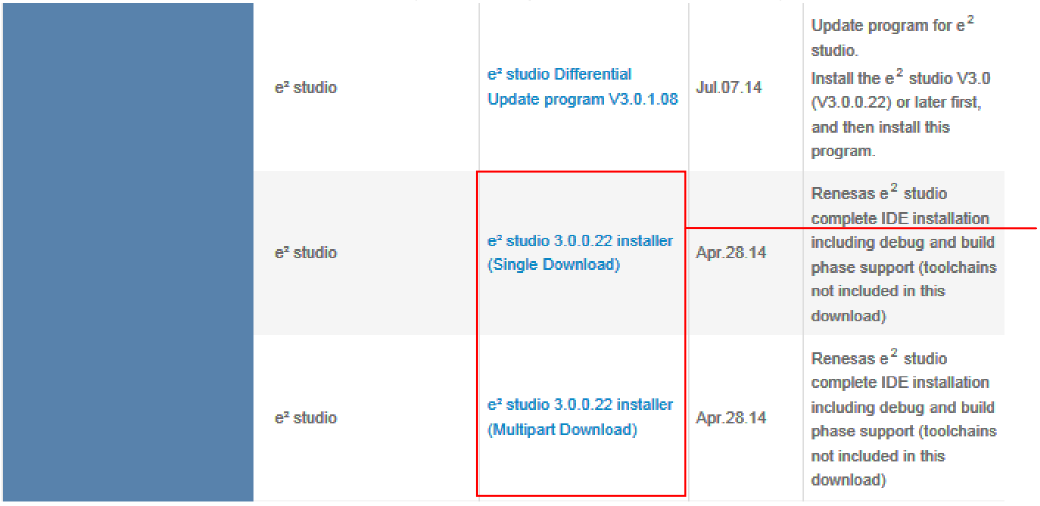
2. Acquiring a Development Environment

2.1 Acquire and Install e^2 studio

The e^2 studio can be downloaded from the Renesas web site.

1. Access the following URL to display the e^2 studio download page.
   http://www.renesas.com/e2studio_download

2. Of the displayed items, click Install the **e^2 studio 3.0.0.22 installer**. (Although there are two versions, one that is broken up into smaller sections, and one that can be downloaded in a single operation, the contents are the same.)
   Next, download the e^2 studio installer by following the instructions displayed.



	e^2 studio	e^2 studio Differential Update program V3.0.1.08	Jul.07.14	Update program for e^2 studio. Install the e^2 studio V3.0 (V3.0.0.22) or later first, and then install this program.
	e^2 studio	e^2 studio 3.0.0.22 installer (Single Download)	Apr.28.14	Renesas e^2 studio complete IDE installation including debug and build phase support (toolchains not included in this download)
	e^2 studio	e^2 studio 3.0.0.22 installer (Multipart Download)	Apr.28.14	Renesas e^2 studio complete IDE installation including debug and build phase support (toolchains not included in this download)

Click either of these links.

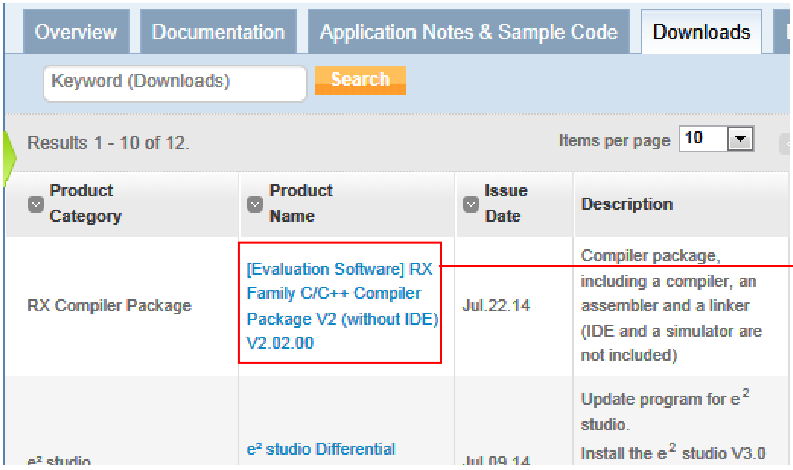
3. Run the downloaded e^2 studio installer to install e^2 studio on your personal computer.
   See the **e2 studio Integrated Development Environment User's Manual: Getting Started Guide** for details on the installation procedure.
   http://documentation.renesas.com/doc/products/tool/doc/r20ut2771ej0200_e2_start_s.pdf

2.2 Acquire a Compiler Package

The RX Family C/C++ Compiler Package, V2.02.00 or later, is required to build this web server system. This section assumes the user does not own the commercial version and will be using the free evaluation version.

1. Access the following URL to display the e^2 studio download page.
   http://www.renesas.com/e2studio_download

2. Of the displayed items, click **[Evaluation Software] RX Family C/C++ Compiler Package V2 (without IDE) V2.02.00**.
   Follow the instructions on the page displayed next to download the compiler installer.



Product Category	Product Name	Issue Date	Description
RX Compiler Package	[Evaluation Software] RX Family C/C++ Compiler Package V2 (without IDE) V2.02.00	Jul.22.14	Compiler package, including a compiler, an assembler and a linker (IDE and a simulator are not included)
e² studio	e² studio Differential	Jul.09.14	Update program for e^2 studio. Install the e^2 studio V3.0

Click this link.

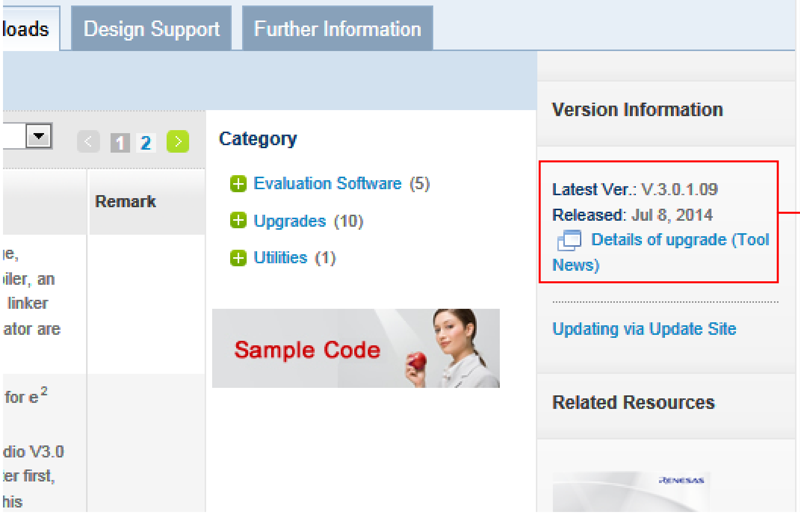
3. Run the downloaded compiler installer to install the compiler on your personal computer.

2.3 Upgrade to Version 3.0.1.09

Upgrade the e^2 studio that is installed on your personal computer to the latest version.

1. Access the following URL to display the e^2 studio download page.
   http://www.renesas.com/e2studio_download

2. Click the version information link on the right side of the displayed page.



loads | Design Support | Further Information

Category

- Evaluation Software (5)
- Upgrades (10)
- Utilities (1)

Remark

ie,
iler, an
linker
ator are

for e^2

dio V3.0
er first,
his

Sample Code

Version Information

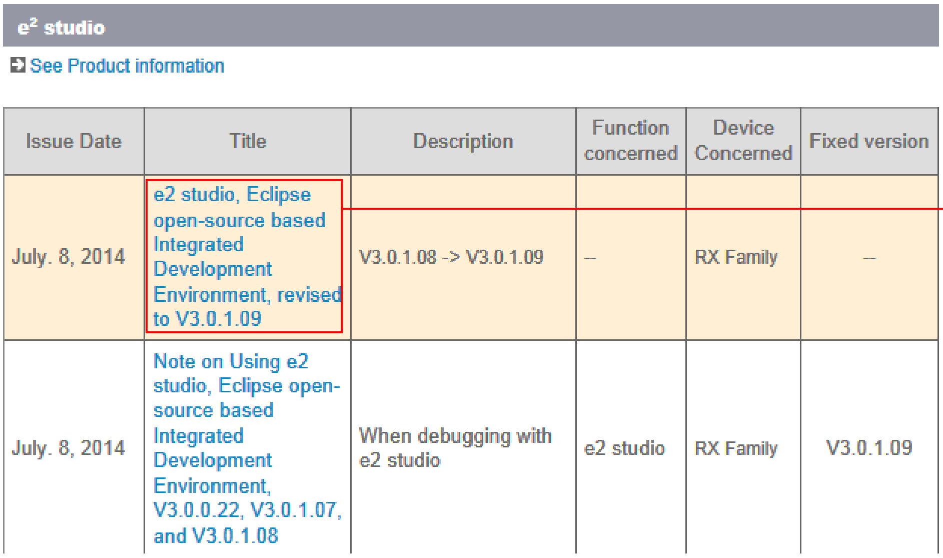
Latest Ver.: V.3.0.1.09
Released: Jul 8, 2014
Details of upgrade (Tool News)

Updating via Update Site

Related Resources

Click this link.

3. Of the displayed items, click the link shown as **e2 studio, Eclipse open-source based Integrated Development Environment, revised to V3.0.1.09**.
   Follow the directions on the displayed page to perform the e^2 studio update.



e^2 studio

See Product information

Issue Date	Title	Description	Function concerned	Device Concerned	Fixed version
July. 8, 2014	e2 studio, Eclipse open-source based Integrated Development Environment, revised to V3.0.1.09	V3.0.1.08 -> V3.0.1.09	–	RX Family	–
July. 8, 2014	Note on Using e2 studio, Eclipse open-source based Integrated Development Environment, V3.0.0.22, V3.0.1.07, and V3.0.1.08	When debugging with e2 studio	e2 studio	RX Family	V3.0.1.09

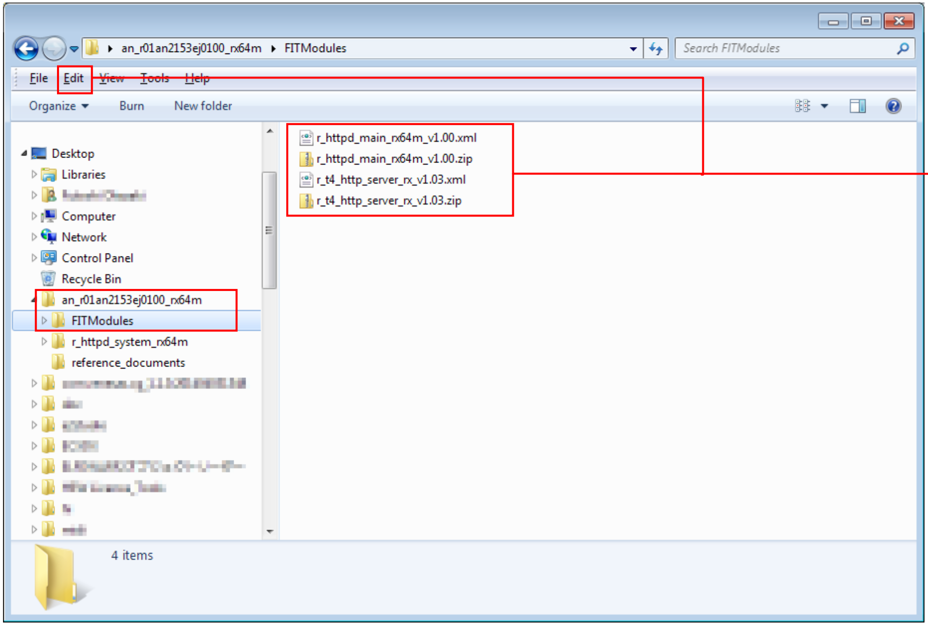
Click this link.

3. Environment Preparation

3.1 Install the FIT Modules

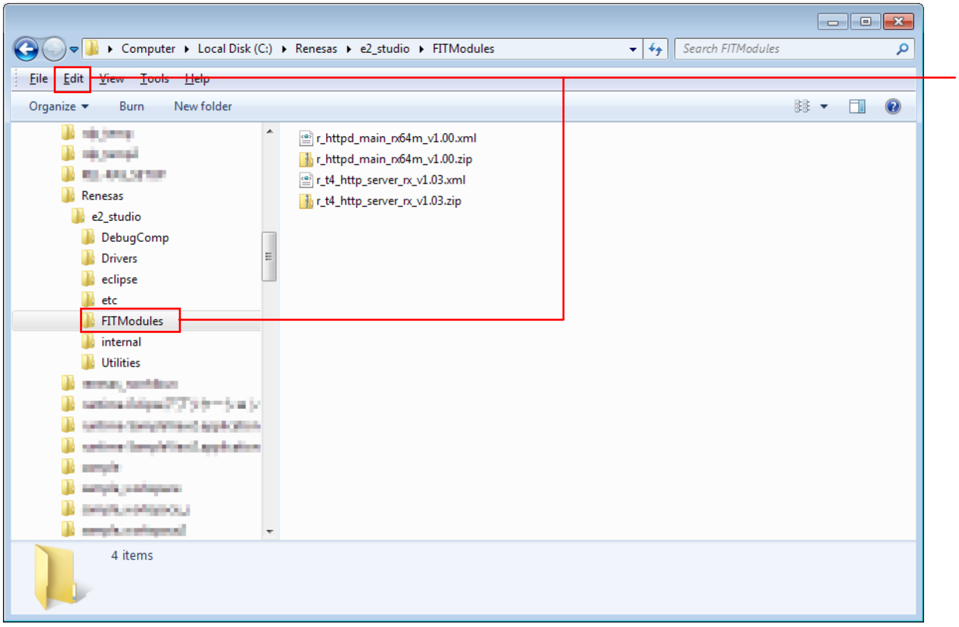
Install the FIT modules used in the web server system in this application note into e^2 studio.

1. Decompress the ZIP file in which this application note is provided into an arbitrary folder.
2. Open the folder into which that ZIP file was decompressed and of the folders in that folder, open the **FITModules** folder.
3. Select all of the files in the **FITModules** folder and click **Copy** in the **Edit** menu.



Select all files and click **Copy** in the **Edit** menu.

4. Open the e^2 studio install folder (Usually, this will be c:/Renesas/e2_studio.) and open the **FITModules** folder in that folder.
5. Click **Paste** on the **Edit** menu.
   The e^2 studio **FITModules** folder will be copied to the FIT modules.


Computer ▸ Local Disk (C:) ▸ Renesas ▸ e2_studio ▸ FITModules
Search FITModules
File
Edit
View
Tools
Help
Organize ▾
Burn
New folder
Renesas
e2_studio
DebugComp
Drivers
eclipse
etc
FITModules
internal
Utilities
r_httpd_main_rx64m_v1.00.xml
r_httpd_main_rx64m_v1.00.zip
r_t4_http_server_rx_v1.03.xml
r_t4_http_server_rx_v1.03.zip
4 items

Open the **FITModules** folder and click **Paste** on the **Edit** menu.
The folder will be copied.

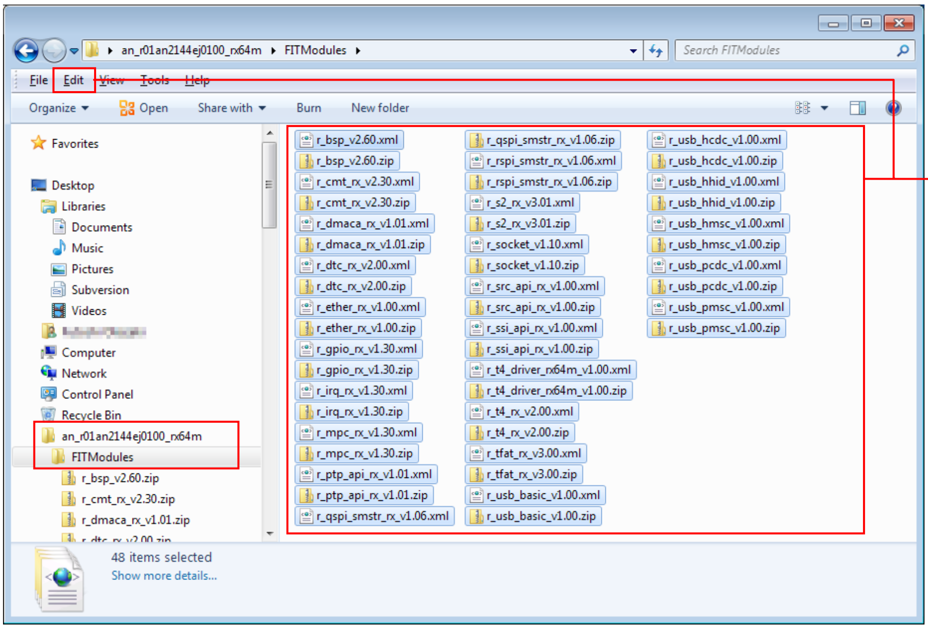
3.2 Install the RX Driver Package

Install the FIT modules included in the RX64M Group RX Driver package in e^2 studio.

1. Download the RX64M Group RX Driver package and decompress the file an_r01an2144ej0100_rx64m.zip into an arbitrary folder.

2. Open the folder that was decompressed and open the **FITModules** folder in that folder.

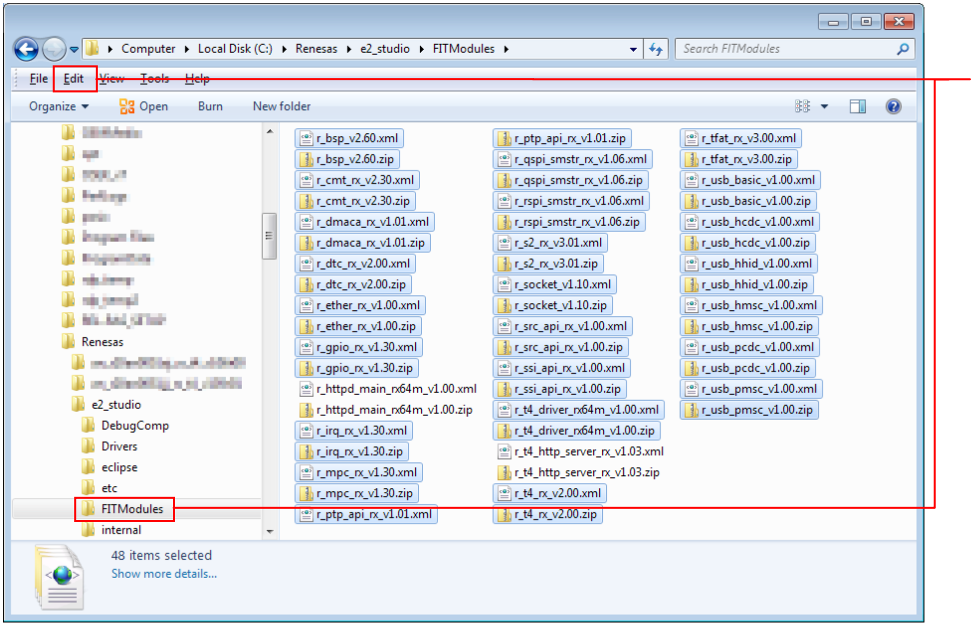
3. Select all the files in the **FITModules** folder, and click **Paste** on the **Edit** menu.



Select all files and click **Copy** in the **Edit** menu.

4. Open the e^2 studio install folder (Usually, this will be c:/Renesas/e2_studio.) and open the **FITModules** folder in that folder.

5. Click **Paste** on the **Edit** menu.
The e^2 studio **FITModules** folder will be copied to the FIT modules.



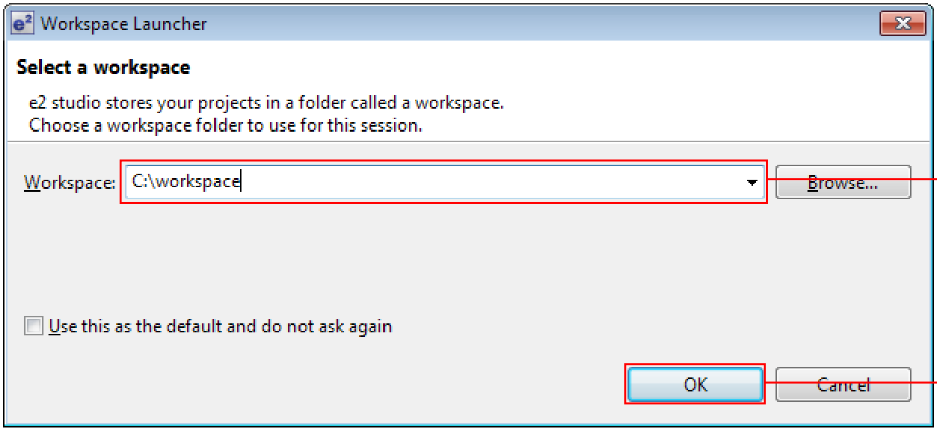
Open the **FITModules** folder and click **Paste** on the **Edit** menu.
The folder will be copied.

4. Building a Project

4.1 Create a Workspace

1. Start e^2 studio.

2. Enter an arbitrary workspace folder in the displayed dialog box and click **OK**.



Workspace Launcher

Select a workspace

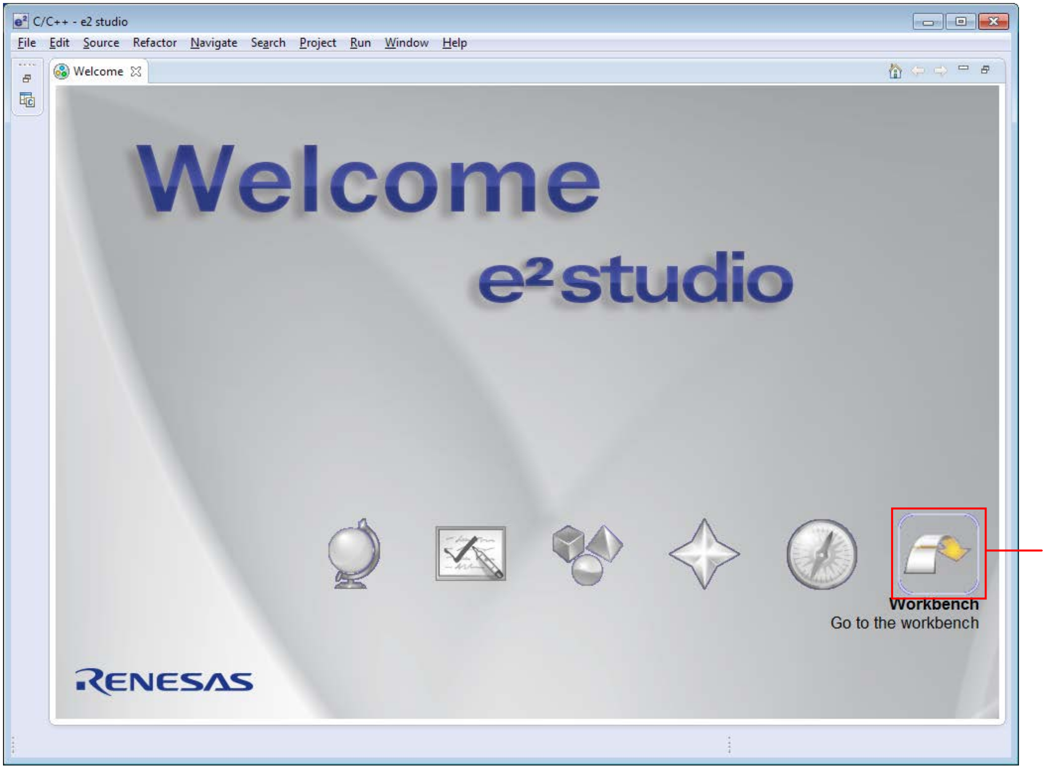
e2 studio stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\workspace

Enter a workspace folder.

Click **OK**.

3. When the following window is displayed, click **Workbench**.

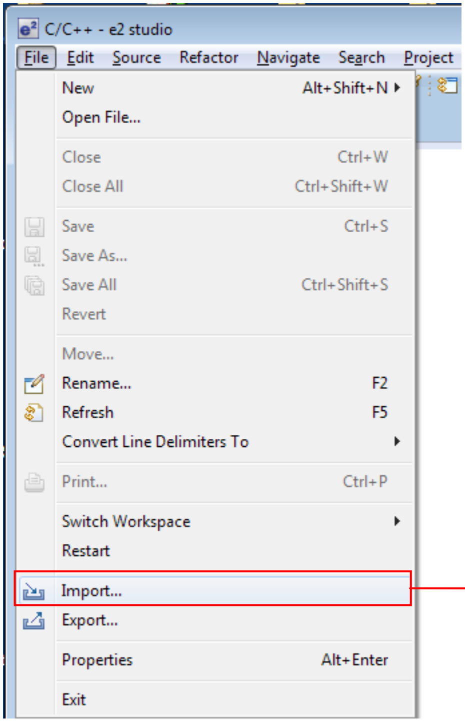


Click **Workbench**.

4.2 Import a Project

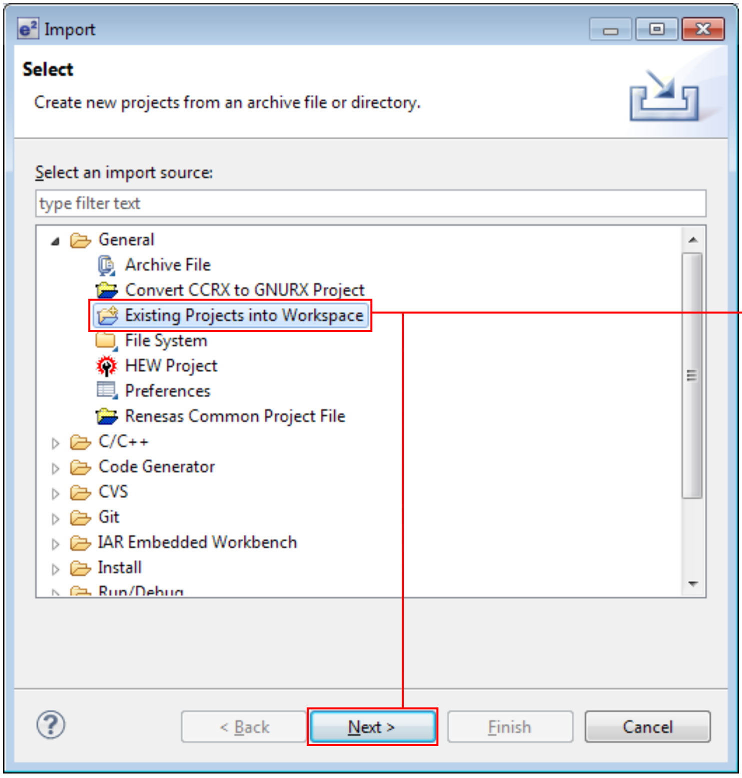
Import the project provided with this application note into the newly created workspace.

1. Select **Import** from the e^2 studio **File** menu.



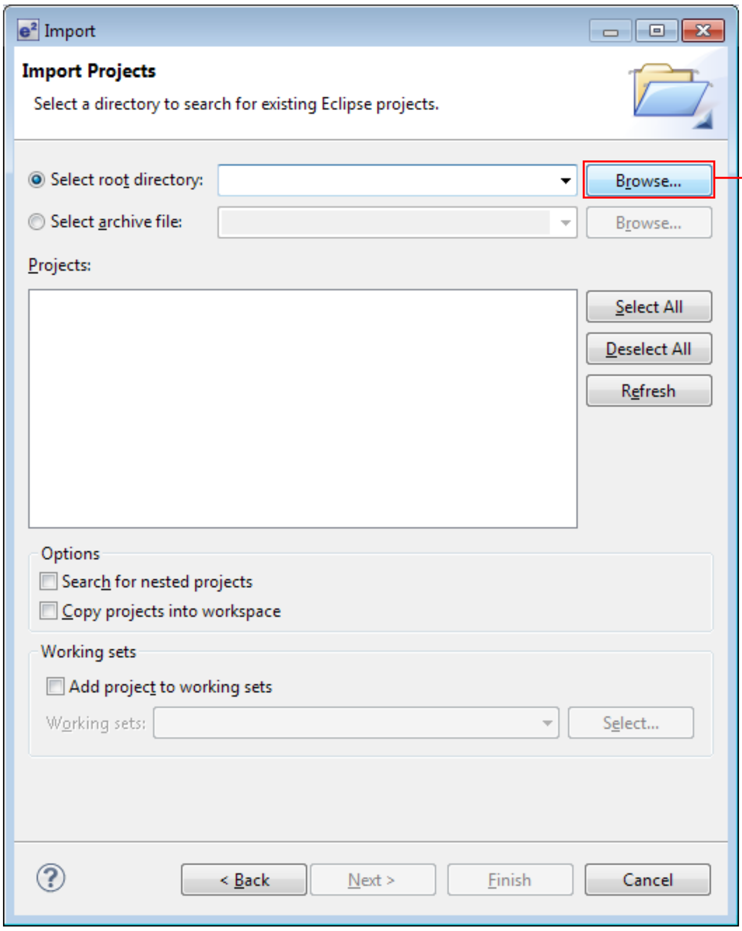
Click here.

2. Select **Existing Projects into Workspace** from **General** and click **Next**.



Select **Existing Projects into Workspace** from **General** and click Next.

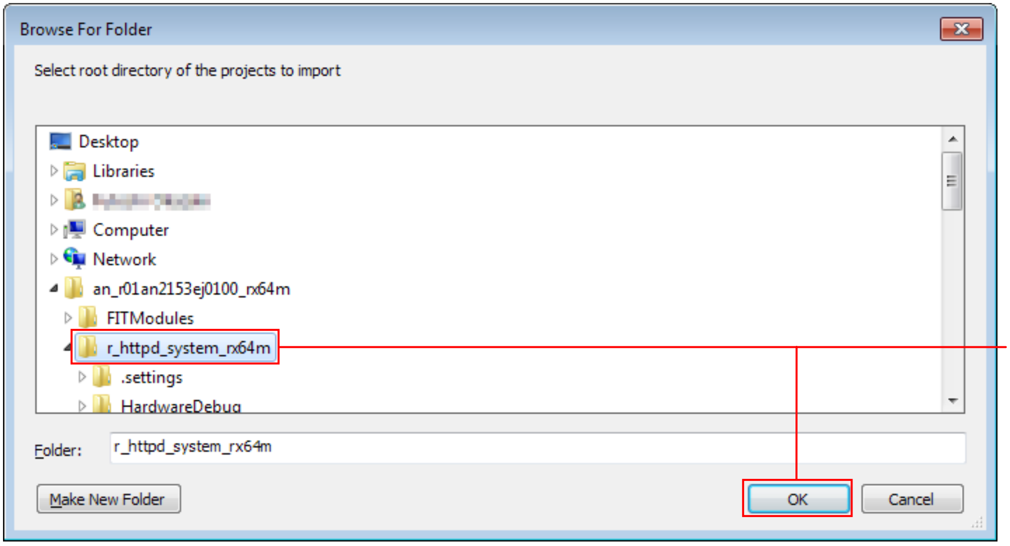
3. Click **Browse**.



● Select root directory:
○ Select archive file:
□ Search for nested projects
□ Copy projects into workspace
□ Add project to working sets
Working sets:

Click here.

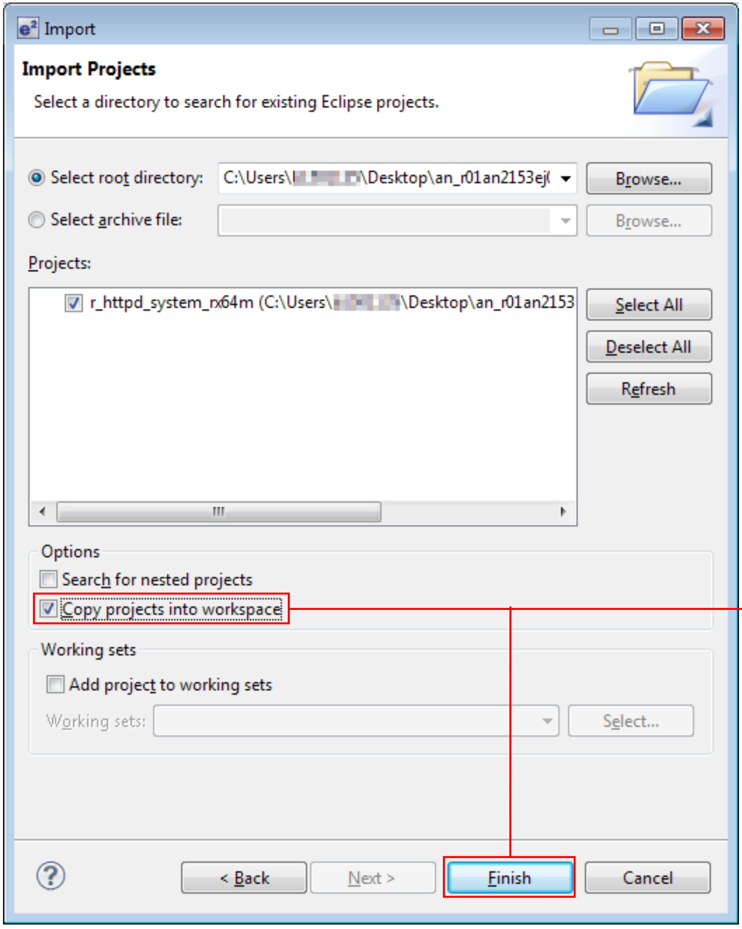
4. Select the project folder associated with this application note and click **OK**.



Folder: r_httpd_system_rx64m

Select this project folder and click **OK**.

5. Check **Copy projects into workspace** and click **Finish**.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: C:\Users\[redacted]\Desktop\an_r01an2153ej(Browse...

○ Select archive file: Browse...

Projects:

☑ r_httpd_system_rx64m (C:\Users\[redacted]\Desktop\an_r01an2153

Options

☐ Search for nested projects

☑ Copy projects into workspace

Working sets

☐ Add project to working sets

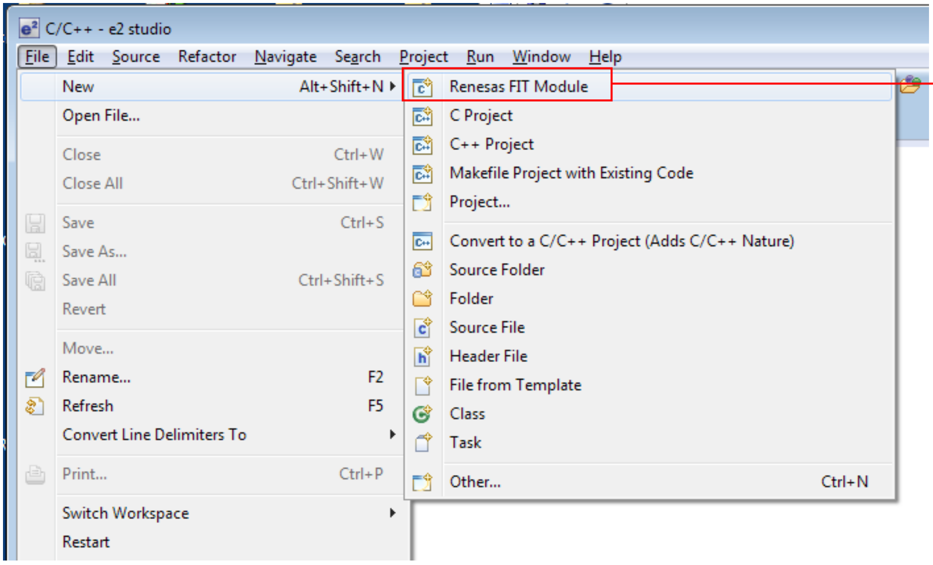
Working sets:

Check this box and click **Finish**.

4.3 Add the Web Server System FIT Modules to the Project

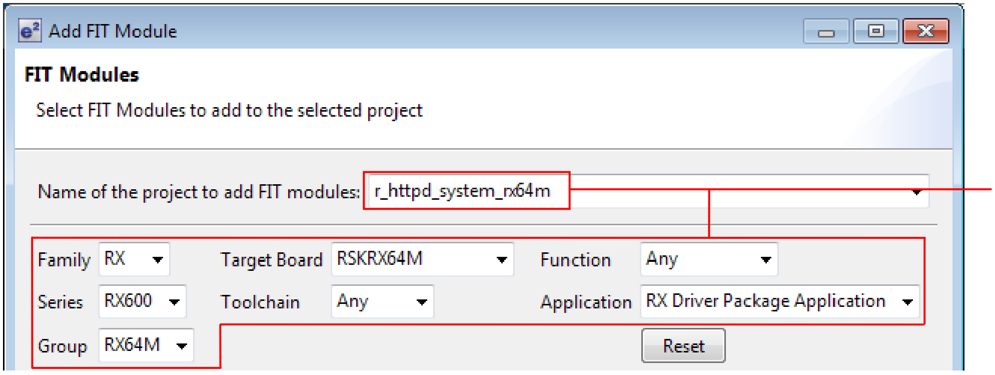
Use the e^2 studio FIT plugin to add the FIT modules used by the web server system to the project.

1. Select **Renesas FIT Module** from **New** in the e^2 studio **File** menu to start the FIT plugin.



Click here.

2. Set the FIT plugin items as shown below.

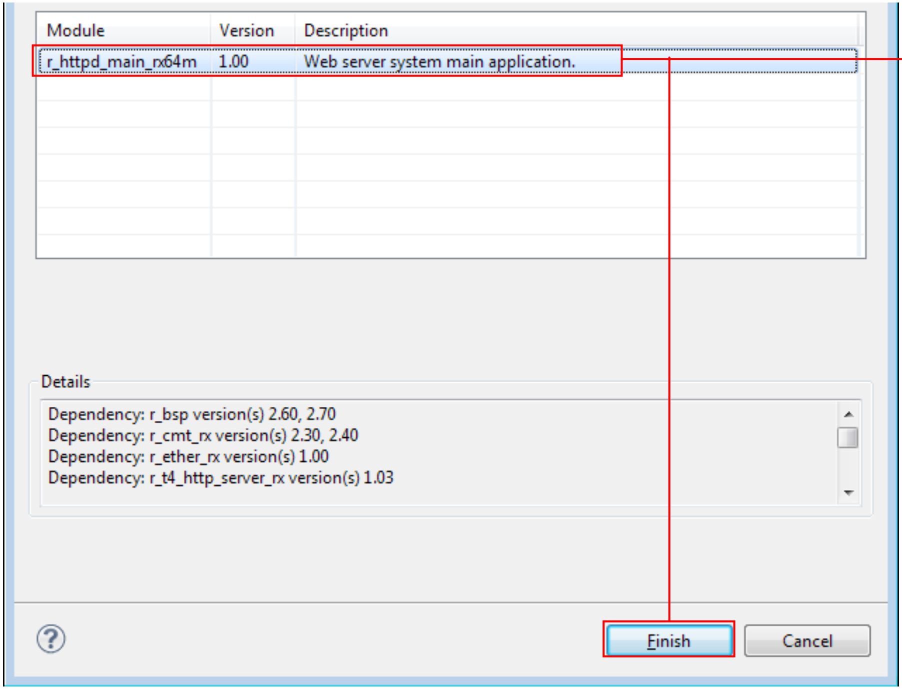


Name of the project to add FIT modules: r_httpd_system_rx64m

Family: RX
Series: RX600
Group: RX64M
Target Board: RSKRX64M
Toolchain: Any
Function: Any
Application: RX Driver Package Application

Set this item.

3. Select **r_httpd_main_rx64m** from the FIT plugin module list and click **Finish**.



Module	Version	Description
r_httpd_main_rx64m	1.00	Web server system main application.

Details

Dependency: r_bsp version(s) 2.60, 2.70
Dependency: r_cmt_rx version(s) 2.30, 2.40
Dependency: r_ether_rx version(s) 1.00
Dependency: r_t4_http_server_rx version(s) 1.03

Select this item
and click **Finish**.

4. A variety of message dialog boxes will be displayed. Click **OK** in all of them.
The above procedure will have installed all the required FIT modules into the project. The project structure after this installation is shown below.

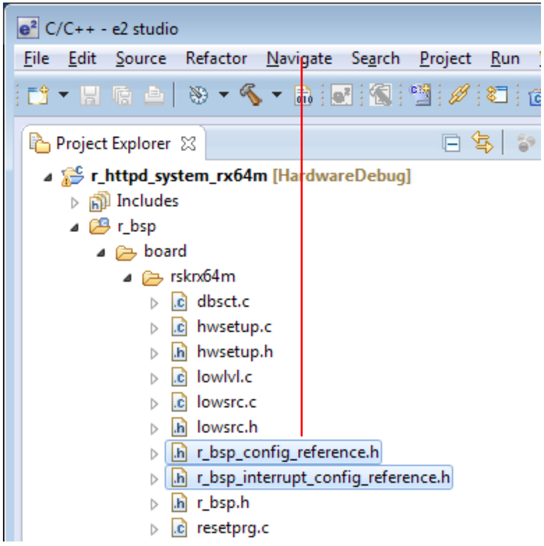
```
r_httpd_system_rx64m        // Web server system project folder
 ├ r_bsp                    // BSP module folder
 ├ r_cmt_rx                 // Compare match timer FIT module folder
 ├ r_ether_rx               // RX Ethernet driver FIT module folder
 ├ r_t4_driver_rx64m        // T4 Ethernet driver interface conversion module folder
 ├ r_t4_http_server_rx      // HTTP server FIT module folder
 ├ r_t4_rx                  // T4 FIT module folder
 ├ r_tfat_rx                // TFAT FIT module folder
 ├ r_usb_basic              // USB Basic Host and Peripheral firmware FIT module folder
 ├ r_usb_hmsc               // USB Host Mass Storage Class driver FIT module folder
 ├ src                      // Main source folder
 ├ HardwareDebug            // Build configuration folder(for debugging)
 ├ doc                      // Web server system document folder
 └ r_config                 // FIT configuration file folder
```

4.4 Set Up Board Support Package (BSP Module)

4.4.1 Copy Configuration File

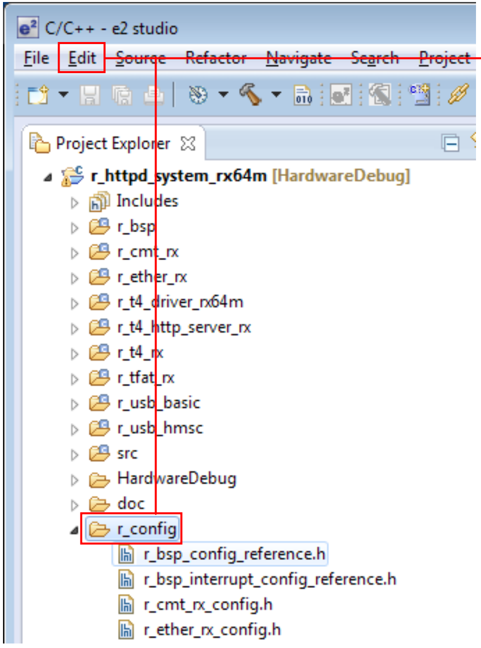
Copy the configuration file for the microcontroller used to the r_config folder.

1. From the e^2 studio project explorer, open **r_bsp/board/rskrx64m** and select two files: **r_bsp_config_reference.h** and **r_bsp_interrupt_config_reference.h**. Then click **Copy** on the **Edit** menu.



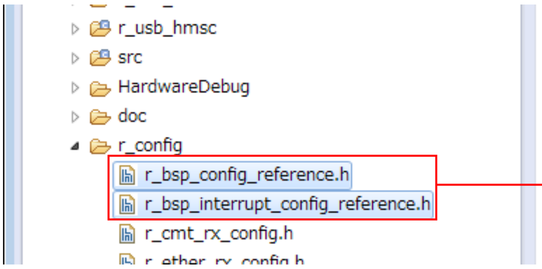
Select the two files and then click **Copy** on the **Edit** menu.

2 Select the **r_config** folder and click **Paste** on the **Edit** menu.



Select the **r_config** folder and click **Paste** on the **Edit** menu.

3. Rename the copied files to **r_bsp_config.h** and **r_bsp_interrupt_config.h**, that is, remove **_reference** from the file names.



Rename these files.

4.4.2 Edit platform.h

Modify platform.h to correspond to the target board being used.

Open r_bsp/platform.h and remove the comment from the include line for the RSKRX64M r_bsp.h file.

r_bsp/platform.h

```
/* RDKRX63N */
//#include "./board/rdkrx63n/r_bsp.h"

/* RDKRX631 */
//#include "./board/rdkrx631/r_bsp.h"

/* RSKRX64M */
#include "./board/rskrx64m/r_bsp.h"

/* RSKRX210 */
//#include "./board/rskrx210/r_bsp.h"

/* HSBRX21AP */
//#include "./board/hsbrx21ap/r_bsp.h"
```

4.5 Modify Configuration

The configuration files for each of the FIT modules that make up the web server system must be modified.

Refer to the manuals and other files in the doc folder for each FIT module for details on the items and their settings in the configuration files.

The places that must be changed in the configuration files to operate this web server system are shown below.

4.5.1 Change Interrupt Stack Size

In this web server system, the main web server processing is performed from the Ethernet controller's interrupt handler. This requires about 2.5 KB of interrupt stack.

Modify the interrupt stack size defined in the r_bsp configuration file as shown below.

r_config/r_bsp_config.h

```
/* Interrupt Stack size in bytes. The Renesas RX toolchain sets the stack
size using the #pragma stacksize directive.
 * If the interrupt stack is the only stack being used then the user will
likely want to increase the default size
 * below.
 */
#pragma stacksize si=0x1000
```

4.5.2 Change Compare Match Timer Driver Settings

Set interrupt priority of compare match timer lower than interrupt priority of the USB driver (IPR=3).

r_config/r_cmt_rx_config.h

```
/* The interrupt priority level to be used for CMT interrupts. */
#define CMT_RX_CFG_IPR         (2)
```

4.5.3 Change USB Driver Settings

Set channel 0 to be unused (USB_NOUSE_PP).

r_config/r_usb_config.h

```
//  #define USB_FUNCSEL_USBIP0_PP       USB_HOST_PP     /* Host Mode */
//  #define USB_FUNCSEL_USBIP0_PP       USB_PERI_PP     /* Peripheral Mode */
    #define USB_FUNCSEL_USBIP0_PP       USB_NOUSE_PP
```

4.5.4 Change T4 Settings

Change the T4 settings as shown below.

Comment out the t4_callback function external reference declaration and add a new external reference declaration for the http_callback function.

r_t4_rx/src/config_tcpudp.c

```
#include "r_t4_itcpip.h"
//extern ER t4_callback(ID cepid, FN fncd , VP p_parblk);
extern ER http_callback(ID cepid, FN fncd , VP p_parblk);
```

Increase the number of TCP reception points to 6 and modify each local point.

r_t4_rx/src/config_tcpudp.c

```
/*** Definition of TCP reception point (only port number needs to be set) ***/
T_TCP_CREP tcp_crep[6] =
{
    /* { attribute of reception point, {local IP address, local port number}} */
    { 0x0000, { 0, 80 }},
    { 0x0000, { 0, 80 }},
    { 0x0000, { 0, 80 }},
    { 0x0000, { 0, 80 }},
    { 0x0000, { 0, 80 }},
    { 0x0000, { 0, 80 }},
};
```

Change the TCP communication end point setting as shown below.

r_t4_rx/src/config_tcpudp.c

```
/***  Definition of TCP communication end point
      (only receive window size needs to be set) ***/
T_TCP_CCEP tcp_ccep[6] =
{
    /* { attribute of TCP communication end point,
         top address of transmit window buffer, size of transmit window
         buffer,top address of receive window buffer, size of receive window
         buffer, address of callback routine }
    */
    { 0, 0, 0, 0, 1460, http_callback },
    { 0, 0, 0, 0, 1460, http_callback },
    { 0, 0, 0, 0, 1460, http_callback },
    { 1, 0, 0, 0, 1460, http_callback },
    { 1, 0, 0, 0, 1460, http_callback },
    { 1, 0, 0, 0, 1460, http_callback },
```

Change in 10 ms to the 2MSL Wait Time.

【r_t4_rx/src/config_tcpudp.c】

```
/***  2MSL wait time (unit:10ms)  ***/
const UH    _tcp_2msl[] =
{
    (1),      /* 10 ms */
    (1),      /* 10 ms */
};
```

4.5.5 Change HTTP Server Settings

Change the CGI_FILE_NAME_TABLE_LIST as shown below.

r_config/r_t4_http_server_rx_config.h

```
/*#define CGI_FILE_NAME_TABLE_LIST \*/
/*  {"cgi_smpl.cgi", NULL},   \*/
extern ER cgi_sample_function(ID cepid, void *res_info);
#define CGI_FILE_NAME_TABLE_LIST \
    {"cgi_smpl.cgi", cgi_sample_function, NULL},   \
```

Change in 6 to maximum number of clients that can be accepted at the same time to match tcp_ccep table of r_t4_rx/src/config_tcpudp.c.

r_config/r_t4_http_server_rx_config.h

```
// set same value number of CEPID in config_tcpudp.c
#define  HTTP_TCP_CEP_NUM  6
```

4.6 Modify Source Code

The places that must be changed in the source code to operate this web server system are shown below.

4.6.1 Allows multiple interrupts

This system uses multiple interrupts.

Allow interrupts before calling the **_process_tcpip** function called in the **lan_inthdr** handler function and the **timer_interrupt** handler function in the **t4_driver.c**.

r_t4_driver_rx64m/src/t4_driver.c

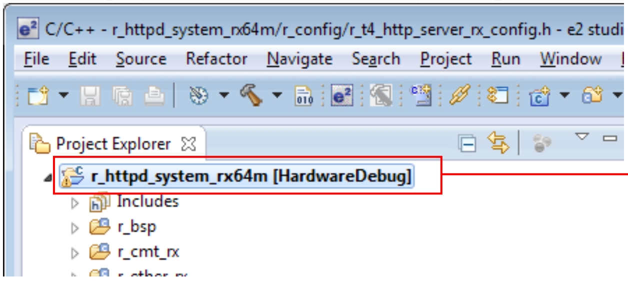
```
/******************************************************************************
Functions (Interrput handler)
******************************************************************************/

void timer_interrupt(void *pdata)
{
 R_BSP_InterruptsEnable();

 if (tcpip_flag == 1)
 {
   _process_tcpip();
   tcpudp_time_cnt++;
 }

 /* for wait function */
 if (wait_timer < 0xFFFF)
 {
   wait_timer++;
 }
}

void lan_inthdr(void *ppram)  // callback from r_ether.c
{
 R_BSP_InterruptsEnable();

 if (tcpip_flag == 1)
 {
   _process_tcpip();
 }
}
```

5. Verify Operation

5.1 Build the Project

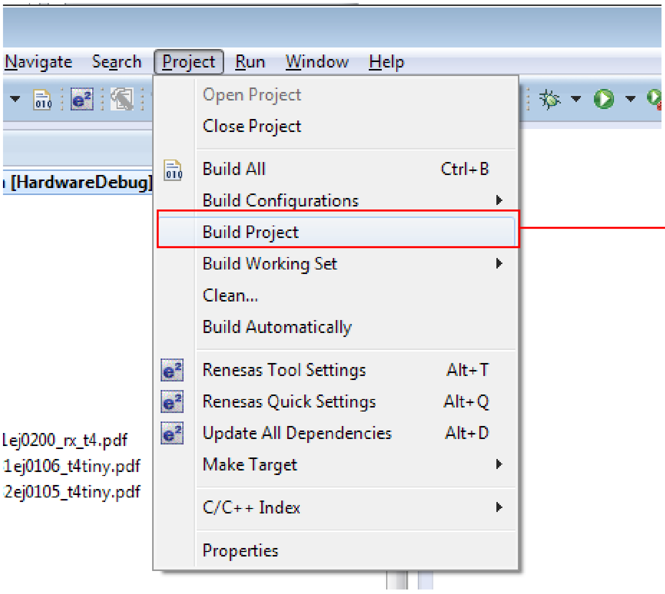
Use the following procedure to build the project and generate a load module.

1. Click the project to build from the **Project Explorer**.


C/C++ - r_httpd_system_rx64m/r_config/r_t4_http_server_rx_config.h - e2 studi
File Edit Source Refactor Navigate Search Project Run Window
Project Explorer
r_httpd_system_rx64m [HardwareDebug]
Includes
r_bsp
r_cmt_rx
r_other_rx

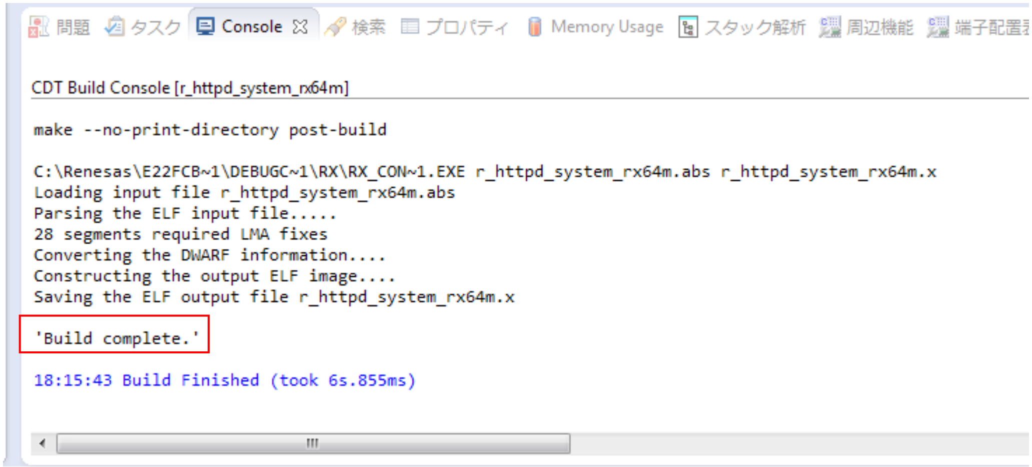
Click here.

2. Click **Build project** from the **Project** menu.



Click here.

3. When “Build complete” is displayed on the **Console panel**, the build will have completed.



```
CDT Build Console [r_httpd_system_rx64m]

make --no-print-directory post-build

C:\Renesas\E22FCB~1\DEBUGC~1\RX\RX_CON~1.EXE r_httpd_system_rx64m.abs r_httpd_system_rx64m.x
Loading input file r_httpd_system_rx64m.abs
Parsing the ELF input file.....
28 segments required LMA fixes
Converting the DWARF information....
Constructing the output ELF image....
Saving the ELF output file r_httpd_system_rx64m.x

'Build complete.'

18:15:43 Build Finished (took 6s.855ms)
```

5.2 Prepare for Debugging

5.2.1 Configure Hardware

The evaluation board must be configured before starting debugging.

A table of the required equipment and its configuration are shown below.

Table 5.2.1.1 Hardware Configuration

No.	Device	Supplementary Information
1	Development PC	Personal computer used for development
2	Evaluation board (Renesas Starter Kit+ for RX64M)	
3	USB memory	Memory that is formatted as either FAT or FAT32.
4	Client PC (web browser)	The development PC can be used for this function.
5	One of the following must be provided as a network environment for connecting the client PC to the RSK (web server). 1. If a switching hub is used a. Switching hub b. LAN cable (straight) × 3 2. If cross cables are used a. LAN cable (cross) × 2	If cross cables and two Ethernet channels are used, then the client PC must have two LAN ports. When only one Ethernet channel is used, the number of LAN cables required will be as follows. 1. If a switching hub is used LAN cable (straight) × 2 2. If cross cables are used LAN cable (cross) × 1

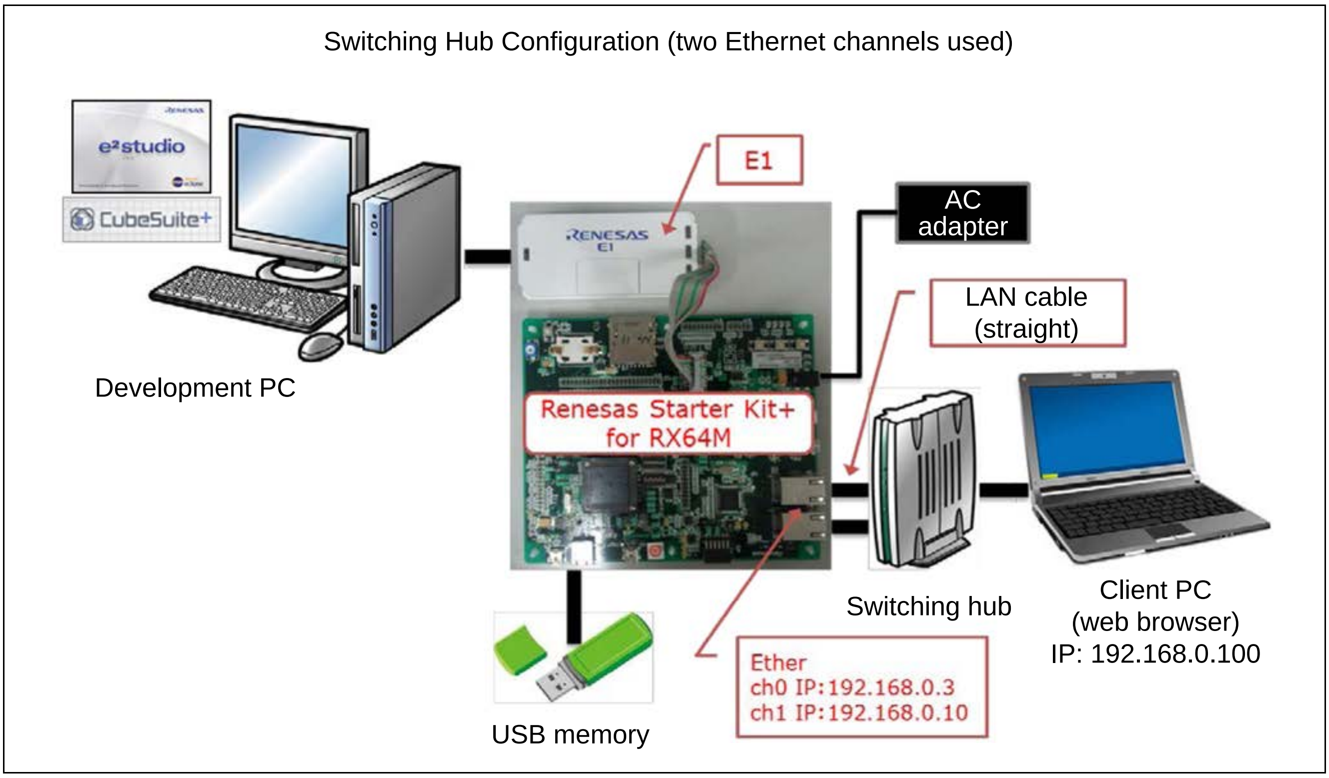


Figure 5.2.1.1 Switching Hub Configuration (Two Ethernet Channels Used)

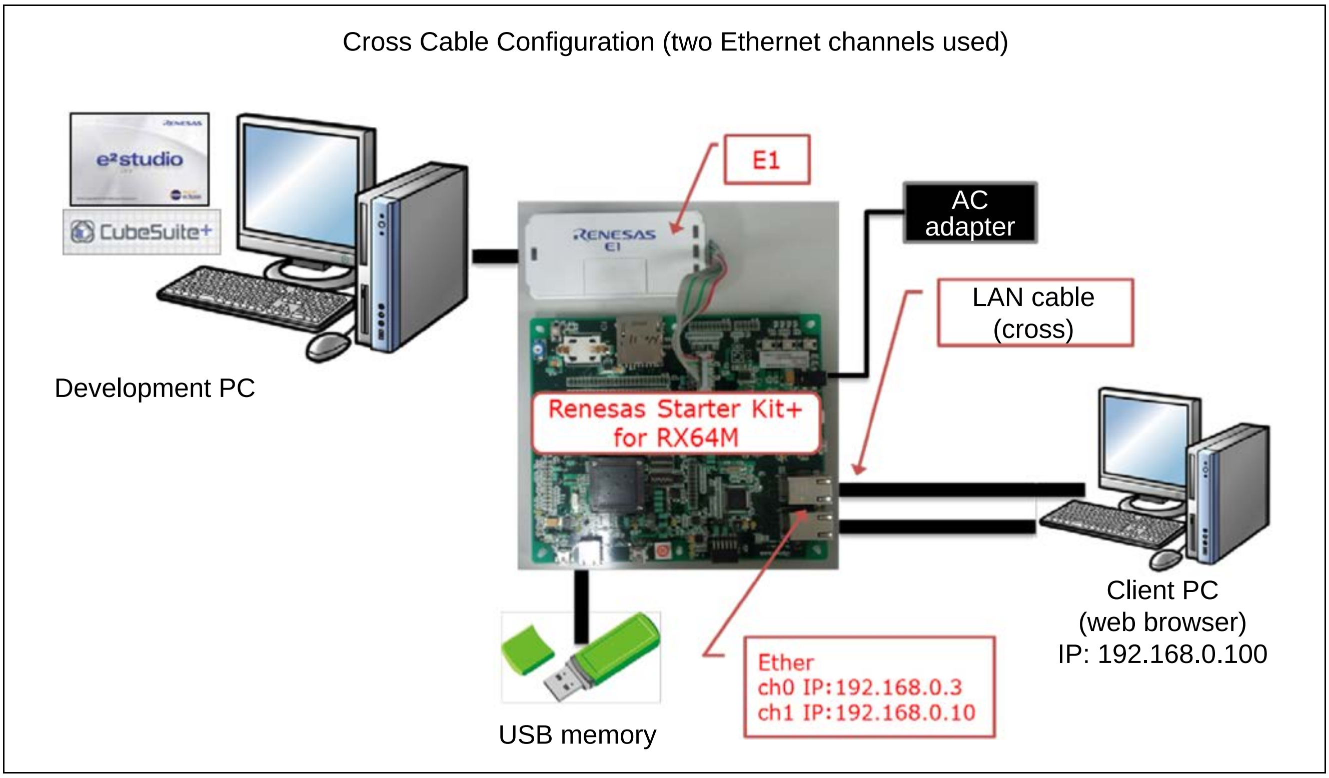

Cross Cable Configuration (two Ethernet channels used)
e²studio
CubeSuite+
Development PC
E1
RENESAS E1
AC adapter
LAN cable (cross)
Renesas Starter Kit+ for RX64M
Client PC (web browser) IP: 192.168.0.100
USB memory
Ether
ch0 IP:192.168.0.3
ch1 IP:192.168.0.10

Figure 5.2.1.2 Cross Cable Configuration (Two Ethernet Channels Used)

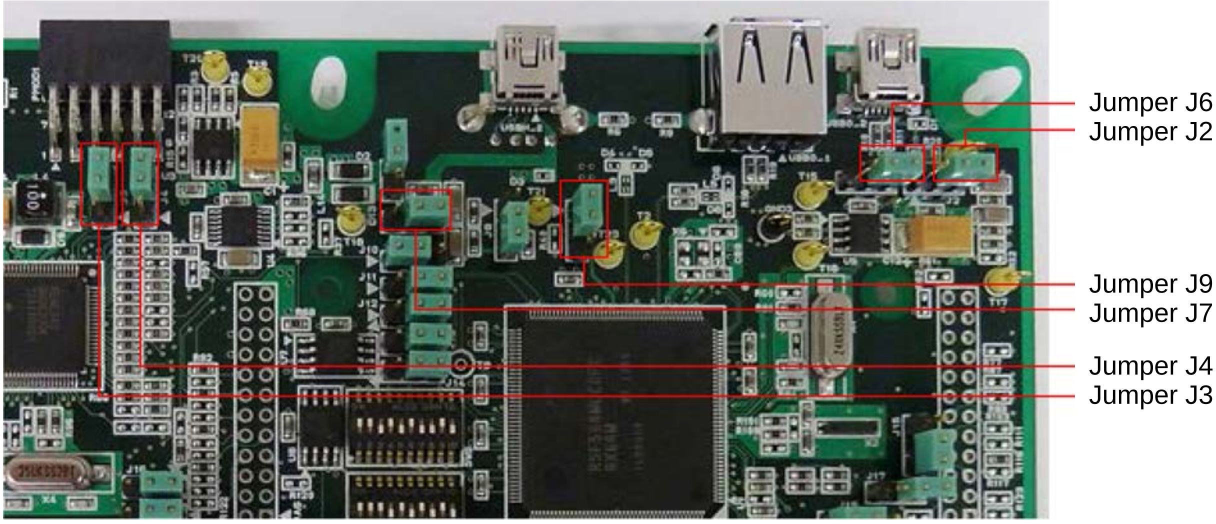
5.2.2 Set Up the Evaluation Board

The evaluation board settings required to operate the web server system are shown below.

1. Set the USB ch0 mode (host/peripheral). Set jumpers J2 and J6 to match the setting of USB_FUNCSEL_USBIP0_PP in r_usb_config.h.
2. Set the USB ch1 mode (host/peripheral). Set jumpers J7 and J9 to match the setting of USB_FUNCSEL_USBIP1_PP in r_usb_config.h.
3. Specify the PHY IC channel used to control the PHY IC from the Ethernet controller. Set jumpers J3 and J4 to match the settings of ETHER_CFG_CH0_PHY_ACCESS and ETHER_CFG_CH1_PHY_ACCESS in r_ether_rx_config.h.

Table 5.2.2.1 Jumper Settings

No.	Setting	Jumper	Setting
1	When use USB0 in host mode. (USB_FUNCSEL_USBIP0_PP = USB_HOST_PP)	J2	Short 1 to 2.
		J6	Short 2 to 3.
	When use USB0 in peripheral mode. (USB_FUNCSEL_USBIP0_PP = USB_PERI_PP)	J2	Short 2 to 3.
		J6	Short 1 to 2.
2	When use USB1 in host mode. (USB_FUNCSEL_USBIP1_PP = USB_HOST_PP)	J7	Short 1 to 2.
		J9	Short 2 to 3.
	When use USB1 in peripheral mode. (USB_FUNCSEL_USBIP1_PP = USB_PERI_PP)	J7	Short 2 to 3.
		J9	Short 1 to 2.
3	Control the PHY IC with ch1.	J3	Short 2 to 3.
		J4	Short 2 to 3.



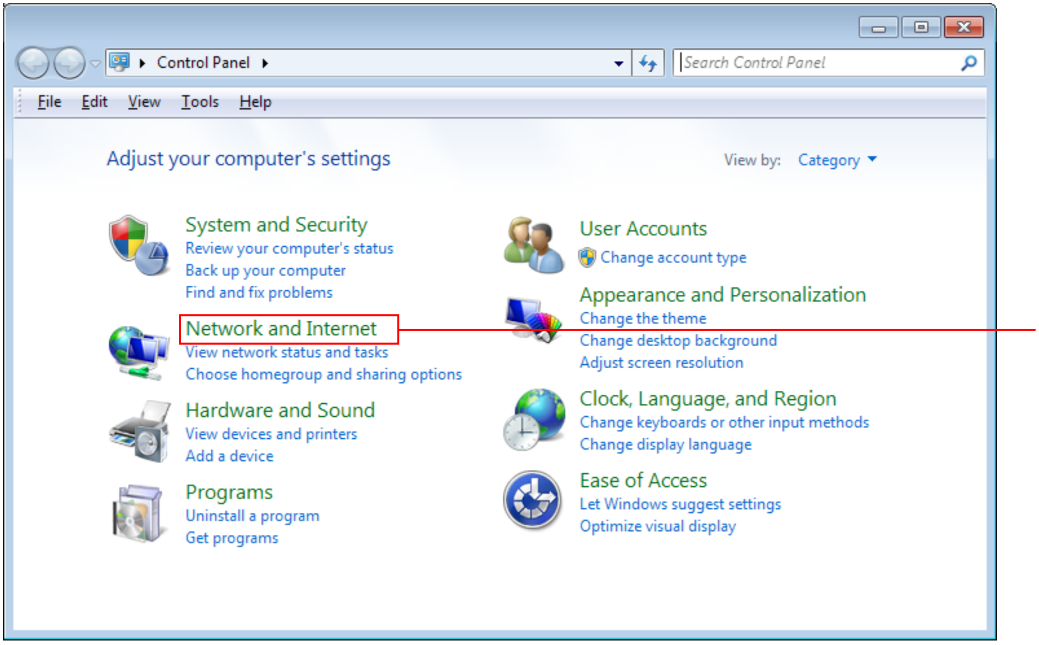
Note: The image and the actual settings differ.

Figure 5.2.2.1 Renesas Starter Kit+ for RX64M Jumper Locations

5.2.3 Set Up Client PC

Set up the network on the client PC. This section shows the procedure when using Windows 7 as an example.

1. Open the **Control Panel** on the client PC and click **Network and Internet**.



Adjust your computer's settings

View by: Category

System and Security
Review your computer's status
Back up your computer
Find and fix problems

Network and Internet
View network status and tasks
Choose homegroup and sharing options

Hardware and Sound
View devices and printers
Add a device

Programs
Uninstall a program
Get programs

User Accounts
Change account type

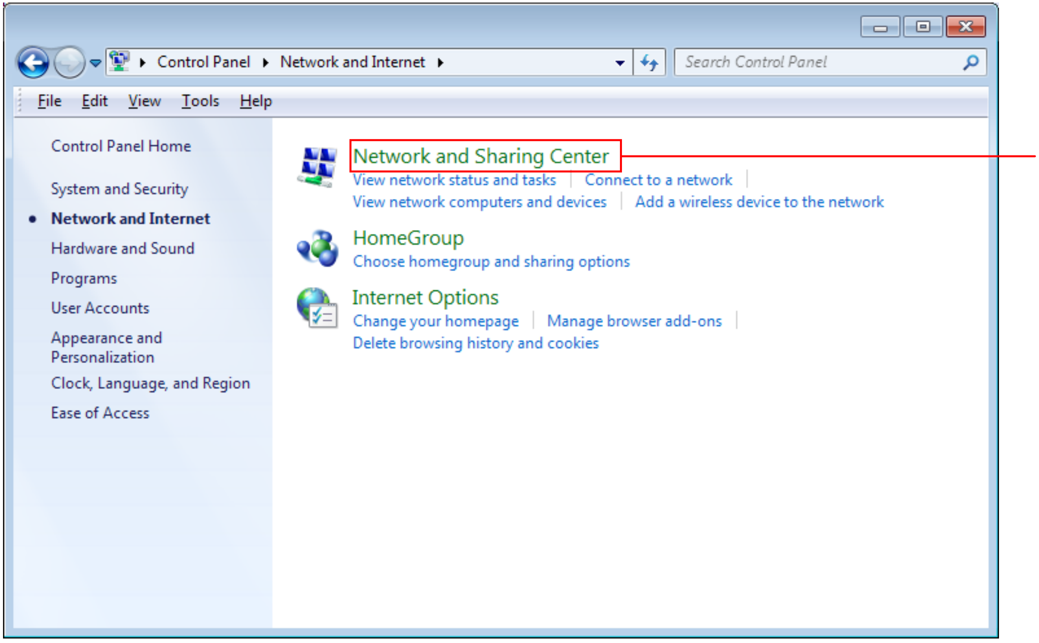
Appearance and Personalization
Change the theme
Change desktop background
Adjust screen resolution

Clock, Language, and Region
Change keyboards or other input methods
Change display language

Ease of Access
Let Windows suggest settings
Optimize visual display

Click here.

2. Click **Network and Sharing Center**.



Control Panel Home

System and Security
Network and Internet
Hardware and Sound
Programs
User Accounts
Appearance and Personalization
Clock, Language, and Region
Ease of Access

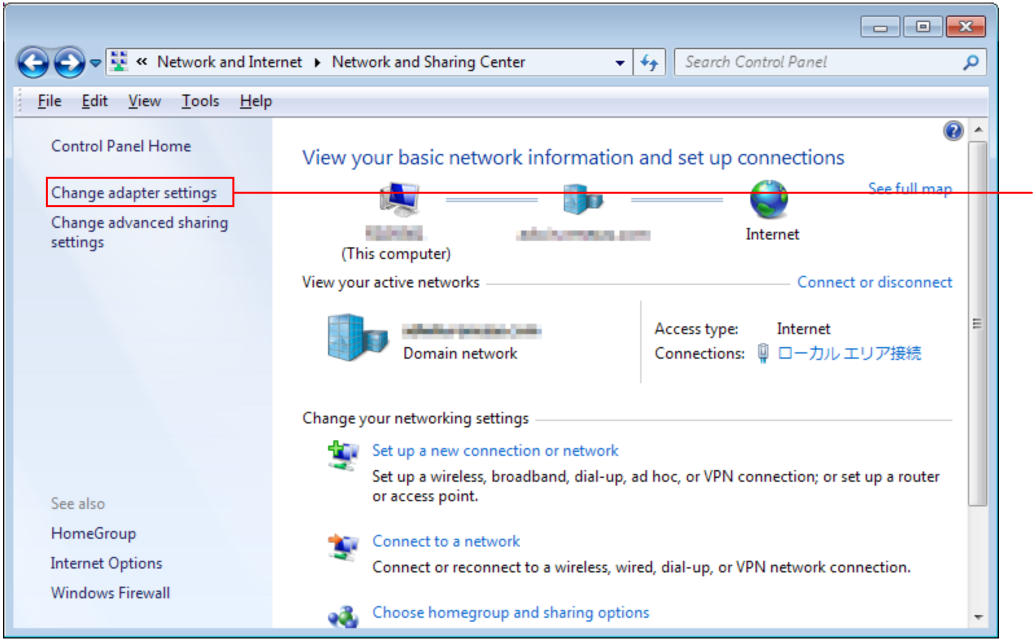
Network and Sharing Center
View network status and tasks | Connect to a network |
View network computers and devices | Add a wireless device to the network

HomeGroup
Choose homegroup and sharing options

Internet Options
Change your homepage | Manage browser add-ons |
Delete browsing history and cookies

Click here.

3. Click **Change adapter settings**.



View your basic network information and set up connections

View your active networks

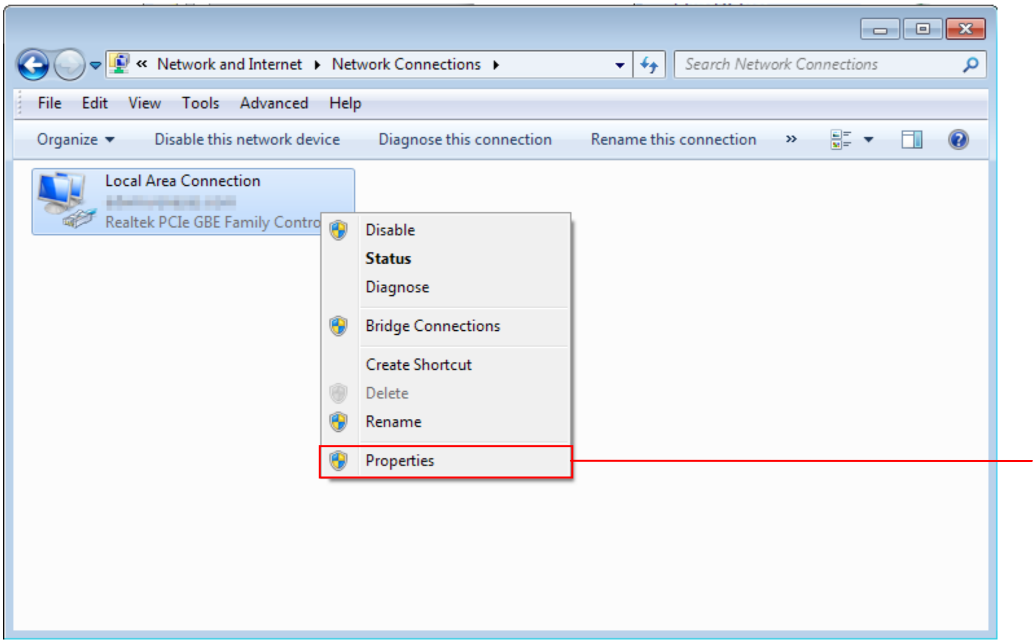
Change your networking settings

Set up a wireless, broadband, dial-up, ad hoc, or VPN connection; or set up a router or access point.

Connect or reconnect to a wireless, wired, dial-up, or VPN network connection.

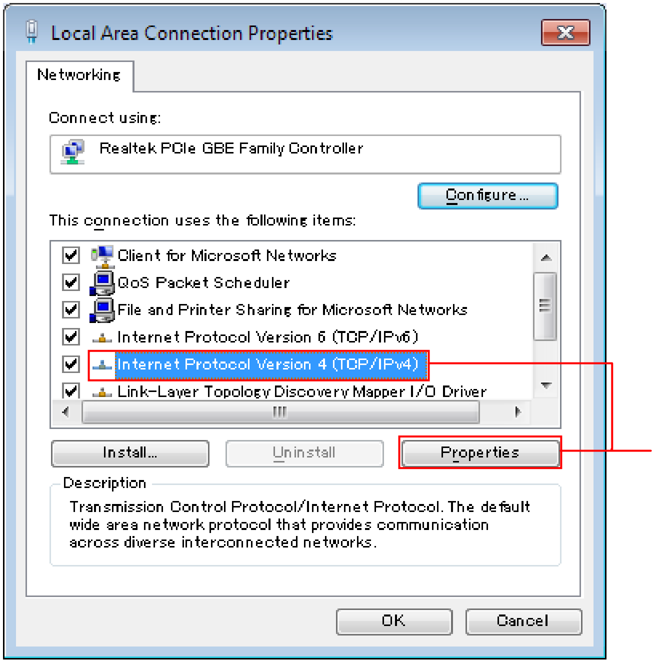
Click here.

4. Right click **Local Area Connection** and select **Properties**.



Click here.

5. Select **Internet Protocol Version 4 (TCP/IPv4)** and click **Properties**.



Local Area Connection Properties

Networking

Connect using:

Realtek PCIe GBE Family Controller

Configure...

This connection uses the following items:

- ☑ Client for Microsoft Networks
- ☑ QoS Packet Scheduler
- ☑ File and Printer Sharing for Microsoft Networks
- ☑ Internet Protocol Version 6 (TCP/IPv6)
- ☑ Internet Protocol Version 4 (TCP/IPv4)
- ☑ Link-Layer Topology Discovery Mapper I/O Driver

Install... Uninstall Properties

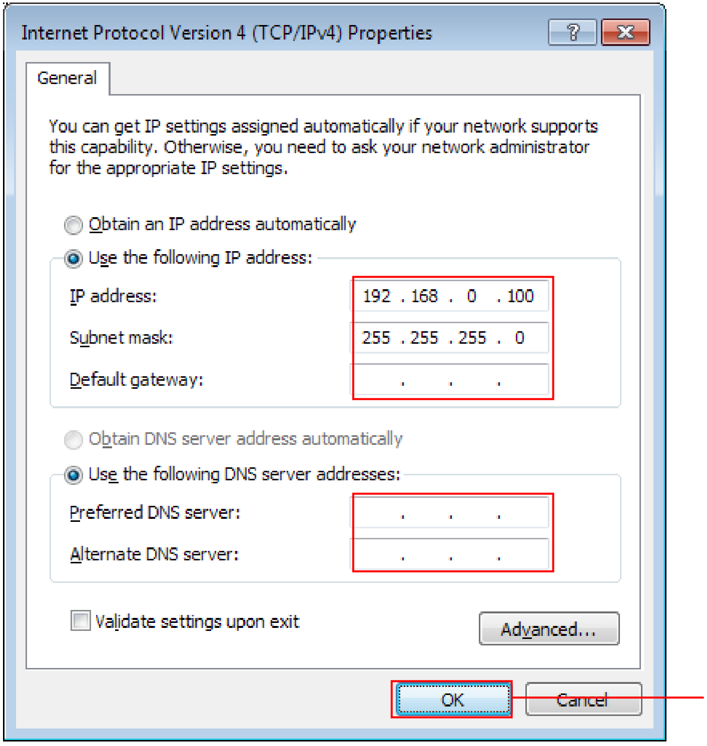
Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

OK Cancel

Click here.

6. The IP address and other settings will be displayed. Set these as shown below and click **OK**.



Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically

● Use the following IP address:

IP address: 192 . 168 . 0 . 100

Subnet mask: 255 . 255 . 255 . 0

Default gateway: . . .

○ Obtain DNS server address automatically

● Use the following DNS server addresses:

Preferred DNS server: . . .

Alternate DNS server: . . .

☐ Validate settings upon exit

Advanced...

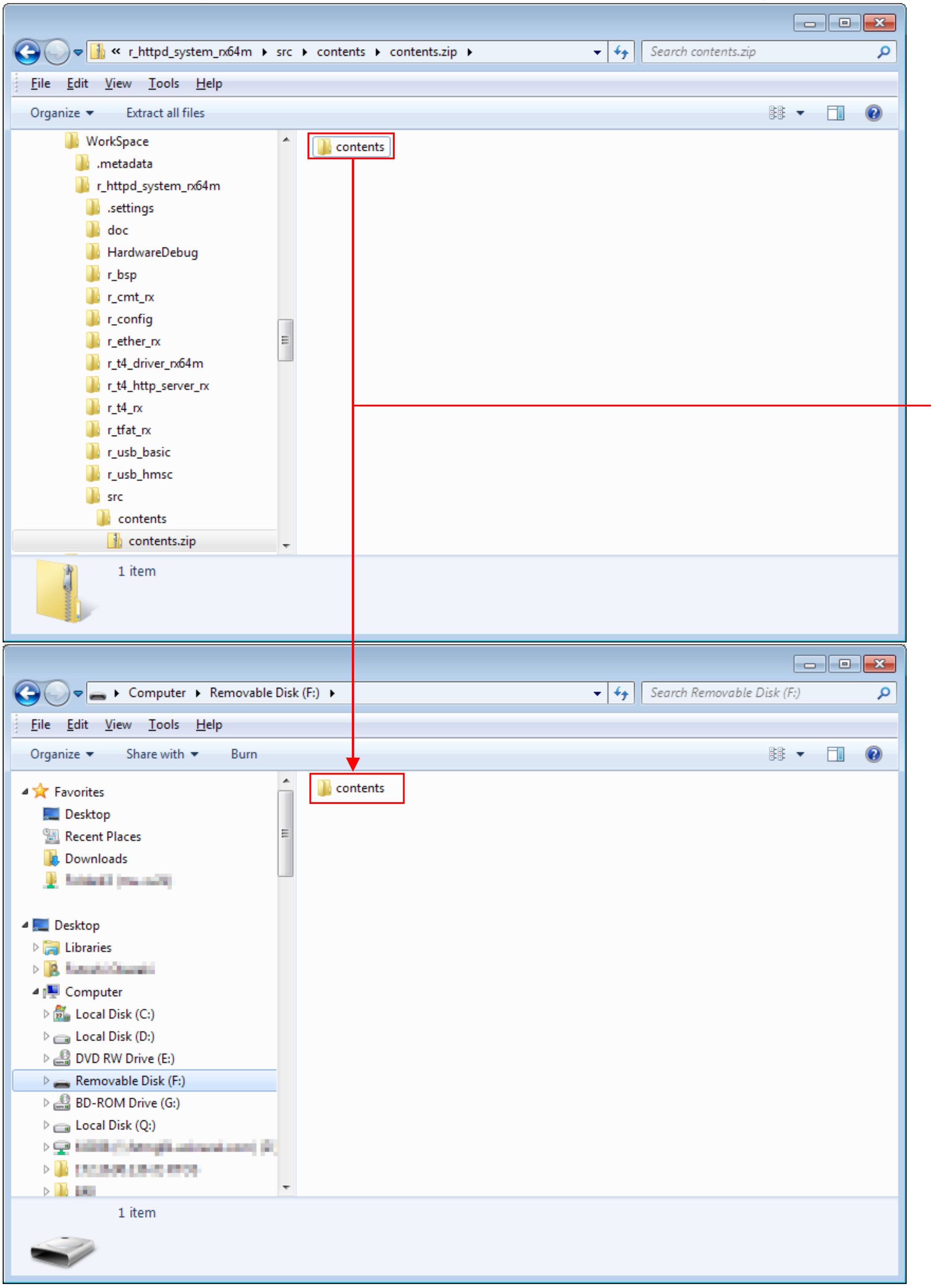
OK Cancel

Click here.

5.2.4 Prepare USB Memory

Store the HTML content on the USB memory.

1. Open the **src** folder in the project and then open the **contents** folder in that folder. Open the **contents.zip** file in the **contents** folder. Copy the **contents** folder in the **demo** folder to the USB memory.

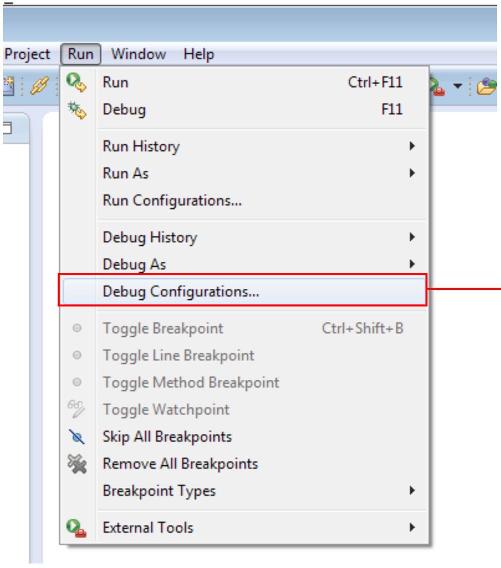


Copy the contents folder.

5.3 Debug the Project

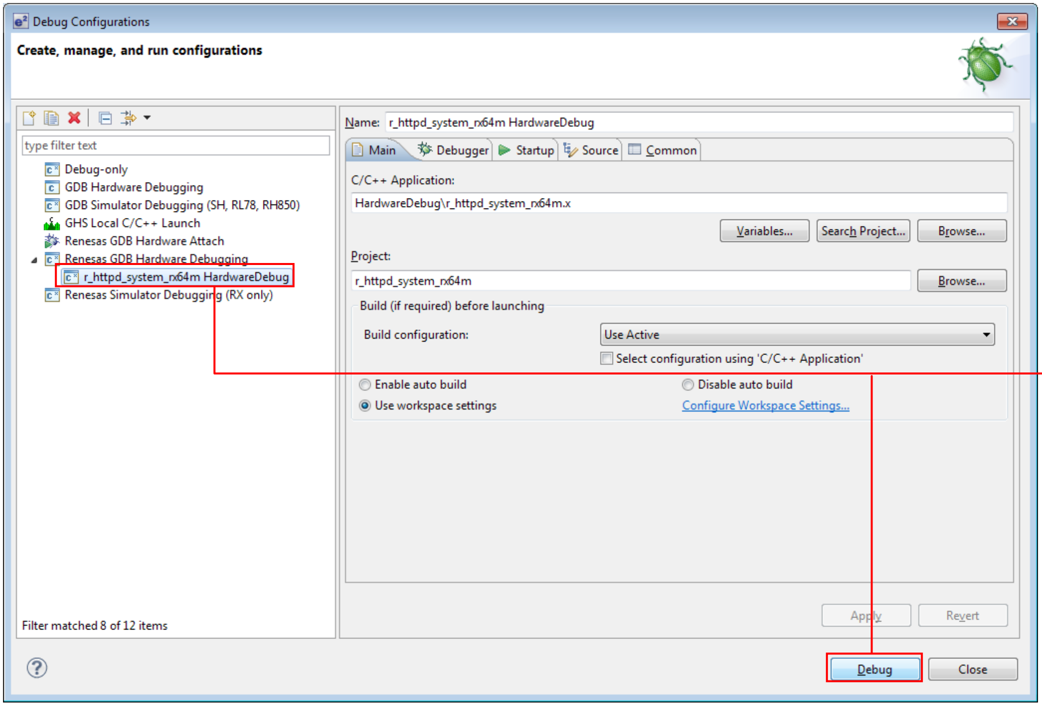
Use the following procedure to start debugging the project.

1. Connect the development PC to the E1 emulator with a USB cable.
2. Connect the evaluation board (Renesas Starter Kit+ for RX64M) to the adapter and turn on the power.
3. Click **Debug Configurations** in the e^2 studio **Run** menu.



Click here.

4. Click **r_httpd_system_rx64m HardwareDebug** under **Renesas GDB Hardware Debugging** and click **Debug.**


Debug Configurations
Create, manage, and run configurations
type filter text
Debug-only
GDB Hardware Debugging
GDB Simulator Debugging (SH, RL78, RH850)
GHS Local C/C++ Launch
Renesas GDB Hardware Attach
Renesas GDB Hardware Debugging
r_httpd_system_rx64m HardwareDebug
Renesas Simulator Debugging (RX only)
Filter matched 8 of 12 items
Main
Debugger
Startup
Source
Common
Variables...
Search Project...
Browse...
Browse...
Configure Workspace Settings...
Apply
Revert
Debug
Close

Name: r_httpd_system_rx64m HardwareDebug

C/C++ Application: HardwareDebug\r_httpd_system_rx64m.x

Project: r_httpd_system_rx64m

Build (if required) before launching

Build configuration: Use Active

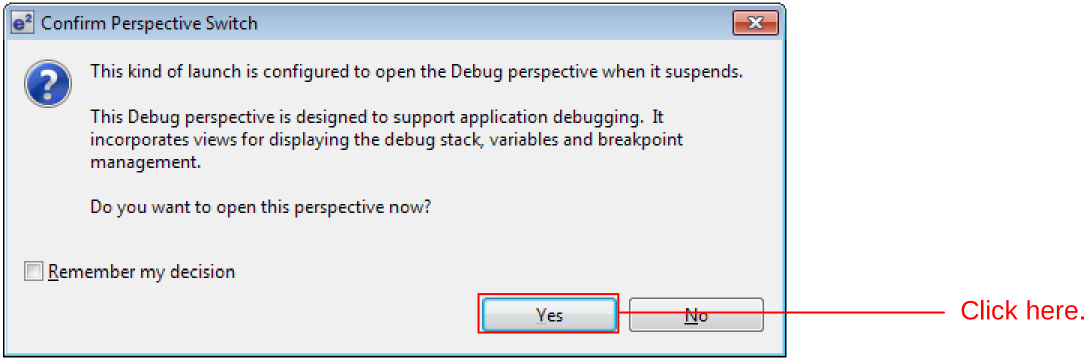
□ Select configuration using 'C/C++ Application'

○ Enable auto build ○ Disable auto build

● Use workspace settings

Click here.

When the following message is displayed, click **Yes**.



Confirm Perspective Switch

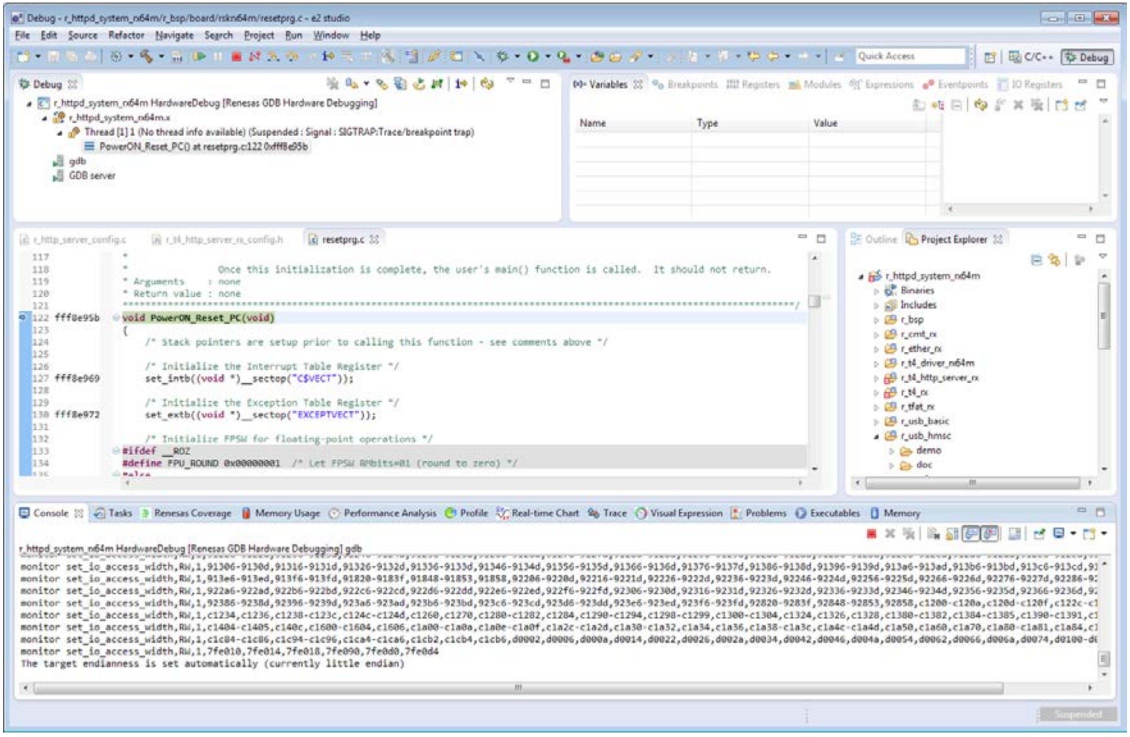
This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

☐ Remember my decision

When the load module download completes, a **Debug** perspective opens.



Debug

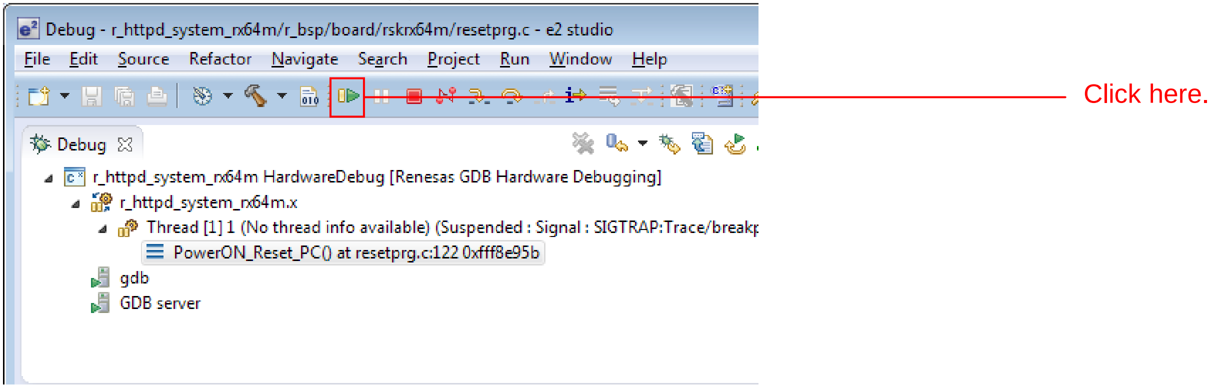
- r_httpd_system_rx64m HardwareDebug [Renesas GDB Hardware Debugging]
  - r_httpd_system_rx64m.x
    - Thread [1] 1 (No thread info available) (Suspended : Signal : SIGTRAP:Trace/breakpoint trap)
      - PowerON_Reset_PC() at resetprg.c:122 0xfff8e95b
  - gdb
  - GDB server

```
             *
             *          Once this initialization is complete, the user's main() function is called.  It should not return.
             * Arguments    : none
             * Return value : none
             ******************************************************************************************************/
fff8e95b     void PowerON_Reset_PC(void)
             {
                 /* Stack pointers are setup prior to calling this function - see comments above */

                 /* Initialize the Interrupt Table Register */
fff8e969         set_intb((void *)__sectop("C$VECT"));

                 /* Initialize the Exception Table Register */
fff8e972         set_extb((void *)__sectop("EXCEPTVECT"));

                 /* Initialize FPSW for floating-point operations */
             #ifdef __ROZ
             #define FPU_ROUND 0x00000001  /* Let FPSW RMbits=01 (round to zero) */
```

r_httpd_system_rx64m HardwareDebug [Renesas GDB Hardware Debugging] gdb

```
monitor set_io_access_width,RW,1,91306-9130d,91316-9131d,91326-9132d,91336-9133d,91346-9134d,91356-9135d,91366-9136d,91376-9137d,91386-9138d,91396-9139d,913a6-913ad,913b6-913bd,913c6-913cd,91
monitor set_io_access_width,RW,1,913e6-913ed,913f6-913fd,91820-9183f,91848-91853,91858,92206-9220d,92216-9221d,92226-9222d,92236-9223d,92246-9224d,92256-9225d,92266-9226d,92276-9227d,92286-92
monitor set_io_access_width,RW,1,922a6-922ad,922b6-922bd,922c6-922cd,922d6-922dd,922e6-922ed,922f6-922fd,92306-9230d,92316-9231d,92326-9232d,92336-9233d,92346-9234d,92356-9235d,92366-9236d,92
monitor set_io_access_width,RW,1,92386-9238d,92396-9239d,923a6-923ad,923b6-923bd,923c6-923cd,923d6-923dd,923e6-923ed,923f6-923fd,92820-9283f,92848-92853,92858,c1200-c120a,c120d-c120f,c122c-c1
monitor set_io_access_width,RW,1,c1234,c1236,c1238-c123c,c124c-c124d,c1260,c1270,c1280-c1282,c1284,c1290-c1294,c1298-c1299,c1300-c1304,c1324,c1326,c1328,c1380-c1382,c1384-c1385,c1390-c1391,c1
monitor set_io_access_width,RW,1,c1404-c1405,c140c,c1600-c1604,c1606,c1a00-c1a0a,c1a0e-c1a0f,c1a2c-c1a2d,c1a30-c1a32,c1a34,c1a36,c1a38-c1a3c,c1a4c-c1a4d,c1a50,c1a60,c1a70,c1a80-c1a81,c1a84,c1
monitor set_io_access_width,RW,1,c1c84-c1c86,c1c94-c1c96,c1ca4-c1ca6,c1cb2,c1cb4,c1cb6,d0002,d0006,d000a,d0014,d0022,d0026,d002a,d0034,d0042,d0046,d004a,d0054,d0062,d0066,d006a,d0074,d0100-d0
monitor set_io_access_width,RW,1,7fe010,7fe014,7fe018,7fe090,7fe0d0,7fe0d4
The target endianness is set automatically (currently little endian)
```

5. Click **Resume** on the toolbar. The program will be executed and a break will occur at the start of the main function.



Debug - r_httpd_system_rx64m/r_bsp/board/rskrx64m/resetprg.c - e2 studio
File Edit Source Refactor Navigate Search Project Run Window Help
Click here.

Debug

- r_httpd_system_rx64m HardwareDebug [Renesas GDB Hardware Debugging]
  - r_httpd_system_rx64m.x
    - Thread [1] 1 (No thread info available) (Suspended : Signal : SIGTRAP:Trace/breakp
      - PowerON_Reset_PC() at resetprg.c:122 0xfff8e95b
  - gdb
  - GDB server

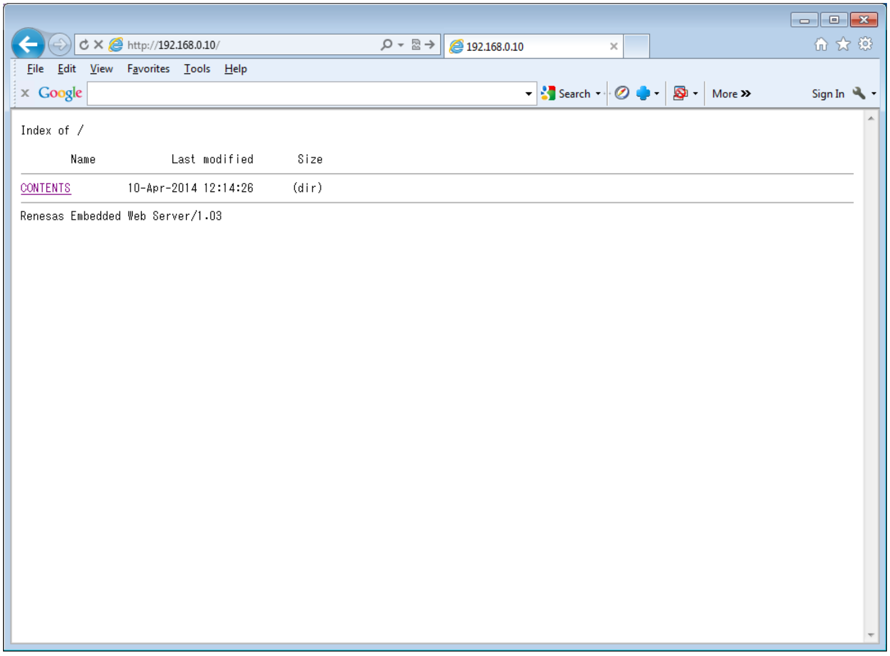
After the break at the start of the main function, click **Resume** on the tool bar again.

6. Start a web browser on the client PC and enter the following address according to which port the LAN cable is connected.

Ethernet Port Number	Web Server Address
0	http://192.168.0.3
1	http://192.168.0.10

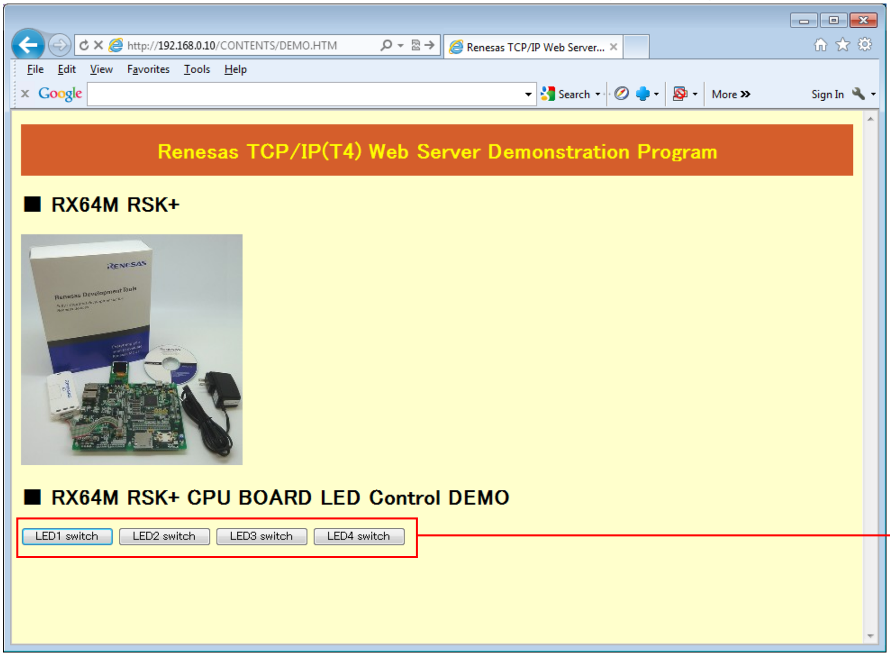
Note: Note that the web address can be changed in the configuration.

A list of files in the root directory on the USB memory will be displayed. The file name is listed in the Name field, the last date on which the file was changed is listed in the Last modified field, and for directories, (dir) is listed as in the size field while for files, the size is shown in bytes. Click Parent Directory to move to the next higher directory.



```
Index of /

        Name            Last modified       Size
CONTENTS          10-Apr-2014 12:14:26     (dir)

Renesas Embedded Web Server/1.03
```

7. Click **CONTENTS** and then click the file **DEMO.HTM**. This will display a page like the one shown below. The LEDs on the board can be controlled (turned on or off) by pressing the **LEDx** button.


http://192.168.0.10/CONTENTS/DEMO.HTM
Renesas TCP/IP Web Server...
File Edit View Favorites Tools Help
Google
Search
More »
Sign In
Renesas TCP/IP(T4) Web Server Demonstration Program
■ RX64M RSK+
■ RX64M RSK+ CPU BOARD LED Control DEMO
LED1 switch
LED2 switch
LED3 switch
LED4 switch

Click here.

6. Web Server Specifications

6.1 Performance Overview

This is a simple web server that is implemented based on the HTTP/1.0 specifications. This web server is intended to serve as a base when a user develops their own web server to be embedded in an end product and that web server will run under M3S-T4-Tiny (referred to as T4 in the remainder of this document). This web server does not included any countermeasures for attacks such as SYN-FLOOD and does not include any security functions. Therefore it is not appropriate for applications in which it is operated as a server connected to the internet waiting on a www port (number 80). This sample program as developed assuming it would be used only in local networks in which malicious actors are not present, such as a network within a business office or factory. Also, the file names it can handle are limited to short file names only.

Note that except for file I/O, this web server operates on microcontroller internal memory only and does not require any special memory. While its processing performance is affected by RAM capacity, this parameter is defined in the program so that it can be set flexibly. In this web server the memory usage is set appropriately for the ROM/RAM capacity of the RX64M microcontroller.

The table below lists the performance of this web server.

Table 6.1.1 Web Server Performance

Item	Performance
ROM size	About 6.6 KB
RAM size	About 36 KB (About 5 KB × number of simultaneous connected clients + α)
Number of simultaneous connections	5 clients (this parameter can be set)
CGI functions	Functions that can remotely control the microcontroller from the web browser.

6.2 Operation Overview

Compared to the web servers (such as Apache) that are widely used on the internet, this web server holds the set of functions implemented to an absolute minimum. Furthermore, it is implemented with nonblocking calls to make it easy to use in embedded application, and the application can perform web server processing simply by calling R_httpd() periodically. The function R_httpd() monitors all communication endpoints (normally called sockets) and transitions to the connection wait state if a socket goes to the disconnected state. Communication processing is performed in the T4 API function _process_tcpip(), and in this web server, this API function is called from timer interrupts and Ethernet interrupts. To report the completion of processing the _process_tcpip() function calls a callback function. HTTP data analysis processing and data generation processing is performed in this callback function.

The processing time required by these interrupt processing operations, including activation of the _process_tcpip() function can vary greatly depending on the performance of the transmit/receive drivers and the implementation of the callback routine. Accordingly, if necessary, operation of the application can be given priority by reducing the priority of these interrupts or by disabling interrupts entirely.

Furthermore, the behavior of this web server can be customized by modifying macro definitions in the configuration file, r_t4_http_server_rx_config.h.

6.3 CGI Functions

This web server provide simplified CGI (Common Gateway Interface) functions. CGI is a mechanism for calling user function in a web browser according to requests from that web browser. In this web browser, when a URL set in advance as a CGI file is requested, the corresponding internal function is called.

6.4 Configuration

The web server's behavior can be customized by modifying the macro definitions in the configuration file (r_t4_http_server_rx_config.h).

- Server header field: HTTPD_VERSION_CODE
  The data stored in the server header field transmitted to the web browser when communicating with the web browser can be specified.
- Root directory: ROOT_DIR
  Which directory in the external memory is taken to be the root directory can be specified.
  Examples: #define ROOT_DIR “”
  #define ROOT_DIR “user”
  #define ROOT_DIR “user/root_dir”
- Display or don’t display index page: INDEXES
  The behavior when a directory is specified by the web browser can be specified.
  When 1 is specified, the response is the directory contents.
  When 0 is specified, the response is the file specified by DEFAULT_FILE_NAME.
- Response file when index page not displayed: DEFAULT_FILE_NAME
  This is the file that is returned when INDEXES is 0.
  If this file cannot be found, the 404 Not Found response will be returned.
- Number of corresponding content types: MAX_EXTENSION
  Specifies the number of definitions in the file extension list for files stored in external memory.
- Corresponding content types: EXTENSION_TYPE_TABLE_LIST
  This is a list of file extensions for files stored in external memory.
  When a file with an extension that is not in this list is transmitted, the file is returned with the settings for the file extension defined at the start of this list.
- Number of register CGI files: MAX_CGI_FILE
- Table of correspondences between CGI file names and internal functions: CGI_FILE_NAME_TABLE_LIST
- Newline code used for index page generation: LF_CODE
- Maximum number of clients that can be accepted at the same time: TCP_CEP_NUM
  This must be set to match the number of endpoints defined in the T4 source file config_tcpudp.c.
- Maximum number of files that can be displayed on the index page: MAX_FILE_LIST
  This must be set so that BODY_BUF_SIZE is not exceeded.
- Receive buffer size: RCV_BUF_SIZE
- Header field transmit buffer size: HDR_BUF_SIZE
- Body field transmit buffer size: BODY_BUF_SIZE

6.5 Files

The table below lists the files in this web server.

Table 6.6.1 Web Server Files

Folder Name	File Name	Description
r_t4_http_server_rx/src	r_http_server.c	Web server source file
	r_http_server_config.c	Web server configuration source file
	r_http_server_config.h	Web server configuration header file

6.6 API Reference

6.6.1 R_httpd

Description

The application calls this function periodically. R_httpd() manages the sockets required for HTTP communication. This function only manages these sockets while the communication itself is performed automatically by T4 interrupt drive.

Usage

#include “r_t4_http_server_rx_if.h”
void R_httpd (void);

Parameters

None

Return Value

None

Remark

None

6.6.2 R_httpd_pending_release_request

Description

Application calls this function when release the CGI pending
Please refer to the section 6.9.1cgi_sample_function.

Usage

```
#include “r_t4_http_server_rx_if.h”
void R_httpd_pending_release_request(ID cepid);
```

Parameters

cepid	input	communication endpoint ID

Return Value

None

Remark

None

6.6.3 R_T4_HTTP_SERVER_GetVersion

Description

Returns the version of this module. The version number is encoded such that the top two bytes are the major version number and the bottom two bytes are the minor version number.
For example, version ‘4.25’, the return value is ‘0x00040019’.

Usage

```
#include “r_t4_http_server_rx_if.h”
uint32_t R_T4_HTTP_SERVER_GetVersion(void);
```

Parameters

None

Return Value

Version number of Web server

Remark

None

6.7 User-Defined Function Reference (File I/O)

This web server calls this set of functions. The user must define the processing performed by these function appropriately for the file system used. Also, this web server uses this data structure and can acquire information from external memory. This web server is defined using TFAT as a sample file system.

Table 6.8.1 User-Defined Functions

Function Name	Function Overview	Function Name	Function Overview
change_dir()	Changes the working directory	file_write()	Writes to a file
file_close()	Closes a file	get_file_info()	Acquires file information
file_delete()	Deletes a file	get_file_list_info()	Acquires a file list
file_open()	Opens a file	get_file_size()	Acquires a file’s size
file_read()	Reads a file	make_dir()	Creates a directory
file_rename()	Renames a file	remove_dir()	Deletes a directory
file_exist()	Verifies that a file exists		

Note: Of the above functions, the ones that this web server does not call are grayed out.

6.7.1 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 Definitions

```
#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 */
```

6.7.2 change_dir

Description

This function sets the directory path specified with the argument to be the working directory. The directory path is specified as a full path name. The information in the working directory is managed by each socket.

Usage

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

Parameters

dir_path	Input	Storage location for the specified directory path

Return Value

-1	The directory does not exist
0	The directory exists

Remark

There are cases where the directory path ends with a "/", and cases where it does not. The presence or absence of the final "/" must be determined according to the file system used.

6.7.3 file_close

Description

This function performs a close operation on the file with the ID value specified in the argument and discards the 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 for the file to be closed

Return Value

-1	Error
0	Normal completion

Remark

None

6.7.4 file_delete

Description

This function deletes the file with the ID value specified in the argument. The file is specified as a full path name starting with the root directory.

Usage

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

Parameters

file_path	Input	Storage location that holds the full path name for the file.

Return Value

-1	Error
0	Normal completion

Remark

None

6.7.5 file_open

Description

This function opens the file specified by the first argument in the mode specified by the second argument. Furthermore it returns as its return value the ID value for the stored management information so that the web server can reference it using that ID. This stored management information must be stored until that ID value is specified 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	Storage location that holds the full path name for the file.
mode_flag	input	File open mode (FILE_WRITE or FILE_READ)

Return Value

-1	Error
0	ID value for the opened file

Remark

The file open state must be stored until that file’s ID value is specified to the file close function.

6.7.6 file_read

Description

This function reads the amount of file data specified by the third argument from the file corresponding to the ID value specified by the first argument to the address specified by the second argument. The file pointer in the management information corresponding to the ID value of the first argument is updated by the amount of data read and saved 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 for the file to be read
buf	Output	Storage address for the file data to be read
read_size	Input	Size of the file data to be read

Return Value

-1	Error
0	Size of data read

Remark

None

6.7.7 file_rename

Description

This function changes the name of the file or directory specified by the first argument to the name specified by the second argument. Both the first and second arguments are full path names from the 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	File or directory to be modified
new_name	Input	Name after modification

Return Value

-1	Error
0	Normal completion

Remark

None

6.7.8 file_exist

Description

This function verifies whether or not the file or directory specified by the first argument exists. The argument is specified as a full path name from the root directory.

Usage

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

Parameters

file_path	Input	File or directory whose existence is to be verified

Return Value

-1	Does not exist
0	Does exist

Remark

None

6.7.9 file_write

Description

This function writes the amount of data specified by the third argument to the file with the ID value specified by the first argument to the address specified by the second argument. The file pointer in the management information corresponding to the ID value of the first argument is updated by the amount of data written and saved until the file close function is called.

Usage

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

Parameters

file_id	Input	ID value for the file to be written
buf	Input	Start address of the data to be written
write_size	Input	Size of data to be written

Return Value

-1	Error
0	Normal completion

Remark

None

6.7.10 get_file_info

Description

This function reads in the file management information for the file corresponding to the ID value specified by the first argument and writes the data information structure specified by the second argument to the files date information.

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 be read
date_info	Output	Storage address of the date information

Return Value

-1	Error
0	Normal completion

Remark

None

6.7.11 get_file_list_info

Description

This function writes the information for either the file or directory stored at directory path specified by the first argument to the file list structure specified by the second argument. The maximum number of information items written out is specified by the third argument and file list read start position is specified by the fourth argument.

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_tread_index);
```

Parameters

dir_path	Input	Storage address for the directory path to read
file_list	Output	Storage address for the read out file list Note that '\0' will be stored at the start of the file name structure at the end of the list.
num_file_list	Input	Maximum number of file list information items to read at one time
read_index	Input	File list read out start position

Return Value

-1	Error
0	Number of file items read out

Remark

When the return value is smaller than num_file_list, it indicates that file list information readout has completed and when it is the same as num_file_list, it indicates that there is still more information to read out. When reading out continued values from the file list, call this function with read_index specified to be the file list read start position.

There are cases where dir_path ends with a "/", and cases where it does not. The presence or absence of the final "/" must be determined according to the file system used.

6.7.12 get_file_size

Description

This function reads the management information for the file corresponding to the ID value specified by the first 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 be read

Return Value

-1	Error
0	File size

Remark

None

6.7.13 make_dir

Description

This function creates the directory specified by the argument. This directory path is specified as a full path name.

Usage

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

Parameters

dir_path	Input	Directory name to be created

Return Value

-1	Error
0	Normal completion

Remark

There are cases where dir_path ends with a "/", and cases where it does not. The presence or absence of the final "/" must be determined according to the file system used.

6.7.14 remove_dir

Description

This function deletes the directory specified by the argument. This directory path is specified as a full path name.

Usage

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

Parameters

dir_path	Input	Directory to be deleted

Return Value

-1	Error
0	Normal completion

Remark

There are cases where dir_path ends with a "/", and cases where it does not. The presence or absence of the final "/" must be determined according to the file system used.

6.8 User-Defined Function Reference (System Timer)

This web server calls these functions. User defines system timer.

Table 6.8.1 User-Defined Functions

Function Name	Function Overview
get_sys_time()	Get pointer to system time

6.8.1 Data Structures

System Time Structure

```
typedef struct sys_time_
{
        uint32_t        sec;
        uint32_t        min;
        uint32_t        hour;
        uint32_t        day;
        uint32_t        month;
        uint32_t        year;
}SYS_TIME;
```

6.8.2 get_sys_time

Description

This function gets pointer to system time.

Usage

```
#include <stdint.h>
#include “r_t4_http_server_rx_config.h”
SYS_TIME *get_sys_time( void );
```

Parameters

None

Return Value

Pointer to system time

Remark

Please specify the variable for system timer.

6.9 Sample CGI Function

6.9.1 cgi_sample_function

Description

CGI function that is defined as CGI_FILE_NAME_TABLE_LIST in "r_t4_http_server_config.h"
The second element (cgi function pointer) of CGI_FILE_NAME_TABLE_LIST will be called when web browser requests the defined cgi file URL. And next, HTTPd will call cgi function.

HTTPd behavior will be changed by the return value.
case: Normal termination
CGI process finishes in this function.
case: Internal error
CGI process errors occur in this function.
case: CGI pending
CGI process does not finish in this function. The third element (cgi function pointer) of CGI_FILE_NAME_TABLE_LIST will be called when user will call R_httpd_pending_release_request() in finishing CGI process.

Usage

```
#include "r_t4_itcpip.h"
#include "r_http_server_config.h"
#include "r_t4_http_server_rx_if.h"
ER cgi_sample_function(ID cepid, void *res_info);
```

Parameters

cepid	Input	Communication endpoint ID for which there was a CGI function execution request
res_info	Input	(HTTPD_RESOURCE_INFO*)res_info->param Parameter associated with the URL for which there was a request from a web browser
	Output	(HTTPD_RESOURCE_INFO*)res_info->res.body HTML character string to be returned as the response
	Output	(HTTPD_RESOURCE_INFO*)res_info->res.body_size Length of the HTML character string to be returned as the response

Return Value

-1	Internal Error
-2	CGI pending
0	Normal completion

Remark

None

7. Main Program Specifications

7.1 Files

The following table lists the files in the main program.

Table 7.1.1 Main Program Files

Folder Name	File Name	Description
src	main.c	Main source file
	led.c	LED initialization processing source file
	led.h	LED initialization processing header file
	r_file_driver.c	Web server file system interface source file
	r_file_driver.h	Web server file system interface header file
	r_http_server_cgi_sample.c	CGI sample source file
	r_sys_time.c	Web server system timer source file
	r_sys_time.h	Web server system timer header file
	r_usb_hmsc_api.c	USB driver call processing source file
	r_usb_hmsc_api.h	USB driver call processing header file

7.2 Modules

The following table lists the modules in the main program.

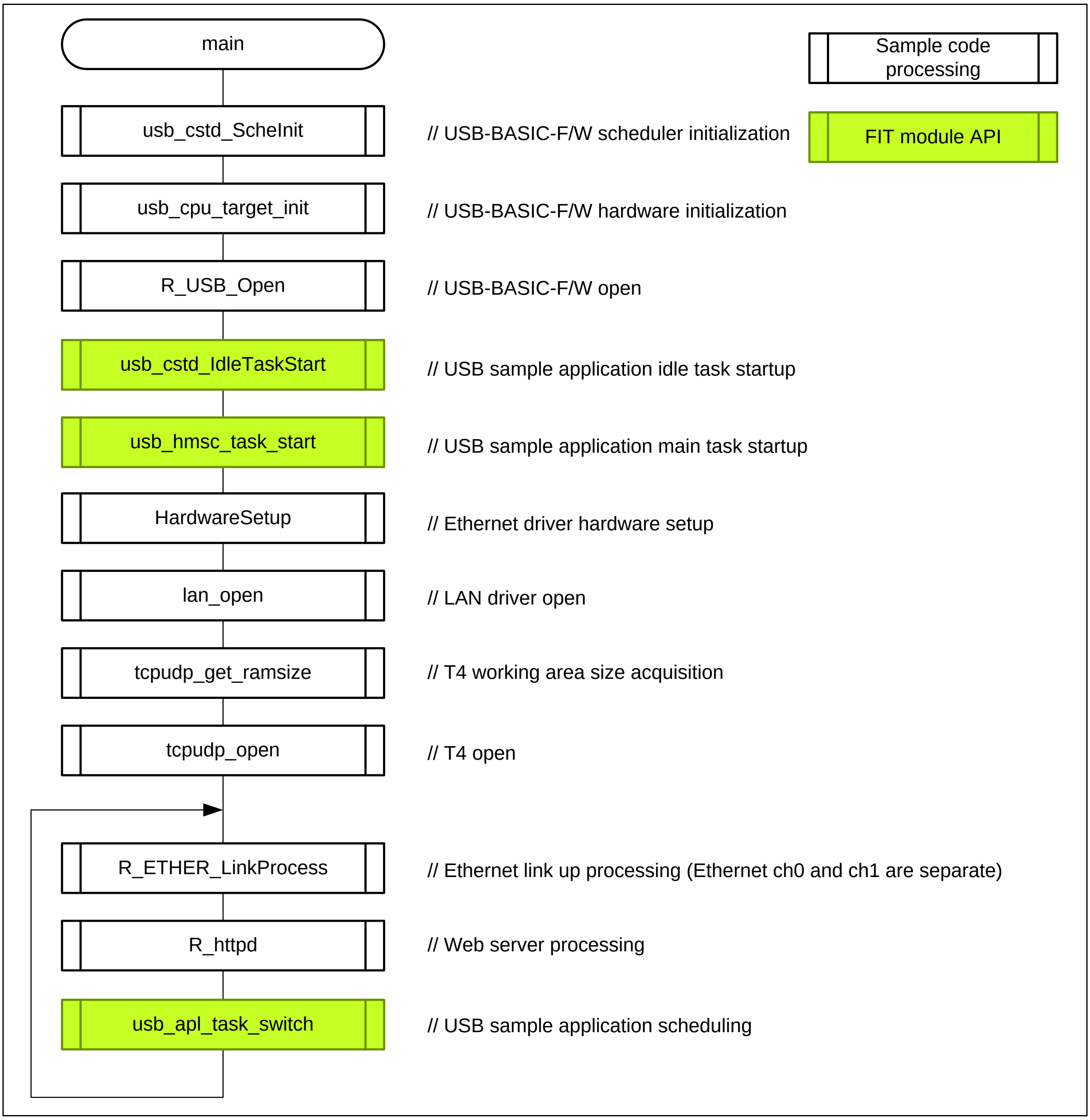
Table 7.2.1 Main Program Modules

File Name	Module Name	Description
main.c	main	Main processing for the main program Calls initialization processing for each of the FIT modules and drives the main processing for the web server, USB driver, and Ethernet driver (uses an infinite loop to implement periodic activation).
r_usb_hmsc_api.c	usb_cstd_IdleTaskStart	Starts the idle task used in low-power mode.
	usb_cstd_IdleTask	Idle task used in low-power mode. Performs no processing in host operation.
	usb_hmsc_task_start	HMSC driver activation processing. Performs USB IP initialization and class driver registration.
	usb_apl_task_switch	Performs task scheduling for the USB drivers in non-OS environments.
	usb_hapl_task_start	Starts the HMSC driver application task.
	usb_hmsc_DummyFunction	HMSC driver dummy function
	usb_hmsc_DriveOpen	HMSC driver open processing
	usb_hapl_registration	Registers HMSC drivers.
	usb_hmsc_apl_init	Initializes HMSC driver application task internal variables.
	usb_hmsc_StrgCommandResult	R_usb_hmsc_StrgDriveSearch() callback processing
	usb_hmsc_SampleAplTask	HMSC driver application task processing. Detects USB memory and mounts the file system.
led.c	led_init	Initialization of LEDs
r_file_driver.c	-	Please refer to the section 6.7 User-Defined Function Reference (File I/O).
r_http_server_cgi_sample.c	-	Please refer to the section 6.9 Sample CGI Function.
r_sys_time.c	-	Please refer to the section 6.8 User-Defined Function Reference (System Timer).

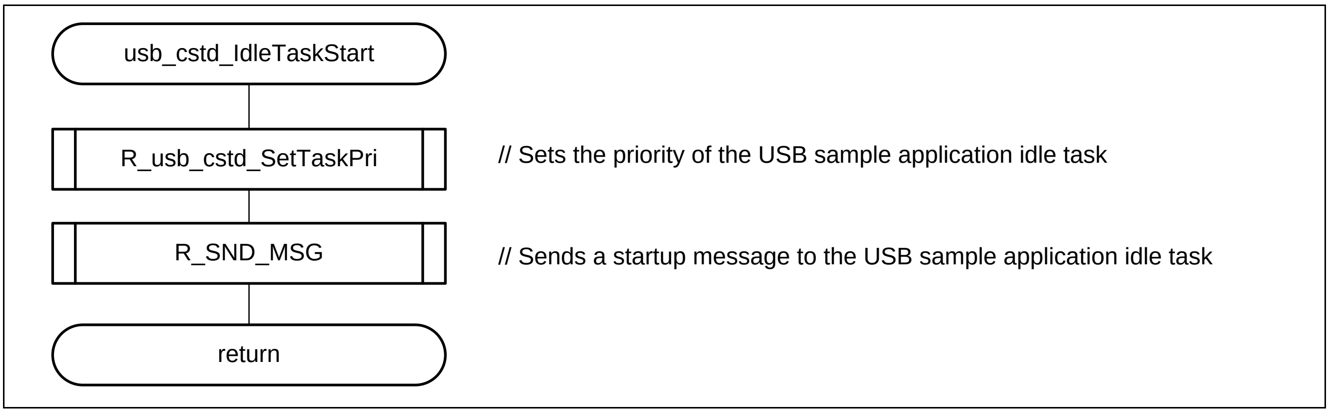
7.3 Flowcharts

This section shows the flowcharts for the modules in the main program.

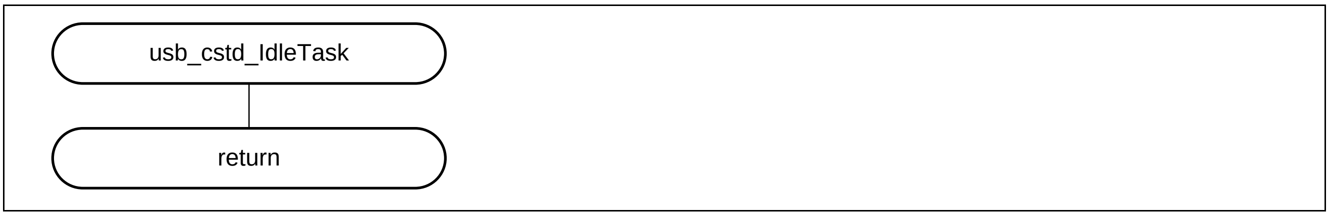
1. main()
   This is the main() function and is first called from the startup routine for the board support package (BSP module). It initializes the drivers and T4 and then periodically calls Ethernet driver link up processing, web server main processing, and USB driver scheduling from an infinite loop.


main
Sample code processing
FIT module API
usb_cstd_ScheInit
// USB-BASIC-F/W scheduler initialization
usb_cpu_target_init
// USB-BASIC-F/W hardware initialization
R_USB_Open
// USB-BASIC-F/W open
usb_cstd_IdleTaskStart
// USB sample application idle task startup
usb_hmsc_task_start
// USB sample application main task startup
HardwareSetup
// Ethernet driver hardware setup
lan_open
// LAN driver open
tcpudp_get_ramsize
// T4 working area size acquisition
tcpudp_open
// T4 open
R_ETHER_LinkProcess
// Ethernet link up processing (Ethernet ch0 and ch1 are separate)
R_httpd
// Web server processing
usb_apl_task_switch
// USB sample application scheduling

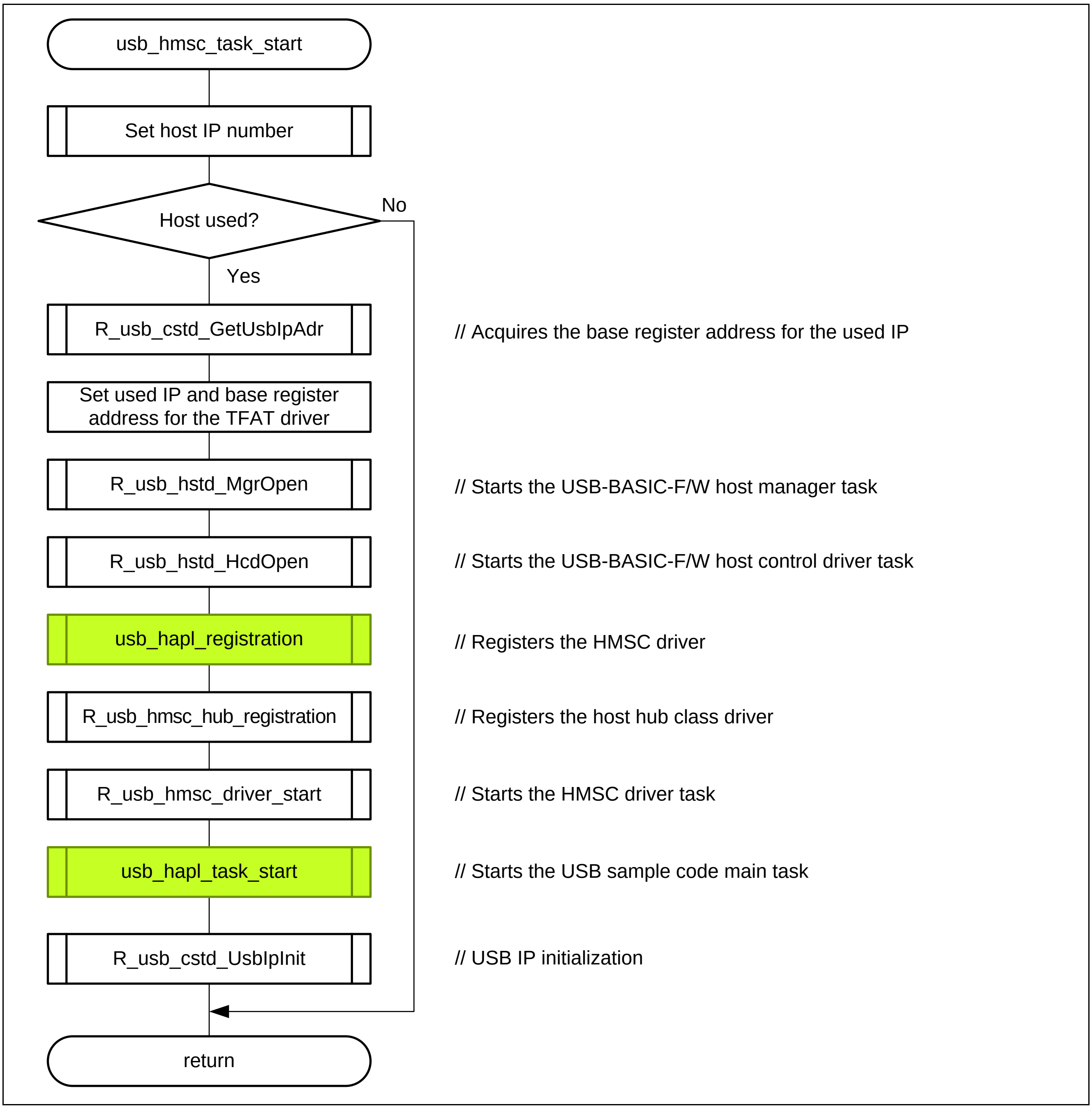
2. usb_cstd_IdleTaskStart
   Starts the USB driver processing idle task.


usb_cstd_IdleTaskStart
R_usb_cstd_SetTaskPri
// Sets the priority of the USB sample application idle task
R_SND_MSG
// Sends a startup message to the USB sample application idle task
return

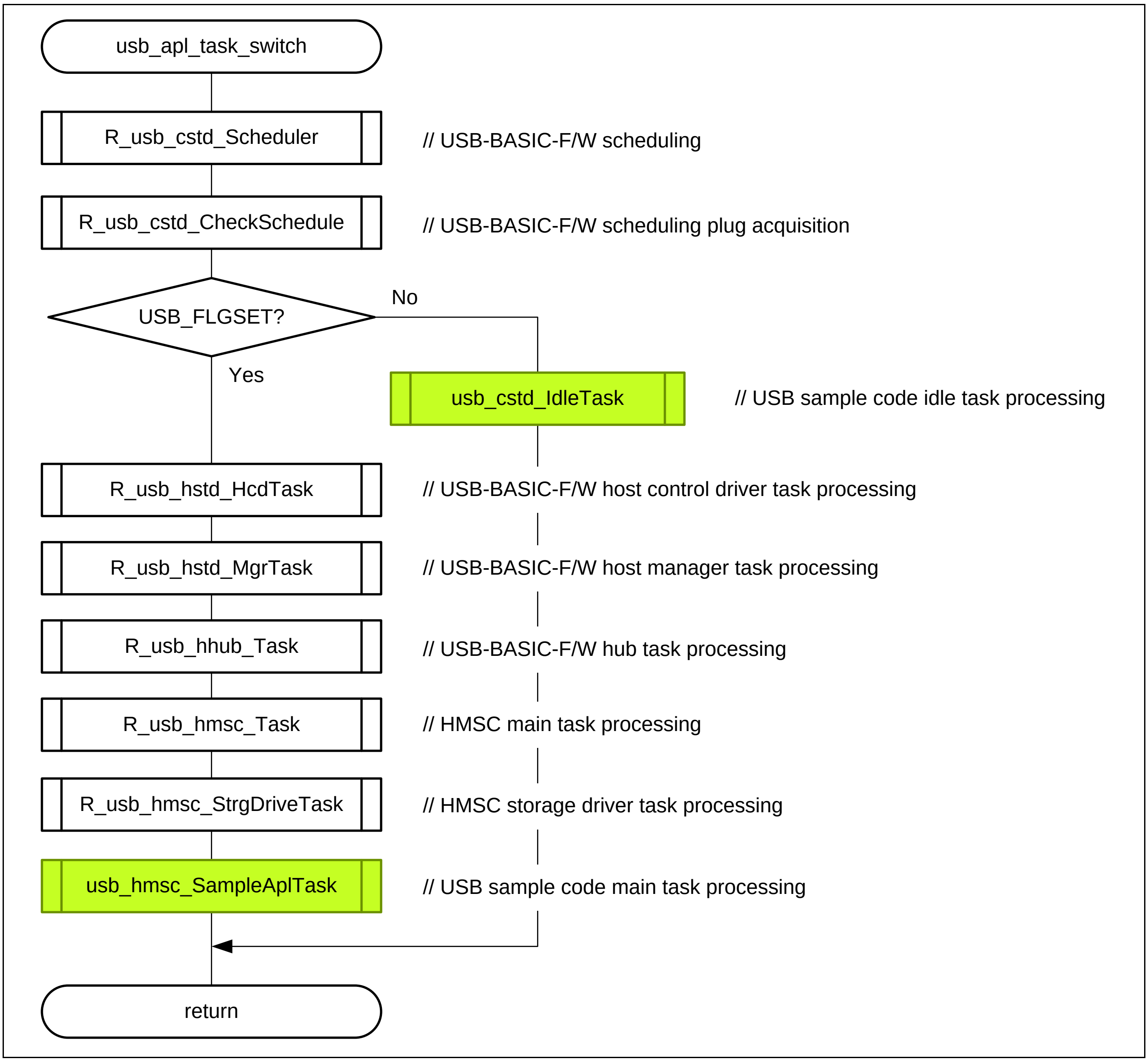
3. usb_cstd_IdleTask
   This is the USB driver processing idle task.



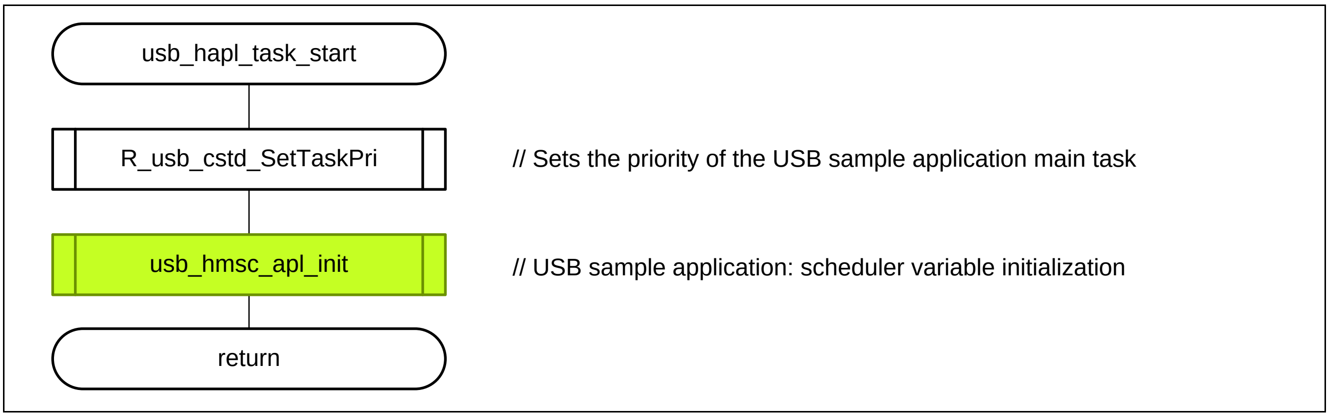
4. usb_hmsc_task_start
   Starts the various tasks within the USB driver, registers class drivers, and starts the USB memory mount processing task.


usb_hmsc_task_start
Set host IP number
Host used?
No
Yes
R_usb_cstd_GetUsbIpAdr
// Acquires the base register address for the used IP
Set used IP and base register address for the TFAT driver
R_usb_hstd_MgrOpen
// Starts the USB-BASIC-F/W host manager task
R_usb_hstd_HcdOpen
// Starts the USB-BASIC-F/W host control driver task
usb_hapl_registration
// Registers the HMSC driver
R_usb_hmsc_hub_registration
// Registers the host hub class driver
R_usb_hmsc_driver_start
// Starts the HMSC driver task
usb_hapl_task_start
// Starts the USB sample code main task
R_usb_cstd_UsbIpInit
// USB IP initialization
return

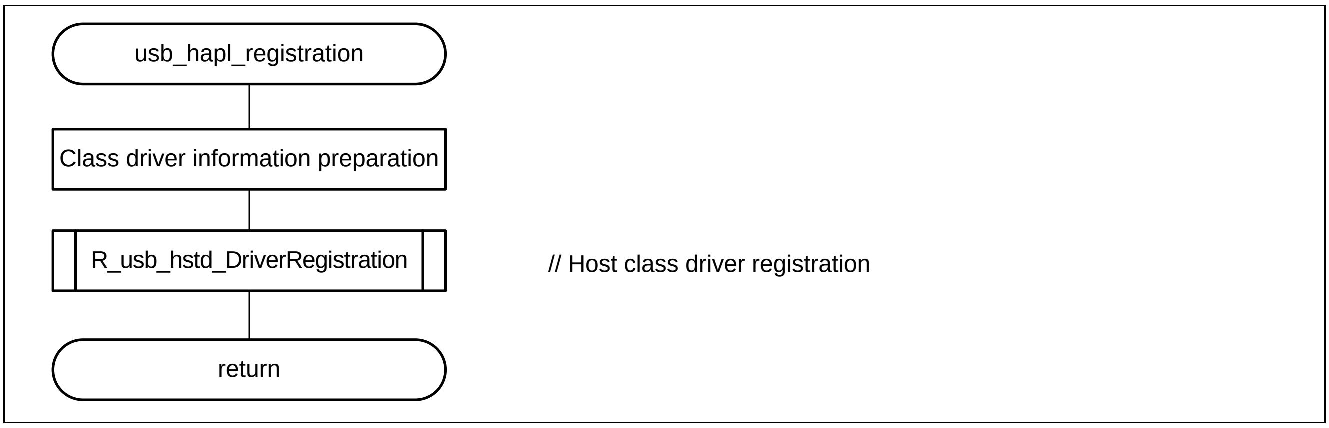
5. usb_apl_task_switch
   Performs the USB driver scheduling.


usb_apl_task_switch
R_usb_cstd_Scheduler
// USB-BASIC-F/W scheduling
R_usb_cstd_CheckSchedule
// USB-BASIC-F/W scheduling plug acquisition
USB_FLGSET?
No
Yes
usb_cstd_IdleTask
// USB sample code idle task processing
R_usb_hstd_HcdTask
// USB-BASIC-F/W host control driver task processing
R_usb_hstd_MgrTask
// USB-BASIC-F/W host manager task processing
R_usb_hhub_Task
// USB-BASIC-F/W hub task processing
R_usb_hmsc_Task
// HMSC main task processing
R_usb_hmsc_StrgDriveTask
// HMSC storage driver task processing
usb_hmsc_SampleAplTask
// USB sample code main task processing
return

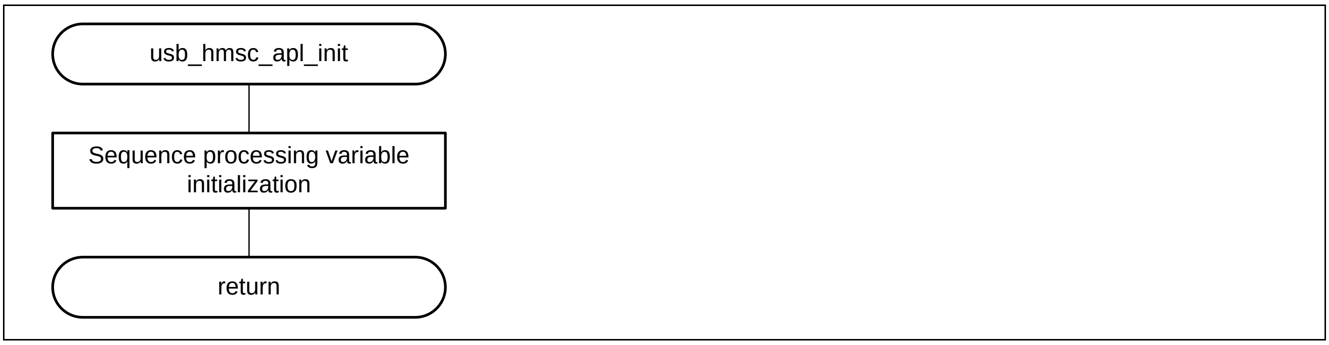
6. usb_hapl_task_start
   Initializes the USB memory mount processing task.


usb_hapl_task_start
R_usb_cstd_SetTaskPri
// Sets the priority of the USB sample application main task
usb_hmsc_apl_init
// USB sample application: scheduler variable initialization
return

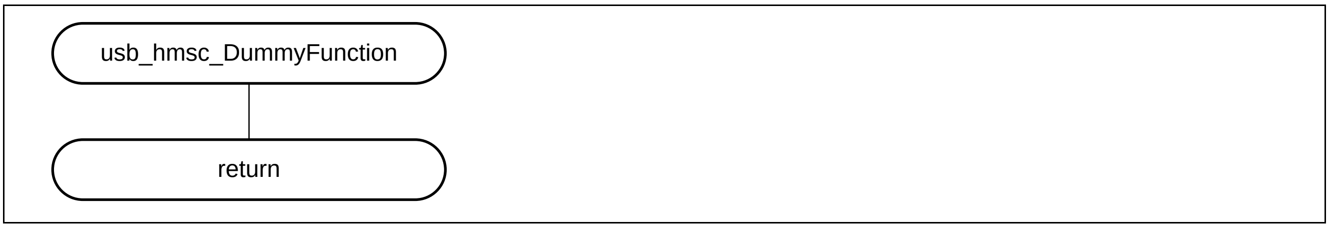
7. usb_hapl_registration
   Performs the class driver registration processing.


usb_hapl_registration
Class driver information preparation
R_usb_hstd_DriverRegistration
// Host class driver registration
return

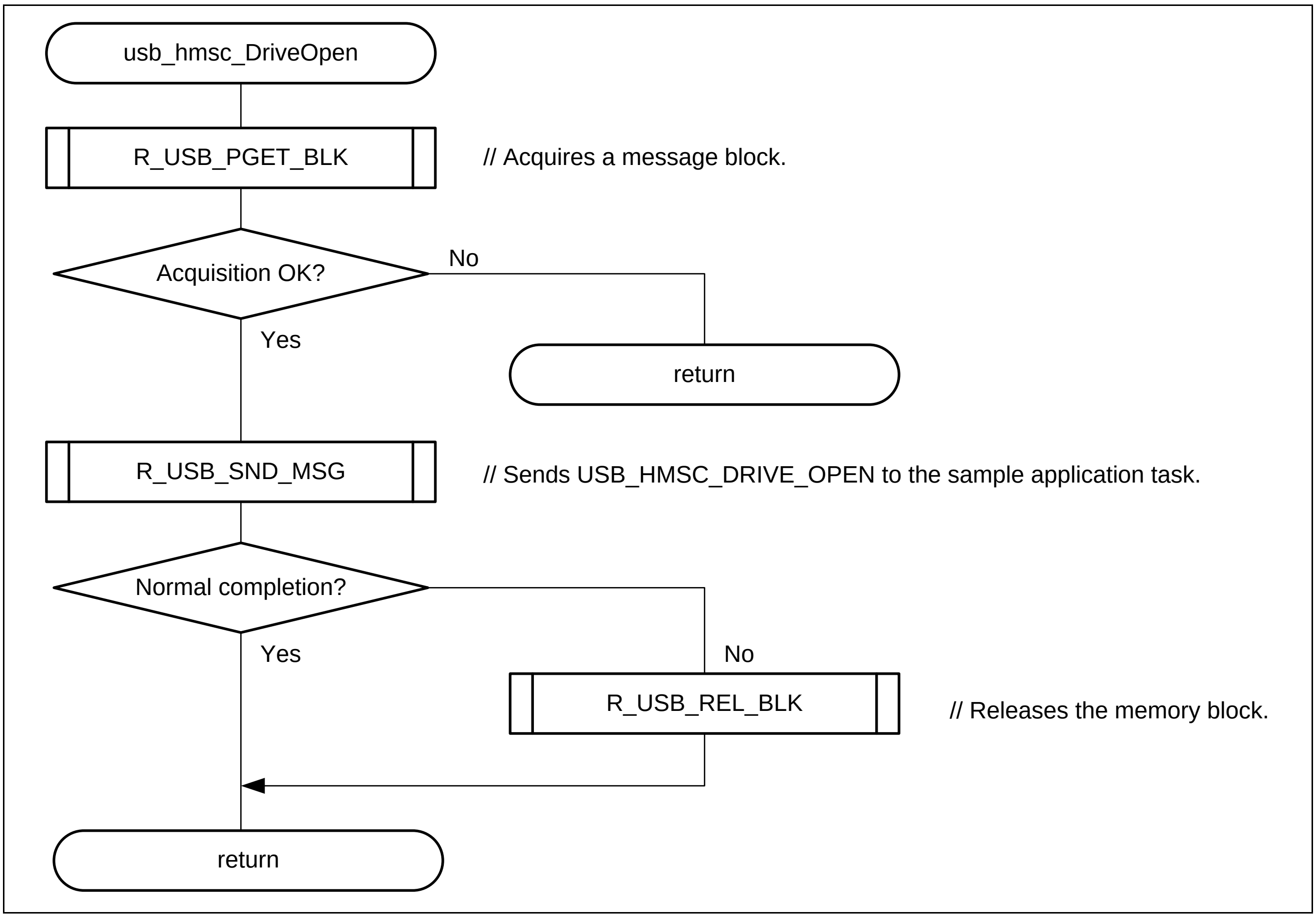
8. usb_hmsc_apl_init
   Initializes the sequence processing variables for the USB memory mount processing task.



9. usb_hmsc_DummyFunction
Dummy function for suspend and resume specified at class driver registration.



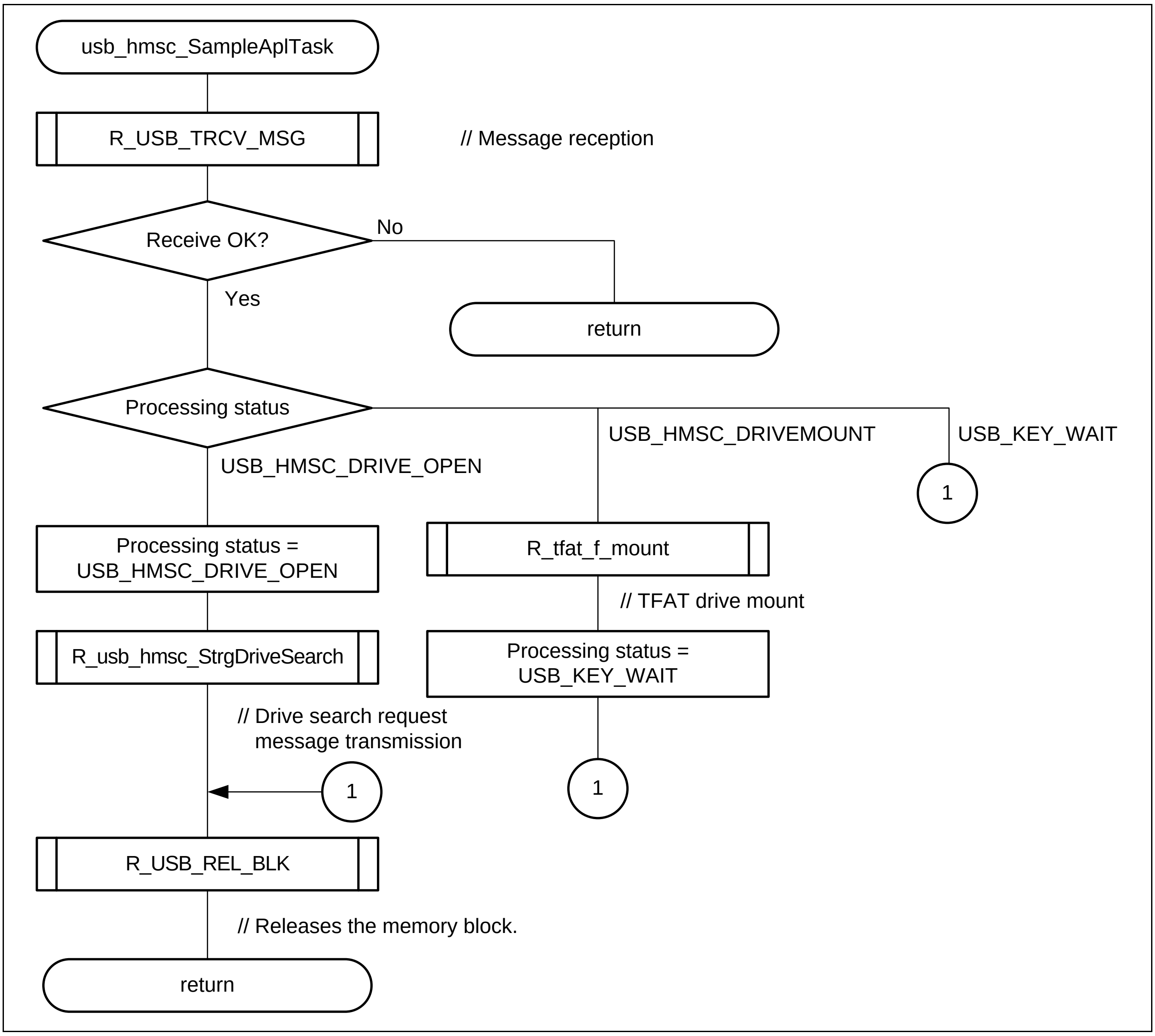
10. usb_hmsc_DriveOpen
This is the callback function called from the USB driver when USB memory is inserted. It sends a USB_HMSC_DRIVE_OPEN message for the sample application task.


usb_hmsc_DriveOpen
R_USB_PGET_BLK
// Acquires a message block.
Acquisition OK?
No
Yes
return
R_USB_SND_MSG
// Sends USB_HMSC_DRIVE_OPEN to the sample application task.
Normal completion?
Yes
No
R_USB_REL_BLK
// Releases the memory block.
return

11. usb_hmsc_SampleAplTask

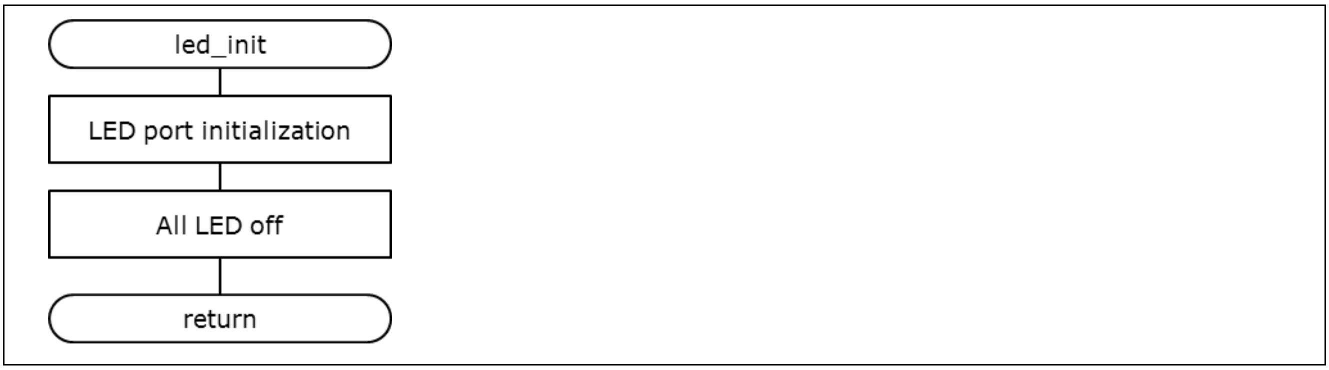
This function performs the sample application task processing. It receives the USB_HMSC_DRIVE_OPEN message issued from the usb_hmsc_DriveOpen function and detects a mountable drive.

Also, it receives the USB_HMSC_DRIVEMOUNT message issued from the usb_hmsc_StrgCommandResult function and performs a mount for the file system.



12. led_init

This function performs the initialization process for using the LED on the Renesas Starter Kit + for RX64M.



8. User-Defined Functions

The user defined functions must be coded by the user to match the user system environment. Some of the user-defined functions are required by the FIT modules.

This package includes the following user-defined function samples. See the corresponding FIT module manual or other documentation for specifications of these user-defined functions.

Table 8.1 User-Defined Functions

User-Defined Function	File Name	FIT Module Name	Document Name/Catalog Number
File system interface	r_file_driver.c	r_t4_http_server_rx	6.7, User-Defined Function Reference (File I/O)
System timer interface	r_sys_time.c	r_t4_http_server_rx	6.8, User-Defined Function Reference (System Timer)

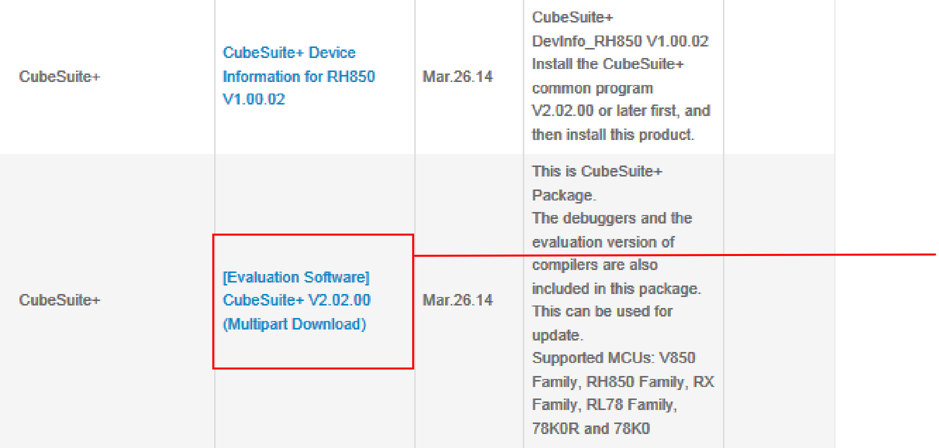
9. When CubeSuite+ is Used

This application note can be evaluated using CubeSuite+. Note that RX Family C/C++ Compiler Package V2.02.00 or later is required to build this application note under CubeSuite+. This section assumes the user does not own the commercial version and will be using the free evaluation version.

9.1 Acquire and Install CubeSuite+

Download CubeSuite+ from the Renesas web site.

1. Access the following URL to display the CubeSuite+ download page.
   http://www.renesas.com/cubesuite+_download
2. Of the displayed items, click **[Evaluation Software] CubeSuite+ V2.02.00**. (Although there are two versions, one that is broken up into smaller sections, and one that can be downloaded in a single operation, the contents are the same.)
   Next, download the CubeSuite+ installer by following the instructions displayed.



CubeSuite+	CubeSuite+ Device Information for RH850 V1.00.02	Mar.26.14	CubeSuite+ DevInfo_RH850 V1.00.02 Install the CubeSuite+ common program V2.02.00 or later first, and then install this product.	
CubeSuite+	[Evaluation Software] CubeSuite+ V2.02.00 (Multipart Download)	Mar.26.14	This is CubeSuite+ Package. The debuggers and the evaluation version of compilers are also included in this package. This can be used for update. Supported MCUs: V850 Family, RH850 Family, RX Family, RL78 Family, 78K0R and 78K0	

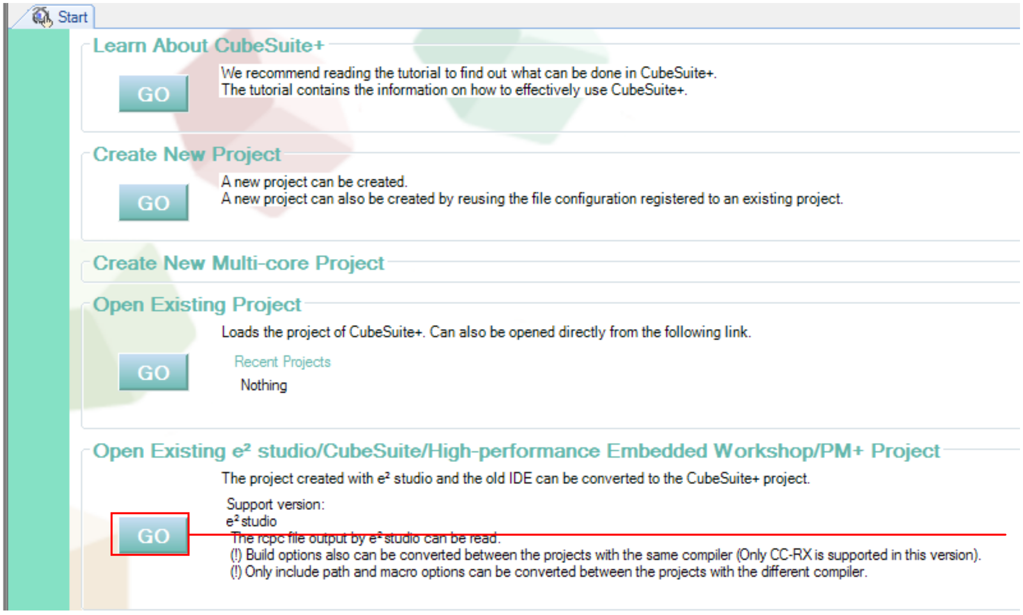
Click this link.

3. Run the downloaded CubeSuite+ installer to CubeSuite+ on your personal computer.
   See the **CubeSuite+ V2.02.00 Integrated Development Environment User’s Manual: Start** for details on the installation procedure.
   http://documentation.renesas.com/doc/products/tool/doc/r20ut2865ej0100_qsst.pdf

9.2 Install the Project

Install the Renesas common project files provided with this application note in CubeSuite+.

1. Decompress the ZIP file in which this application note is provided into an arbitrary folder.
2. Start CubeSuite+ and from the start screen, click **GO** under **Open Existing e^2 studio/CubeSuite/High-performance Embedded Workshop/PM+ project**.



Learn About CubeSuite+

GO

We recommend reading the tutorial to find out what can be done in CubeSuite+.
The tutorial contains the information on how to effectively use CubeSuite+.

Create New Project

GO

A new project can be created.
A new project can also be created by reusing the file configuration registered to an existing project.

Create New Multi-core Project

Open Existing Project

GO

Loads the project of CubeSuite+. Can also be opened directly from the following link.

Recent Projects
Nothing

Open Existing e² studio/CubeSuite/High-performance Embedded Workshop/PM+ Project

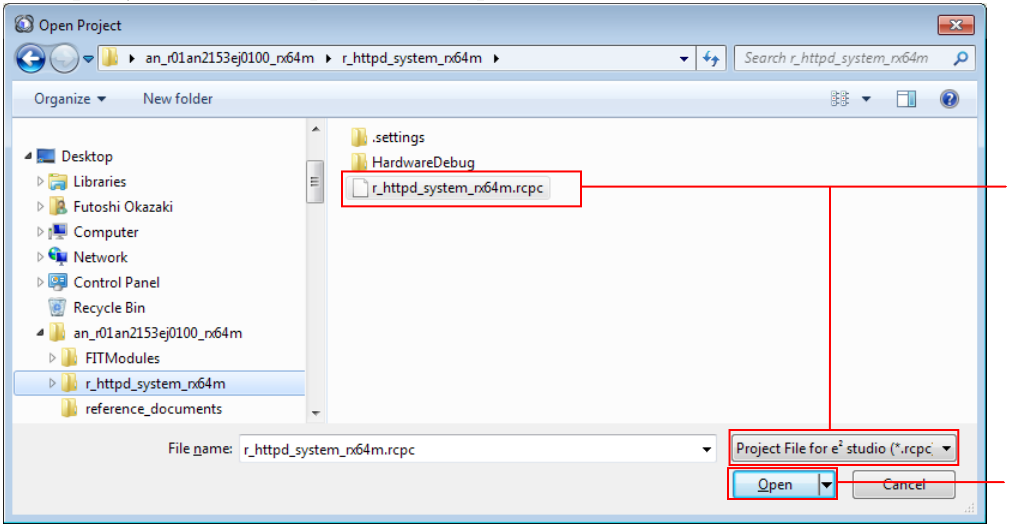
GO

The project created with e² studio and the old IDE can be converted to the CubeSuite+ project.

Support version:
e² studio
The rcpc file output by e² studio can be read.
(!) Build options also can be converted between the projects with the same compiler (Only CC-RX is supported in this version).
(!) Only include path and macro options can be converted between the projects with the different compiler.

Click **GO**.

3. Open the folder decompressed in step 1 above and of those entries, open **Web server system project (h_httpd_system_rx64m** folder**)**. From there, select **Renesas common project files (h_httpd_system_rx64m.rcpc)** and click **Open**.



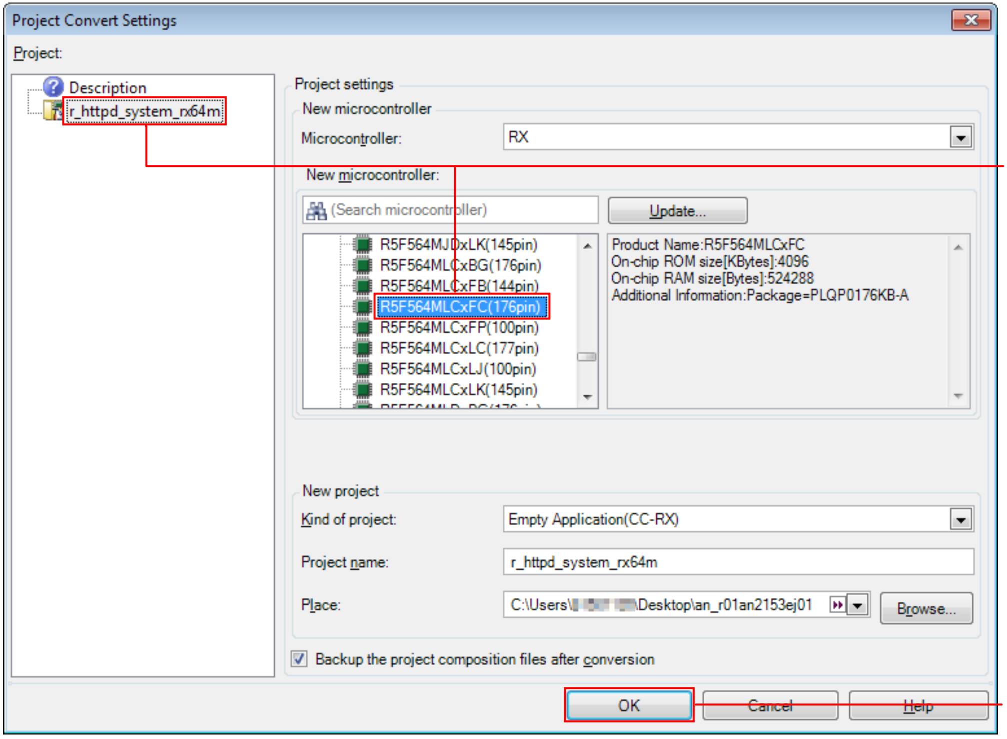
File name: r_httpd_system_rx64m.rcpc

Project File for e² studio (*.rcpc)

Select this item.

Click **Open**.

4. After selecting the project from the project tree, select the items as shown below and click **OK**. Note that **Microcontroller used** must be selected to match the device actually mounted in the evaluation board used.



Project Convert Settings

Project:

- Description
- r_httpd_system_rx64m

Project settings

New microcontroller

Microcontroller: RX

New microcontroller:

(Search microcontroller) Update...

- R5F564MJDxLK(145pin)
- R5F564MLCxBG(176pin)
- R5F564MLCxFB(144pin)
- R5F564MLCxFC(176pin)
- R5F564MLCxFP(100pin)
- R5F564MLCxLC(177pin)
- R5F564MLCxLJ(100pin)
- R5F564MLCxLK(145pin)

Product Name:R5F564MLCxFC
On-chip ROM size[KBytes]:4096
On-chip RAM size[Bytes]:524288
Additional Information:Package=PLQP0176KB-A

New project

Kind of project: Empty Application(CC-RX)

Project name: r_httpd_system_rx64m

Place: C:\Users\[redacted]\Desktop\an_r01an2153ej01 Browse...

☑ Backup the project composition files after conversion

Select this item.

Click **OK**.

5. The project will be converted and the converted project opened. Also, the e^2 studio project will be backed up.

9.3 Add the FIT Modules to the Project

Add the FIT modules included in this application note and the RX64M Group Driver Package to the project.

The added FIT modules are listed in the table below.

Table 9.3.1 Added FIT Modules

Type	Module	FIT Module Name	Version
Board Support Package	Board support package (BSP module)	r_bsp	2.60
Device Driver	Compare match timer (CMT)	r_cmt_rx	2.30
Device Driver	Ethernet controller (ETHERC)	r_ether_rx	1.00
Middleware	M3S-T4-Tiny interface conversion module	r_t4_driver_rx64m	1.00
Middleware	TCP/IP protocol stack (M3S-T4-Tiny)	r_t4_rx	2.00
Middleware	FAT file system (M3S-TFAT-Tiny)	r_tfat_rx	3.00
Device Driver	USB basic firmware	r_usb_basic	1.00
Device Driver	USB host mass storage class	r_usb_hmsc	1.00
Application	HTTP server	r_t4_http_server_rx	1.03
Application	Web server system main program	r_httpd_main_rx64m	1.00

See the “RX Family: Adding Firmware Integration Technology Modules to CubeSuite+ Projects” document for the methods for adding FIT modules to a project.

http://documentation.renesas.com/doc/products/mpumcu/apn/rx/r01an1826ej0100_rx.pdf

10. Supplement

10.1 USB Driver Limitations

When both USB channels ch0 and ch1 are set to host mode, only ch0 can recognize USB memory. To use ch1 in host mode, set ch0 to either unused or peripheral mode.

r_config/r_usb_config.h

```
/* Select USB mode(Host or Periphera) per each USB IP */
//  #define USB_FUNCSEL_USBIP0_PP      USB_HOST_PP    /* Host Mode */
//  #define USB_FUNCSEL_USBIP0_PP      USB_PERI_PP    /* Peripheral Mode */
    #define USB_FUNCSEL_USBIP0_PP      USB_NOUSE_PP

    #define USB_FUNCSEL_USBIP1_PP      USB_HOST_PP    /* Host Mode */
//  #define USB_FUNCSEL_USBIP1_PP      USB_PERI_PP    /* Peripheral Mode */
//  #define USB_FUNCSEL_USBIP1_PP      USB_NOUSE_PP
```

Figure 10.1.1 When Using ch1 in Host Mode

10.2 Web Server System Limitations

After program operation, if the USB memory is removed it will not be recognized if it is reinserted. The program should be restarted.

10.3 Notes on Using the Free Evaluation Version of the RX Family C/C++ Compiler Package

There is a usage period limitation and certain usage limitations on the free evaluation version of the RX Family C/C++ Compiler Package. If the usage period is exceeded, load modules may not be generated correctly due to the usage limitations.

See the page on evaluation software on the Renesas web site at the link below.

http://www.renesas.com/products/tools/evaluation_software/index.jsp

Website and Support

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

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

Revision History

		Description	
Rev.	Date	Page	Summary
1.00	Sep 1, 2014	—	First edition issued

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 document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

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

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

2. Processing at Power-on

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

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

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

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

4. Clock Signals

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

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

5. Differences between Products

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

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

Notice

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

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

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

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

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

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

Renesas Electronics Canada Limited
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: +49-211-6503-0, Fax: +49-211-6503-1327

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

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

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022/9044

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

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

Renesas Electronics Malaysia Sdn.Bhd.
Unit 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.
12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 4.0