**Introduction**

This document is an application note of the industrial network solution using TCP/IP protocol stack (referred to as “M3S-T4-Tiny”).

This application note includes sample code for a main program that uses Network Middleware Modules (DHCP client, DNS client, HTTP server, FTP server), when used in combination with the RX64M, RX71M Group RX Driver Package, allows the construction of various network systems.

**Target Device**

RX64M Group (Renesas Starter Kit+ RX64M)
RX71M Group (Renesas Starter Kit+ RX71M)

**RX Driver Package Used**

RX64M, RX71M Group RX Driver Package Ver.1.02 (R01AN2606EJ)

When using this application note with your product, careful evaluation is recommended.

And 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 ................................................................................................................. 3
  1.1 This Application Note ......................................................................................... 3
  1.2 Operating Environment .................................................................................... 3
  1.3 Module Structure ............................................................................................... 4
  1.4 File Structure ..................................................................................................... 6
  1.5 Projects ............................................................................................................... 7

2. Acquiring a Development Environment .............................................................. 8
  2.1 Acquire e² studio .............................................................................................. 8
  2.2 Acquire a Compiler Package ............................................................................ 8

3. Building a Project ................................................................................................... 9
  3.1 Create a Workspace ......................................................................................... 9
  3.2 Create a Project ............................................................................................... 10
  3.3 Import a Project ............................................................................................... 12
  3.4 Modify Configuration ...................................................................................... 16
  3.4.1 Change Configuration ............................................................................... 16
  3.4.2 Change Project Setting ............................................................................ 24

4. Verify Operation .................................................................................................. 27
  4.1 Build the Project ............................................................................................ 27
  4.2 Prepare for Debugging ................................................................................... 29
  4.2.1 Configure Hardware .................................................................................. 29
  4.2.2 Figure 4-4 No DNS server and No DHCP server – LAN Cable (cross) Configuration Set Up the Evaluation Board ........................................................................... 31
  4.2.3 Set Up Client PC ....................................................................................... 35
  4.2.4 Prepare USB Memory ................................................................................ 39
  4.3 Debug the Project ........................................................................................... 40
  4.3.1 Verify operation of the HTTP server ......................................................... 43
  4.3.2 Verify operation of the FTP server ......................................................... 46

5. Network Middleware Specifications .................................................................. 47

6. Main Program Specifications .......................................................................... 48
  6.1 Files .................................................................................................................. 48
  6.2 Modules ........................................................................................................... 49
  6.3 Flowcharts ....................................................................................................... 50

7. User-Defined Functions .................................................................................... 59

8. Supplement ......................................................................................................... 60
  8.1 USB Driver Limitations .................................................................................. 60
  8.2 This Application Limitations ......................................................................... 60
  8.3 FTP Server and DHCP Client Limitations .................................................... 60
  8.4 Notes on Using the Free Evaluation Version of the RX Family C/C++ Compiler Package .......................................................... 60
1. Overview

1.1 This Application Note

This application note describes the procedure for main program evaluation by combining the Board Support Package (referred to as “BSP”), USB driver (Host Mass Storage Class Driver “USB HMSC” and “Basic Firmware”), M3S-TFAT-Tiny FAT file system (referred to as “TFAT”), Ethernet driver, TCP/IP protocol stack (M3S-T4-Tiny), and Network Middleware of FIT modules included in the RX64M, RX71M Group RX Driver Package.

This application note operates on the Renesas Starter Kit+ for RX64M, Renesas Starter Kit+ for RX71M (referred to as “RSK” in the remainder of this document).

1.2 Operating Environment

This application note operates in the following environment.

Table 1-1 Operating Environment

<table>
<thead>
<tr>
<th>Microcontroller</th>
<th>RX64M Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation board</td>
<td>Renesas Starter Kit+ RX64M</td>
</tr>
<tr>
<td>Integrated development environment (IDE)</td>
<td>e2 studio, V4.1.0 or later</td>
</tr>
<tr>
<td>Cross tools</td>
<td>RX Family C/C++ Compiler Package V2.03.00 or later</td>
</tr>
<tr>
<td>Emulator</td>
<td>E1 (included in the Renesas Starter Kit+ for RX64M), E20</td>
</tr>
<tr>
<td>RX Driver Package</td>
<td>RX64M, RX71M Group RX Driver Package Ver1.02 (R01AN2606EJ)*</td>
</tr>
</tbody>
</table>

Table 1-2 Operating Environment

<table>
<thead>
<tr>
<th>Microcontroller</th>
<th>RX71M Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation board</td>
<td>Renesas Starter Kit+ RX71M</td>
</tr>
<tr>
<td>Integrated development environment (IDE)</td>
<td>e2 studio, V4.1.0 or later</td>
</tr>
<tr>
<td>Cross tools</td>
<td>RX Family C/C++ Compiler Package V2.03.00 or later</td>
</tr>
<tr>
<td>Emulator</td>
<td>E1 (included in the Renesas Starter Kit+ for RX71M), E20</td>
</tr>
<tr>
<td>RX Driver Package</td>
<td>RX64M, RX71M Group RX Driver Package Ver1.02 (R01AN2607EJ)*</td>
</tr>
</tbody>
</table>

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.
1.3 Module Structure

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

![Diagram](image-url)
### Table 1-3  Modules

<table>
<thead>
<tr>
<th>Type</th>
<th>Module</th>
<th>FIT Module Name</th>
<th>Rev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Board Support Package</td>
<td>Board Support Package (BSP)</td>
<td>r_bsp</td>
<td>3.00</td>
</tr>
<tr>
<td>Device Driver</td>
<td>Compare Match Timer (CMT)</td>
<td>r_cmt_rx</td>
<td>2.60</td>
</tr>
<tr>
<td>Device Driver</td>
<td>USB Basic Firmware</td>
<td>r_usb_basic</td>
<td>1.10</td>
</tr>
<tr>
<td>Device Driver</td>
<td>USB Host Mass Storage Class</td>
<td>r_usb_hmsc</td>
<td>1.10</td>
</tr>
<tr>
<td>Device Driver</td>
<td>Ethernet controller (ETHERC)</td>
<td>r_ether_rx</td>
<td>1.02</td>
</tr>
<tr>
<td>Middleware</td>
<td>TCP/IP M3S-T4-Tiny for Embedding (Note 1)</td>
<td>r_t4_rx</td>
<td>2.02</td>
</tr>
<tr>
<td>Interface</td>
<td>Interface conversion module for Ethernet Driver and</td>
<td>r_t4_driver_rx</td>
<td>1.02</td>
</tr>
<tr>
<td></td>
<td>Embedded System M3S-T4-Tiny</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Middleware</td>
<td>DHCP client using the embedded TCP/IP M3S-T4-Tiny</td>
<td>r_t4_dhcp_client_rx</td>
<td>1.03</td>
</tr>
<tr>
<td></td>
<td>Module</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Middleware</td>
<td>DNS client using the embedded TCP/IP M3S-T4-Tiny</td>
<td>r_t4_dns_client_rx</td>
<td>1.02</td>
</tr>
<tr>
<td></td>
<td>Module</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Middleware</td>
<td>FTP server using the embedded TCP/IP M3S-T4-Tiny</td>
<td>r_t4_ftp_server_rx</td>
<td>1.03</td>
</tr>
<tr>
<td></td>
<td>Module</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Middleware</td>
<td>Web server using the embedded TCP/IP M3S-T4-Tiny</td>
<td>r_t4_http_server_rx</td>
<td>1.04</td>
</tr>
<tr>
<td></td>
<td>Module</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interface</td>
<td>File driver for FTP server and Web server Module</td>
<td>r_t4_file_driver_rx</td>
<td>1.01</td>
</tr>
<tr>
<td>Middleware</td>
<td>M3S-TFAT-Tiny (FAT file system)</td>
<td>r_tfat_rx</td>
<td>3.02</td>
</tr>
<tr>
<td>Interface</td>
<td>M3S-TFAT-Tiny Memory Driver Interface Module</td>
<td>r_tfat_driver_rx</td>
<td>1.02</td>
</tr>
<tr>
<td>Application</td>
<td>Main program</td>
<td>src</td>
<td>1.01</td>
</tr>
</tbody>
</table>

Note 1: This package contains the TCP/IP M3S-T4-Tiny evaluation version. For commercial version, refer to the following URL.

http://www.renesas.com/mw/t4
1.4 File Structure

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

![File Structure Diagram]

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.

“Network Solution project (r_network_solution_rx64m)” or “Network Solution project (r_network_solution_rx71m)” under the “workspace” folder is the project that built this application. Operation of the application can be verified by importing the project to the e² studio workspace.

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” (R01AN1723EU) describes the method for including the FIT modules, as a FIT plugin, in an e² studio project. The document “Adding Firmware Integration Technology Modules to CS+ Projects” (R01AN1826EJ) describes the method for including the FIT modules in a CS+ project.
1.5 Projects

This application note includes an e² studio project for building and evaluating this application. These projects register both a build structure that stores the build settings and a debug structure that stores debug settings.

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

<table>
<thead>
<tr>
<th>Table 1-4 Project Settings</th>
<th>Structure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build structure</td>
<td>HardwareDebug</td>
<td>This structure is used to generate a load module with debugging information included. Main settings</td>
</tr>
<tr>
<td></td>
<td>(Debug on hardware)</td>
<td></td>
</tr>
<tr>
<td>Debug structure</td>
<td>HardwareDebug (E1)</td>
<td>Used for hardware debugging over an E1 emulator using a load module generated by HardwareDebug (Debug on hardware).</td>
</tr>
</tbody>
</table>

The table below lists the target-specific setting.

<table>
<thead>
<tr>
<th>Table 1-5 Target-specific setting</th>
<th>Items</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool chain version</td>
<td>v2.03.00</td>
<td></td>
</tr>
<tr>
<td>Debug Hardware</td>
<td>E1 (RX)</td>
<td></td>
</tr>
<tr>
<td>Data endian</td>
<td>Little-endian data</td>
<td></td>
</tr>
<tr>
<td>Select Target</td>
<td>R5F564MLCxFc (RX64M LQFP 176pin) or R5F571MLCxFc (RX71M LQFP 176pin)</td>
<td></td>
</tr>
<tr>
<td>Renesas RTOS Support (Note 1)</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Note 1: For use, OS environment download is required.
2. Acquiring a Development Environment

2.1 Acquire e² studio
Access the following URL and download the e² studio.

http://www.renesas.com/e2studio_download

This document requires you to use e² studio V4.1.0 or later. If the version older than V4.1.0 is used, some functions of the e² studio may not be available. For download, obtain the latest version of the e² studio on the website.

2.2 Acquire a Compiler Package
Access the following URL and download the RX Family C/C++ Compiler Package.

http://www.renesas.com/e2studio_download
3. Building a Project

This application note includes environment-built project. The procedure to import a Project using e² studio Smart browser is described below.

3.1 Create a Workspace

1. Start e² studio.

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

3. When the following window is displayed, click [Workbench].
3.2 Create a Project
When using the Smart browser function, the target project or file needs to be selected. To use this function, create the project that specified the target device (Note 1).

Note 1: The project to be created here is a dummy to use the Smart browser. For the setting of a Project to be imported, refer to Table 1-5 Target-specific setting.

1. Click [File (F)], [New (N)], then [C Project] to create new C project. Start [Create New project wizard].

![Image of C Project creation]

2. Input any project name and select [Renesas RXC Toolchain]. Click [Next (N)].

![Image of selecting Renesas RXC Toolchain]
3. Set Select Target ; for RX64M :R5F564Mxxxxx, for RX71M :R5F571Mxxxxx. For other items, any setting is OK (Note 1). When the setting is completed, click [Finish(F)].

Note 1 : Setting for dummy project. For the setting of the project environment to be imported using the smart browser, refer to Table 1-5 Target-specific setting.

4. Click [OK].
3.3 Import a Project

Import the project of Main program in the workspace created.

1. Select the project created in ”3.2 Create a Project” from Project explorer.

2. Click on [Renesas Views] → [e2 Solution Tool kit] → [Smart browser] to start the Smart browser.

3. Click on the [Application Note] on the [Smart browser] tabbed page.
4. Click [Update].

5. Select the application note and right-click. Then, click on [Sample code (Project import)] in the context menu (Note 1).

Note 1: If authentication by My Renesas has never been performed, "My Renesas" dialog opens when downloading the file. Enter your mail address and password registered in the Renesas website.
6. Click [Agree].

7. Save the application note.
8. Select either [r_network_solution_rx64m] or [r_network_solution_rx71m] displayed in [Project(P)], and click [Finish(F)].

9. The figure below shows when [r_network_solution_rx64m] is selected. Delete the project (shown as “sample” here) created to use the Smart browser as this is not required.
3.4 Modify Configuration

In this project, the configuration file setting and project setting for each FIT module are changed to configure the application. The detail is shown as follows.

Refer to this information when building new project. To use the project imported, go to “4. Verify Operation”.

3.4.1 Change Configuration

The configuration files for each FIT module configuring this application require modification.

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

The places to be changed in the configuration files are shown below.

(1) Change Interrupt Stack Size

In this application, the TCP/IP 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

```c
/* 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
```

(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

```c
/* The interrupt priority level to be used for CMT interrupts. */
#define CMT_RX_CFG_IPR (2)
```
(3) Change USB Driver Settings
Set channel 0 to Unused (USB_NOUSE_PP) and set channel 1 to Host (USB_HOST_PP).

```c
/*
   Select USB mode to USB IP(USBb)
   USB_HOST_PP  : USB Host Mode
   USB_PERI_PP  : USB Peripheral Mode
   USB_NOUSE_PP : Not Used (USBb)
*/
#define USB_FUNCSEL_USBIP0_PP USB_NOUSE_PP

/*
   Select USB mode to USB IP(USBAa/USBA)
   USB_HOST_PP  : USB Host Mode
   USB_PERI_PP  : USB Peripheral Mode
   USB_NOUSE_PP : Not Used (USBAa/USBA)
*/
#define USB_FUNCSEL_USBIP1_PP USB_HOST_PP
```

```c
#define USB_MEDIA_INITIALIZE(data1) R_tfat_disk_initialize((uint8_t)data1)
```
(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, ftp_callback, ftp_data_callback and dns_callback function.

```
r_t4_rx/src/config_tcpudp.c

#include "r_t4_itcip.h"
//extern ER t4_callback(ID cepid, FN fnccd, VP p_parblk);
extern ER http_callback(ID cepid, FN fnccd, VP p_parblk);
extern ER ftp_callback(ID cepid, FN fnccd, VP p_parblk);
extern ER ftp_data_callback(ID cepid, FN fnccd, VP p_parblk);
extern ER dns_callback(ID cepid, FN fnccd, VP p_parblk);
```

Change the TCP reception point setting as shown below.

```
r_t4_rx/src/config_tcpudp.c

/*** Definition of TCP reception point (only port number needs to be set) ***/
T_TCP_CREP tcp_crep[18] =
{
    /* HTTP server use setting below. */
    { 0x0000, { 0, 80 }},
    { 0x0000, { 0, 80 }},
    { 0x0000, { 0, 80 }},
    { 0x0000, { 0, 80 }},
    { 0x0000, { 0, 80 }},
    /* FTP server use setting below. */
    { 0x0000, { 0, 21 }},
    { 0x0000, { 0, 20 }}, /* The port number in the active mode is 20, and the port number in the passive mode is 1024. */
    { 0x0000, { 0, 21 }},
    { 0x0000, { 0, 20 }}, /* The port number in the active mode is 20, and the port number in the passive mode is 1025. */
    { 0x0000, { 0, 21 }},
    { 0x0000, { 0, 20 }}, /* The port number in the active mode is 20, and the port number in the passive mode is 1026. */
    { 0x0000, { 0, 21 }},
    { 0x0000, { 0, 20 }}, /* The port number in the active mode is 20, and the port number in the passive mode is 1024. */
    { 0x0000, { 0, 21 }},
    { 0x0000, { 0, 20 }}, /* The port number in the active mode is 20, and the port number in the passive mode is 1025. */
    { 0x0000, { 0, 21 }},
};
```
Change the TCP communication end point setting as shown below.

**r_t4_rx/src/config_tcpudp.c**

```c
/*** Definition of TCP communication end point
   (only receive window size needs to be set) ***/
T_TCP_CCEP tcp_ccep[18] =
{
    /* 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 */
    /* HTTP server use setting below. */
    { 0, 0, 0, 0, 1460, http_callback },
    { 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 },
}

/* FTP server use setting below. */
{/ 0, 0, 0, 0, 128, ftp_callback }, / FTP control connection */
{ 0, 0, 0, 0, 1460, ftp_data_callback }, /* FTP data connection */
{ 0, 0, 0, 0, 128, ftp_callback }, /* FTP control connection */
{ 0, 0, 0, 0, 1460, ftp_data_callback }, /* FTP data connection */
{ 0, 0, 0, 0, 128, ftp_callback }, /* FTP control connection */
{ 0, 0, 0, 0, 1460, ftp_data_callback }, /* FTP data connection */
{ 1, 0, 0, 0, 128, ftp_callback }, /* FTP control connection */
{ 1, 0, 0, 0, 1460, ftp_data_callback }, /* FTP data connection */
{ 1, 0, 0, 0, 128, ftp_callback }, /* FTP control connection */
{ 1, 0, 0, 0, 1460, ftp_data_callback }, /* FTP data connection */
{ 1, 0, 0, 0, 128, ftp_callback }, /* FTP control connection */
{ 1, 0, 0, 0, 1460, ftp_data_callback }, /* FTP data connection */
{ 1, 0, 0, 0, 128, ftp_callback }, /* FTP control connection */
{ 1, 0, 0, 0, 1460, ftp_data_callback }, /* FTP data connection */
};
```

Change the 2MSL Wait Time from 1 min. to 10 ms.

**r_t4_rx/src/config_tcpudp.c**

```c
/*** 2MSL wait time (unit:10ms) ***/
const UH _tcp_2msl[] =
{
    (1),  /* 10 ms */
    (1),  /* 10 ms */
};
```
Change the UDP communication end point setting as shown below.

```
T_UDP_CCEP udp_ccep[1] =
{
   /* only setting port number */
   { 0x0000, { 0, 1365 }, dns_callback },
};
```

(5) Change HTTP Server Settings
Set the MAX_CGI_FILE to 2.

```
#define MAX_CGI_FILE 2
```

Change the CGI_FILE_NAME_TABLE_LIST as shown below.

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

Change the number of communication endpoints used in the HTTP server to 6 to match tcp_ccep table of r_t4_rx/src/config_tcpudp.c.

```
#define HTTP_TCP_CEP_NUM 6
```

(6) Change DNS client Settings
Change IP address of the DNS server used to verify the operation (Following is example).

```c
r_config/r_t4_dns_client_rx_config.h

/**************************************************************************
** Configuration Options
**************************************************************************/
#define DNS_IP_ADDR_1 172,30,11,5 /* DNS server IP address (primary) */
#define DNS_IP_ADDR_2 10,29,72,33 /* DNS server IP address (secondary) */
```

(7) Change FTP server Settings
Change the position of the TCP communication end point used in FTP server to match TCP communication end point setting of the T4.

```c
r_config/r_t4_ftp_server_rx_config.h

#include "r_t4_http_server_rx_config.h"
/**************************************************************************
** Configuration Options
**************************************************************************/
#define FTP_TCP_CEP_NUM 12 /* Adapt myself to the communication endpoint number of config_tcpudp.c */

#define FTP_START_TCP_CEP HTTP_TCP_CEP_NUM /* starting position of the communication endpoint in config_tcpudp.c */
```
(8) Change File driver for FTP server and Web server Settings
Change the setting to use the USB.

r_config/r_t4_file_driver_rx_config.h

```c
/* Only one is effective.  
   (priority is given to SERVER_FILE_DRV_USE_EXTERNAL)  
   Setting to 1 : used 
   0 : unused  
*/
define SERVER_FILE_DRV_USE_EXTERNAL (1)  // USB, SD card etc
#define SERVER_FILE_DRV_USE_INTERNAL  (0)  // internal-RAM / external-SDRAM
```

(9) Change M3S-TFAT-Tiny Memory Driver Settings
Change the setting to use the USB.

r_config/r_tfat_driver_rx_config.h

```c
/* Number of logical drives to be used.  
   Setting to 0     : unused memory  
   other : number of logical drives  
   (USB and SDHI can be used together.)  
*/
define TFAT_USB_DRIVE_NUM          (1)  
define TFAT_SDHI_DRIVE_NUM         (0)  
define TFAT_USB_MINI_DRIVE_NUM     (0)  

/* allocate a drive number  
   <valid define>  
   TFAT_CTRL_USB : for USB  
   TFAT_CTRL_SDHI : for SDHI  
   NULL           : unallocated drive  
   MAX 10 drives(TFAT module spec)  
   ex.)  
define TFAT_DRIVE_ALLOC_NUM_0    TFAT_CTRL_USB  
define TFAT_DRIVE_ALLOC_NUM_1    TFAT_CTRL_SDHI  
define TFAT_DRIVE_ALLOC_NUM_2    TFAT_CTRL_SDHI  
define TFAT_DRIVE_ALLOC_NUM_3    TFAT_CTRL_USB  
*/
define TFAT_DRIVE_ALLOC_NUM_0    TFAT_CTRL_USB  
define TFAT_DRIVE_ALLOC_NUM_1    NULL
```
(10) Change Ethernet Settings
Change the PHY IC channel used to control the PHY IC form Ethernet controller to ch0.

r_config/rEther_rx_config.h

/* The register bus of PHY0/1 for ETHER0/1 select
   0 = The access of the register of PHY uses ETHER0.
   1 = The access of the register of PHY uses ETHER1.
*/
#define ETHER_CFG_CH0_PHY_ACCESS (0)
#define ETHER_CFG_CH1_PHY_ACCESS (0)
3.4.2 Change Project Setting
The setting when building is changed from default setting to the content shown in Table 3-1 Changed build setting.

To check the project setting, use the following procedure.

1. Select the project for the e² studio ([r_network_solution_rx64m] or [r_network_solution_rx71m]) and right-click. Then, click [Property (R)].

Click here.
2. Click on [C/C++ Build], and [Setting].

Click here.

Click here.
3. Refer to the contents of Table 3-1 Changed build setting on the [Tool Setting] tab.

**Table 3-1 Changed build setting**

<table>
<thead>
<tr>
<th>Items</th>
<th>Changed contents</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Compiler</strong></td>
<td>Add include path to &quot;Include file directory&quot;</td>
<td>Set an include path that requires the setting for each FIT module. It is automatically set when including each FIT module using “FIT Configurator”.</td>
</tr>
<tr>
<td><strong>Linker</strong></td>
<td>For Little endian: Add ${workspace_loc}/${ProjName}/r_tfat_rx600_little.lib&quot; (Note 1) For Big endian: Add ${workspace_loc}/${ProjName}/r_tfat_rx600_big.lib&quot; (Note 1)</td>
<td>Setting is required when using TFAT. (required when using FAT file system driver) It's not automatically set even when using &quot;FIT Configurator&quot;. Setting by the user is required when creating new project. (Note 2)</td>
</tr>
<tr>
<td></td>
<td>For Little endian: Add ${workspace_loc}/${ProjName}/r_t4_rx/lib/T4_Library_rxv2_ether_little.lib&quot; (Note 1) For Big endian: Add ${workspace_loc}/${ProjName}/r_t4_rx/lib/T4_Library_rxv2_ether_big.lib&quot; (Note 1)</td>
<td>Setting is required when using Ethernet. It's not automatically set even when using &quot;FIT Configurator&quot;. Setting by the user is required when creating new project. (Note 2)</td>
</tr>
<tr>
<td><strong>Linker</strong></td>
<td>Add B ETHERNET BUFFERS_1 section, B RX_DESC_1 section, and B TX_DESC_1 section to RAM capacity</td>
<td>Setting is required when using Ethernet. It's not automatically set even when using &quot;FIT Configurator&quot;. Setting by the user is required when creating new project.</td>
</tr>
</tbody>
</table>

<Setting examples>

<table>
<thead>
<tr>
<th>Address</th>
<th>Section name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>B ETHERNET BUFFERS_1</td>
</tr>
<tr>
<td></td>
<td>B RX_DESC_1</td>
</tr>
<tr>
<td></td>
<td>B TX_DESC_1</td>
</tr>
</tbody>
</table>

(Note 1) The setting change is required when creating the project that includes each FIT module. For the setting, refer to the manuals, etc. in the doc folder of each FIT module.

(Note 2) Automatic setting is available in e² studio V4.1 or later. However, as the FIT module used this time does not meet the specification, the automatic setting function is not available. This will be supported in the latest FIT module.
4. **Verify Operation**

4.1 **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**.

   ![Image of Project Explorer](image1.png)

2. Click `[Build project]` from the **Project** menu.

   ![Image of Project menu](image2.png)
3. When “Build complete” is displayed on the Console panel, the build will have completed.

```
C:enesas\270817\100X0X0C1\UX\UX\COM=1.ERX r_network_solution_rx64m.exe r_network_solution_rx64m.exe Parsing the ELF input file...  
28 segments required UMA fixes  
Converting the GNOME Information...  
Saving the ELF output file r_network_solution_rx64m.exe

18:42:40 Build finished (took 33s.084ms)
```
4.2 Prepare for Debugging

4.2.1 Configure Hardware

The evaluation board must be configured before starting debugging.

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

<table>
<thead>
<tr>
<th>No.</th>
<th>Device</th>
<th>Supplementary Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Development PC</td>
<td>PC used for development</td>
</tr>
<tr>
<td>2</td>
<td>Evaluation board</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Renesas Starter Kit+ for RX64M or Renesas Starter Kit+ for RX71M</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>USB memory</td>
<td>Memory that is formatted as either FAT or FAT32.</td>
</tr>
<tr>
<td>4</td>
<td>Client PC (Web browser / FTP Client )</td>
<td>The development PC can be used for this function.</td>
</tr>
<tr>
<td>5</td>
<td>DHCP server, DNS server</td>
<td>When verify operation of the DHCP client and DNS client.</td>
</tr>
<tr>
<td>6</td>
<td>One of the following must be provided as a network environment.</td>
<td>For example of the network environment, Please refer to the following.</td>
</tr>
<tr>
<td></td>
<td>• Router × 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Switching hub × 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• LAN cable (straight or cross) × 1 to 3</td>
<td></td>
</tr>
</tbody>
</table>

If you have a DNS server and DHCP server

Router Configuration (with DHCP server function)

[Diagram]

Figure 4-1 DNS server and DHCP server - Router Configuration (with DHCP server function)
If you have a DNS server and DHCP server

Switching Hub Configuration

If you don't have a DNS server and DHCP server

Switching Hub Configuration

Figure 4-2 DNS server and DHCP server – Switching Hub Configuration

Figure 4-3 No DNS server and No DHCP server – Switching Hub Configuration
If you don’t have a DNS server and DHCP server
LAN Cable (cross) Configuration

---

Figure 4-4  No DNS server and No DHCP server – LAN Cable (cross) Configuration
(1) **Set Up USB Configuration**

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`.

### Table 4-2 Jumper Settings

<table>
<thead>
<tr>
<th>No.</th>
<th>Setting</th>
<th>Jumper 1</th>
<th>Setting contents</th>
<th>The setting used to verify the operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>When use USB0 in host mode. (USB_FUNCSEL_USBIP0_PP = USB_HOST_PP)</td>
<td>J2</td>
<td>Short 1 to 2.</td>
<td>Unused</td>
</tr>
<tr>
<td></td>
<td>When use USB0 in peripheral mode. (USB_FUNCSEL_USBIP0_PP = USB_PERI_PP)</td>
<td>J6</td>
<td>Short 2 to 3.</td>
<td>Unused</td>
</tr>
<tr>
<td>2</td>
<td>When use USB1 in host mode. (USB_FUNCSEL_USBIP1_PP = USB_HOST_PP)</td>
<td>J7</td>
<td>Short 1 to 2.</td>
<td>Used</td>
</tr>
<tr>
<td></td>
<td>When use USB1 in peripheral mode. (USB_FUNCSEL_USBIP1_PP = USB_PERI_PP)</td>
<td>J9</td>
<td>Short 2 to 3.</td>
<td>Used</td>
</tr>
</tbody>
</table>

Note: The image and the actual settings differ.
(2) **Set Up Ethernet Configuration**

1. 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 *rEther_rx_config.h*.
2. If Ethernet controller (ET0) is used, set the switches and jumpers.
3. If Ethernet controller (ET1) is used, set the jumpers.

<table>
<thead>
<tr>
<th>Table 4-3 Ethernet Setting list</th>
</tr>
</thead>
<tbody>
<tr>
<td>No.</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Figure 4-6  Renesas Starter Kit+ for RX64M/RX71M Jumper locations [Ethernet Controller PHY IC]

Figure 4-7  Renesas Starter Kit+ for RX64M/RX71M Jumper locations [Ethernet Controller ET0/ET1]

(Note) ET0 : Red, ET1 : Orange
4.2.3 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].

2. Click [Network and Sharing Center].
3. Click [Change adapter settings].

4. Right click [Local Area Connection] and select [Properties].
5. Select [Internet Protocol Version 4 (TCP/IPv4)] and click [Properties].

Click here.
6. The IP address and other settings will be displayed. Set these as shown below and click [OK] depending on the existence of a DHCP server.

(a) If exist the DHCP server (Obtain an IP address automatically).

(b) If the DHCP server does not exist (Fixed IP address).
4.2.4 Prepare USB Memory

Store the HTML content on the USB memory.

1. Open the src folder in the project, then open the contents folder in the folder. Open the contents.zip file in the contents folder. Copy the contents folder to the USB memory.
4.3 **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 or Renesas Starter Kit+ for RX71M) to the adapter and turn on the power.

3. Click [Debug Configurations] in the e² studio Run menu.

![Click here.](image-url)
4. Click on [r_network_solution_rx64m HardwareDebug] or [r_network_solution_rx71m HardwareDebug] under [Renesas GDB Hardware Debugging] and click on [Debug].

Click on the [Debugger] tab, and the [Connection Setting] tab.

Change [EXTAL frequency] to 24.0000, then, [Power target from the emulator] to [No (Note 1)].

When completed, click on [Debug (D)].

Note 1 : The setting is used when using an external power supply. Select [Yes] for Power target from the emulator.

When the following message is displayed, click [Yes].
When the load module download completes, a **Debug** perspective opens.

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

![Screenshot of Debug perspective]

6. After the break at the start of the main function, click [Resume] on the tool bar again.
4.3.1 Verify operation of the HTTP server

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

<table>
<thead>
<tr>
<th>Ethernet Port Number</th>
<th>DHCP server</th>
<th>Web Server Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not existing</td>
<td><a href="http://192.168.0.3">http://192.168.0.3</a> (can be changed in the configuration file)</td>
</tr>
<tr>
<td>1</td>
<td>Not support</td>
<td><a href="http://192.168.0.10">http://192.168.0.10</a></td>
</tr>
</tbody>
</table>

Note: Note that the http server address can be changed in the configuration file.

The procedure for confirmation about IP address obtained from the DHCP server is shown below.
When complete the connection, open the `src/main.c`
Set the breakpoint to the `nop()` function in `main.c`.

![Set breakpoint.](image-url)
Execute a Program to confirm the program stopped at the set breakpoint. Then, confirm the variables “dhcp” in the watch window.

The information set to each member of the variables “dhcp”:
- ipaddr : IP address
- maskaddr : Subnetmask
- gwaddr : Gateway address
- dnsaddr : Primary DNS server IP address
- dnsaddr2 : Secondary DNS server address

A list of files in the root directory on the USB memory is displayed. The file name is listed in the Name field, the last update date is listed in the Last modified field. For size, if it is directories, (dir) is listed and the size is shown in bytes. Click on [Parent Directory] to move to the next higher directory.
2. Click [CONTENTS] and then click the file [DEMO.HTM]. This will display a page like the one shown below.

![Web Server Demonstration Program](image)

3. Confirm LED control using CGI function
   Press the [LEDx switch] button to control (turn on or off) LEDs on the board.

![LED Control DEMO](image)

4. Confirm DNS client function using CGI function
   Input any domain name into the "Input domain name" text box and push "Conversion" button.

![DNS DEMO](image)

Web browser shows the IP address confirmation page as follows.
4.3.2 Verify operation of the FTP server

1. Start FTP client software (e.g. FFFTP).

2. Use the user name and password that is defined to "r_t4_ftp_server_rx_config.h", connect to IP address below.

<table>
<thead>
<tr>
<th>Ethernet Port Number</th>
<th>DHCP server</th>
<th>FTP Server Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not existing</td>
<td><a href="http://192.168.0.3">http://192.168.0.3</a> (can be changed in the configuration file.)</td>
</tr>
<tr>
<td></td>
<td>Existing</td>
<td>Obtain from DHCP server</td>
</tr>
<tr>
<td>1</td>
<td>Not support</td>
<td>Not support</td>
</tr>
</tbody>
</table>

A file listing on the root directory in the USB memory is displayed as follows.

Note: This shows the example using the FTP Client software (Japanese version).
5. **Network Middleware Specifications**

For the specifications of the DHCP client, DNS client, HTTP server and FTP server, refer to the document attached to each FIT module.
6. Main Program Specifications

6.1 Files

The main program is included in the project. The source files of the main program is included in the `src` folder. The files of the main program are listed below.

<table>
<thead>
<tr>
<th>Folder Name</th>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>src</td>
<td>main.c</td>
<td>Main source file</td>
</tr>
<tr>
<td></td>
<td>led.c</td>
<td>LED initialization processing source file</td>
</tr>
<tr>
<td></td>
<td>led.h</td>
<td>LED initialization processing header file</td>
</tr>
<tr>
<td></td>
<td>r_http_server_cgi_sample.c</td>
<td>CGI sample source file</td>
</tr>
<tr>
<td></td>
<td>r_sys_time.c</td>
<td>HTTP server system timer source file</td>
</tr>
<tr>
<td></td>
<td>r_sys_time.h</td>
<td>HTTP server system timer header file</td>
</tr>
<tr>
<td></td>
<td>r_usb_hmsc_api.c</td>
<td>USB driver call processing source file</td>
</tr>
<tr>
<td></td>
<td>r_usb_hmsc_api.h</td>
<td>USB driver call processing header file</td>
</tr>
<tr>
<td>src/contents</td>
<td>contents.zip</td>
<td>ZIP file of HTML content to be stored in the USB memory</td>
</tr>
</tbody>
</table>
6.2 Modules

The following table lists the modules in the main program.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Module Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>main.c</td>
<td>main</td>
<td>Main processing for the main program. Calls initialization processing for each of the FIT modules and drives the main processing for the DNS client, HTTP server, FTP server, USB driver, and Ethernet driver (uses an infinite loop to implement periodic activation).</td>
</tr>
<tr>
<td>set_tcpudp_env</td>
<td></td>
<td>Setting the network information which obtained from DHCP server to the environment variable of T4.</td>
</tr>
<tr>
<td>usb_cpu_FunctionUSB0IP</td>
<td></td>
<td>Port initialization and Pin setting of USB ch0.</td>
</tr>
<tr>
<td>usb_cpu_FunctionUSB1IP</td>
<td></td>
<td>Port initialization and Pin setting of USB ch1.</td>
</tr>
<tr>
<td>r_usb_hmsc_api.c</td>
<td>usb_cstd_IdleTaskStart</td>
<td>Starts the idle task used in low-power mode.</td>
</tr>
<tr>
<td></td>
<td>usb_cstd_IdleTask</td>
<td>Idle task used in low-power mode. Performs no processing in host operation.</td>
</tr>
<tr>
<td></td>
<td>usb_hmsc_task_start</td>
<td>HMSC driver activation processing. Performs USB IP initialization and class driver registration.</td>
</tr>
<tr>
<td></td>
<td>usb_apl_task_switch</td>
<td>Performs task scheduling for the USB drivers in non-OS environments.</td>
</tr>
<tr>
<td></td>
<td>usb_hapl_task_start</td>
<td>Starts the HMSC driver application task.</td>
</tr>
<tr>
<td></td>
<td>usb_hmsc_DummyFunction</td>
<td>HMSC driver dummy function</td>
</tr>
<tr>
<td></td>
<td>usb_hmsc DriveOpen</td>
<td>HMSC driver open processing</td>
</tr>
<tr>
<td></td>
<td>usb_hapl_registration</td>
<td>Registers HMSC drivers.</td>
</tr>
<tr>
<td></td>
<td>usb_hmsc_apl_init</td>
<td>Initializes HMSC driver application task internal variables.</td>
</tr>
<tr>
<td>r_http_server_cgi_smalple.c</td>
<td>R_usb_hmsc_StrgDriveSearch() callback processing</td>
<td></td>
</tr>
<tr>
<td>r_sys_time.c</td>
<td>get_sys_time</td>
<td>Please refer to the attached document to HTTP server module (RX Family Web server using the embedded TCP/IP M3S-T4-Tiny Module Firmware Integration Technology / R20AN0075EJ).</td>
</tr>
<tr>
<td></td>
<td>start_system_time</td>
<td>Start the timer for HTTP server.</td>
</tr>
<tr>
<td></td>
<td>update_sys_time</td>
<td>Interrupt handler of the timer for HTTP server.</td>
</tr>
<tr>
<td></td>
<td>stop_system_time</td>
<td>Stop the timer for HTTP server.</td>
</tr>
</tbody>
</table>

| led.c              | led_init               | Initialization of LEDs                                                      |
|                   |                        | Please refer to the attached document to HTTP server module (RX Family Web server using the embedded TCP/IP M3S-T4-Tiny Module Firmware Integration Technology / R20AN0075EJ). |

| r_http_server_cgi_smalple.c |                        | Please refer to the attached document to HTTP server module (RX Family Web server using the embedded TCP/IP M3S-T4-Tiny Module Firmware Integration Technology / R20AN0075EJ). |
| r_sys_time.c            | get_sys_time           | Please refer to the attached document to HTTP server module (RX Family Web server using the embedded TCP/IP M3S-T4-Tiny Module Firmware Integration Technology / R20AN0075EJ). |
|                         | start_system_time      | Start the timer for HTTP server.                                            |
|                         | update_sys_time        | Interrupt handler of the timer for HTTP server.                            |
|                         | stop_system_time       | Stop the timer for HTTP server.                                            |
6.3 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, DNS client, HTTP server and FTP server main processing, and USB driver scheduling from an infinite loop.
(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.

```
usb_cstd_IdleTask

return
```
(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
  | R_usb_cstd_GetUsbIpAddr
  | R_usb_hstd_MgrOpen
  | R_usb_hstd_HcdOpen
  | usb_hapi_registration
  | R_usb_hmsc_hub_registration
  | R_usb_hmsc_driver_start
  | usb_hapi_task_start
  | R_usb_cstd_UsdIpInit
  | return

// Acquires the base register address for the used IP

// Starts the USB-BASIC-FW hostmgr task

// Starts the USB-BASIC-FW hostcontrol driver task

// Registers the HMSC driver

// Registers the host hub class driver

// Starts the HMSC driver task

// Starts the USB sample code main task

// USB IP initialization
```
(5) **usb_apl_task_switch**

Performs the USB driver scheduling.

---

(6) **usb_hapl_task_start**

Initializes the USB memory mount processing task.
(7) **usb_hapl_registration**  
Performs the class driver registration processing.

```c
usb_hapl_registration
Class driver information preparation
R_usb_hstdc_DriverRegistration // Host class driver registration
return
```

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

```c
usb_hmsc_apl_init
Sequence processing variable initialization
return
```

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

```c
usb_hmsc_DummyFunction
return
```
(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
    Acquisition OK?
      No
      return
      Yes
        R_USB_SND_MSG
          Normal completion?
            No
              return
            Yes
              R_USB_REL_BLK
                // Sends USB_HMSC_DRIVE_OPEN to the sample application task.
                // Releases the memory block.
```
(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.

![Diagram of usb_hmsc_SampleAplTask](image)

(12) **led_init**

This function performs the initialization process for using the LED on the Renesas Starter Kit + for RX64M or the Renesas Starter Kit + for RX71M.

![Diagram of led_init](image)
(13) set_tcpudp_env
Setting the network information which obtained from DHCP server to the environment variable of T4.

```
set_tcpudp_env

Set the the environment variable(tcpudp_env) of the T4.
Set the primary address of the DNS server
Set the secondary address of the DNS server
return
```

(14) usb_cpu_FunctionUSB0IP, usb_cpu_FunctionUSB1IP
Performs I/O port and Pin functions initialization for using the USB ch0 and ch1.

```
usb_cpu_FunctionUSB0IP

Setting the I/O port and Pin functions for USB ch0.
If unused:
Do nothing.
If a host mode:
Set the ch0 to host mode.
If a peripheral mode:
Set the ch0 to peripheral mode.
return
```

```
usb_cpu_FunctionUSB1IP

Setting the I/O port and Pin functions for USB ch1.
If unused:
Do nothing.
If a host mode:
Set the ch1 to host mode.
If a peripheral mode:
Set the ch1 to peripheral mode.
return
```

(15) start_system_time
Start the compare match timer (10ms interval) for the system time of HTTP server.

```
start_system_time

R_CMT_CreatePeriodic
// Start the compare match timer of 10ms periods.
return
```

(16) update_sys_time
This function is an interrupt handler that counts up of the system time of HTTP server.

```
update_sys_time

Counts up of the system time at 10ms periods.
return
```
(17) **stop_system_time**

Stop the compare match timer for the system time of HTTP server.

```
stop_system_time
R_CMT_Stop // Stop the compare match timer.
return
```
7. 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 7-1 User-Defined Functions

<table>
<thead>
<tr>
<th>User-Defined Function</th>
<th>File Name</th>
<th>FIT Module Name</th>
<th>Document Name/Catalog Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>System timer interface</td>
<td>r_sys_time.c</td>
<td>r_t4_http_server_rx</td>
<td>RX Family Web server using the embedded TCP/IP M3S-T4-Tiny Module Firmware Integration Technology / R20AN0075EJ (4.2 System Timer)</td>
</tr>
</tbody>
</table>
8. Supplement

8.1 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. For the setting examples, refer to “3.4.1(3) Change USB Driver Settings.”

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

8.3 FTP Server and DHCP Client Limitations
FTP server and DHCP client can be operated at Ethernet ch0 only.

8.4 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 exceeds, 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
## Revision History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Sep 1, 2014</td>
<td>—</td>
<td>First edition issued</td>
</tr>
<tr>
<td>1.03</td>
<td>Feb 29, 2016</td>
<td>—</td>
<td>Added DHCP client, DNS client and FTP server. Added RX71M. Changed an application note to R01AN2608EJ</td>
</tr>
</tbody>
</table>
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.