RX Family

Lightweight IP (lwIP) Driver Using Firmware Integration Technology

Introduction

This application note describes the lightweight IP (lwIP) driver FIT module (r_lwip_driver_rx), which is compliant with the firmware integration technology (FIT) concept.

In this application note, software of the lightweight IP (lwIP) driver FIT module is generally referred to as “the lwIP driver FIT module” or “this FIT module”.

Target Devices

RX65N group
RX72M group
RX72N group
RX64M group
RX71M group
RX66N group

When applying the material covered in this application note on another microcontroller, modify the descriptions or code to suit the specifications of the target microcontroller and extensively evaluate operation following the modifications.

Documents for Reference

1 Firmware Integration Technology User’s Manual (R01AN1833)
2 RX Family: Board Support Package Module Using Firmware Integration Technology (R01AN1685)
3 Adding Firmware Integration Technology Modules to Projects (R01AN1723)
4 Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)
5 Renesas e^2 studio Smart Configurator User's Guide (R20AN0451)
6 RX Family: Ethernet Module Using Firmware Integration Technology (R01AN2009)
7 RX Family: CMT Module Using Firmware Integration Technology (R01AN1856)
8 RX family: TSIP (Trusted Secure IP) Module Firmware Integration Technology (R20AN0371)
9 lwIP - A Lightweight TCP/IP stack (link)
10 Lightweight IP stack (link)
11 lwip-tcpip GitHub (link)
12 RX Family: Lightweight IP (lwIP) Module Using Firmware Integration Technology (R20AN0789)

Contents

1. Overview 4
1.1 What is the “lwIP Driver FIT Module”? 4
1.2 Overview of the lwIP Driver FIT Module 4
1.2.1 Configuration of the Software 4
1.2.2 Overview of API Functions 6

2. API Information 7
2.1 Hardware Requirements 7
2.2 Software Requirements 7
2.3 Supported Toolchains 7
2.4 Interrupt Vectors to be Used 7
2.5 Header File 7
2.6 Integer Types 7
2.7 Settings for Compilation 8
2.8 Code Size 9
2.9 Arguments 10
2.10 Return Values 10
2.11 Including the FIT Module in Your Project 11
2.12 for, while, and do while Statements 12

3. API Functions 13
3.1 R_LWIP_DRIVER_Open() 13
3.2 R_LWIP_DRIVER_Close() 15
3.3 R_LWIP_DRIVER_EthernetClose() 16
3.4 R_LWIP_DRIVER_EthernetLinkCheck() 18
3.5 r_lwip_driver_ethernetif_init() 20
3.6 R_LWIP_DRIVER_Input() 22
3.7 r_lwip_driver_low_level_output() 24
3.8 r_lwip_driver_get_rand() 26

4. Restrictions 27
4.1 Restrictions 27

5. Appendix 28
5.1 Environments for Confirming Operation 28
5.2 Sample Projects 29
5.3 Support for Multi-Interface Usage 30
5.3.1 Implementing Processing Required for Multiple Interfaces 30
5.3.1.1 Defining the Channel Number Macro for the Network Interface for Use through the Second Channel 30
5.3.1.2 Creating the r_lwip_driver_ethernetif_init_2() Function 31
5.3.1.3 Creating the r_lwip_driver_low_level_init_2() Function 32

6. Troubleshooting 34
6.1 Troubleshooting 34

7. Reference Documents 36

Revision History 37

1. Overview

1.1 What is the “lwIP Driver FIT Module”?

This FIT module is used by including it as a set of API functions for use in a project. For details on how to include the FIT module, see section 2.11, Including the FIT Module in Your Project.

1.2 Overview of the lwIP Driver FIT Module

This FIT module is for use with the lwIP FIT module (r_lwip_rx, referred to as lwIP) as the upper layer and supports Ethernet communications in conformance with the TCP/IP. This FIT module provides a porting layer for the lwIP FIT module, which is configured from the lwIP open-source software (OSS), on RX group microcontrollers.

1.2.1 Configuration of the Software

Figure 1 illustrates the configuration of the software.

This FIT module (the lwIP driver FIT module) operates in combination with the lwIP FIT module.

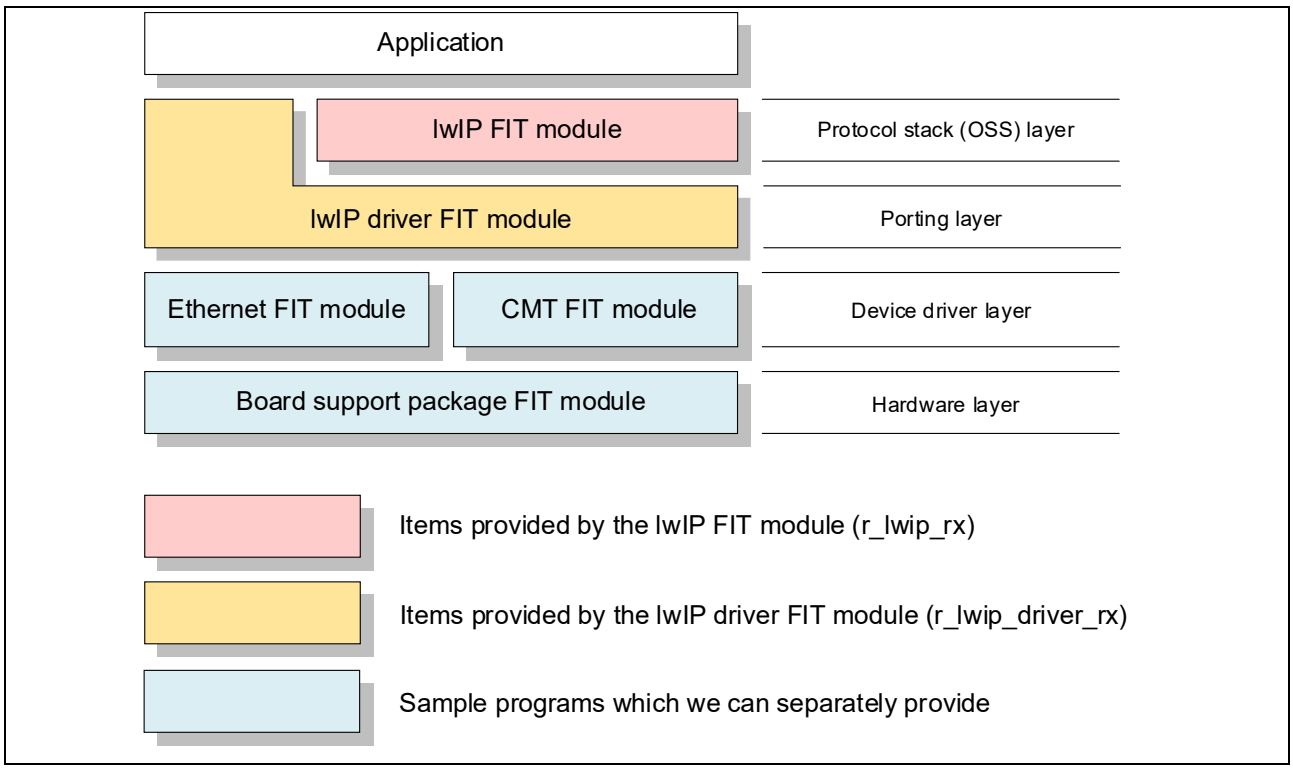

Figure 1 Schematic Diagram of the Software

(1) lwIP driver FIT module (r_lwip_driver_rx)
Target software covered in this application note, which is referred to as this FIT module. It serves as the porting layer for the lwIP FIT module (lwIP).

(2) lwIP FIT module (r_lwip_rx)
A module which implements the lwIP, an open-source TCP/IP stack. It is packaged as a FIT module. Refer to the documentation under the heading “Documents for Reference” on the first page for details.

(3) Ethernet FIT module (r_ether_rx)
This module handles the transfer of Ethernet frames. A sample program is available. For details on how to get it, refer to the corresponding documentation under the heading “Documents for Reference” on the first page.

(4) CMT FIT module (r_cmt_rx)
This module uses the compare match timer (CMT) units to measure times within the lwIP FIT module (lwIP). A sample program is available. For details on how to get it, refer to the corresponding documentation under the heading “Documents for Reference” on the first page.

(5) Board support package FIT module (r_bsp)
This module sets up the microcontroller. A sample program is available. For details on how to get it, refer to the corresponding documentation under the heading “Documents for Reference” on the first page.

1.2.2 Overview of API Functions

Table 1.1 lists the API functions which are defined in this FIT module.

Table 1.1 API Functions for Use with the lwIP Driver FIT Module

API Function	Description
R_LWIP_DRIVER_Open	Makes the initial settings for this FIT module.
R_LWIP_DRIVER_Close	Releases resources that had been acquired by this FIT module.
R_LWIP_DRIVER_EthernetClose	Disables transmit and receive functionality on the ETHERC.
R_LWIP_DRIVER_EthernetLinkCheck	Checks the state of the Ethernet link.
r_lwip_driver_ethernetif_init	Makes the initial settings for the user-defined netif structure and handles the initial setup of the Ethernet FIT module. Direct calls of this function by user software are prohibited.
R_LWIP_DRIVER_Input	Reads a received Ethernet frame.
r_lwip_driver_low_level_output	Writes an Ethernet frame for transmission. Direct calls of this function by user software are prohibited.
r_lwip_driver_get_rand	Generates a random number.

2. API Information

The operation of this FIT module has been confirmed under the conditions listed below.

2.1 Hardware Requirements

The microcontroller to be used must include the following modules.

- Ethernet controller (ETHERC)
- DMA controller for the Ethernet controller (EDMAC)
- Compare match timer (CMT)
- Trusted Secure IP (TSIP) *1

Note 1. Needed when using the MCU incorporating a TSIP module.

2.2 Software Requirements

This FIT module is dependent on the following FIT modules.

- r_bsp (Board support package FIT module, see document 2)
- r_ether_rx (Ethernet FIT module, see document 6)
- r_cmt_rx (CMT FIT module, see document 7)
- r_lwip_rx (lwIP FIT module, see document 12)
- r_tsip_rx (TSIP FIT module, see document 8) *1

Note 1. Needed when using the MCU incorporating a TSIP module.

2.3 Supported Toolchains

The operation of this FIT module has been confirmed with the toolchains indicated in section 5.1, Environments for Confirming Operation.

2.4 Interrupt Vectors to be Used

None

2.5 Header File

All API calls of this FIT module are defined in r_lwip_driver_rx_if.h.

2.6 Integer Types

This FIT module uses ANSI C99 integer types, which are defined in stdint.h.

2.7 Settings for Compilation

The configuration options for this FIT module are defined in r_lwip_driver_rx_config.h.

The option names and settings are described in Table 2.1.

Table 2.1 Configuration Options (r_lwip_driver_rx_config.h)

Configuration Options in r_lwip_driver_rx_config.h	
LWIP_DRIVER_CFG_ETH_MAC_ADDR0 *The default setting is 0x74.	These options set the MAC address. Use the LWIP_DRIVER_CFG_ETH_MAC_ADDR0 to LWIP_DRIVER_CFG_ETH_MAC_ADDR5 options to set the address. The LWIP_DRIVER_CFG_ETH_MAC_ADDR0 option is the first octet. Values in the range from 0x00 to 0xFF can be set for each option.
LWIP_DRIVER_CFG_ETH_MAC_ADDR1 *The default setting is 0x90.	
LWIP_DRIVER_CFG_ETH_MAC_ADDR2 *The default setting is 0x50.	
LWIP_DRIVER_CFG_ETH_MAC_ADDR3 *The default setting is 0x10.	
LWIP_DRIVER_CFG_ETH_MAC_ADDR4 *The default setting is 0xFE.	
LWIP_DRIVER_CFG_ETH_MAC_ADDR5 *The default setting is 0x79.	
LWIP_DRIVER_CFG_ETH_DRV_CH *The default setting is 0.	Sets the channel number of the ETHERC to be used by the lwIP. The default setting is for channel 0 of the ETHERC. Set the value corresponding to the channel of the ETHERC that is to be controlled. Specifically, set 0 or 1 for devices with two ETHERC channels or 0 for other devices.
LWIP_DRIVER_CFG_MTU *The default setting is 1500.	Sets the maximum unit for transfer. Values in the range from 576 to 1500 can be set.
LWIP_DRIVER_CFG_SEND_MAX_LOOP *The default setting is 20.	Sets the number of times to retry processing until the transmission buffer has empty space when this FIT module is unable to acquire it because it is full.
LWIP_DRIVER_CFG_SEND_DELAY_US *The default setting is 50.	Sets the interval in microseconds for retrying processing until the transmission buffer has empty space when this FIT module is unable to acquire it because it is full.

2.8 Code Size

Table 2.2 lists the ROM and RAM sizes and the maximum stack usage for this FIT module.

The configuration options listed in section 2.7, Settings for Compilation, during the build process determine the actual ROM (code and constants) and RAM (global data) sizes.

The values listed in Table 2.2 have been confirmed under the following conditions.

FIT module revision: r_lwip_driver_rx rev1.00
Compiler version: Renesas Electronics C/C++ Compiler Package for RX Family V3.07.00
(The “-lang = c99” option was added to the default settings of the integrated development environment.)
Compilation options: Default settings

Table 2.2 Code Size

Code Sizes of ROM, RAM, and Stack			
Device	**Category**	**Memory Used**	**Remarks**
RX65N	ROM	Approximately 2k bytes	-
	RAM	12 bytes	-
	Maximum stack usage	Approximately 500 bytes	This is a value measured using the sample project included with the package for this FIT module (see section 5.2). This value includes the stack usage of other FIT modules that this FIT module calls.

2.9 Arguments

The following structure is used to provide the arguments for the API functions of this FIT module.

The following structure is defined in the netif.h file of the lwIP FIT module (lwIP).

- netif /* General-purpose data structure for the lwIP FIT module (lwIP) */

2.10 Return Values

The following enumerated type is used to provide the return values for the API functions of this FIT module.

The following enumerated type is defined in the r_lwip_driver_rx_if.h file.

```
typedef enum    /* Error codes for the API functions of this FIT module */
{
    LWIP_DRV_SUCCESS,           /* Successful completion */
    LWIP_DRV_ERR_OPEN,          /* The R_LWIP_DRIVER_Open() function has already been executed. */
    LWIP_DRV_ERR_NOT_OPEN,      /* The R_LWIP_DRIVER_Open() function has not been executed. */
    LWIP_DRV_ERR_ARG,           /* An argument is incorrect. */
    LWIP_DRV_ERR_CMT            /* An error has occurred in the CMT FIT module. */
} e_lwip_drv_return_t;
```

The following enumerated type is defined in the err.h file of the lwIP FIT module (lwIP).

- err_enum_t /* Definition of the err_t return values for use by the API functions of the lwIP FIT module (lwIP) */

2.11 Including the FIT Module in Your Project

This FIT module must be included in each project where it is to be used. Renesas recommends using the Smart Configurator in the way described in (1), (3), or (5) below. However, the Smart Configurator only supports this for some RX devices. If the RX device in use is not supported, use the method described in either (2) or (4).

(1) Including the FIT module in your project by using the Smart Configurator in the e^2 studio
Using the Smart Configurator in the e^2 studio allows the automatic inclusion of the FIT module in your project. Refer to the application note *RX Smart Configurator User's Guide: e² studio* (R20AN0451) for details.

(2) Including the FIT module in your project by using the FIT Configurator in the e^2 studio
Using the FIT Configurator in the e^2 studio allows the automatic inclusion of the FIT module in your project. Refer to the application note *RX Family: Adding Firmware Integration Technology Modules to Projects* (R01AN1723) for details.

(3) Including the FIT module in your project by using the Smart Configurator in CS+
Using the standalone version of the Smart Configurator in CS+ allows the automatic inclusion of the FIT module in your project. Refer to the application note *RX Smart Configurator User's Guide: CS+* (R20AN0470) for details.

(4) Including the FIT module in your project in CS+
Manually include the FIT module in your project in CS+. Refer to the application note *RX Family: Adding Firmware Integration Technology Modules to CS+ Projects* (R01AN1826) for details.

(5) Including the FIT module in your project by using the Smart Configurator with the IAR Embedded Workbench (IAREW)
Using the standalone version of the Smart Configurator allows the automatic inclusion of the FIT module in your project. Refer to the application note *RX Smart Configurator User's Guide: IAREW* (R20AN0535) for details.

2.12 for, while, and do while Statements

This FIT module uses various types of loop processing with the use of for, while, or do while statements in specific processing such as waiting for registers to reflect written values. Such loop processing is written under or next to comments with “WAIT_LOOP” as a keyword. Accordingly, when embedding your own fail-safe processing in loop processing, you can use “WAIT_LOOP” to search for the relevant processing.

The listings below show examples of loop processing.

```
Example of a while statement:
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
   /* The delay period needed is to make sure that the PLL has stabilized. */
}

Example of a for statement:
/* Initialize reference counters to 0. */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
   g_protect_counters[i] = 0;
}

Example of a do while statement:
/* Reset completion waiting */
do
{
   reg = phy_read(ether_channel, PHY_REG_CONTROL);
   count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /* WAIT_LOOP */
```

3. API Functions

3.1 R_LWIP_DRIVER_Open()

This function makes the initial settings for this FIT module.

Format

```
e_lwip_drv_return_t R_LWIP_DRIVER_Open(
    uint32_t *p_cmt_ch,
    uint16_t ms
)
```

Parameters

p_cmt_ch (OUT)	This parameter holds the CMT channel number acquired by the CMT FIT module.
ms (IN)	Specify a value in milliseconds for use in obtaining the interval between interrupts generated by the CMT. The values specifiable as this argument are from 1 to 1000 and by which 1000 is divisible; that is, values such that the result of dividing 1000 by the value is an integer.

Return Values

LWIP_DRV_SUCCESS	/* Successful completion */
LWIP_DRV_ERR_OPEN	/* This function has already been executed. */
LWIP_DRV_ERR_ARG	/* An argument is incorrect. */
LWIP_DRV_ERR_CMT	/* An error has occurred in the CMT FIT module. */

Properties

Its prototype is declared in r_lwip_driver_if.h.

Description

This function calls the R_CMT_CreatePeriodic() function for use with the CMT FIT module and the R_ETHER_Initial() function for use with the Ethernet FIT module to make the initial settings.

The result of dividing 1000 by the setting of the second argument (1000/ms) is set as the integer type for the interval between interrupts generated by the CMT. We recommend setting the second argument to a value by which 1000 is divisible to avoid an error in the precision of counting.

Reentrant

Not usable

Thread Safety

Not supported

Example

```
e_lwip_drv_return_t rx_lwip_drv_ret;
uint32_t cmt_ch;

#if  BSP_CFG_MCU_PART_ENCRYPTION_INCLUDED == true ¥
      ||  BSP_CFG_MCU_PART_FUNCTION == (0x11)
R_TSIP_Open(NULL,NULL);
#endif /*  BSP_CFG_MCU_PART_ENCRYPTION_INCLUDED == true¥
       ||  BSP_CFG_MCU_PART_FUNCTION == (0x11) */

rx_lwip_drv_ret = R_LWIP_DRIVER_Open(&cmt_ch, 1);

/* Initialize the lwIP stack */
lwip_init();
```

Special Note

Be sure to execute this function before calling API functions of the lwIP FIT module and other API functions for use with this FIT module.

In addition, if the MCU used is equipped with a Trusted Secure IP (TSIP) module, be sure to call R_TSIP_Open() function of the TSIP FIT (r_tsip_rx) module before executing this function.

3.2 R_LWIP_DRIVER_Close()

This function releases resources that had been acquired by the lwIP driver FIT module.

Format

```
e_lwip_drv_return_t R_LWIP_DRIVER_Close(
    uint32_t cmt_ch
)
```

Parameter

cmt_ch (IN)	Specify the CMT channel number given in the R_LWIP_DRIVER_Open() function call.

Return Values

LWIP_DRV_SUCCESS	/* Successful completion */
LWIP_DRV_ERR_NOT_OPEN	/* The R_LWIP_DRIVER_Open() function has not been executed. */
LWIP_DRV_ERR_CMT	/* An error has occurred in the CMT FIT module. */

Properties

Its prototype is declared in r_lwip_driver_if.h.

Description

This function calls the R_CMT_Stop() function of the CMT FIT module to release the resources it is using.

Reentrant

Not usable

Thread Safety

Not supported

Example

```
e_lwip_drv_return_t rx_lwip_drv_ret;
uint32_t cmt_ch = 0;

rx_lwip_drv_ret = R_LWIP_DRIVER_Close(cmt_ch);
```

Special Note

Call this function after having called the R_LWIP_DRIVER_EthernetClose() function to disable transmit and receive functionality on the ETHERC in use.

3.3 R_LWIP_DRIVER_EthernetClose()

This function disables transmit and receive functionality on the ETHERC.

Format

```
e_lwip_drv_return_t R_LWIP_DRIVER_EthernetClose(
    struct netif *netif
)
```

Parameter

netif (IN)	A user-declared netif variable

Return Values

LWIP_DRV_SUCCESS	/* Successful completion */
LWIP_DRV_ERR_ARG	/* An argument is incorrect. */

Properties

Its prototype is declared in r_lwip_driver_if.h.

Description

This function calls the R_ETHER_CheckWrite() function of the Ethernet FIT module to cause waiting for the completion of transmission through the Ethernet channel specified by the name[1] member of the netif structure (netif->name[1]) set as the argument of this function. After that, the function calls the R_ETHER_Close_ZC2()
function to release the resources corresponding to the given Ethernet channel.

Be sure to call this function before calling the R_LWIP_DRIVER_Close() function.

When multiple network interfaces are in use, call this API function to release the resources for the Ethernet FIT module corresponding to all Ethernet channels before calling the R_LWIP_DRIVER_Close() function.

Reentrant

Not usable

Thread Safety

Not supported

Example

```
e_lwip_drv_return_t rx_lwip_drv_ret;

rx_lwip_drv_ret = R_LWIP_DRIVER_EthernetClose(&gnetif);

rx_lwip_drv_ret = R_LWIP_DRIVER_Close(cmt_ch);
```

Special Notes

This API function can be used when the Ethernet FIT module and PHY-LSI are to be re-initialized following an error in Ethernet transfer.

After the call of this function, use the appropriate API functions of the lwIP FIT module (lwIP) to initialize the Ethernet interface. The listing below shows an example of code implemented to close the Ethernet FIT module, ETHERC module, and Ethernet PHY-LSI, and then to initialize the Ethernet interface.

```
R_LWIP_DRIVER_EthernetClose(&gnetif);
netif_remove(&gnetif);
netif_add(&gnetif, &ipaddr, &netmask, &gw, NULL, &r_lwip_driver_ethernetif_init,
&netif_input);

/* Check if the ethernet link up */
if(ETHER_SUCCESS == R_ETHER_CheckLink_ZC (LWIP_DRIVER_CFG_ETH_DRV_CH))
{
   /* Link is up */
   netif_set_default(&gnetif);
   netif_set_up(&gnetif);

}
```

3.4 R_LWIP_DRIVER_EthernetLinkCheck()

This function checks for the link state of the Ethernet PHY-LSI.

Format

```
void R_LWIP_DRIVER_EthernetLinkCheck(
    struct netif *netif
)
```

Parameter

netif (IN)	A user-declared netif variable

Return Values

None

Properties

Its prototype is declared in r_lwip_driver_if.h.

Description

This function reads the state of the Ethernet PHY-LSI and handles the following processing according to the state of the Ethernet link that is read.

If the state read is such that the Ethernet link is up but the link held by the lwIP FIT module (lwIP) is down, the function calls the netif_set_link_up() function for the lwIP to bring the link held by the lwIP up. On the other hand, if the state read is such that the Ethernet link is down but the link held by the lwIP is up, the function calls the netif_set_link_down() function for the lwIP to bring the link held by the lwIP down.

To update the state of the link held by the lwIP, repeatedly call this function from the main loop.

Table 3.1 Processing by This Function

	State of the Ethernet Link	
State of the lwIP Link	**Up**	**Down**
Up	No processing	Calling the netif_set_link_down() function
Down	Calling the netif_set_link_up() function	No processing

Reentrant

Not usable

Thread Safety

Not supported

Example

```
/* Global */
struct netif gnetif;
uint32_t EthernetLinkTimer = 0;

void Ethernet_Link_Periodic_Handle(struct netif *netif)
{
    /* Check the ethernet link every 100ms */
    if (sys_now() - EthernetLinkTimer >= 100)
    {
        EthernetLinkTimer = sys_now();
        R_LWIP_DRIVER_EthernetLinkCheck(netif);
    }
}

void main()
{
    uint32_t counts = 0;

    netif_set_default(&gnetif);
    netif_set_up(&gnetif);

    while(1)
    {
        Ethernet_Link_Periodic_Handle(&gnetif);

        sys_check_timeouts();

        if (netif_is_link_up(&gnetif) && counts == 0)
        {
            counts++;
            tcp_echoclient_connect();
        }
    }
}
```

Special Note

The name[1] member of the netif structure set as the argument of this function (netif->name[1]) specifies the channel of the Ethernet PHY-LSI for which this API function is to check the link state. The value of netif->name[1] is initialized by the r_lwip_driver_ethernetif_init() function. Direct changes to the value of netif->name[1] by user software are prohibited.

3.5 r_lwip_driver_ethernetif_init()

This function makes the initial settings for the netif structure and handles the initial setup of the Ethernet FIT module.

Format

```
err_t r_lwip_driver_ethernetif_init(
    struct netif *netif
)
```

Parameter

netif (IN/OUT)	A user-declared netif structure

Return Values

ERR_OK	/* Successful completion */
ERR_IF	/* An error has occurred in the Ethernet FIT module. */

Properties

Its prototype is declared in r_lwip_driver_if.h.

Description

This function makes the initial setting for the structure to be set as the netif argument. It also calls the R_ETHER_Open_ZC2() function of the Ethernet FIT module to initialize the module.

This function sets the interface standard for use by the Ethernet FIT module between the ETHERC and Ethernet PHY-LSI. It uses the ETHER_CFG_MODE_SEL macro of the Ethernet FIT module to select either the media independent interface (MII) or the reduced media independent interface (RMII) as the interface standard to be used.

Direct calls of this function by user software are prohibited. Execute the netif_add() function of the lwIP FIT module (lwIP) with the pointer to this function specified as the sixth argument; this function will then be executed from within the netif_add() function.

Reentrant

Not usable

Thread Safety

Not supported

Example

```
netif_add(&gnetif, &ipaddr, &netmask, &gw, NULL, &r_lwip_driver_ethernetif_init,
&netif_input);
```

Special Notes

Use the pin-setting window of the Smart Configurator to make the ETHERC pin settings.

When multiple network interfaces are to be used (multi-interface usage), this function does not handle processing to initialize the network interface for the second channel. Refer to the implementation of this function to implement the processing required to initialize the network interface for the second channel. At that time, set a value that is different from that set for the first channel in the name[1] member of the netif structure (netif->name[1]). For details on the case where multiple interfaces are to be used, see section 5.3.

3.6 R_LWIP_DRIVER_Input()

This function reads a received Ethernet frame.

Format

```
void R_LWIP_DRIVER_Input(
    struct netif *netif
)
```

Parameter

netif (IN)	A user-declared netif structure

Return Values

None

Properties

Its prototype is declared in r_lwip_driver_if.h.

Description

This function calls the R_ETHER_Read_ZC2() function of the Ethernet FIT module to read a received frame. When a frame has been received, the function calls the function (netif->input()), which is set as the input member of the netif structure set as the argument of this function.

The netif_add() function of the lwIP FIT module (lwIP) is used to register a function as the input member of the netif structure. Select the function to be used for the processing of received frames from among the network interface functions provided by the lwIP (netif_input_fn) and set the pointer to the function selected for execution as the seventh argument of the netif_add() function. For details on the netif_add() function, refer to document 9 under the heading “Documents for Reference” on the first page.

Reentrant

Not usable

Thread Safety

Not supported

Example

```
/* Infinite loop */
while (1)
{
    /* Read a received packet from the Ethernet buffers and send it
       to the lwIP for handling */
    R_LWIP_DRIVER_Input(&gnetif);

    /* Handle timeouts */
    sys_check_timeouts();

    Ethernet_Link_Periodic_Handle(&gnetif);

#if LWIP_DHCP
    DHCP_Periodic_Handle(&gnetif);
#endif
}
```

Special Note

Repeatedly call this function from the main loop.

3.7 r_lwip_driver_low_level_output()

This function writes an Ethernet frame for transmission.

Format

```
err_t r_lwip_driver_low_level_output(
    struct netif *netif,
    struct pbuf *p
)
```

Parameters

netif (IN/OUT)	A user-declared netif structure
p (IN)	Pointer to data for transmission

Return Values

ERR_OK	/* Successful completion */
ERR_IF	/* An error has occurred in the Ethernet FIT module. */
ERR_ARG	/* The Ethernet channel number set as an argument is incorrect. */

Properties

Its prototype is declared in r_lwip_driver_if.h.

Description

This function calls the R_ETHER_Write_ZC2_GetBuf() and R_ETHER_Write_ZC2_SetBuf() functions of the Ethernet FIT module to write an Ethernet frame for transmission to the transmission buffer of the Ethernet FIT module, and handles processing to start transmission.

Direct calls of this function by user software are prohibited. This function is registered as the linkoutput member of the netif structure (netif->linkoutput) by using the r_lwip_driver_ethernetif_init() function and is executed during processing by the lwIP FIT module (lwIP).

Reentrant

Not usable

Thread Safety

Not supported

Examples

None

Special Note

When multiple network interfaces are to be used, this function is registered as the linkoutput member of the netif structure by using the function that handles processing to initialize the network interface for the second channel (the r_lwip_driver_ethernetif_init_2() function in the example given in section 5.3.1.2). See section 5.3.1.2 for details.

3.8 r_lwip_driver_get_rand()

This function generates a random integer.

Format

```
uint32_t r_lwip_driver_get_rand(
    void
)
```

Parameters

None

Return Values

32-bit unsigned integer /* Generated random number */

Properties

Its prototype is declared in r_lwip_driver_if.h.

Description

If the MCU to be used incorporates a Trusted Secure IP (TSIP) module, this function handles random number generation processing using true random number generator (TRNG) built in the TSIP module.

If the MCU to be used does not incorporate a TSIP module, this function handles a software pseudo-random number generation process using the standard C library function rand().

Reentrant

Not usable

Thread Safety

Not supported

4. Restrictions

4.1 Restrictions

The restrictions listed below apply to this FIT module.

- Only little endian is supported.
- IPv6 is not supported.
- Only raw or native (bare metal) API functions for use with the lwIP are supported (only functions subject to the “NO_SYS = 1” setting are supported).
  Although the lwIP FIT module (lwIP) supports the following three types of API function, this FIT module only supports raw or native API functions, which are usable when the NO_SYS lwIP configuration setting is 1. For details on these API functions, refer to the API list (APIs) section of document 9 under the heading “Documents for Reference” on the first page.
  - Raw or native API functions: Supported by this FIT module
  - Netconn API functions: Not supported
  - Socket API functions: Not supported
- The netif->name[1] member of the netif structure is used to manage the channel number of the Ethernet PHY-LSI for use with this driver. Set the value to the ASCII code for 0 (0x30) or 1 (0x31). For details on the case where multiple network interfaces are to be used, see section 5.3.1.2.

5. Appendix

5.1 Environments for Confirming Operation

Table 5.1 lists the environments used in confirming operation of this FIT module.

Table 5.1 Environments for Confirming Operation

Item		Description
Integrated development environment		e^2 studio Ver. 2025-01 from Renesas Electronics
Compiler	CC-RX	C/C++ Compiler for RX Family V3.07.00 from Renesas Electronics Compilation options: The following option was added to the default settings of the integrated development environment. -lang = c99
	GCC	GCC for Renesas RX 8.3.0.202411
Endian for operation		Little
Revision number of this FIT module		Rev. 1.00
Board used		Renesas CK-RX65N v2 (part number: RTK5CK65N0S08xxxBE) Renesas RX72N Envision Kit (part number: RTK5RX72N0C00000BJ) Renesas Starter Kit+ for RX72M (part number: RTK5572MNxSx0000BE)
FIT	BSP FIT	Ver. 7.52
	Ethernet FIT	Ver. 1.23
	CMT FIT	Ver. 5.70
	LWIP FIT	Ver. 1.00
Protocols	Internet layer	IPv4, ICMP, IGMP, ARP
	Transport layer	TCP, UDP
	Application layer	DHCP

5.2 Sample Projects

The sample projects listed in Table 5.2 are bundled with this FIT module. Download the package for the FIT module from the following URL to get the sample projects.

https://www.renesas.com/en/search?keywords=R20AN0788

Table 5.2 Sample Projects*1

Supported Protocols	Project Name	Applicable Kit (and Device)
TCP client ICMP (ping) reception	rx65n_ck_v2_lwip_driver_tcp_client	CK-RX65N v2
	rx72n_evk_lwip_driver_tcp_client	Envision Kit for the RX72N
	rx72m_rsk_lwip_driver_tcp_client	Renesas Starter Kit+ (RSK+) for the RX72M (for use with multiple (2-channel) Ethernet interfaces)*2
UDP client ICMP (ping) reception	rx65n_ck_v2_lwip_driver_udp_client	CK-RX65N v2
	rx72n_evk_lwip_driver_udp_client	Envision Kit for the RX72N
	rx72m_rsk_lwip_driver_udp_client	Renesas Starter Kit+ (RSK+) for the RX72M (for use with multiple (2-channel) Ethernet interfaces)*2

*1: The compiler used was the C/C++ Compiler Package for RX Family (CC-RX) from Renesas Electronics.

*2: Even when multiple network interfaces are to be used (multi-interface usage), the API functions to be used only handle processing for the first channel.
The lwIP FIT module (lwIP) does support multiple interfaces. This FIT module, however, does not provide the API functions for processing to initialize and close the second ETHERC channel. Therefore, users need to implement the functions for handling these types of processing as described in sections 3.2 and 3.5. Refer to the sample code in the RSK+ for the RX72M (for use with multiple (2-channel) Ethernet interfaces), which is provided as part of this FIT module, to implement the functions.

5.3 Support for Multi-Interface Usage

The lwIP FIT module (lwIP) can use multiple network interfaces (multi-interface usage). Some RX family MCUs, such as the RX72M, have two ETHERC and Ethernet-PHY management channels, so they allow the use of two Ethernet channels by the lwIP.

This FIT module, however, does not include processing to allow use of the second channel as a network interface, which is required when the lwIP is to use multiple interfaces. Therefore, implement the required processing by referring to the guide to implementation given in section 5.3.1 and the demo project for multiple interfaces for use with the RSK+ for the RX72M (see section 5.2).

5.3.1 Implementing Processing Required for Multiple Interfaces

This section describes how to implement the processing that is required for the lwIP FIT module (lwIP) to use multiple interfaces. Copy and rename some API and internal functions for use with this FIT module as a basis for creating the API and internal functions for use with the second network interface channel.

In the examples in this section, the first and second channels for use as the network interfaces are referred to as channels 0 and 1, respectively (see section 5.3.1.1).

Points to note:

1. Implement the macro definition and functions to be added in the user application in the ways described in the guide in this section. This is because an attempt to implement them in the source code for this FIT module itself may lead to the code which the user has added to the implementation contents being overwritten by the source code of this FIT module package due to automatic code generation by the Smart Configurator, resulting in deletion of the additions.
2. When two channels are to be used as the network interfaces, number them 0 and 1. In the case where the number of the first channel is 0, that of the second channel is 1.
   Up to two channels can be used as Ethernet interfaces. The number (one or two) depends on the specifications of the Ethernet FIT module.

5.3.1.1 Defining the Channel Number Macro for the Network Interface for Use through the Second Channel

- Define an ETHER_CH_2 macro in the user application.
- The second channel for use as a network interface has the number 1 in this example.

```
#define ETHER_CH_2      (1)
```

5.3.1.2 Creating the r_lwip_driver_ethernetif_init_2() Function

- Copy the r_lwip_driver_ethernetif_init() function and rename the copied function. The name of the function after renaming is r_lwip_driver_ethernetif_init_2 in this example.
- Change the setting in netif->name[1] to “(ETHER_CH_2 + '0')” (line 13 in the listing below).

```
err_t r_lwip_driver_ethernetif_init_2(struct netif *netif)
{
   err_t err = ERR_OK;

   LWIP_ASSERT("netif != NULL", (netif != NULL));

  #if LWIP_NETIF_HOSTNAME
   /* Initialize interface hostname */
   netif->hostname = "LWIP_HOST_NAME";
  #endif /* LWIP_NETIF_HOSTNAME */

   netif->name[0] = 'c';
   netif->name[1] = (char) (ETHER_CH_2 + '0');
   /* We directly use etharp_output() here to save a function call.
    * You can instead declare your own function an call etharp_output()
    * from it if you have to do some checks before sending (e.g. if link
    * is available...) */
   netif->output     = etharp_output;
   netif->linkoutput = r_lwip_driver_low_level_output;

   /* initialize the hardware */
   err = r_lwip_driver_low_level_init(netif);

   return err;
}
```

5.3.1.3 Creating the r_lwip_driver_low_level_init_2() Function

- Copy the r_lwip_driver_low_level_init() function and rename the copied function. The name of the function after renaming is r_lwip_driver_low_level_init_2 in this example.
- Change the channel number to be set as the eth_ch variable to “ETHER_CH_2” (line 5 in the listing below).
- Change the MAC address settings in netif->hwaddr to those for the second channel (lines 11 to 16).
- Comment out the settings of the eth_param.ether_callback.pcb_func and eth_param.ether_int_hnd.pcb_int_hnd callback functions for use with the Ethernet FIT module (lines 25 to 42).
  As this operation proceeds within the processing for the first channel by the r_lwip_driver_low_level_init() function, specific operations for the second channel are not required.

```
static err_t r_lwip_driver_low_level_init_2(struct netif *netif)
{
    ether_param_t eth_param = { 0 };
    ether_return_t ether_ret;
    uint32_t eth_ch = ETHER_CH_2;

    /* Clear the link status flags */
    g_link_change[eth_ch] =  LWIP_DRIVER_LINK_CHANGE_FLAG_OFF;
    g_link_status[eth_ch] =  LWIP_DRIVER_LINK_STATUS_FLAG_OFF;

    /* set MAC hardware address length */
    netif->hwaddr_len = ETH_HWADDR_LEN;

    /* set MAC hardware address */
    netif->hwaddr[0] = XXXX_ADDR0;
    netif->hwaddr[1] = XXXX_ADDR1;
    netif->hwaddr[2] = XXXX_ADDR2;
    netif->hwaddr[3] = XXXX_ADDR3;
    netif->hwaddr[4] = XXXX_ADDR4;
    netif->hwaddr[5] = XXXX_ADDR5;

    /* maximum transfer unit */
    netif->mtu = LWIP_DRIVER_CFG_MTU;

    /* device capabilities */
    /* don't set NETIF_FLAG_ETHARP if this device is not an ethernet one */
    netif->flags |= NETIF_FLAG_BROADCAST | NETIF_FLAG_ETHARP;

//    eth_param.ether_callback.pcb_func = r_lwip_driver_rx_ether_cb;
//
//    ether_ret = R_ETHER_Control(CONTROL_SET_CALLBACK, eth_param);
//
//    if (ETHER_SUCCESS != ether_ret)
//    {
//        return ERR_IF;
//    }
//
//    /* Set callback function called by EINT status interrupts. */
//    eth_param.ether_int_hnd.pcb_int_hnd = r_lwip_driver_rx_ether_int_cb;
//
//    ether_ret = R_ETHER_Control(CONTROL_SET_INT_HANDLER, eth_param);
//
//    if (ETHER_SUCCESS != ether_ret)
//    {
//        return ERR_IF;
//    }

    eth_param.channel = eth_ch;
```

```
    R_ETHER_Control(CONTROL_POWER_ON, eth_param);
    --- Omitted ---
    return ERR_OK;
}
```

6. Troubleshooting

6.1 Troubleshooting

(1) Q: I added this FIT module to my project, but when I attempted to build it, an error message, “Could not open source file ‘platform.h’,” appeared.

A: The FIT module may not have been correctly added to your project. Refer to the appropriate document of the two listed below to confirm how to add the FIT module to a project.

- When CS+ is in use
  Application note *RX Family: Adding Firmware Integration Technology Modules to CS+ Projects* (R01AN1826)
- When the e^2 studio is in use
  Application note *RX Family: Adding Firmware Integration Technology Modules to Projects* (R01AN1723)

When this FIT module is to be used, the board support package FIT module (BSP module) must also be added to the project. For details on how to add the BSP module, refer to the application note *RX Family: Board Support Package Module Using Firmware Integration Technology* (R01AN1685).

(2) Q: I added this FIT module to my project, but when I attempted to build it, an error message indicating that the configuration settings were incorrect appeared.

A: The settings in the r_lwip_driver_rx_config.h configuration header file may have been incorrect. Check the file and set correct values. See section 2.7, Settings for Compilation for details.

(3) Q: When I attempt to transmit large amounts of data, the function registered with the linkoutput member of the netif structure (netif->linkoutput) may return ERR_IF, an lwIP error, as the return value (see sections 3.7 and 5.3.1.2).

A: This is probably because the amount of data for transmission is too large for the transmission buffer to be allocated within the specified retry period, resulting in an error. In such cases, adjusting the following configuration settings to have greater values may rectify the error. Try again after using the Smart Configurator to set greater values for the following configuration settings for the Ethernet FIT module and this FIT module.

- Ethernet FIT module
  - ETHER_CFG_EMAC_TX_DESCRIPTORS
- lwIP driver FIT module (this FIT module)
  - LWIP_DRIVER_CFG_SEND_MAX_LOOP
  - LWIP_DRIVER_CFG_SEND_DELAY_US

Note that setting a greater value for the LWIP_DRIVER_CFG_SEND_DELAY_US macro may lower the communications throughput. Therefore, we recommend setting it to the smallest possible value of those in the range that provide a solution to the error.

(4) Q: I edited the files in the smc_gen/r_[driver name]_rx folder for the project, but when I attempted to build the project, the results of my editing were deleted (the edited files were returned to their states before editing).

A: With the default setting for the Smart Configurator, automatic code generation (deployment of the FIT module package and execution of the configurations for the configuration header file in the smc_gen/r_config folder) proceeds in response to a specific event such as building. Accordingly, you can consider the automatic code generation to have overwritten the results of your editing, thus deleting them.
The information displayed in the [Help] window by following the procedure below can be used to suppress automatic code generation by the Smart Configurator.

Select [Help Contents] from the [Help] menu bar for the e^2 studio. From [Contents] in the opened [Help] window, refer to [e^2 studio User Guide] > [Building Projects] > [Smart Configurator] > [Code Generation].

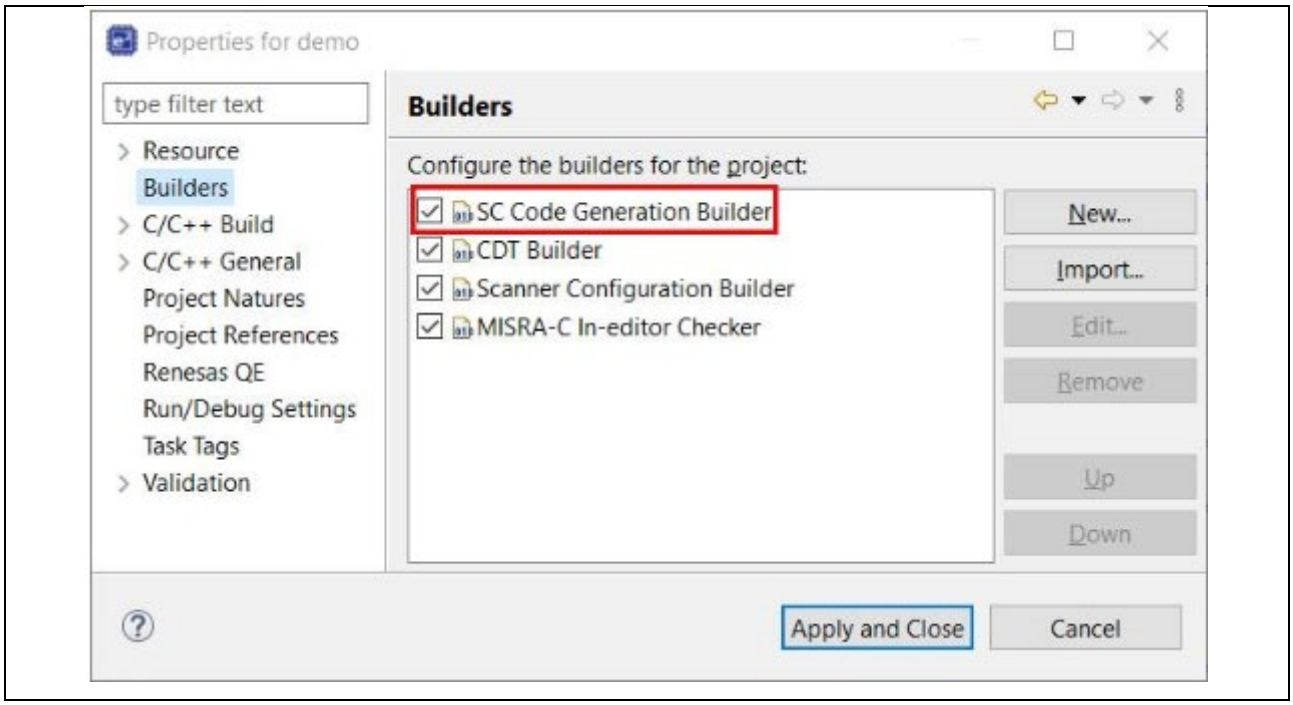

Figure 2 Setting to Stop Automatic Code Generation by the Smart Configurator

(5) Q: When lwip_init() function of lwIP is called, an infinite loop occurs in the R_TSIP_GenerateRandomNumber() function of the TSIP FIT module that is called during the process.

A: This may be because R_TSIP_Open() function of the TSIP FIT module has not been executed. Call the R_TSIP_Open() function from the user application before calling R_LWIP_DRIVER_Open() function of this FIT module.

7. Reference Documents

User's Manual: Hardware
(The latest version can be downloaded from the Renesas Electronics Web site.)

Technical Updates/Technical News
(The latest information can be downloaded from the Renesas Electronics Web site.)

User's Manual: Development Environment
RX Family CC-RX Compiler User's Manual (R20UT3248)
(The latest version can be downloaded from the Renesas Electronics Web site.)

GNU-RX Compiler Manual
(The latest version can be downloaded from the following Web site.)
https://llvm-gcc-renesas.com/ja/gnu-tools-manuals/gnu-compiler/

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jun.20.25	-	First edition issued.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

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

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

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

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

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

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

Contact information

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

Trademarks

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