RX62N Group/RX63N Group

Sample Program Using Embedded TCP/IP M3S-T4-Tiny

Introduction

This document explains TCP/IP and driver that operated on RX62N and RX63N. This documents name is "Introduction Guide".

The TCP/IP protocol stack for embedded system is Tiny TCP/IP library “RX Family TCP/IP for Embedded system M3S-T4-Tiny” (which called The T4). T4 is provided as library format and user can develop own system with this library to use TCP/IP function.

The driver program for using T4 include cmt driver, system timer, ethernet driver, t4 driver and common driver.

We prepared sample programs for each CPU board included in the Renesas Starter Kit+. This sample program shows how to setup CPU board, PC settings, network connections to confirm correct sample program behavior.

Please refer to the following URL to know the latest information about T4.

https://www.renesas.com/mw/t4
echo server sample:R20AN0051

T4 is provided as Firmware Integration Technology (FIT) Module. Please refer to the URL to know FIT outline.
https://www.renesas.com/us/en/products/software-tools/software-os-middleware-driver/softwarepackage/fit.html

Target Device

RX62N Group

RX63N Group

Target Compilers

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

For details of the confirmed operation contents of each compiler, refer to "2 Operation Confirmation Conditions".

Contents

1. Overview

1.1 Overview

The sample program provides TCP/IP protocol, device driver (common driver, CMT driver, Ethernet driver, System timer and T4 driver) and sample for RX62N group, RX63N group.

2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2-1 Operation Confirmation Conditions for RX62N

Item	**Contents**
MCU Used	R5F562N8BDBG (RX62N Group)
Operating frequencies	- Main clock: 12 MHz - PLL: 12 MHz (main clock) - System clock (ICLK): 96 MHz (Main clock multiply by 8) - Peripheral module clock (PCLK): 48 MHz (Main clock multiply by 4) - External bus clock (BCLK): 24 MHz (Main clock multiply by 2)
Operating voltage	3.3V
Integrated development environment	Renesas Electronics e^2 studio Version 7.6.0 IAR Embedded Workbench for Renesas RX 4.12.1
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 4.8.4.201902 Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.12.1 Compiler option: The default settings of the integrated development environment.
iodefine.h version	Version 1.4
Endian	little endian/big endian
Operating mode	Single-chip mode
Processer mode	Supervisor mode
Sample program version	Version 1.00
Board used	Renesas Starter Kit+ for RX62N (R0K5562Nxxxxxxx)

Table 2-2 Operation Confirmation Conditions for RX63N

Item	Contents
MCU Used	R5F563NFDDFC (RX63N Group)
Operating frequencies	- Main clock: 12 MHz - PLL: 192 MHz (main clock divided by 1 and multiplied by 16) - System clock (ICLK): 96 MHz (PLL divided by 2) - Peripheral module clock A (PCLKA): 96 MHz (PLL divided by 2) - Peripheral module clock B (PCLKB): 48 MHz (PLL divided by 4) - External bus clock (BCLK): 48 MHz (PLL divided by 4) - FlashIF clock (FCLK): 48 MHz (PLL divided by 4) - IEBUS clock (IECLK): 48 MHz (PLL divided by 4)
Operating voltage	3.3V
Integrated development environment	Renesas Electronics e^2 studio Version 7.6.0 IAR Embedded Workbench for Renesas RX 4.12.1
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 4.8.4.201902 Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.12.1 Compiler option: The default settings of the integrated development environment.
iodefine.h version	Version 1.8A
Endian	Big endian/little endian
Operating mode	Single-chip mode
Processer mode	Supervisor mode
Sample program version	Version 1.00
Board used	Renesas Starter Kit+ for RX63N (R0K50563Nxxxxxx)

3. Related Documents

RX Family TCP/IP for Embedded system M3S-T4-Tiny Introduction Guide Firmware Integration Technology (R20AN0051)

4. Hardware Configuration

4.1 Pin setting example for using RSK+RX62N/RSK+RX63N

Table 4-1 shows pin setting example for using RSK+RX62N/RSK+RX63N. Figure 4-1 and Figure 4-2 show examples of connection to the PHY-LSI.

Table 4-1 Pin setting example for using RSK+RX62N/RSK+RX63N

Case of Using MII Mode	Case of Using RMII Mode[*3]	I/O Port
ET_TX_CLK		PC4
ET_RX_CLK	REF50CK	P76
ET_TX_EN	RMII_TXD_EN	P80
ET_ETXD3		PC6
ET_ETXD2		PC5
ET_ETXD1	RMII_TXD1	P82
ET_ETXD0	RMII_TXD0	P81
ET_TX_ER		PC3
ET_RX_DV		PC2
ET_ERXD3		PC0
ET_ERXD2		PC1
ET_ERXD1	RMII_RXD1	P74
ET_ERXD0	RMII_RXD0	P75
ET_RX_ER	RMII_RX_ER	P77
ET_CRS	RMII_CRS_DV	P83
ET_COL		PC7
ET_MDC		P72
ET_MDIO		P71
ET_LINKSTA		P54 *1
ET_EXOUT		- *2
ET_WOL		- *2

Notes: 1. Setting is not required if the setting of #define ETHER_CFG_USE_LINKSTA is 0.

Notes: 2. Setting is not required because these pin are not used in Ethernet driver module.

Notes: 3. RMII mode can only be used with RSK + RX63N.

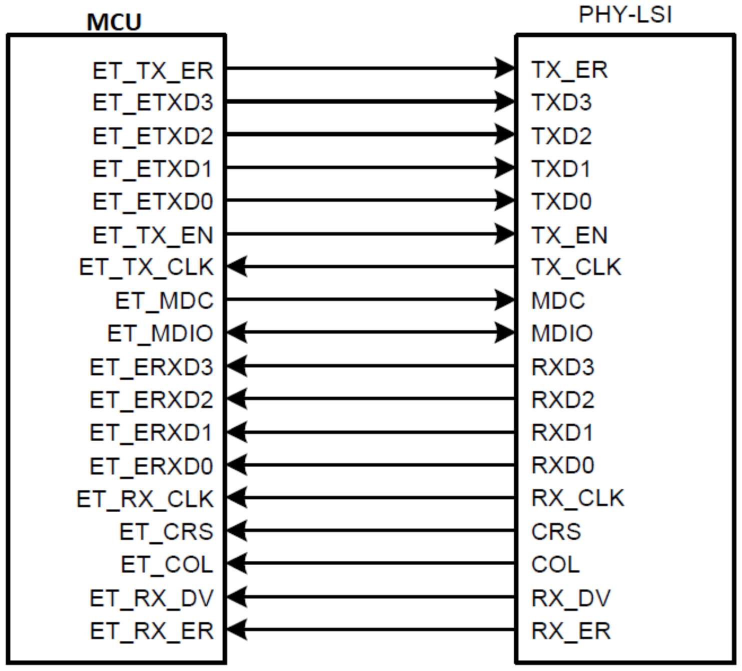


Figure 4-1 Example of connection between RX62N, RX63N and PHY(MII)

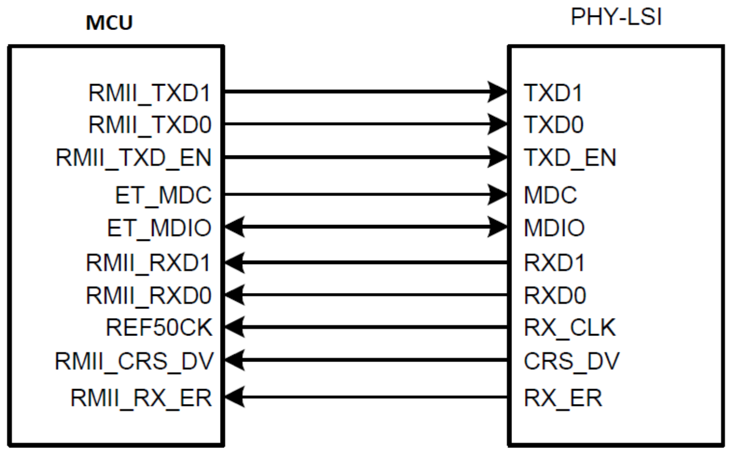


Figure 4-2 Example of connection between RX62N, RX63N and PHY(RMII)

5. Software

5.1 Sample Program Structure

This sample program includes following files.

Table 5-1 Sample Program Structure

File/Directory name			detail
r01an5161xx0100-rx62nrx63n-t4-connectivity			
	workspace		
		r01an5161_rx62n_t4_connectivity_ccrx	CCRX sample project for RX62N
		r01an5161_rx63n_t4_connectivity_ccrx	CCRX sample project for RX63N
		r01an5161_rx62n_t4_connectivity_gcc	GCC sample project for RX62N
		r01an5161_rx63n_t4_connectivity_gcc	GCC sample project for RX63N
		r01an5161_rx62n_t4_connectivity_iar	IAR sample project for RX62N
		r01an5161_rx63n_t4_connectivity_iar	IAR sample project for RX63N
	r01an5161jj0100-rx62nrx63n-t4-connectivity.pdf		Introduction Guide(Japanese)
	r01an5161ej0100-rx62nrx63n-t4-connectivity.pdf		Introduction Guide(English this document)
	readme_j.txt		readme file(Japanese)
	readme_e.txt		readme file(English)

For example, the CCRX sample project for RX63N will contain the files listed in table Table 5-2 below.

Table 5-2 Sample project for RX63N Structure

File/Directory name	detail
r01an5161_rx63n_t4_connectivity_ccrx	
doc	Document
r01an5161jj0100-rx62nrx63n-t4-connectivity.pdf	Introduction Guide(Japanese)
r01an5161ej0100-rx62nrx63n-t4-connectivity.pdf	Introduction Guide(English this document)
generate	MCU reset program
r01an5161_src	Sample program source
main.c	main function
echo_srv_sample.h	echo server sample program header file
r_pincfg	Pin setting
mcu_initial	Clock setting, non-existent port setting, stop peripheral functions that operate after reset
r_t4_rx	T4 module
tcp_blocking_sample	TCP Blocking Call
echo_srv.c	TCP blocking server echo
tcp_nonblocking_cancel_sample	TCP Non-blocking cancel Call
echo_srv.c	TCP non-blocking cancel server echo
tcp_nonblocking_sample	TCP Non-blocking Call
echo_srv.c	TCP non-blocking server echo
udp_blocking_sample	UDP Blocking Call
echo_srv.c	UDP blocking server echo
udp_nonblocking_sample	UDP Non-blocking Call
echo_srv.c	UDP non-blocking server echo
driver	Driver folder
cmt_driver	CMT driver module
common_driver	Common driver module
ether_driver	Ether driver module
systime_driver	System timer module
t4_driver	T4 driver module
r_driver_rx_config.h	The driver configuration file
HardwareDebug	Debug folder
Release	Release folder
.settings	Project settings folder
.cprojcet	Cproject file
.project	Project file
r01an5161_rx63n_t4_connectivity.x.launch	Launch file
r01an5161_rx63n_t4_connectivity.rcpc	RCPC file

The folder to which the contents of r_t4_rx is extracted will contain the files listed in table Table 5-3 below.

Table 5-3 Structure of the T4 FIT Modules

File/Directory name	detail
T4 FIT Module body (r_t4_rx)	
T4 Library (lib)	
cc-rx	Renesas CC-RX
T4_Library_ether_ccrx_rxv1_big.lib	T4 Library (RXV1 core, Big endian, for Ethernet)
T4_Library_ether_ccrx_rxv1_ether_little.lib	T4 Library (RXV1 core, Little endian, for Ethernet)
T4_Library_ether_ccrx_rxv1_ether_big_debug.lib	T4 Library includes debug information. (RXV1 core, Big endian, for Ethernet/for QE for TCP/IP)
T4_Library_ether_ccrx_rxv1_ether_little_debug.lib	T4 Library includes debug information. (RXV1 core, Little endian, for Ethernet/for QE for TCP/IP)
GCC	
libT4_Library_ether_gcc_rxv1_big.a	T4 Library (RXV1 core, Big endian, for Ethernet)
libT4_Library_ether_gcc_rxv1_little.a	T4 Library (RXV1 core, Little endian, for Ethernet)
IAR	
T4_Library_ether_iar_rxv1_big.a	T4 Library (RXV1 core, Big endian, for Ethernet)
T4_Library_ether_iar_rxv1_little.a	T4 Library (RXV1 core, Little endian, for Ethernet)
r_t4_itcpip.h	T4 Library header file
r_stdint.h	Standard data type header file
r_mw_version.h	Middleware version header file
T4 Document (doc)	
ja	
r20uw0031jj0111-t4tiny.pdf	User's Manual (Japanese)
r20uw0032jj0108-t4tiny.pdf	Ethernet Driver Interface Specification (Japanese)
r20an0051jj0209-rx-t4.pdf	T4 Introduction Guide (Japanese)
en	
r20uw0031ej0111-t4tiny.pdf	User's Manual (English)
r20uw0032ej0108-t4tiny.pdf	Ethernet Driver Interface Specification (English)
r20an0051ej0209-rx-t4.pdf	T4 Introduction Guide (English)
T4 Library make environment (make_lib)	
make_lib.zip	T4 Library make environment (includes source code)
T4 config reference (ref)	
config_tcpudp_reference.tpl	T4 Config file (template)
r_t4_rx_config_reference.h	T4 Config header(reference)
src	
config_tcpudp.c	T4 Config file
readme.txt	readme

The folder to which the contents of driver is extracted will contain the files listed in table Table 5-4 below.

Table 5-4 Structure of the Driver Modules

File/Directory name						**detail**
driver						
	cmt_driver					
		src				
			cmt_driver.c			CMT driver source file
		cmt_if.h				CMT driver interface file
	common_driver					
		src				
			drv_locking.c			Locking source file
			r_rx_compiler.h			File that unifies the function definition of each compiler
			r_rx_inrtinsic_funtions.c			A file that defines built-in functions that are not implemented by the GCC / IAR compiler
			r_rx_inrtinsic_funtions.h			File that unifies the definition of intrinsic functions of each compiler
		common_driver.h				Common driver interface file
	ether_driver					
		src				
			phy			
				phy.c		Ethernet PHY device driver
				phy.h		Ethernet PHY device driver interface file
			targets			
				rx63n		
					ether_setting_rx63n.c	Interrupt and PHY management interface setting file
			ether_driver.c			Ethernet driver source file
			ether_private.h			Ethernet internal macro, structure and function define
		ether_if.h				Ethernet driver interface
	systime_driver					
		src				
			sys_time_driver.c			System timer source file
			sys_time_private.h			System timer internal macro, structure and function define
		sys_time_if.h				System timer interface
	t4_driver					
		src				
			ether_callback.c			Callback function for Ethernet source file
			t4_driver.c			t4 driver source file
			timer.c			timer source file
			timer.h			timer interface file
	r_driver_rx_config.h					driver configuration file

5.2 Sample source

Each project files including source file below.

- TCP blocking call (tcp_blocking directory)
- TCP Non-blocking cancel call (tcp_nonblocking_cancel directory)
- TCP Non-blocking call (tcp_nonblocking directory)
- UDP blocking call (udp_blocking directory)
- UDP Non-blocking call (udp_nonblocking directory)

The sample program has a common main function. The main function calls an echo_srv() function. Five above patterns are implementations of the echo back server and please use one pattern of project.

5.2.1 Flow of the TCP Echo Back Server Function (for Blocking Call)

Flow of the Echo Back Server Function

Cf. Figure 5.2 Flow of the TCP Echo Back Server Function (for Blocking Call).

5.2.2 Flow of the TCP Echo Back Server Function (for Non-blocking Call)

Flow of the Echo Back Server Function

Cf. Figure 5.3 Flow of the TCP Echo Back Server Function (for Non-Blocking Call).

Flow of the Callback Function

Cf. Figure 5.4 Flow of the TCP Callback Function (for Non-Blocking Call).

5.2.3 Flow of the UDP Echo Back Server Function (for Blocking Call)

Flow of the Echo Back Server Function

Cf. Figure 5.5 Flow of the UDP Echo Back Server Function (for Blocking Call).

5.2.4 Flow of the UDP Echo Back Server Function (for Non-blocking Call)

Flow of the Echo Back Server Function

Cf. Figure 5.6 Flow of the UDP Echo Back Server Function (for Non-Blocking Call).

Flow of the Callback Function

Cf. Figure 5.7 Flow of the UDP Callback Function (for Non-Blocking Call)

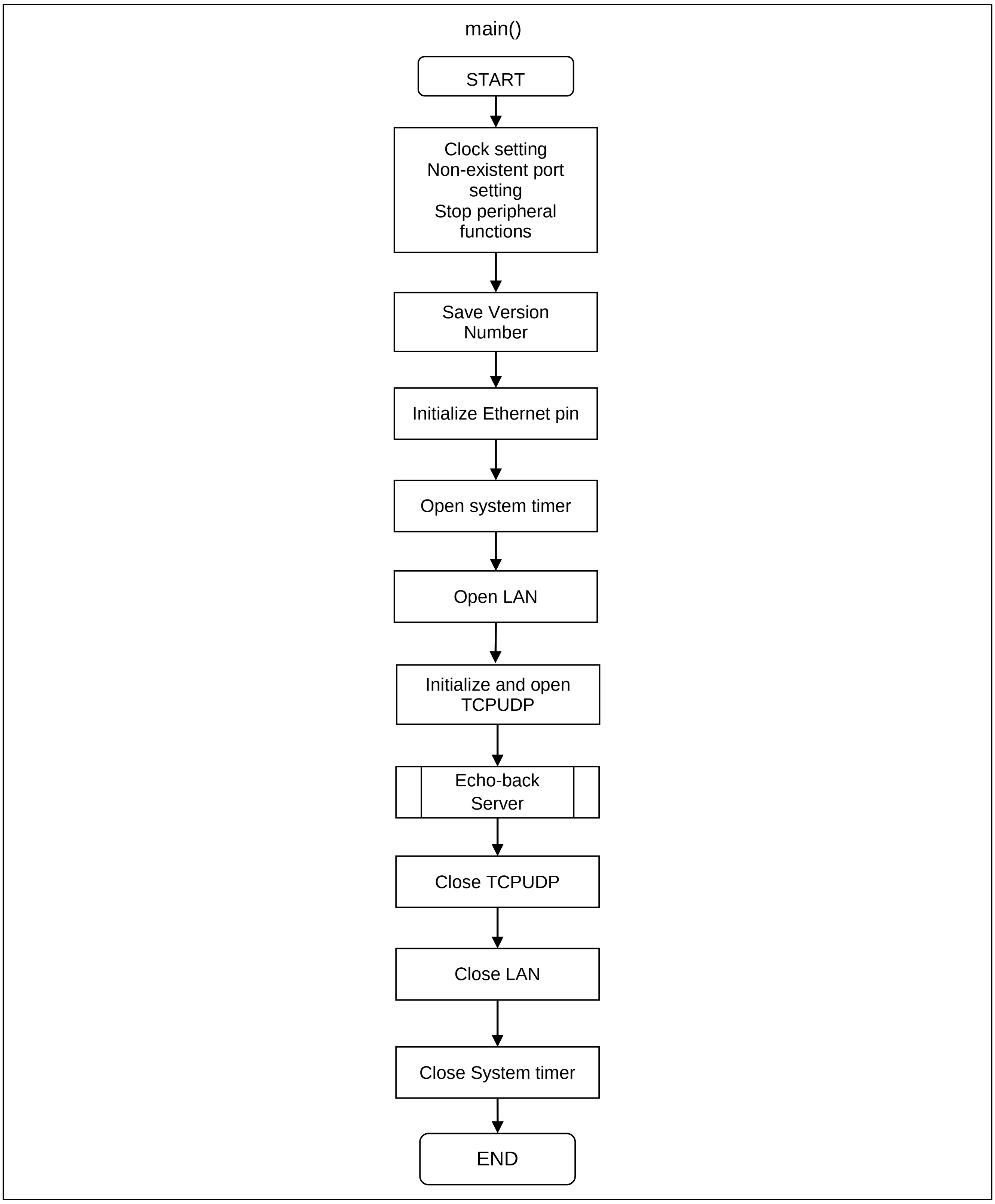


Figure 5.1 Flow of the Main Function

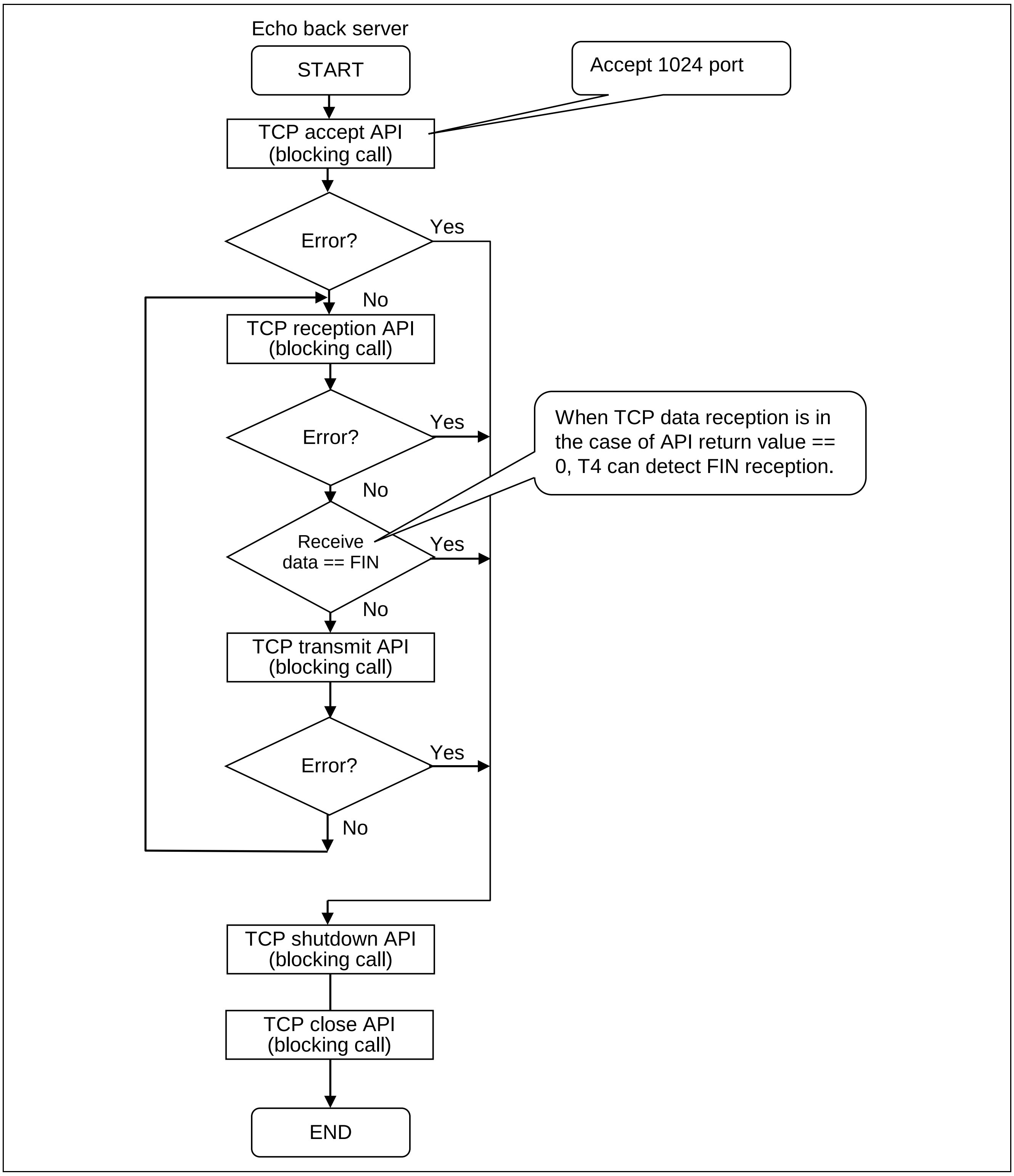

Echo back server
START
Accept 1024 port
TCP accept API (blocking call)
Error?
Yes
No
TCP reception API (blocking call)
Error?
Yes
No
When TCP data reception is in the case of API return value == 0, T4 can detect FIN reception.
Receive data == FIN
Yes
No
TCP transmit API (blocking call)
Error?
Yes
No
TCP shutdown API (blocking call)
TCP close API (blocking call)
END

Figure 5.2 Flow of the TCP Echo Back Server Function (for Blocking Call)

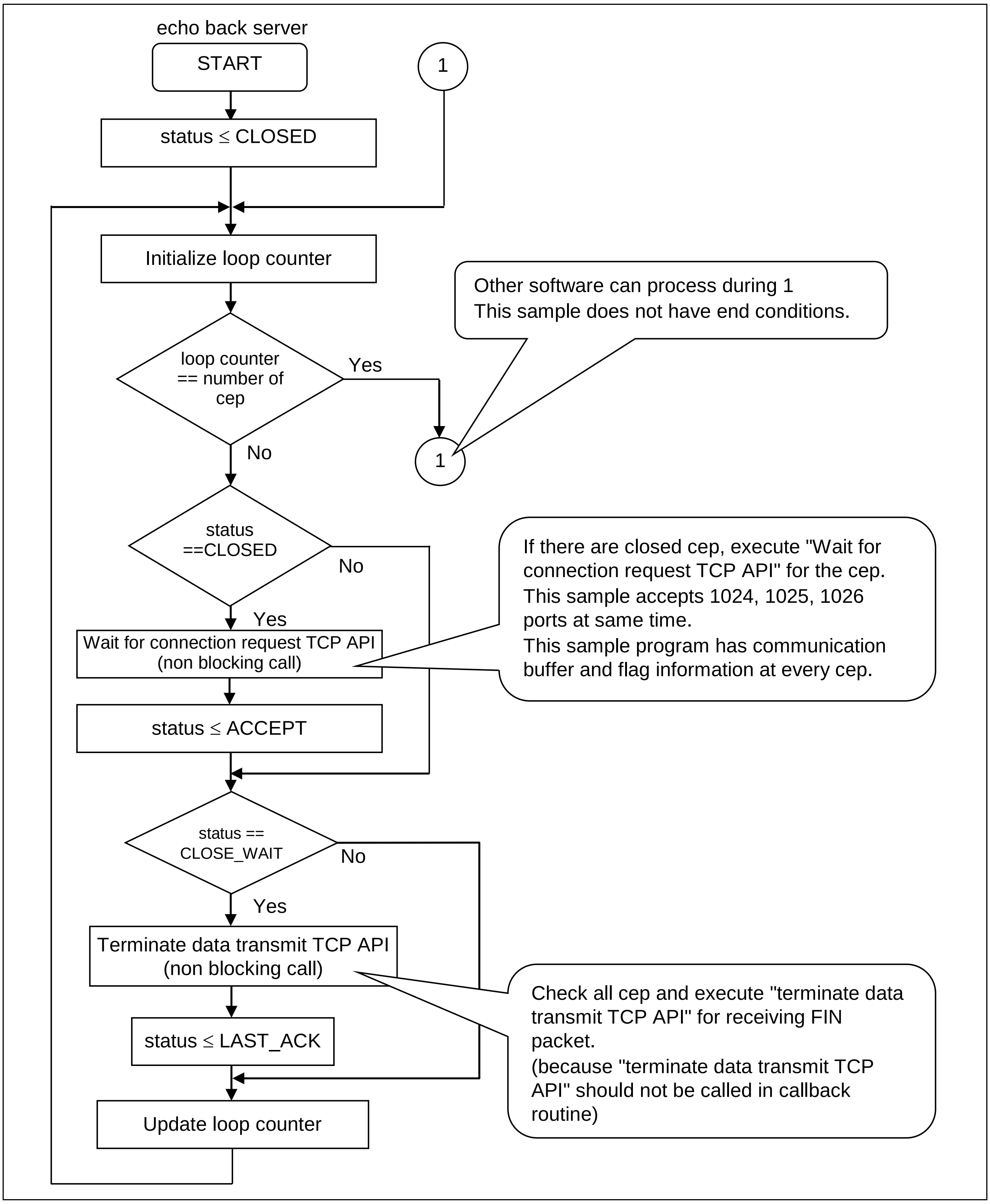

echo back server
START
1
status ≤ CLOSED
Initialize loop counter
Other software can process during 1
This sample does not have end conditions.
loop counter == number of cep
Yes
No
1
status ==CLOSED
No
Yes
If there are closed cep, execute "Wait for connection request TCP API" for the cep.
This sample accepts 1024, 1025, 1026 ports at same time.
This sample program has communication buffer and flag information at every cep.
Wait for connection request TCP API (non blocking call)
status ≤ ACCEPT
status == CLOSE_WAIT
No
Yes
Terminate data transmit TCP API (non blocking call)
Check all cep and execute "terminate data transmit TCP API" for receiving FIN packet.
(because "terminate data transmit TCP API" should not be called in callback routine)
status ≤ LAST_ACK
Update loop counter

Figure 5.3 Flow of the TCP Echo Back Server Function (for Non-Blocking Call)

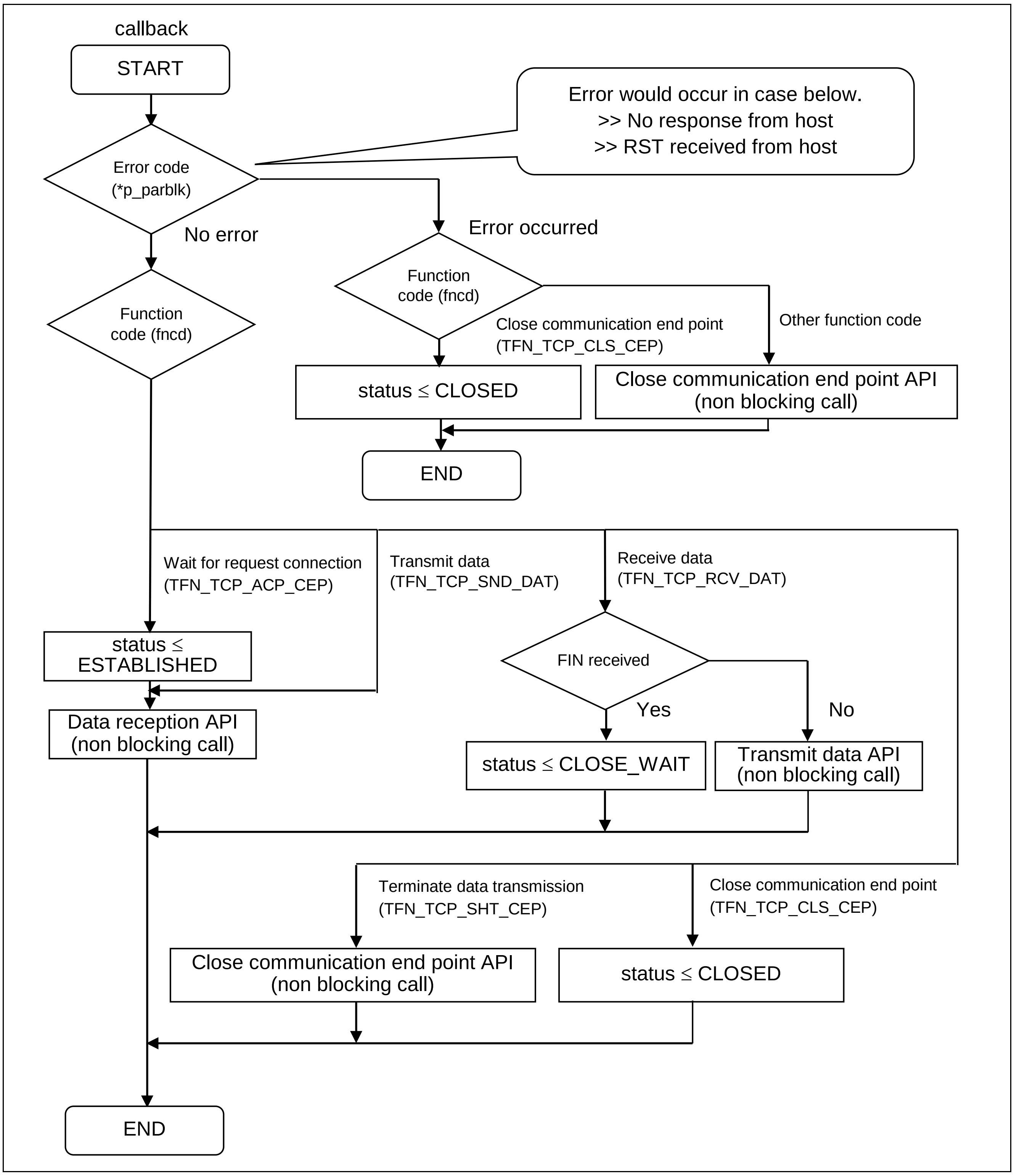

callback
START
Error would occur in case below.
>> No response from host
>> RST received from host
Error code (*p_parblk)
No error
Error occurred
Function code (fncd)
Function code (fncd)
Close communication end point (TFN_TCP_CLS_CEP)
Other function code
status ≤ CLOSED
Close communication end point API (non blocking call)
END
Wait for request connection (TFN_TCP_ACP_CEP)
Transmit data (TFN_TCP_SND_DAT)
Receive data (TFN_TCP_RCV_DAT)
status ≤ ESTABLISHED
FIN received
Yes
No
Data reception API (non blocking call)
status ≤ CLOSE_WAIT
Transmit data API (non blocking call)
Terminate data transmission (TFN_TCP_SHT_CEP)
Close communication end point (TFN_TCP_CLS_CEP)
Close communication end point API (non blocking call)
status ≤ CLOSED
END

Figure 5.4 Flow of the TCP Callback Function (for Non-Blocking Call)

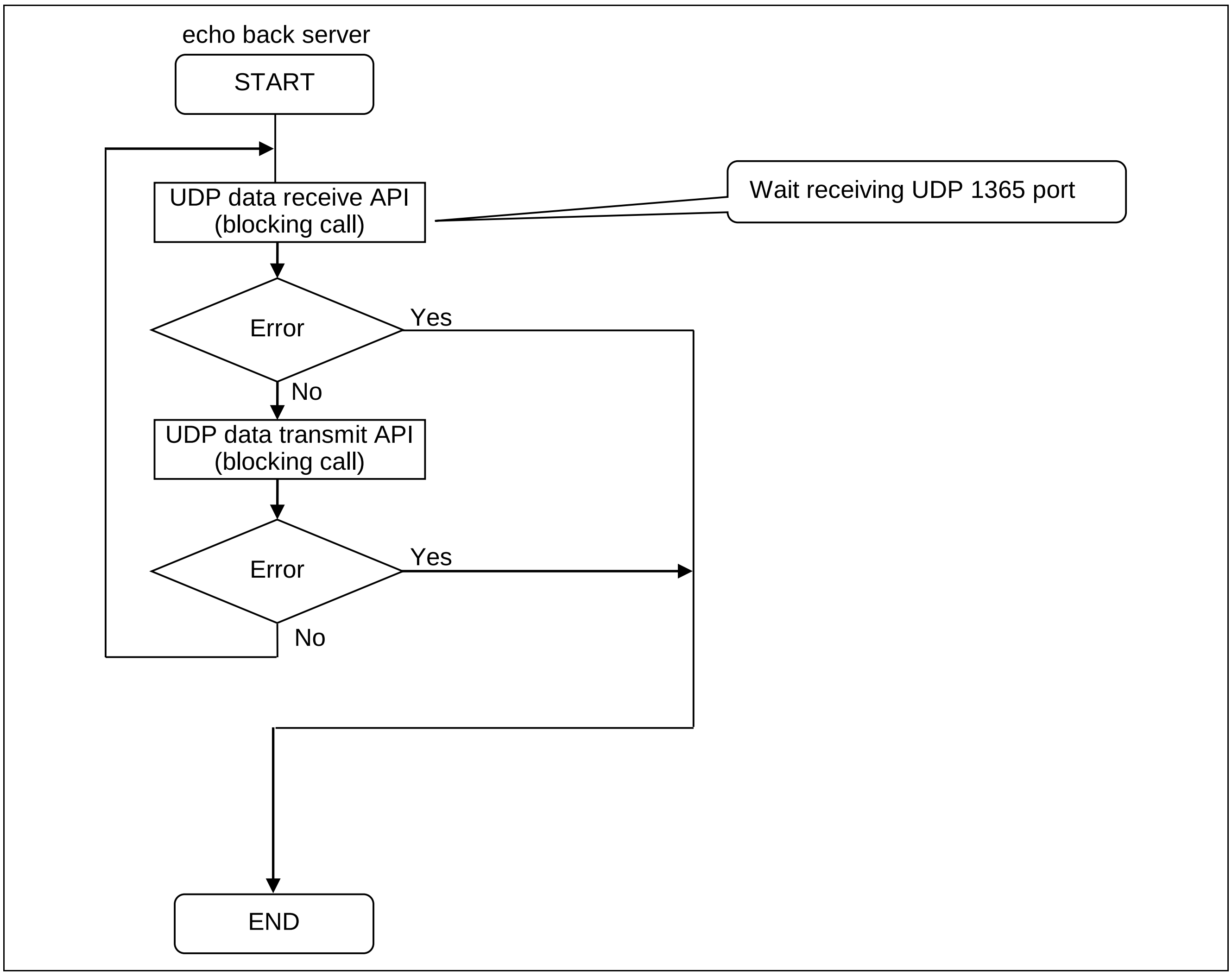

echo back server
START
UDP data receive API (blocking call)
Wait receiving UDP 1365 port
Error
Yes
No
UDP data transmit API (blocking call)
Error
Yes
No
END

Figure 5.5 Flow of the UDP Echo Back Server Function (for Blocking Call)

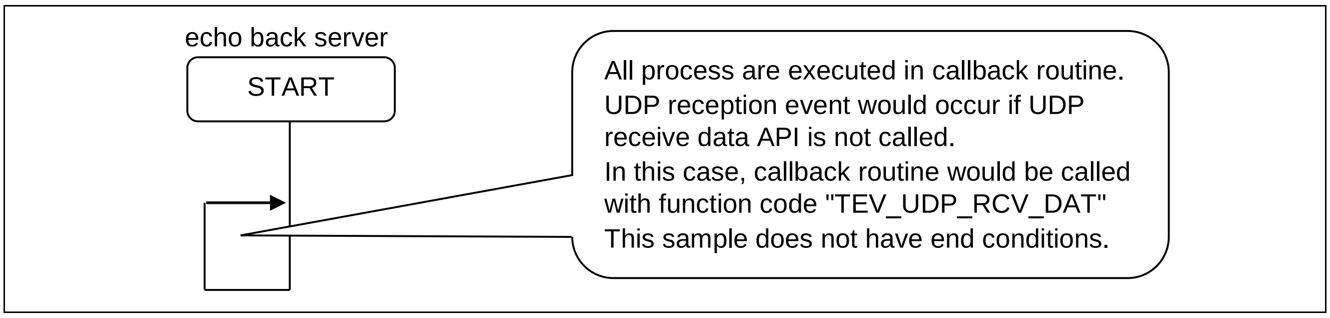

echo back server
START
All process are executed in callback routine.
UDP reception event would occur if UDP receive data API is not called.
In this case, callback routine would be called with function code "TEV_UDP_RCV_DAT"
This sample does not have end conditions.

Figure 5.6 Flow of the UDP Echo Back Server Function (for Non-Blocking Call)

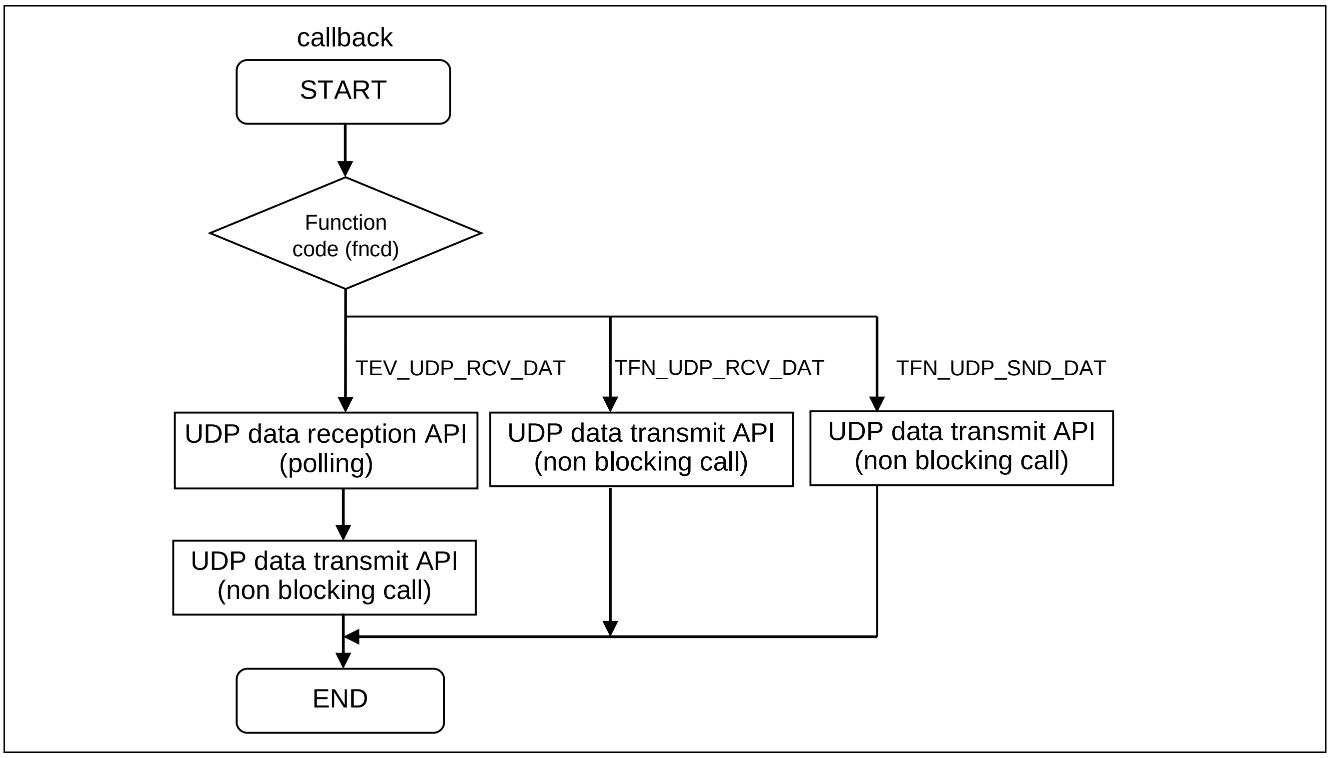

callback
START
Function code (fncd)
TEV_UDP_RCV_DAT
TFN_UDP_RCV_DAT
TFN_UDP_SND_DAT
UDP data reception API (polling)
UDP data transmit API (non blocking call)
UDP data transmit API (non blocking call)
UDP data transmit API (non blocking call)
END

Figure 5.7 Flow of the UDP Callback Function (for Non-Blocking Call)

5.3 Environment for RX62N/RX63N sample program

Sample program projects for RX62N, RX63N are provided in the package, and the projects can import to e^2 studio and IAR Embedded Workbench.

5.3.1 Software Structure

Figure 5.8 shows the configuration diagram of each driver.

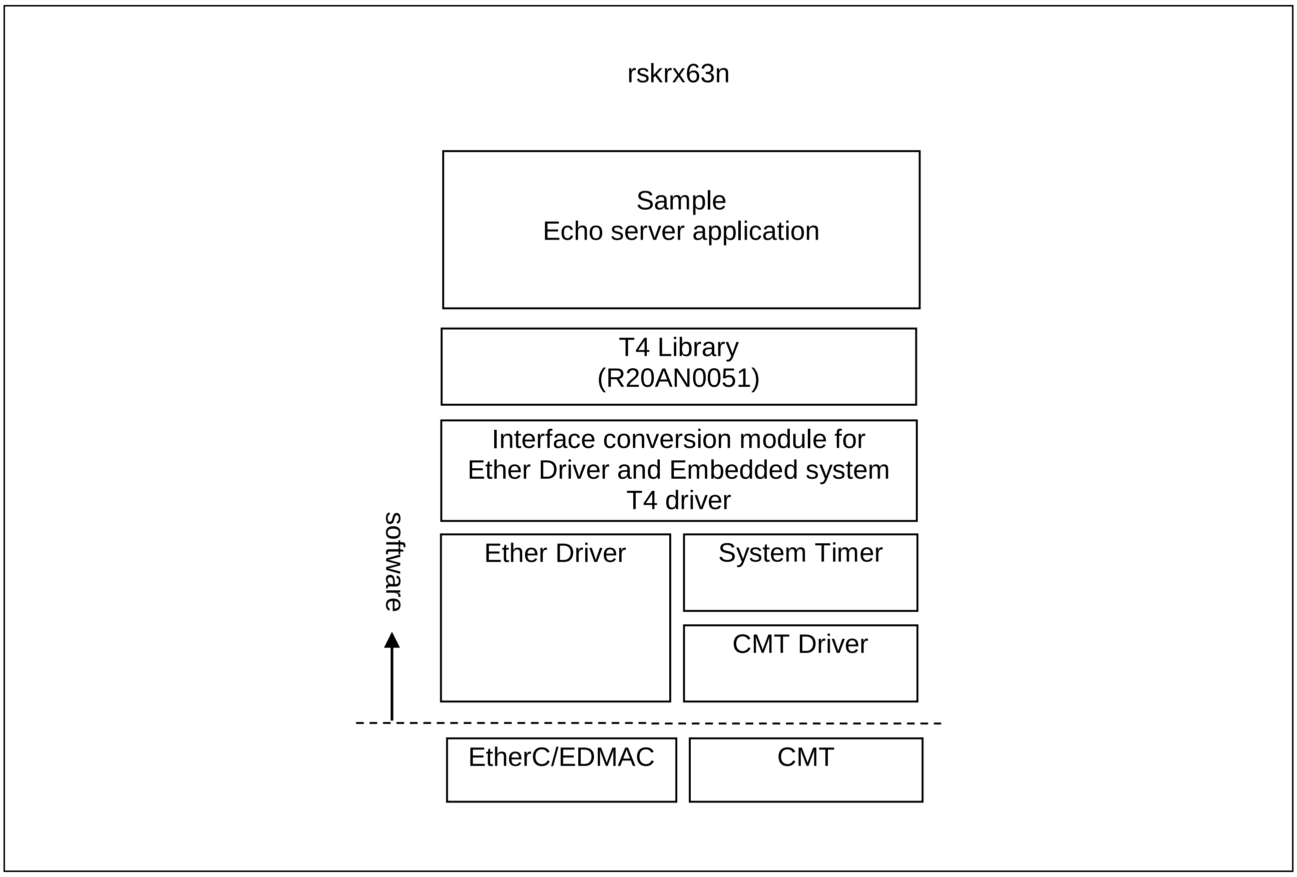


Figure 5.8 Configuration diagram of each driver

5.3.2 Method of converting e^2 studio to CS+ project

e^2 studio project can be converted to CS+ project to use *.rcpc file included in e^2 studio project. This section shows method of converting (in this example using sample program projects for RX63N).

- Start CS+ for CC, push the "GO" button in "e^2 studio / CubeSuite / …".
- Select "e^2 studio project file (*.rcpc) " and, open the *.rcpc file.
- "Project convert settings" window would open, and please select project in the project tree.
- Project settings on the right side of project tree, please select MCU "RX63N" -> "R5F563NFDxFC" and push the "OK" button. CS+ outputs the converted project.
- Please select "CC-RX" in the project tree.
- Please select "RXv2 architecture" in the "common option" tab -> "CPU" -> "command set architecture"
- echo_srv.c is registered in the each folders (project tree -> file -> src).
  -> TCP-blocking, TCP-non-blocking, UDP-blocking, UDP-nonblocking sample.
  Please remove echo_srv.c from the project tree excluding you need to work.
- Build the project
- Please set the debug tools fitting for your environment. User can confirm working sample program after this.

5.4 Confirm sample program

How to confirm Ethernet sample program

5.4.1 Environment

(1) Connecting the hardware

Setup Hardware connections

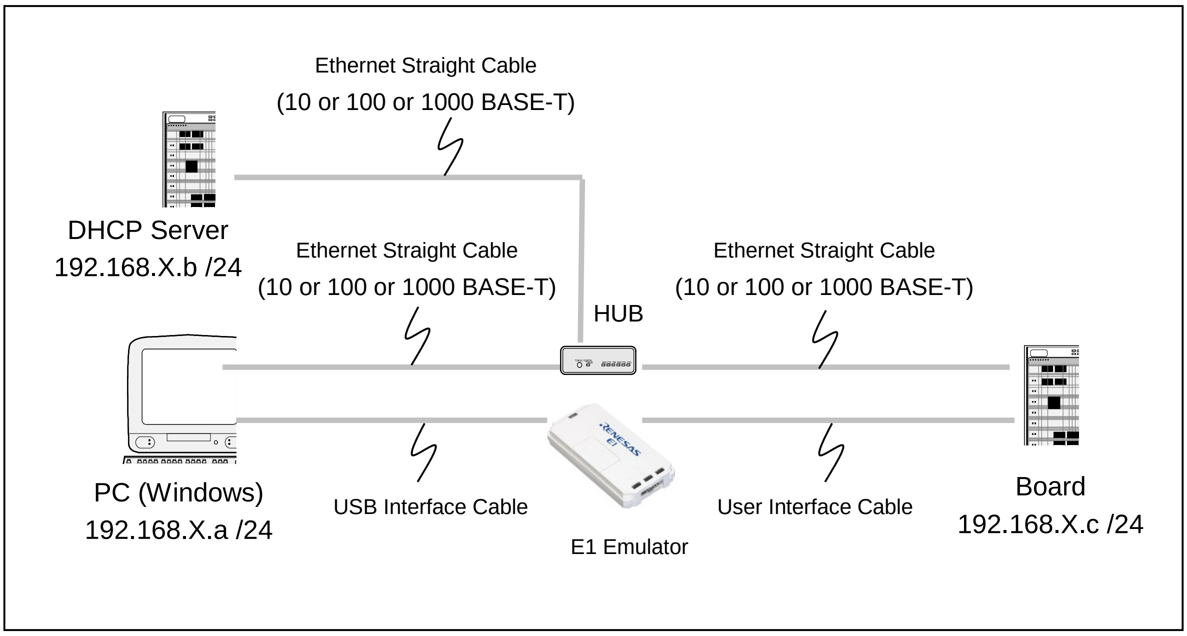


Figure 5.9 Ethernet sample program environment

We have confirmed using the Ethernet-switch product introduced in below.

- NETGEAR: GS108E

This Ethernet-switch has the function called “port mirroring function”, this function provides monitoring function for Ethernet. This Ethernet-switch can realize packet monitoring environment if user uses normally Ethernet-switch.

For example, please refer to the figure below, the board A transfers data to board B, normally Ethernet-switch filters packet and only outputs to the port connected to board B. If “port mirroring function” exists on Ethernet-switch, it copies data board B port and port mirroring port.

This function provides to monitor for peer-to-peer communication.

We recommend “Wireshark” for packet monitor. Please use “promiscuous mode” for peer-to-peer communication.

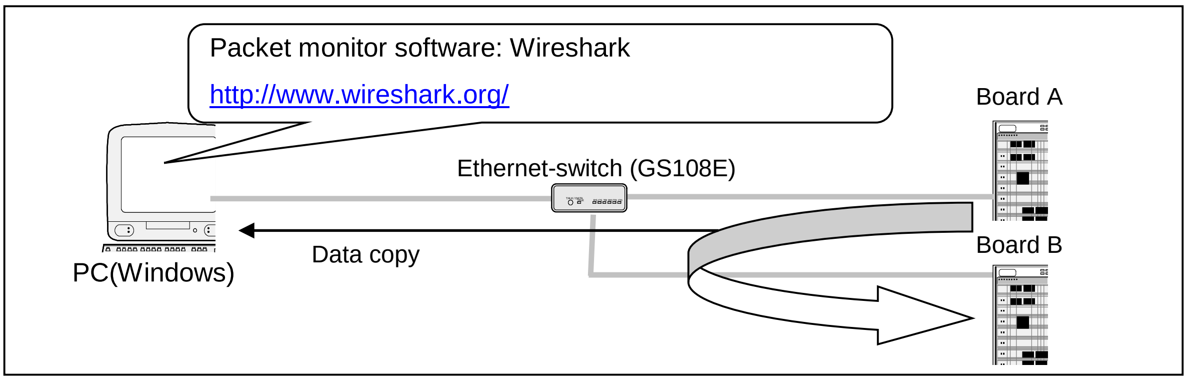


Figure 5.10 Ethernet sample program confirmation environment

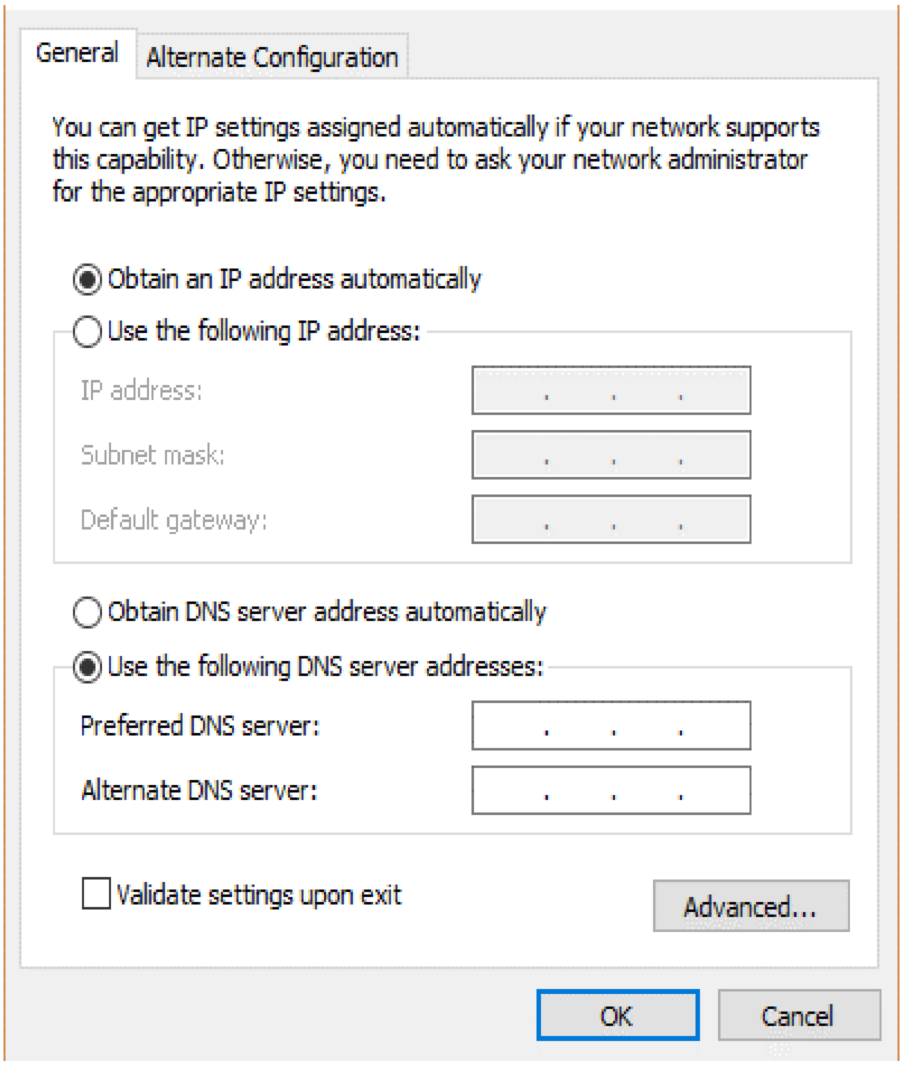
(2) PC setting

Windows 7:

Control Panel -> Network -> Adaptor setting -> Local Network Connection

Network Tab -> Internet Protocol Version 4 (TCP/IPv4) -> Property

Please select the "Auto IP address configuration" in following dialog.



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

- [✓ Obtain an IP address automatically]
- Use the following IP address:
  - IP address: . . .
  - Subnet mask: . . .
  - Default gateway: . . .
- Obtain DNS server address automatically
- [✓ Use the following DNS server addresses:]
  - Preferred DNS server: . . .
  - Alternate DNS server: . . .

☐ Validate settings upon exit

Figure 5.11 Internet Protocol Version 4 (TCP/IPv4) Property

After setting, please push OK button.

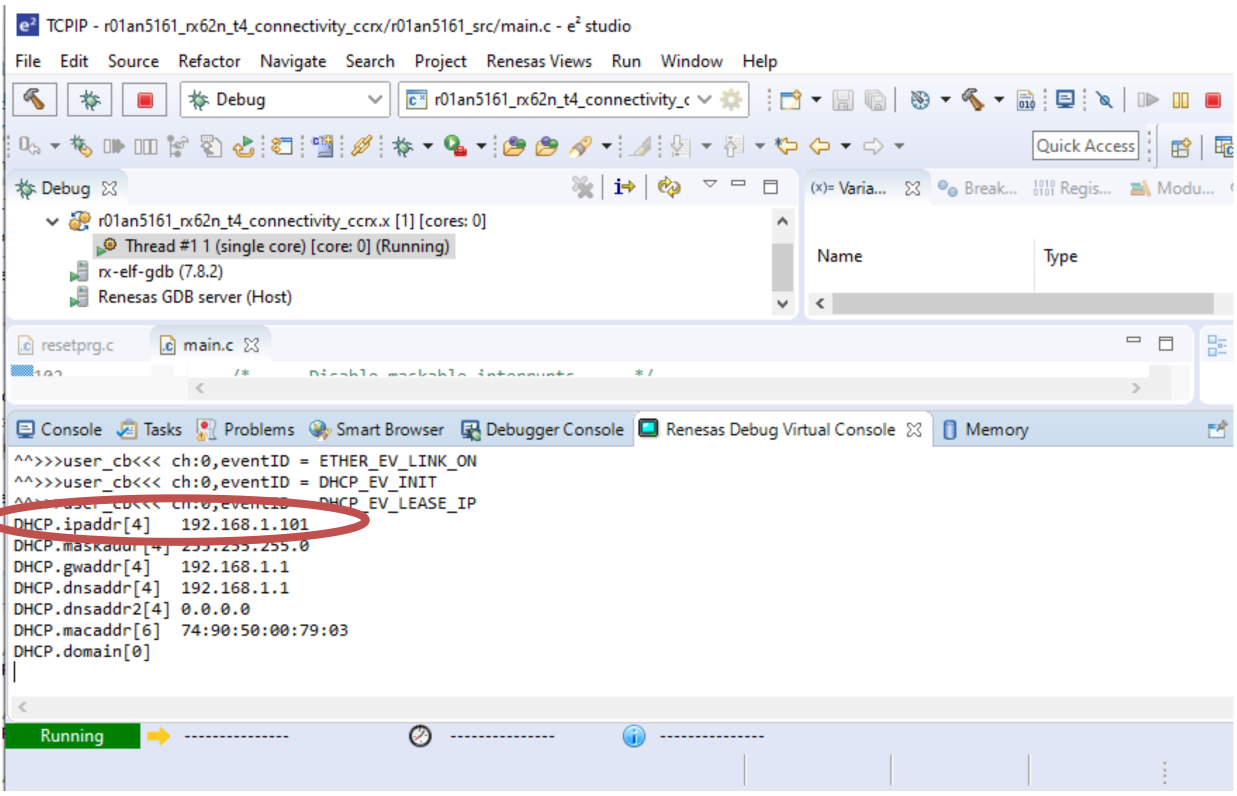
(3) Start Sample program (sample folder)

- Start e^2 studio and open the sample program.
- From the [Project] menu, click the [Build Project].
- Connect E1 Emulator, and from the [Run] menu, click the [Debug].
- The program is run by clicking button in the [Debug] view , or pressing [F8] key.

(4) Confirm IP Address for MCU

IP address will be allocated by DHCP server when sample program executes.

User can confirm the allocated IP address value using Renesas Debug Virtual Console on e^2 studio. In the GCC sample project, the IP address is not displayed on the Renesas debug virtual console. Check directly in the debug window.


TCPIP - r01an5161_rx62n_t4_connectivity_ccrx/r01an5161_src/main.c - e² studio
File Edit Source Refactor Navigate Search Project Renesas Views Run Window Help
Debug
r01an5161_rx62n_t4_connectivity_c
Quick Access
Debug
r01an5161_rx62n_t4_connectivity_ccrx.x [1] [cores: 0]
Thread #1 1 (single core) [core: 0] (Running)
rx-elf-gdb (7.8.2)
Renesas GDB server (Host)
Varia... Break... Regis... Modu...
Name
Type
resetprg.c
main.c
Console Tasks Problems Smart Browser Debugger Console Renesas Debug Virtual Console Memory
Running

```
^^>>>user_cb<<< ch:0,eventID = ETHER_EV_LINK_ON
^^>>>user_cb<<< ch:0,eventID = DHCP_EV_INIT
^^>>>user_cb<<< ch:0,eventID = DHCP_EV_LEASE_IP
DHCP.ipaddr[4]   192.168.1.101
DHCP.maskaddr[4] 255.255.255.0
DHCP.gwaddr[4]   192.168.1.1
DHCP.dnsaddr[4]  192.168.1.1
DHCP.dnsaddr2[4] 0.0.0.0
DHCP.macaddr[6]  74:90:50:00:79:03
DHCP.domain[0]
```

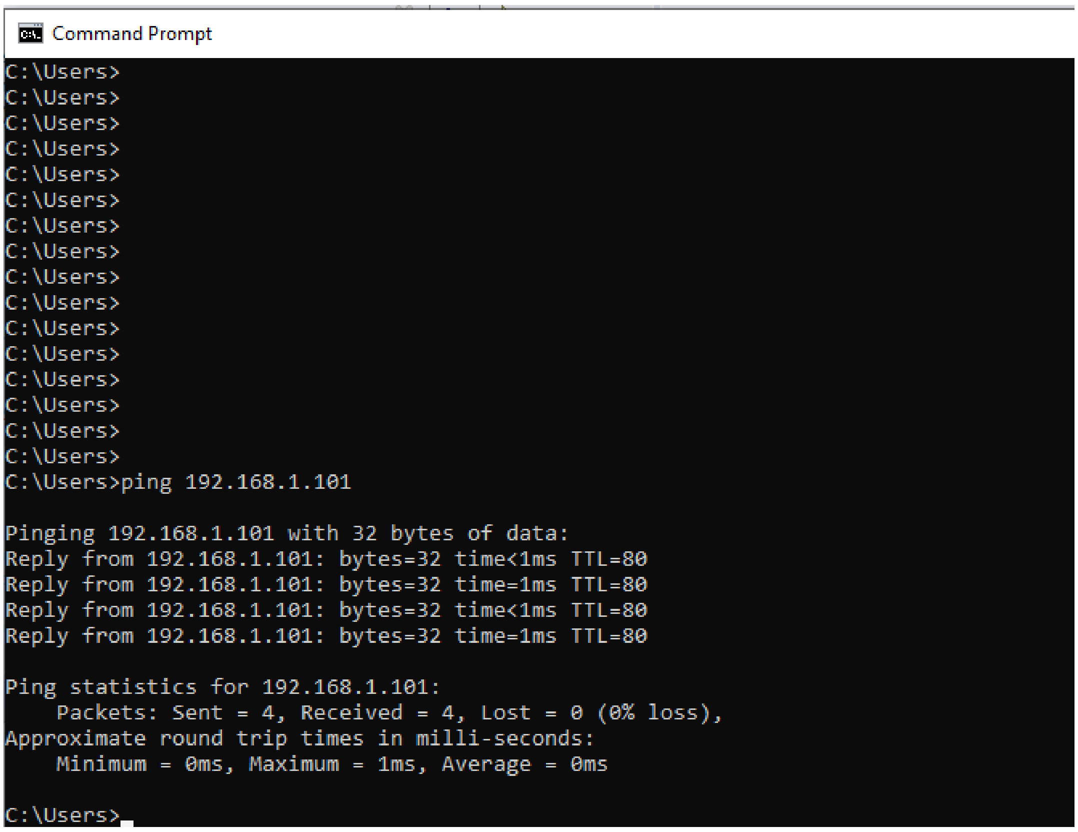
Figure 5.12 Example about Display of IP address

Figure Example about display of IP Address shows IP address is 192.168.1.101 is allocated to MCU.

Please execute ipconfig in command prompt in order to confirm PC (Windows) IP address if need.

(5) Communication inspection

Execute ping command to MCU in command prompt.



Command Prompt

```
C:\Users>
C:\Users>
C:\Users>
C:\Users>
C:\Users>
C:\Users>
C:\Users>
C:\Users>
C:\Users>
C:\Users>
C:\Users>
C:\Users>
C:\Users>
C:\Users>
C:\Users>
C:\Users>
C:\Users>ping 192.168.1.101

Pinging 192.168.1.101 with 32 bytes of data:
Reply from 192.168.1.101: bytes=32 time<1ms TTL=80
Reply from 192.168.1.101: bytes=32 time=1ms TTL=80
Reply from 192.168.1.101: bytes=32 time<1ms TTL=80
Reply from 192.168.1.101: bytes=32 time=1ms TTL=80

Ping statistics for 192.168.1.101:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Users>
```

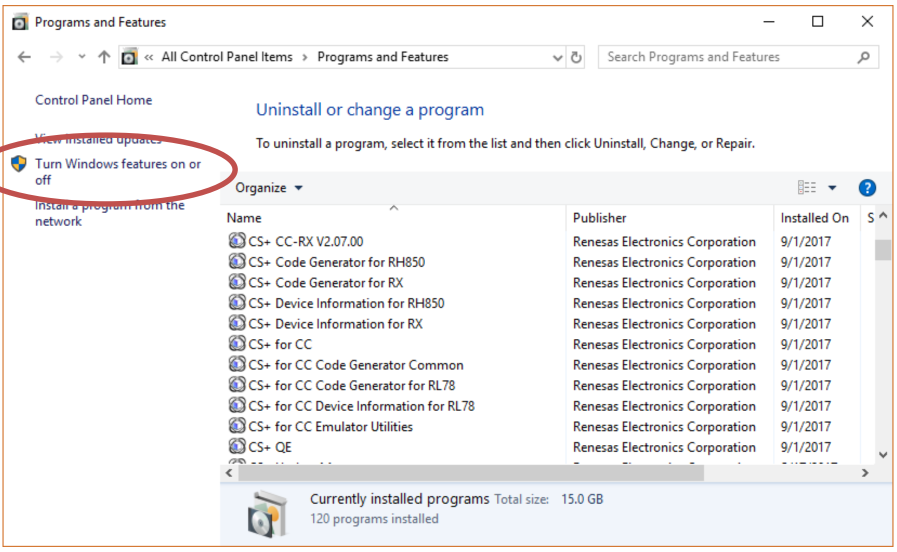
Figure 5.13 Example about Execution of ping

5.4.2 Confirm TCP connection

Execute telnet in command prompt

(1) Enable telnet(Windows7 only)

- Start -> Control Panel -> Program and Function



Programs and Features

All Control Panel Items › Programs and Features

Search Programs and Features

Control Panel Home

View installed updates

Turn Windows features on or off

Install a program from the network

Uninstall or change a program

To uninstall a program, select it from the list and then click Uninstall, Change, or Repair.

Organize ▾

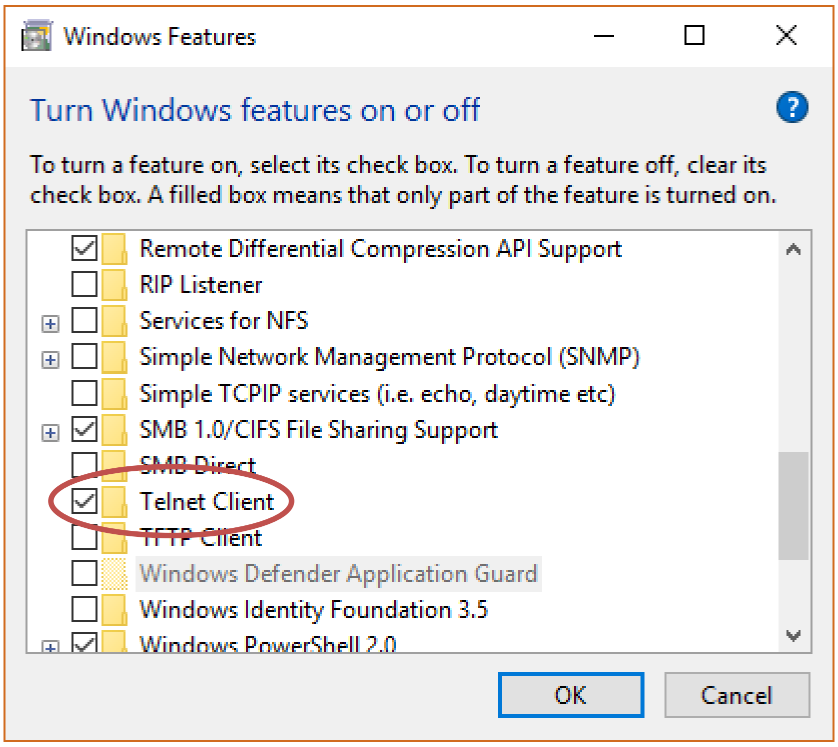
Name	Publisher	Installed On
CS+ CC-RX V2.07.00	Renesas Electronics Corporation	9/1/2017
CS+ Code Generator for RH850	Renesas Electronics Corporation	9/1/2017
CS+ Code Generator for RX	Renesas Electronics Corporation	9/1/2017
CS+ Device Information for RH850	Renesas Electronics Corporation	9/1/2017
CS+ Device Information for RX	Renesas Electronics Corporation	9/1/2017
CS+ for CC	Renesas Electronics Corporation	9/1/2017
CS+ for CC Code Generator Common	Renesas Electronics Corporation	9/1/2017
CS+ for CC Code Generator for RL78	Renesas Electronics Corporation	9/1/2017
CS+ for CC Device Information for RL78	Renesas Electronics Corporation	9/1/2017
CS+ for CC Emulator Utilities	Renesas Electronics Corporation	9/1/2017
CS+ QE	Renesas Electronics Corporation	9/1/2017

Currently installed programs Total size: 15.0 GB

120 programs installed

Figure 5.14 Program and Function

- Please check Telnet client



Windows Features

Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- [✓] Remote Differential Compression API Support
- [ ] RIP Listener
- [ ] Services for NFS
- [ ] Simple Network Management Protocol (SNMP)
- [ ] Simple TCPIP services (i.e. echo, daytime etc)
- [✓] SMB 1.0/CIFS File Sharing Support
- [ ] SMB Direct
- [✓] Telnet Client
- [ ] TFTP Client
- [ ] Windows Defender Application Guard
- [ ] Windows Identity Foundation 3.5
- [✓] Windows PowerShell 2.0

OK Cancel

Figure 5.15 Telnet Client

(2) Single LAN Port

Execute the command shown below at the MS-DOS prompt of your computer.

[TCP blocking call sample program]
telnet 192.168.X.c 1024

[TCP none-blocking call sample program (multiple communication end point can be used at same time)]
telnet 192.168.X.c 1024
telnet 192.168.X.c 1025

(3) Several LAN Port

Execute one of the following depending on the execution environment of the sample program.
To establish connections, run the following command at the MS-DOS prompt on the PC.

[TCP blocking call sample program]
telnet 192.168.X.c 1024
telnet 192.168.X.d 1025

[TCP none-blocking call sample program(multiple communication end point can be used at same time)]
telnet 192.168.X.c 1024
telnet 192.168.X.c 1025

telnet 192.168.X.d 1026
telnet 192.168.X.d 1027

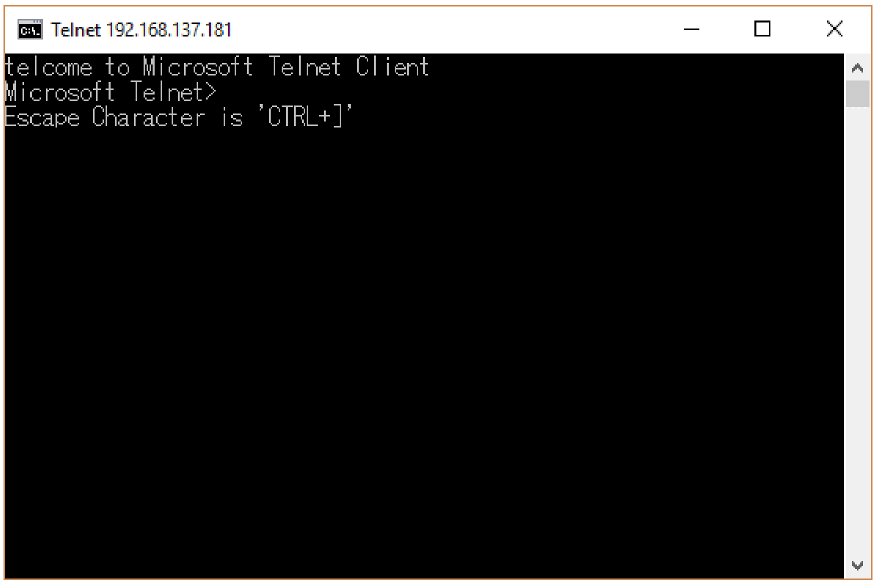
(4) End of communication

Please input “telnet 192.168.X.c 1024” in command prompt. (192.168.X.c is the IP address allocated to MCU.)

Please input any keyboard input.

It is OK to confirm the data echo-back.

Please input Ctrl + “]” and next, input “quit[enter key]” makes disconnection.



```
telcome to Microsoft Telnet Client
Microsoft Telnet>
Escape Character is 'CTRL+]'
```

Figure 5.16 Disconnect

5.4.3 Confirm UDP connection

User uses PC free tool that can generate UDP packet.

(1) Preparation of UDP software

Get free software that can send and receive UDP data at the following site.:
Socket Debugger Free https://www.udom.co.jp/sdg/
(This tool includes Japanese character)

(2) Single LAN Port

User uses PC free tool that can generate UDP packet. Setting is below.

[UDP blocking call sample program]
Destination IP address: 192.168.X.c port number 1365

[UDP none-blocking call sample program(multiple communication end point can be used at same time)]
Destination IP address: 192.168.X.c port number 1365
Destination IP address: 192.168.X.c port number 1366

(3) Several LAN Port

User uses PC free tool that can generate UDP packet. Setting is below.

[UDP blocking call sample program]
Destination IP address: 192.168.X.c port number 1365
Destination IP address: 192.168.X.d port number 1366

[UDP none-blocking call sample program(multiple communication end point can be used at same time)]
Destination IP address: 192.168.X.c port number 1367
Destination IP address: 192.168.X.c port number 1368

Destination IP address: 192.168.X.d port number 1369
Destination IP address: 192.168.X.d port number 1370

(4) End of communication

Since UDP does not establish a connection, there is no communication termination command.

5.5 Configuration Overview

The configuration option settings of this module are located in r_t4_rx_config.h and r_driver_rx_config.h. The r_t4_rx_config.h's option names and setting values are in Section 2.7 of R20AN0051EJ0209. The r_driver_rx_config.h's option names and setting values are listed in the table below:

Configuration options in r_driver_rx_config.h	
DRV_MCU_RX63_ALL Note: Default value = 1	Select whether the RX63N group is used. If not use RX63N group, please comment out this macro.
DRV_MCU_RX62_ALL Note: Default value = 1	Select whether the RX62N group is used. If not use RX62N group, please comment out this macro.
DRV_PCLKB_HZ Note: Default value = 48000000	Setting the frequency of PCLKB of RX63N For the setting range, refer to RX63N Group UMH.
DRV_PCLK_HZ Note: Default value = 48000000	Setting the frequency of PCLK of RX62N For the setting range, refer to RX62N Group UMH.
CMT_RX_CFG_IPR Note: Default value = 5	Interrupt priority level used for CMT interrupts
ETHER_CFG_MODE_SEL Note: Default value = 0	Sets the interface between ETHERC and the Ethernet PHY-LSI. If set to 0, MII (Media Independent Interface) is selected. If set to 1, RMII (Reduced Media Independent Interface) is selected.
ETHER_CFG_CH0_PHY_ADDRESS Note: Default value = 31	Specify the PHY-LSI address used by ETHERC channel 0. Specify a value between 0 and 31.
ETHER_CFG_EMAC_RX_DESCRIPTORS Note: Default value = 1	Sets the number of receive descriptors. This must be set to a value 1 or greater
ETHER_CFG_EMAC_TX_DESCRIPTORS Note: Default value = 1	Sets the number of transmit descriptors. This must be set to a value 1 or greater
ETHER_CFG_BUFSIZE Note: Default value = 1,536	Specify the size of the transmit buffer or receive buffer. The buffer is aligned with 32-byte boundaries, so specify a value that is a multiple of 32 bytes.
ETHER_CFG_EINT_INT_PRIORITY Note: Default value = 2	Sets the priority level of the EINT interrupt. This must be set to a value in the range 1 to 15.
ETHER_CFG_CH0_PHY_ACCESS Note: Default value = 0	Specify the PHY access channel used by ETHERC channel 0. When 0 is specified, ETHERC0 is used for PHY register access When 1 is specified, ETHERC1 is used for PHY register access.
ETHER_CFG_PHY_MII_WAIT Note: Default value = 8	Specify the loop count of software loop used for read or write in PHY-LSI. Set the number of loops according to the PHY-LSI to be used. Specify a value of 1 or greater.
ETHER_CFG_PHY_DELAY_RESET Note: Default value = 0x00020000	Specify the loop count used for timeout processing of PHY-LSI reset completion wait. Set the number of loops according to the PHY-LSI to be used.

ETHER_CFG_LINK_PRESENT Note: Default value = 0	Specify the polarity of the link signal output by the PHY-LSI. When 0 is specified, link-up and link-down correspond respectively to the fall and rise of the LINKSTA signal. When 1 is specified, link-up and link-down correspond respectively to the rise and fall of the LINKSTA signal.
ETHER_CFG_USE_LINKSTA Note: Default value = 0	Specify whether or not to use the PHY-LSI status register instead of the LINKSTA signal when a change in the link status is detected. When 0 is specified, the PHY-LSI status register is used. When 1 is specified, the LINKSTA signal is used.
ETHER_CFG_USE_PHY_KSZ8041NL Note: Default value = 0	Specify whether or not the KSZ8041NL PHY-LSI from Micrel is used. When 0 is specified, the KSZ8041 is not used. When 1 is specified, the KSZ8041 is used.

5.6 Section setting

5.6.1 CCRX for Renesas RX section setting example

The following settings are required in the e^2 studio linker setting.

- Add the section.

Adding the Section

Add B_ETHERNET_BUFFERS_1, B_RX_DESC_1 and B_TX_DESC_1 section to the RAM.

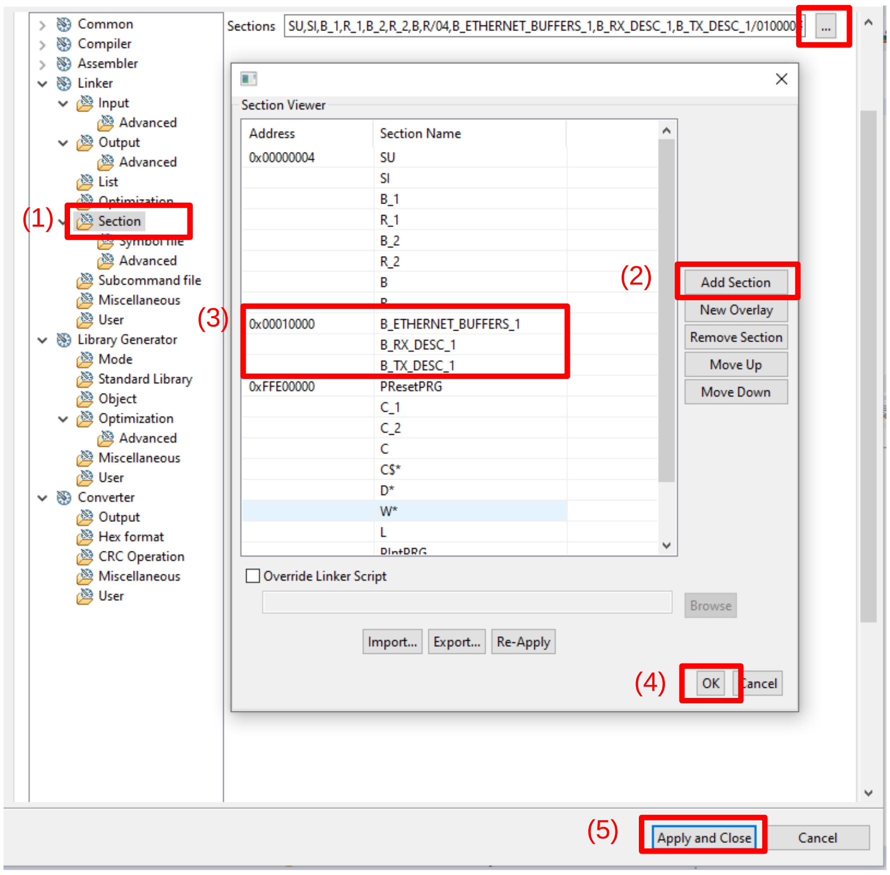
(1) Click the Section.

(2) Click the Add section button.

(3) Type Address and the Section Name.

(4) Click the OK button.

(5) Click the Apply and Close button.



Sections SU,SI,B_1,R_1,B_2,R_2,B,R/04,B_ETHERNET_BUFFERS_1,B_RX_DESC_1,B_TX_DESC_1/010000

Section Viewer

Address	Section Name
0x00000004	SU
	SI
	B_1
	R_1
	B_2
	R_2
	B
0x00010000	B_ETHERNET_BUFFERS_1
	B_RX_DESC_1
	B_TX_DESC_1
0xFFE00000	PResetPRG
	C_1
	C_2
	C
	C$*
	D*
	W*
	L

☐ Override Linker Script

5.6.2 GCC for Renesas RX section setting example

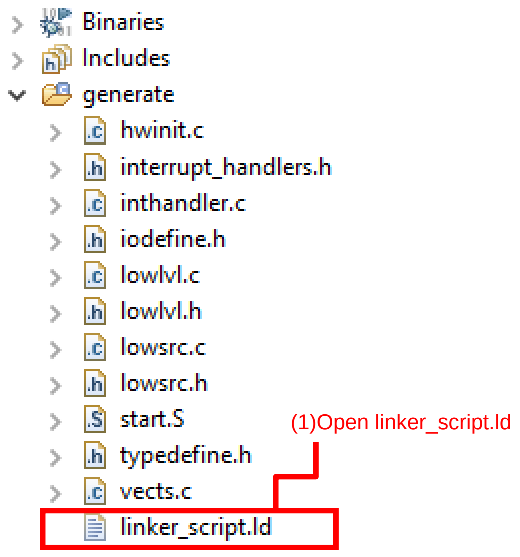
Edit the linker_script.ld file and add sections and symbols.

Add Sections and Symbols

Add the following code.

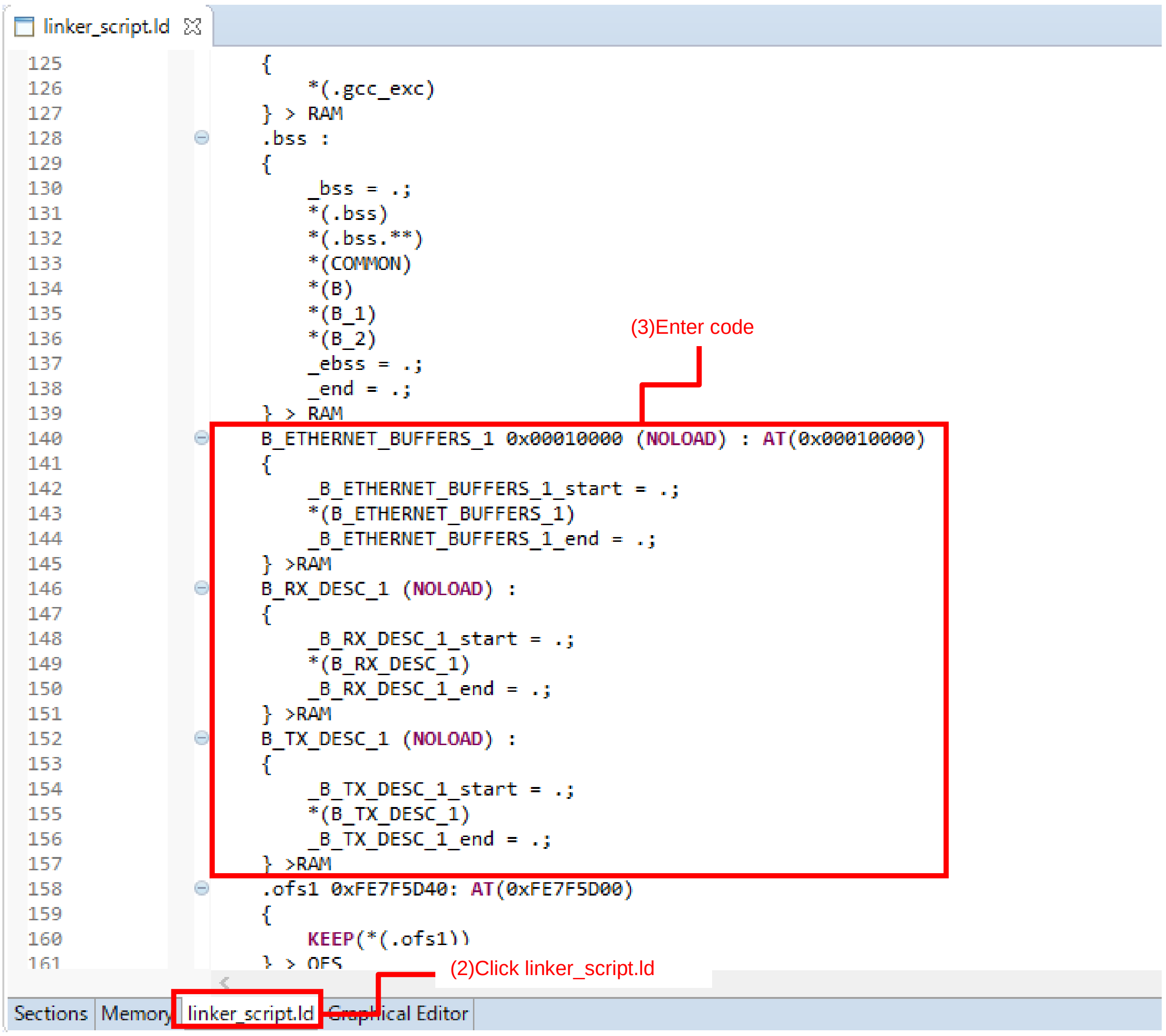
```
       B_ETHERNET_BUFFERS_1 0x00010000 (NOLOAD) : AT(0x00010000)
       {
              _B_ETHERNET_BUFFERS_1_start = .;
              *(B_ETHERNET_BUFFERS_1)
              _B_ETHERNET_BUFFERS_1_end = .;
       } >RAM
       B_RX_DESC_1 (NOLOAD) :
       {
              _B_RX_DESC_1_start = .;
              *(B_RX_DESC_1)
              _B_RX_DESC_1_end = .;
       } >RAM
       B_TX_DESC_1 (NOLOAD) :
       {
              _B_TX_DESC_1_start = .;
              *(B_TX_DESC_1)
              _B_TX_DESC_1_end = .;
       } >RAM
```

(1) Open "linker_script.ld" from Project Explorer.



(2) Click linker_script.ld.

(3) Enter code.



```
    {
        *(.gcc_exc)
    } > RAM
    .bss :
    {
        _bss = .;
        *(.bss)
        *(.bss.**)
        *(COMMON)
        *(B)
        *(B_1)
        *(B_2)
        _ebss = .;
        _end = .;
    } > RAM
    B_ETHERNET_BUFFERS_1 0x00010000 (NOLOAD) : AT(0x00010000)
    {
        _B_ETHERNET_BUFFERS_1_start = .;
        *(B_ETHERNET_BUFFERS_1)
        _B_ETHERNET_BUFFERS_1_end = .;
    } >RAM
    B_RX_DESC_1 (NOLOAD) :
    {
        _B_RX_DESC_1_start = .;
        *(B_RX_DESC_1)
        _B_RX_DESC_1_end = .;
    } >RAM
    B_TX_DESC_1 (NOLOAD) :
    {
        _B_TX_DESC_1_start = .;
        *(B_TX_DESC_1)
        _B_TX_DESC_1_end = .;
    } >RAM
    .ofs1 0xFE7F5D40: AT(0xFE7F5D00)
    {
        KEEP(*(.ofs1))
    } > OFS
```

5.6.3 IAR C/C++ Compiler for Renesas RX section setting example

Edit the icf file and add section settings.

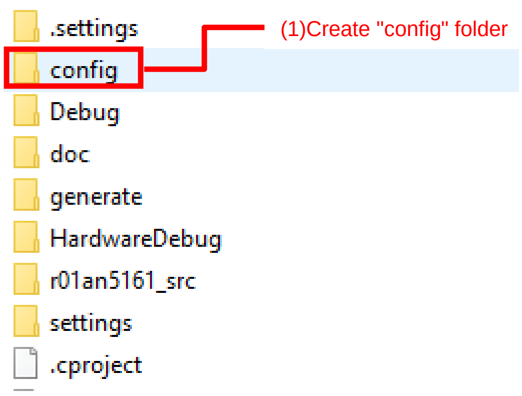
The icf file to be edited depends on the target device of the project, so please confirm and edit the upper 8 digits of the model name of the device to be used.

As an example, edit "lnkr5f563nf.icf" with RX63N (R5F563NFDxFC).

The following is an example of editing on the RX63N (R5F563NFDxFC).

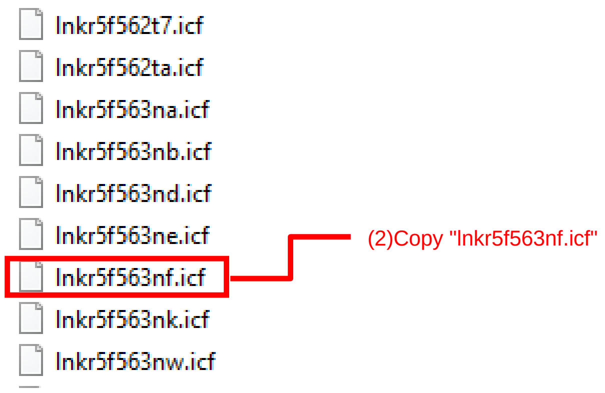
Add section settings.

(1) Create "config" folder in project folder.



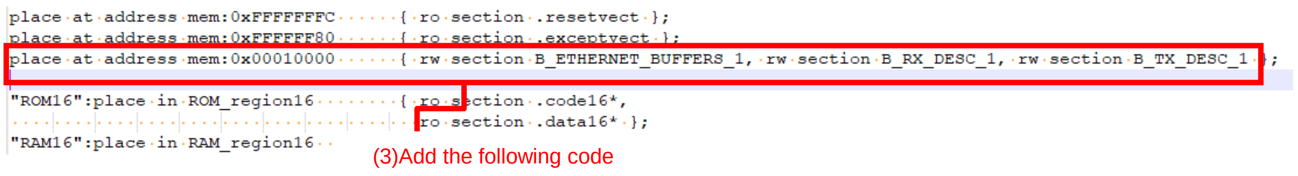
(2) Copy "lnkr5f563nf.icf" from "\rx\config" where IAR C/C++ Compiler for Renesas RX ("EWRX") is installed to the "config" folder in the project folder.

The installation default is "C: \Program Files (x86)\IAR Systems\Embedded Workbench 8.3".



(3) Open the copied "lnkr5f563nf.icf" file and add the following code.

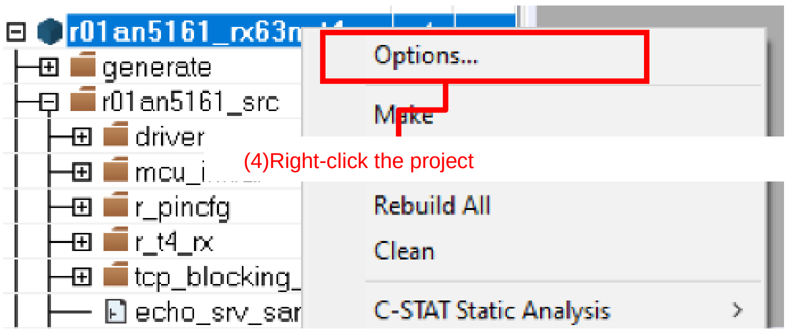
```
place at address mem:0x00010000      { rw section B_ETHERNET_BUFFERS_1, rw section
B_RX_DESC_1, rw section B_TX_DESC_1 };
```



```
place at address mem:0xFFFFFFFC      { ro section .resetvect };
place at address mem:0xFFFFFF80      { ro section .exceptvect };
place at address mem:0x00010000      { rw section B_ETHERNET_BUFFERS_1, rw section B_RX_DESC_1, rw section B_TX_DESC_1 };

"ROM16":place in ROM_region16        { ro section .code16*,
                                       ro section .data16* };
"RAM16":place in RAM_region16
```

(4) Open the project from EWRX, right-click the project in the workspace and open options.

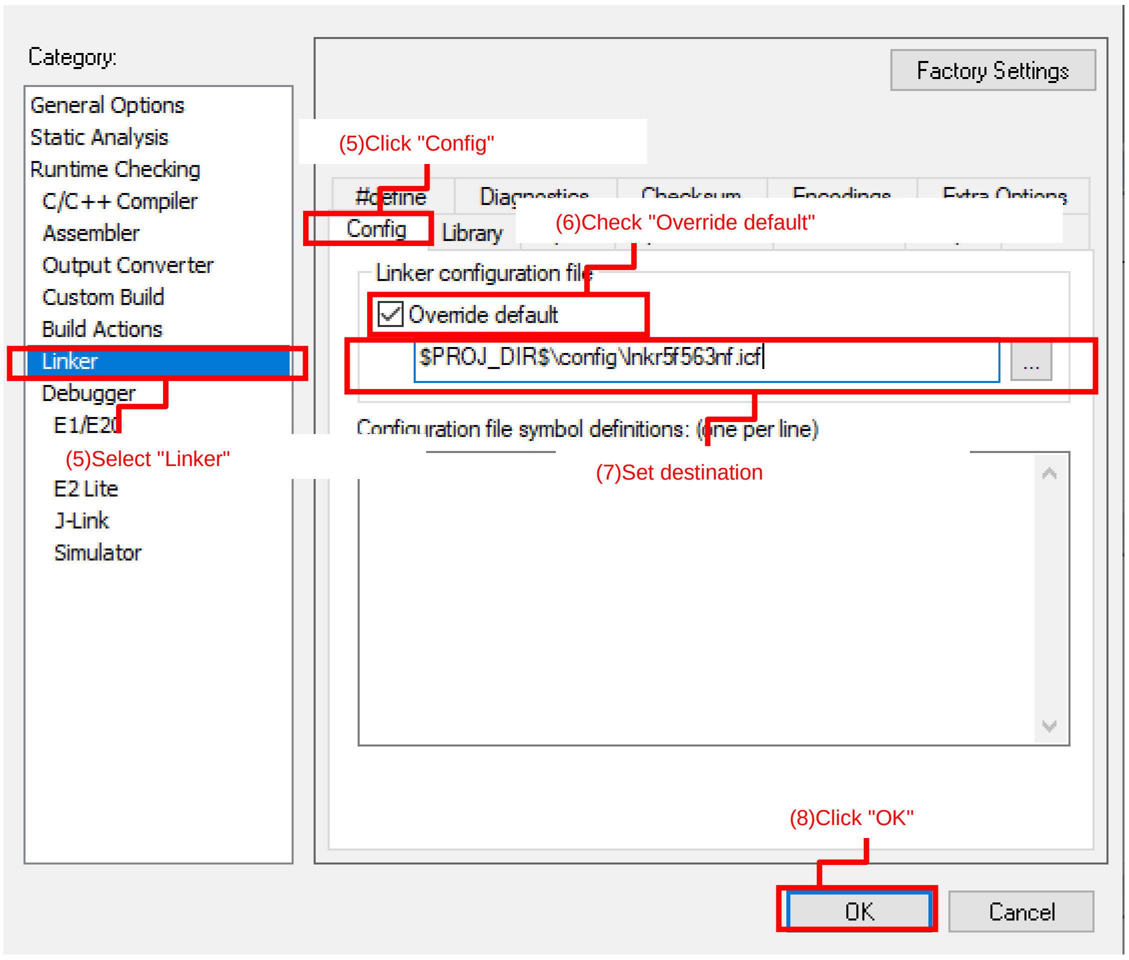


(5) Select "Category: Linker" and click "Config".

(6) Check "Override default".

(7) Set the file edited in step (3) as a reference destination.

(8) Click "OK".



Linker configuration file: [✓] Override default, $PROJ_DIR$\config\lnkr5f563nf.icf

5.7 Option-Setting Memory

Table 5-5 lists the Option-Setting Memory Configured in the Sample Code. When necessary, set a value suited to the user system.

Table 5-5 Option-Setting Memory Configured in the Sample Code

Symbol	Address	Setting Value	Contents
OFS0	FFFF FF8Fh to FFFF FF8Ch	FFFF FFFFh	The IWDT is stopped after a reset. The WDT is stopped after a reset.
OFS1	FFFF FF8Bh to FFFF FF88h	FFFF FFFFh	The voltage monitor 0 reset is disabled after a reset. HOCO oscillation is disabled after a reset.
MDES	FFFF FF83h to FFFF FF80h	FFFF FFFFh	Little endian

Notes:

1. Option-Setting memory only for RX63N.

6. Sample code

Sample code can be downloaded from the Renesas Electronics website.

7. Notes

7.1 T4 Library

(1) Specify the size of 15bit or less for the third argument "INT len" of tcp_rcv_dat() and tcp_snd_dat().

(2) Specify the size of 15bit or less for argument "TMO tmout" of tcp_acp_cep(), tcp_con_cep(), cp_cls_cep(), tcp_rcv_dat(), tcp_snd_dat(), udp_snd_dat() and udp_rcv_dat().

(3) This library can be used with Microcontroller Options fint_register=0 (Fast interrupt vectorregister [None]). The default for this option is fint_register=0.

7.2 Sample Program

(1) The sample program for little endian mode is only included.

(2) The MAC address of the sample program is stored in _myethaddr variable of config_tcpudp.c.
Change an initial value of the _myethaddr (MAC address) variable if necessary according to the system.

Revision History

Rev.	Date	Description	
		Page	**Summary**
1.00	Dec 22, 19	-	First released.

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.

Corporate Headquarters

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

Trademarks

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

Contact information

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