μNet3 Users Guide

9th Edition eForce Co., Ltd

Introduction

μNet3 (Micro Net Cube) is a TCP/IP protocol stack which is integrated for real-time operating system μ **C3 (Micro C Cube)** of our company**.**

μ Net3 is easy to understand that is memory saving, designed to implement flexible interfaces.

The position of this document

This document is a manual of μ Net3/Compact and μ Net3/Standard, regarding to the method of using real-time OS, please make reference to "μ C3/Compact Users Guide" and "μ C3/Standard Users Guide.

The contents of this document may be changed without prior notice.

Revision History

Modified items in 2nd edition

Chapter	Contents
2.1.24, 4.1.3	Updates MTU
2.1.26, 3.1.1	Updates IP reassembly
2.1.21, 5.4	Updates callback function
4.3	Updates the setup of network initialization task, MTU, number of network buffer operated by configurator

Modified items in 3rd edition

Chapter	Contents
1.3, 4.2, 5.4	Adds development procedure, configuration and API of Standard version
2.2.1, 4.1.5, 4.1.6, 5.4	Updates due to network devices are available to associate with socket arbitrarily
2.3, 4.1.3	Updates due to adding template network device

Modified items in 4th edition

Chapter	Contents
4.2	Updates due to adding items of configuration setup

Modified items in 5th edition

Chapter	Contents
4.2	Support configurator ver 3
6.3	HTTP server update with additional functions

Modified items in 6th edition

Chapter	Contents
2.3, 6.5, 6.6., 6.7	Modified by adding a network app
3.2	Modified by adding the loopback interface
3.1.4	Modified by adding TCP Keep-Alive feature
3.1.2, 5.2	Modified by adding ACD feature
3.4, 6.8	Correction due to the non-use of the standard library String system

Modified items in 7th edition

Chapter	Contents
4.3.1	Updates due to adding items of configuration setup for µNet3/Standard

Modified items in 8th edition

Chapter	Contents
5.4	Change return value cre_soc() at over capable generating number of socket

Modified items in 9th edition

Chapter	Contents
5.4, 7.3	Add a description of the error code EV_ADDR
6.3	Corrected HTTP server control information structure

Table of contents

Chapter 1: What is μNet3?

1.1 Features

μNet3 is a compact TCP/IP protocol stack which is optimized for one-chip microcomputer. Besides, in order to facilitate the installation, we are now adopting the original API which is very easy to understand.

1.2 Key Functions

- Supports IPv4, ARP, ICMP, IGMPv2, UDP, TCP Protocol
- Possibility of using functions such as DHCP client, DNS client, FTP server, HTTP server
- Possibility of setting up TCP/IP by Configurator (Compact version)
- Supports TCP Fast retransmit/ Fast recovery algorithm
- Supports IP reassembly and fragmentation
- Supports plural network interfaces

1.3 Development Procedure

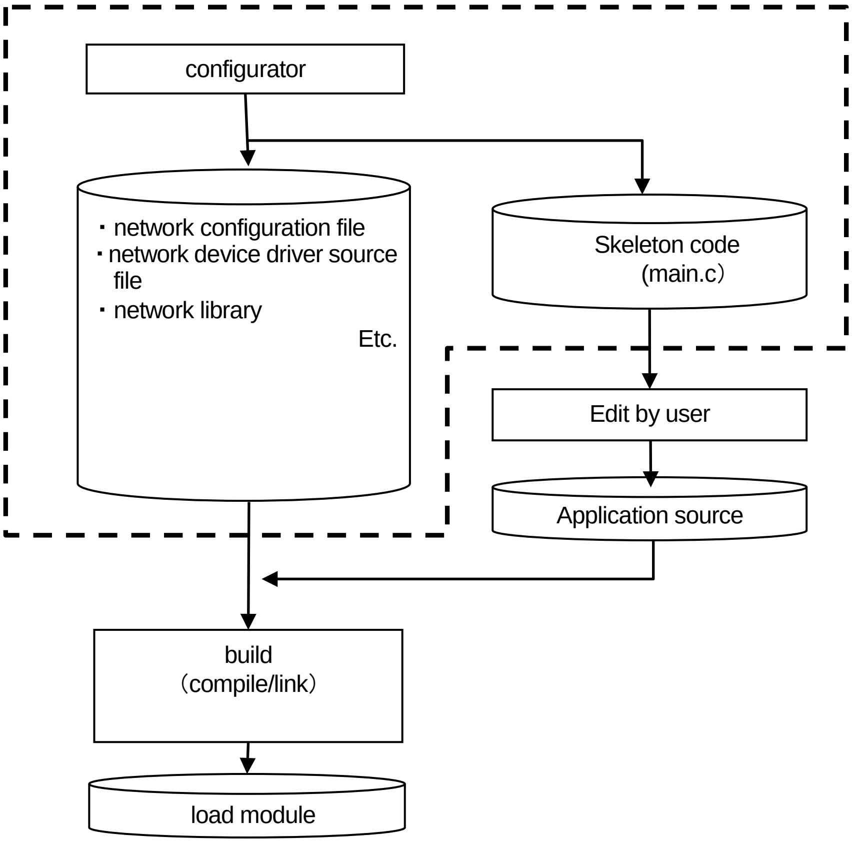
The development procedure used μNet3 is shown in the figure.

In case of using Compact version, at first we will input information relating to TCP/IP into configurator. The configurator will generate a code based on input informations. The generated code has file and skeleton code to be used without modification. This skeleton code is created in order to support describing necessary application program.

In case of using Standard version, we will describe configurations such as network configuration (IP address,socket definition),device driver configuration (MAC address, device I/F definition, features specific to driver), μNet3 initialization routine calling in template source code net_cfg.c.

After describing application program, we build and creat a load module.

※The configurator can run the kernel configuration at the same time, but we do not mention in this document. Please make appropriate reference to “**μC3/Compact Users Guide”.**



The figure of the development procedure

All parts are surrounded by in the figure of the development procedure is created automatically by generating source function of configurator in Compact version. By Standard version, configuration inputs the information as designed, then integrate into net_cfg.c file. Please refer to **Chapter 4 Configuration** for more details.

1.4 Tutorial using sample

It will explain about using samples are packaged in μNet3/Compact to creat the program. In μNet3/Standard, the similar program that wascreated previously is also packaged.

Sample description

This time, you can use Sample¥ARMv4T¥IAR_LPC2478_STK.NET in Sample folder (**please make appropriate changes to Sample¥XXX.NET folder which suits your usage environment)**. It uses this sample to create the program combines HTTP server and DHCP client

・DHCP client

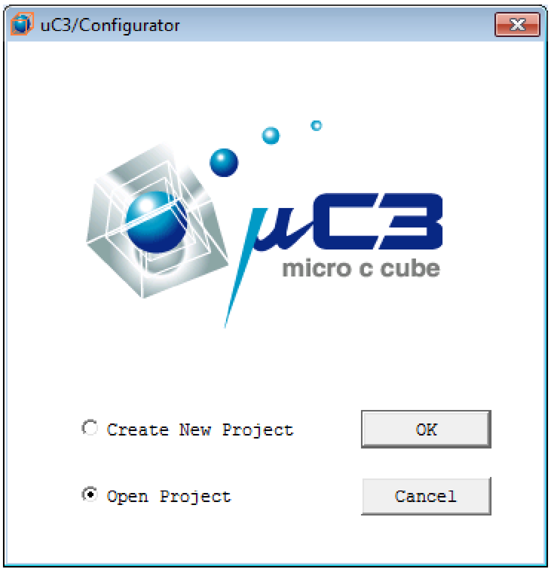
Acquire dynamic IP address from a DHCP server, assign to local host.

・HTTP server

Changes the blink interval of LED to 100 msec from Web browser

1. 4. 1　Configuration file reading

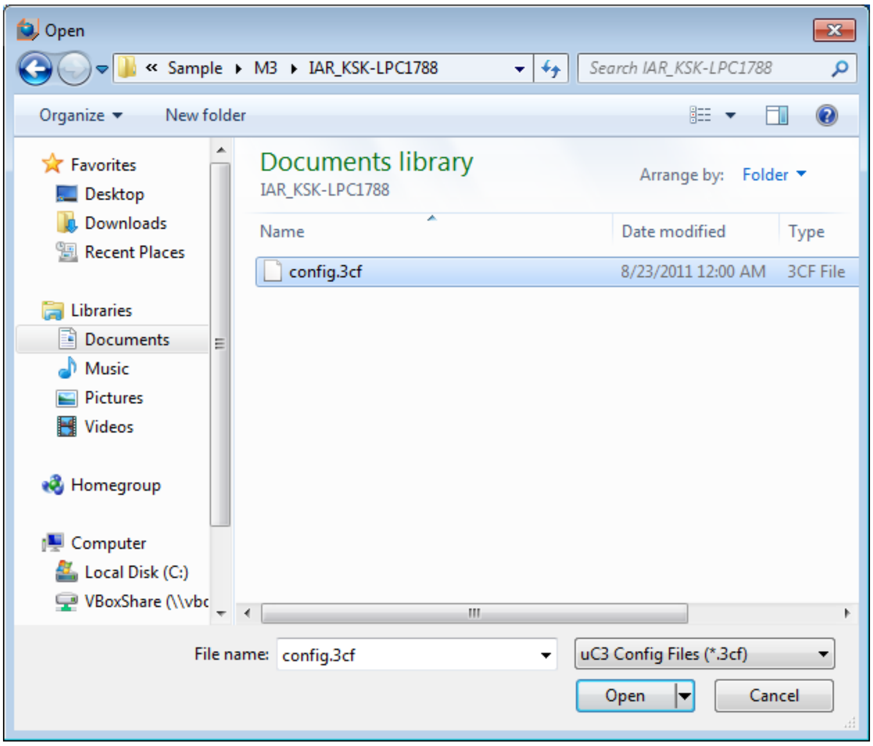
Start up file Config¥uC3conf.exe and then read the file which has finished configuration.


uC3/Configurator

○ Create New Project

● Open Project

Choose　“Open existing project”　and then click　“OK”　button.


Open

« Sample ▸ M3 ▸ IAR_KSK-LPC1788

Documents library
IAR_KSK-LPC1788

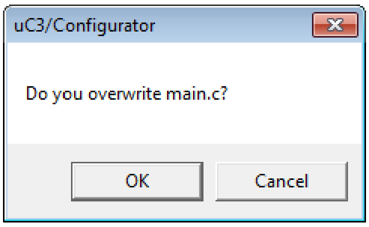
Arrange by: Folder

Name	Date modified	Type
config.3cf	8/23/2011 12:00 AM	3CF File

File name: config.3cf

uC3 Config Files (*.3cf)

Choose file　config_net.3cf　and then click　“Open “button

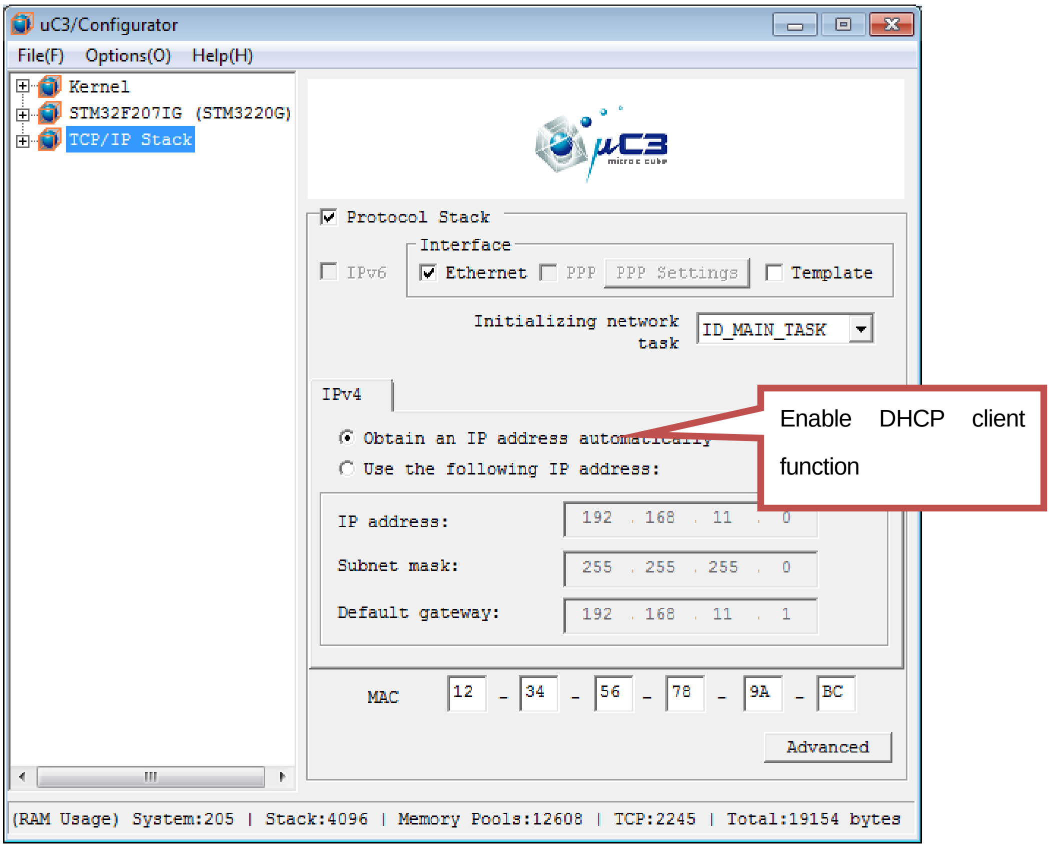

uC3/Configurator

Do you overwrite main.c?

If confirm dialog asking to change CPU appears, click “No” button

1. 4. 2 Setup using uC3/configurator

Run configuration of μ Net3/Compact. This time, configuration has already been done before so it don’t have to change the setup.


uC3/Configurator

☑ Protocol Stack

Interface: □ IPv6 ☑ Ethernet □ PPP PPP Settings □ Template

Initializing network task: ID_MAIN_TASK

IPv4

● Obtain an IP address automatically

○ Use the following IP address:

IP address: 192 . 168 . 11 . 0

Subnet mask: 255 . 255 . 255 . 0

Default gateway: 192 . 168 . 11 . 1

MAC 12 - 34 - 56 - 78 - 9A - BC

(RAM Usage) System:205 | Stack:4096 | Memory Pools:12608 | TCP:2245 | Total:19154 bytes

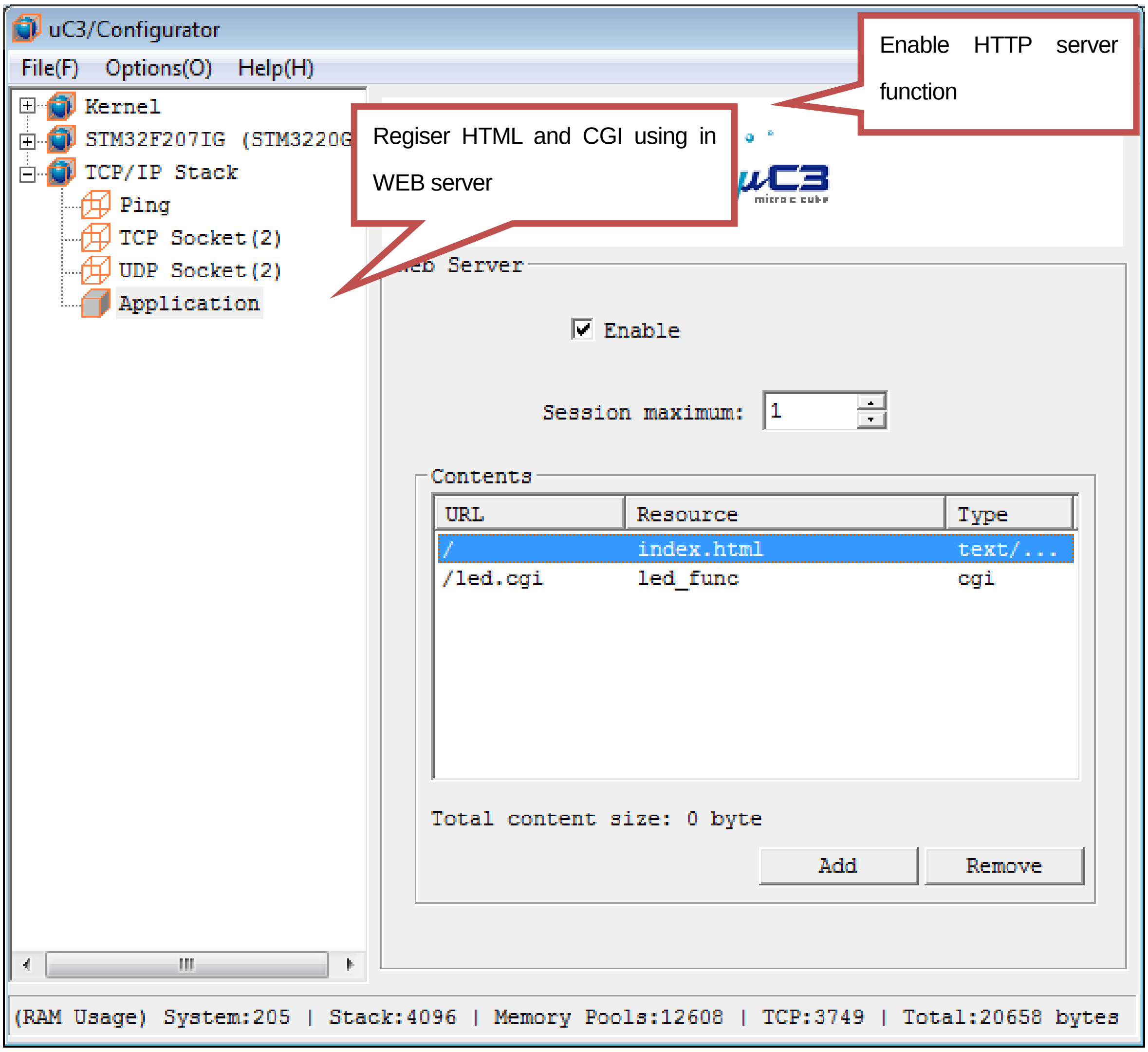
Enable DHCP client function

Choose “TCP/IP stack” by mouse click on the left tree. On this screen, primarily set up IP address of target side.

Now, please make sure that the option “Get IP address automatically” has been chosen.

On the left tree, choose "Application" by mouse click. On this screen, primarily register files of HTML or JPG using in WEB (HTTP) server and function name of CGI program. In WEB server, CGI function is supported, LED blinking uses this function to run. If using CGI fucntion, It is possible to call the function which is specified to meet the requirement from the browser. Here, it sets URL as /led.cgi and specify the led_func() function is able to call.

In case of choosing "Use" WEB server, please make sure HTML : index.html and CGI function : led_func are recorded in "Content".



Enable HTTP server function

Regiser HTML and CGI using in WEB server

Web Server

☑ Enable

Session maximum: 1

Contents

URL	Resource	Type
/	index.html	text/...
/led.cgi	led_func	cgi

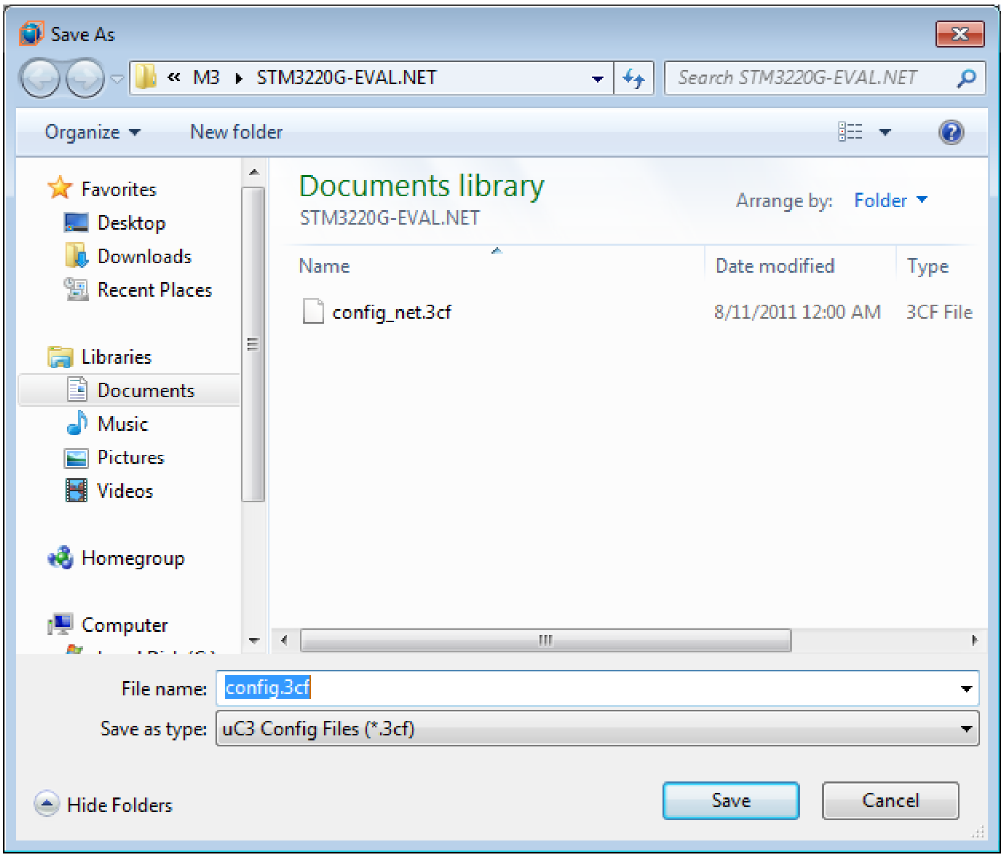
Total content size: 0 byte

(RAM Usage) System:205 | Stack:4096 | Memory Pools:12608 | TCP:3749 | Total:20658 bytes

1. 4. 3 Saving configurator setup

This time, we do not change configuration setup of μNet3/Compact but we will save the configurator setup in order to confirm the content established by configurator.

Choose “Save” from “File” menu.



Name	Date modified	Type
config_net.3cf	8/11/2011 12:00 AM	3CF File

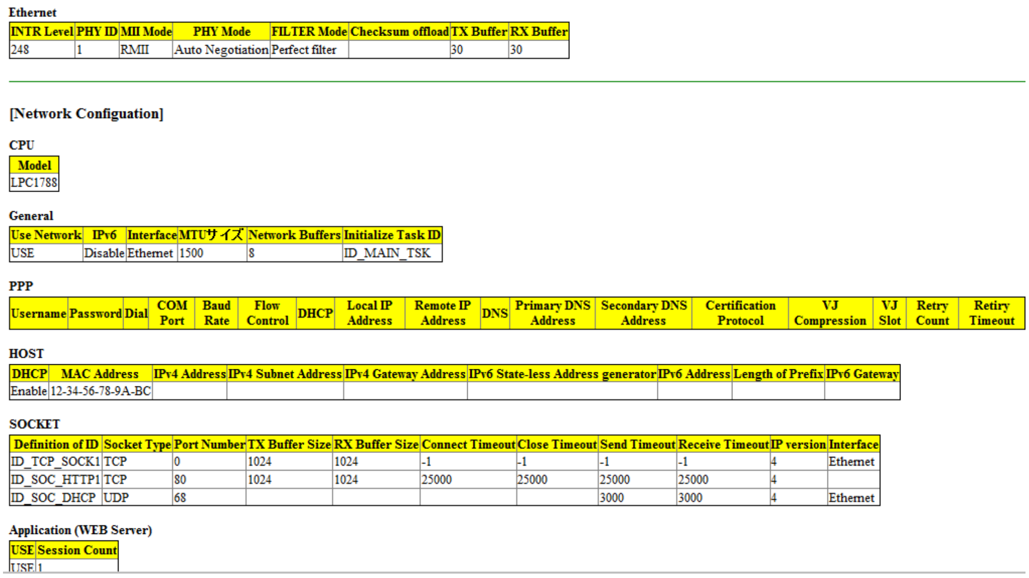
File name: config.3cf

Save as type: uC3 Config Files (*.3cf)

This time, just overwrite save without change any setup. Click “Yes” button when over write attention message appears.

Upon act of saving, it can save the setup of configurator. And at the same time, also by this act, the project file (config_net.3cf) and the file with file extension has changed to .xml are saved.

After opening config_net.xml by browser, we can confirm on browser screen all the information established by the configurator. An example of setup by network is as below.



Ethernet

INTR Level	PHY ID	MII Mode	PHY Mode	FILTER Mode	Checksum offload	TX Buffer	RX Buffer
248	1	RMII	Auto Negotiation	Perfect filter		30	30

[Network Configuation]

CPU

Model
LPC1788

General

Use Network	IPv6	Interface	MTUサイズ	Network Buffers	Initialize Task ID
USE	Disable	Ethernet	1500	8	ID_MAIN_TSK

PPP

Username	Password	Dial	COM Port	Baud Rate	Flow Control	DHCP	Local IP Address	Remote IP Address	DNS	Primary DNS Address	Secondary DNS Address	Certification Protocol	VJ Compression	VJ Slot	Retry Count	Retiry Timeout

HOST

DHCP	MAC Address	IPv4 Address	IPv4 Subnet Address	IPv4 Gateway Address	IPv6 State-less Address generator	IPv6 Address	Length of Prefix	IPv6 Gateway
Enable	12-34-56-78-9A-BC							

SOCKET

Definition of ID	Socket Type	Port Number	TX Buffer Size	RX Buffer Size	Connect Timeout	Close Timeout	Send Timeout	Receive Timeout	IP version	Interface
ID_TCP_SOCK1	TCP	0	1024	1024	-1	-1	-1	-1	4	Ethernet
ID_SOC_HTTP1	TCP	80	1024	1024	25000	25000	25000	25000	4	
ID_SOC_DHCP	UDP	68					3000	3000	4	Ethernet

Application (WEB Server)

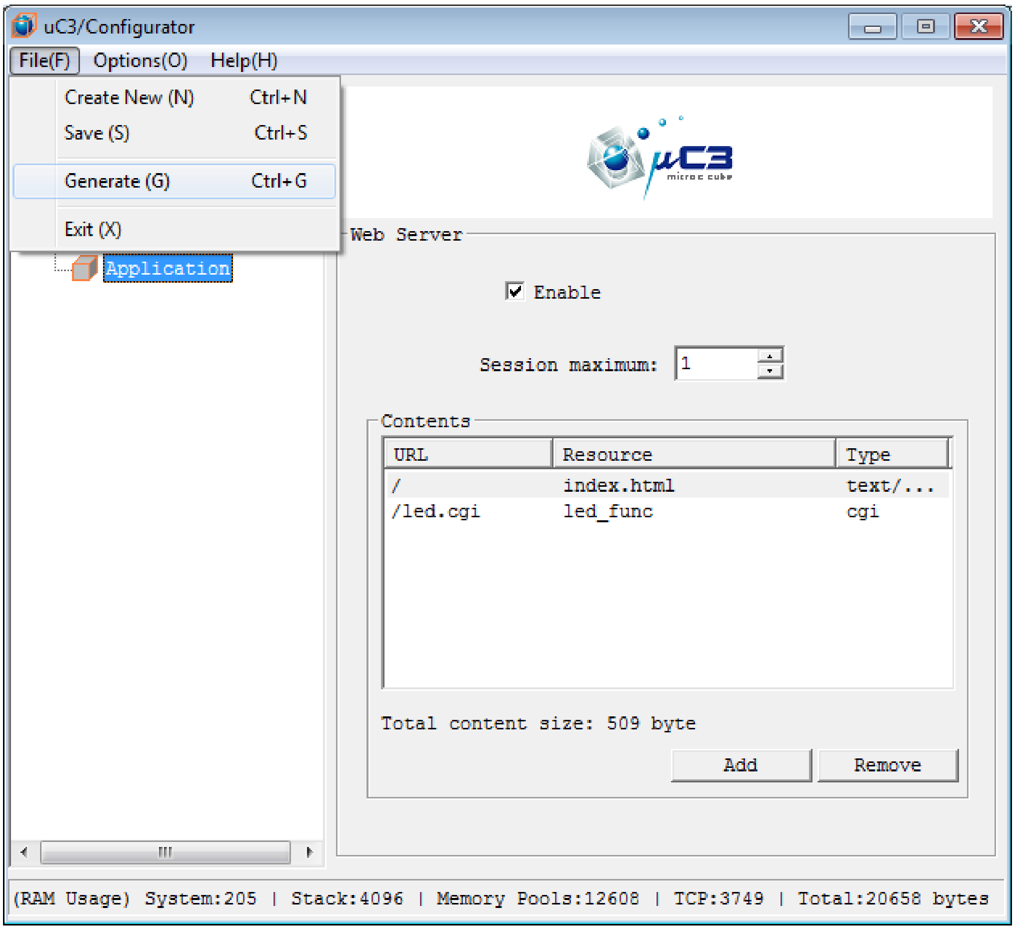
USE	Session Count
USE	1

Content table was generated by configurator

(This XML information is only Japanese in current version.)

1. 4. 4 Source code generation

Generate source code.



Web Server

☑ Enable

Session maximum: 1

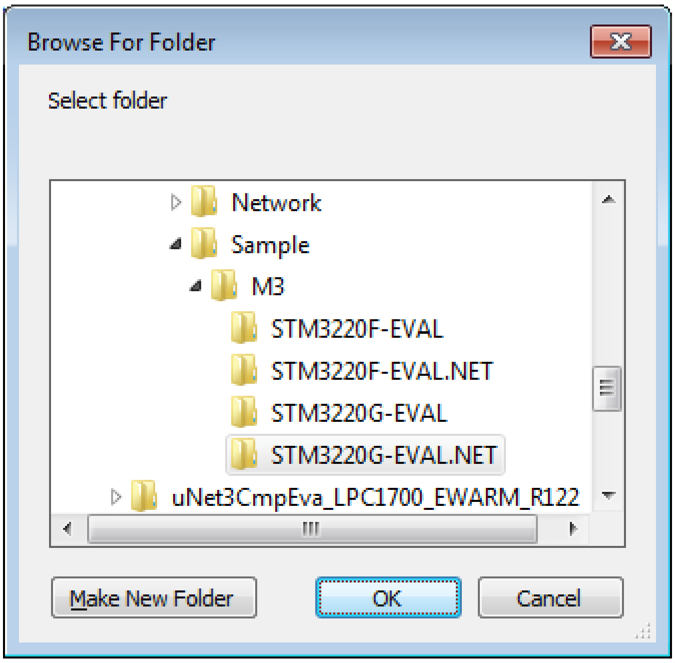
Contents

URL	Resource	Type
/	index.html	text/...
/led.cgi	led_func	cgi

Total content size: 509 byte

(RAM Usage) System:205 | Stack:4096 | Memory Pools:12608 | TCP:3749 | Total:20658 bytes

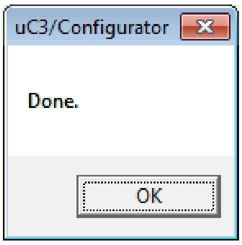
Choose “Generate source” on File menu.



Browse For Folder

Select folder

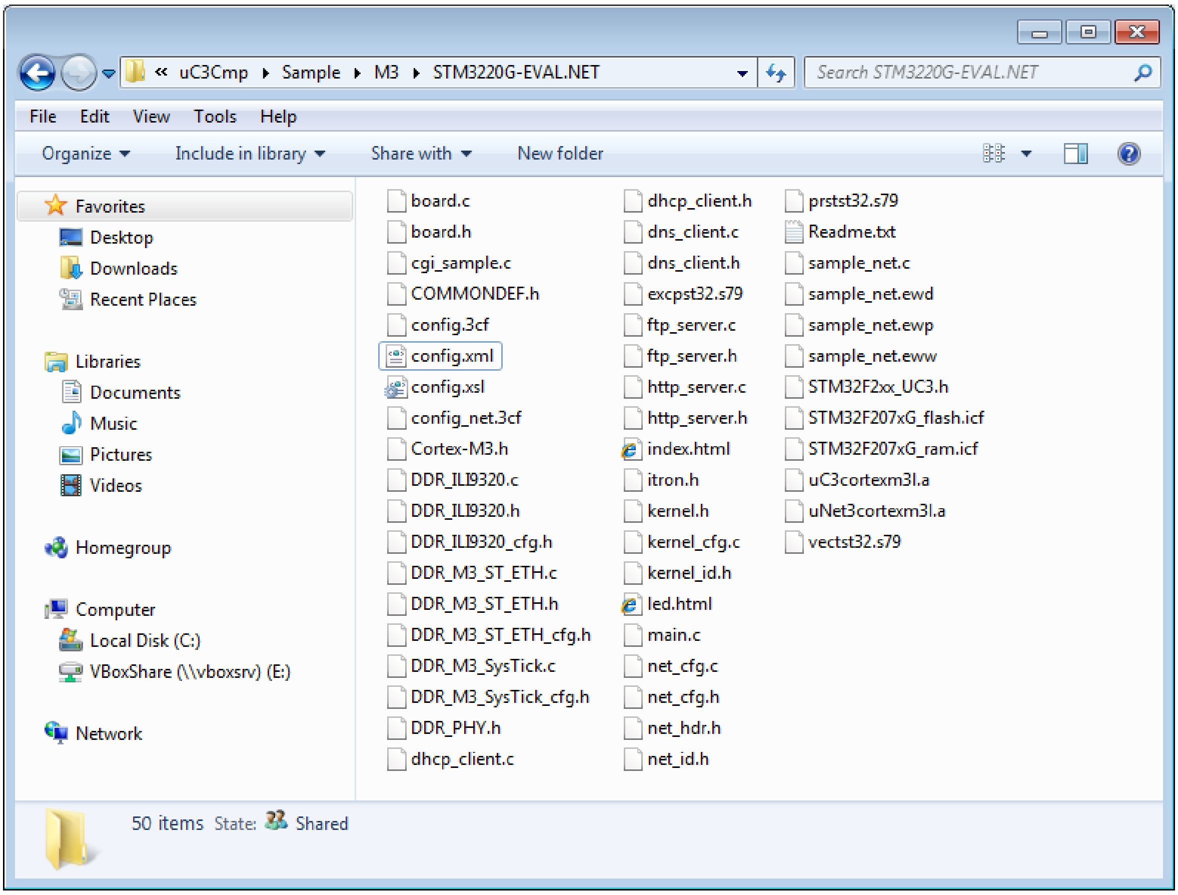
Before generating source, please choose Sample¥ARMv4T¥IAR_LPC2478_STK.NET.



uC3/Configurator

Done.

If compete dialog box appears on screen, please click “OK” button. Then, source code generation has completed. Generated source has TCP/IP protocol stack library, configuration file… etc. like the map of generated file.



board.c
board.h
cgi_sample.c
COMMONDEF.h
config.3cf
config.xml
config.xsl
config_net.3cf
Cortex-M3.h
DDR_ILI9320.c
DDR_ILI9320.h
DDR_ILI9320_cfg.h
DDR_M3_ST_ETH.c
DDR_M3_ST_ETH.h
DDR_M3_ST_ETH_cfg.h
DDR_M3_SysTick.c
DDR_M3_SysTick_cfg.h
DDR_PHY.h
dhcp_client.c
dhcp_client.h
dns_client.c
dns_client.h
excpst32.s79
ftp_server.c
ftp_server.h
http_server.c
http_server.h
index.html
itron.h
kernel.h
kernel_cfg.c
kernel_id.h
led.html
main.c
net_cfg.c
net_cfg.h
net_hdr.h
net_id.h
prstst32.s79
Readme.txt
sample_net.c
sample_net.ewd
sample_net.ewp
sample_net.eww
STM32F2xx_UC3.h
STM32F207xG_flash.icf
STM32F207xG_ram.icf
uC3cortexm3l.a
uNet3cortexm3l.a
vectst32.s79

50 items State: Shared

The map of generated file

1. 4. 4 Creat Program

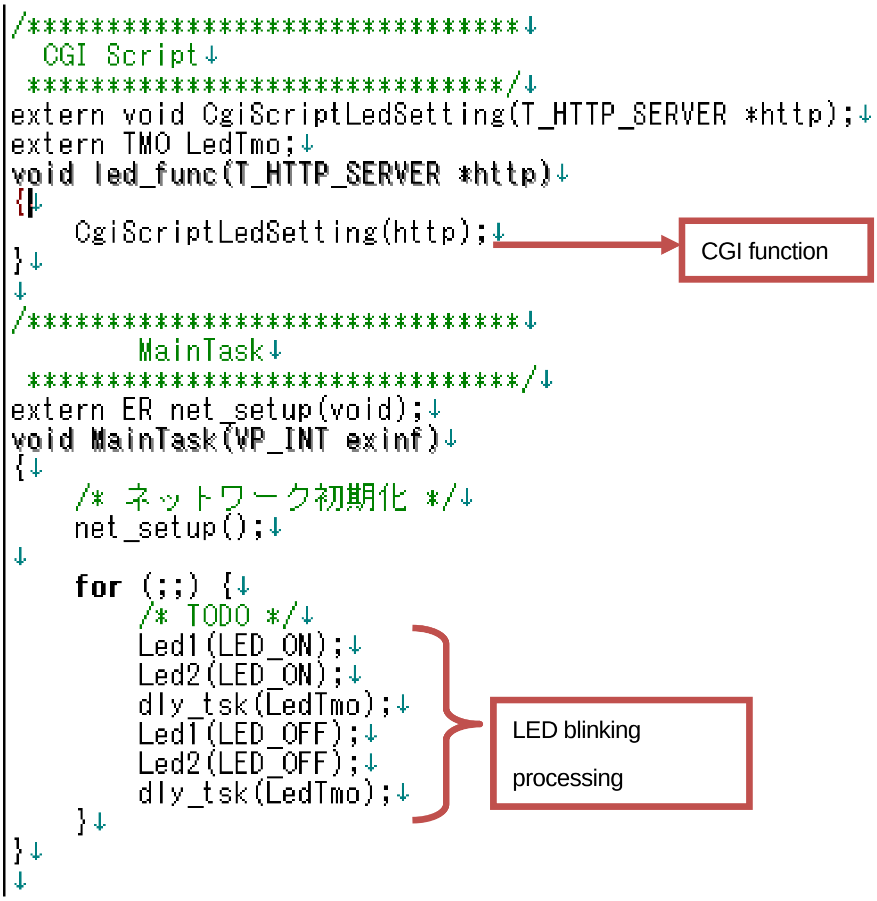
Start up IAR Embedded Workbench, and then read sample_net.eww. In the future, describing the program in skeleton code main.c has been generated by configurator, but this time we use sample_net.c in program which has finished describing in advance.

Creat CGI program

Receive the setting value which has been sent here by the application using CGI function. CGI is a system used to start the program on demand from browser, after that, accepts the setting value from browser and then send a response back to the browser.

CgiScriptLedSetting() function which was described in sample code cgi_ sample.c will set the setting value of interval time of LED blinking which was sent from browser in LedTmo variable. Then describe call processing of CgiScriptLedSetting() in led_func(), describe LED blinking processing in MainTask (Source list).

Source list



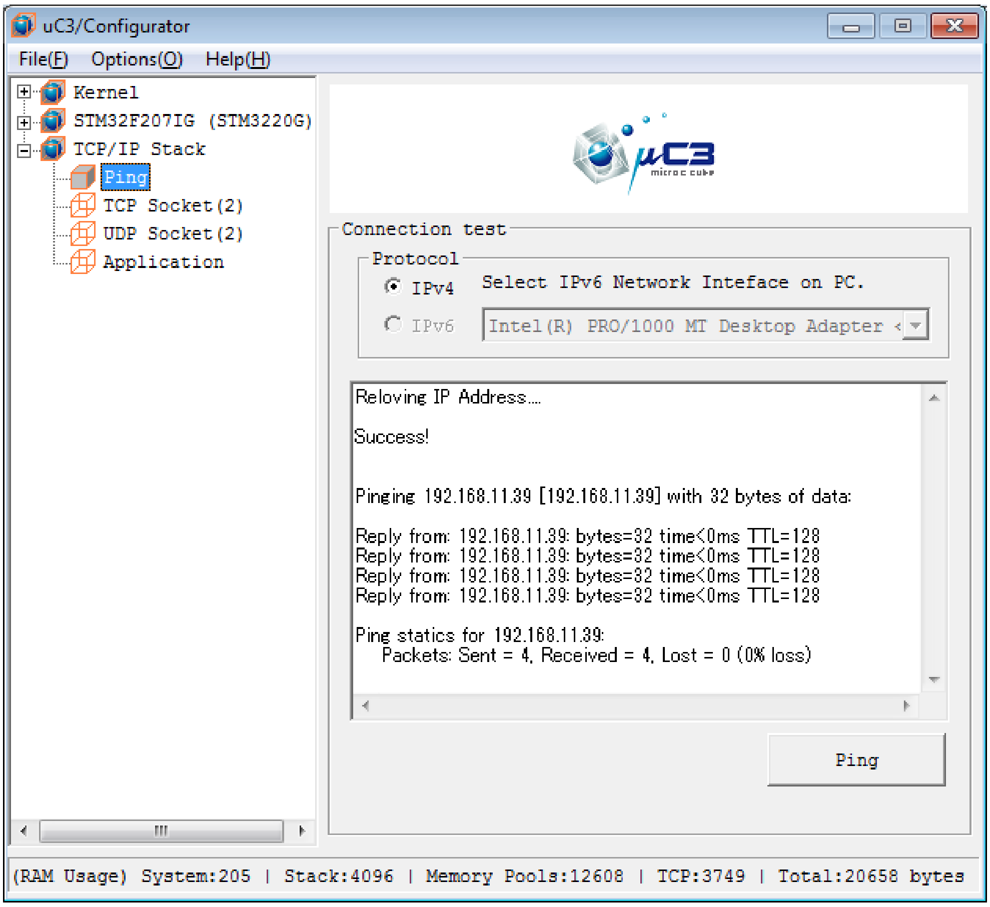
```
/*********************************
  CGI Script
 *********************************/
extern void CgiScriptLedSetting(T_HTTP_SERVER *http);
extern TMO LedTmo;
void led_func(T_HTTP_SERVER *http)
{
    CgiScriptLedSetting(http);
}

/*********************************
        MainTask
 *********************************/
extern ER net_setup(void);
void MainTask(VP_INT exinf)
{
    /* ネットワーク初期化 */
    net_setup();

    for (;;) {
        /* TODO */
        Led1(LED_ON);
        Led2(LED_ON);
        dly_tsk(LedTmo);
        Led1(LED_OFF);
        Led2(LED_OFF);
        dly_tsk(LedTmo);
    }
}

```

1. 4. 5 WEB server execution

In IAR Embedded Workbench, choose “Flash Debug” in construction of build objects, after that, build it and create load module.

Write the load module in Flash memory, if the program executes correctly, the LED on board will blink. At first, use the communication test function of configurator to check whether TCP/IP stack is working properly or not. Choose “Communication test”, if then click “Execute communication test” button, can Ping to target based on all the contents set by configurator. If the communication works normally, a reply message as “Reply from xxx.xxx.xxx.xxx.....” is displayed like the **communication test picture**. Because of that message, we can confirm that the target responses properly or not. Next, start up browser. If input the established IP address directly on browser, HTML screen set up by configurator is displayed (WEB page picture). Here, if we input the interval, the LED will blink at specified time interval.

※ **In case PC has been installed virus security software, even though the target program is working properly, communication test may fail. At that time, we will execute the communication test after neutralize all the settings of virus security software.**

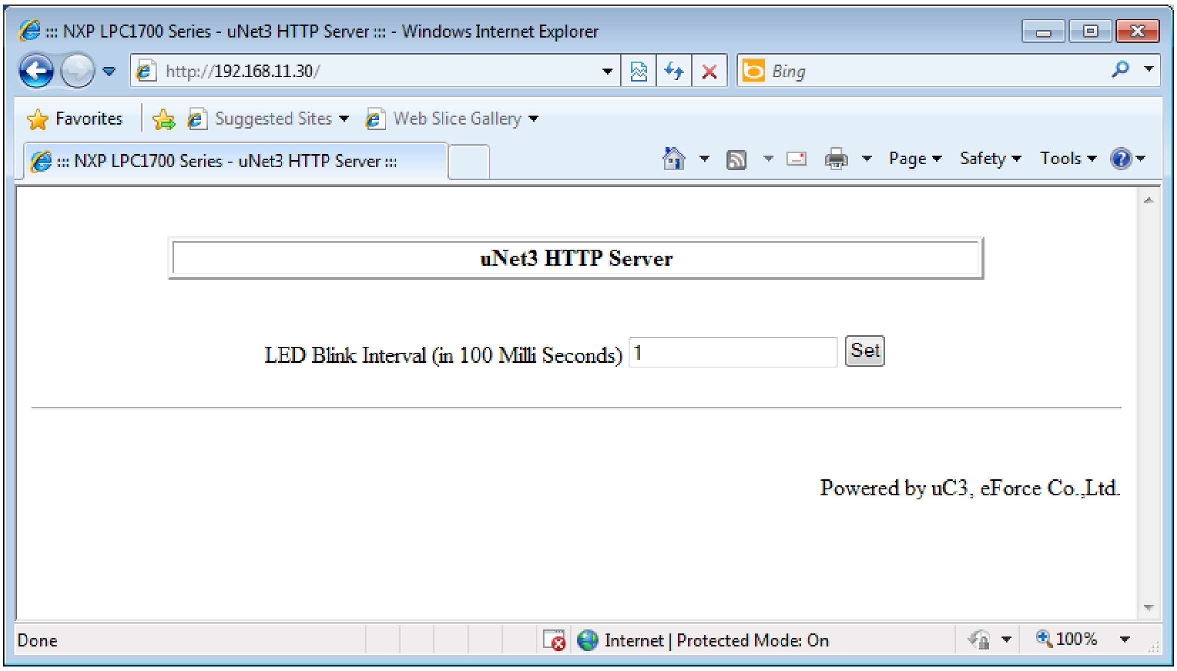


● IPv4 Select IPv6 Network Inteface on PC.
○ IPv6 Intel(R) PRO/1000 MT Desktop Adapter

```
Reloving IP Address....

Success!


Pinging 192.168.11.39 [192.168.11.39] with 32 bytes of data:

Reply from: 192.168.11.39: bytes=32 time<0ms TTL=128
Reply from: 192.168.11.39: bytes=32 time<0ms TTL=128
Reply from: 192.168.11.39: bytes=32 time<0ms TTL=128
Reply from: 192.168.11.39: bytes=32 time<0ms TTL=128

Ping statics for 192.168.11.39:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
```

(RAM Usage) System:205 | Stack:4096 | Memory Pools:12608 | TCP:3749 | Total:20658 bytes

Communication test picture


::: NXP LPC1700 Series - uNet3 HTTP Server ::: - Windows Internet Explorer
http://192.168.11.30/
Bing
Favorites
Suggested Sites
Web Slice Gallery
::: NXP LPC1700 Series - uNet3 HTTP Server :::
Page
Safety
Tools
Done
Internet | Protected Mode: On
100%

uNet3 HTTP Server

LED Blink Interval (in 100 Milli Seconds) 1 Set

Powered by uC3, eForce Co.,Ltd.

WEB page picture

Chapter 2: Basic concepts of μNet3

2. 1 Glossary

2. 1. 1 Protoco

Protocol is a set of rules that determines the method and the procedure of transmitting data between networks. μNet3/Compact adopts this protocol (=communication rule). These rules are called Request For Comments (abbreviation: RFC)” ,its specification is published.

2. 1. 2 Protocol stack

Choose a necessary protocol in order to implement functions on network,and a protocol stack is a prescribed hierarchy of software layers. The following figure show the hiearachy in µNet3.

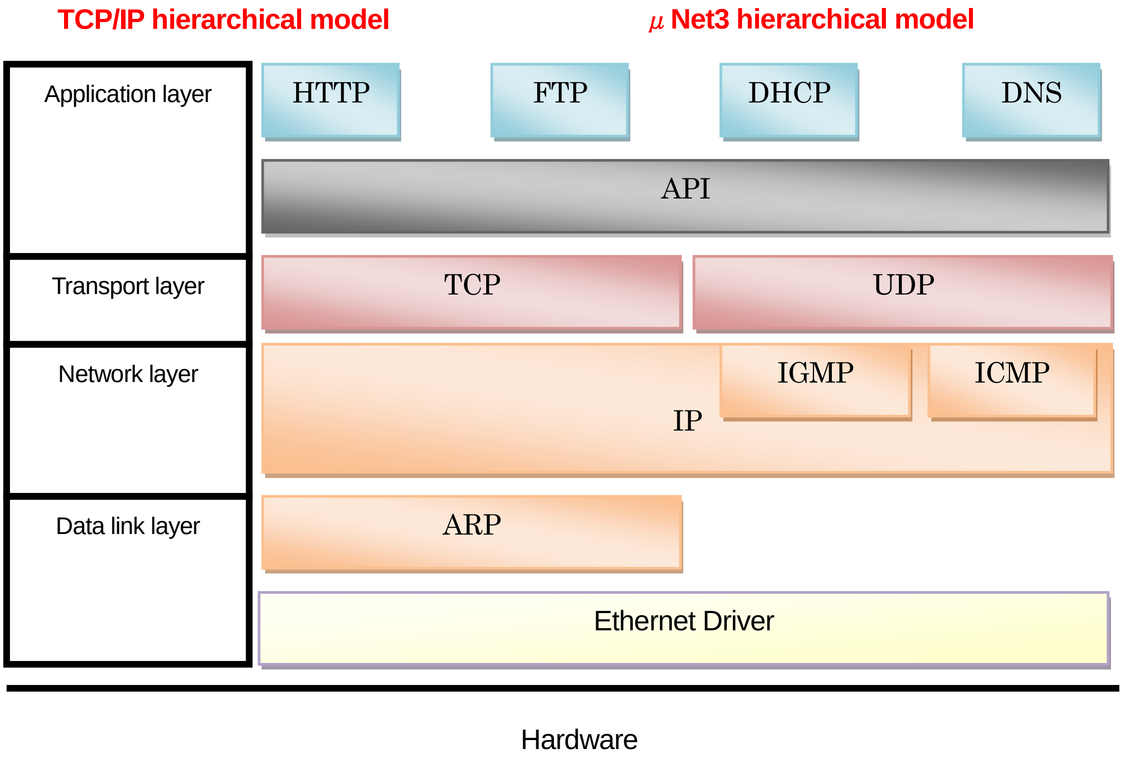


Figure of TCP/IP hierarchical model and μNet3 hierarchical model

2. 1. 3 IP (Internet Protocol) address

Each node on the network has a specific logical number, it is called “IP address”. IP address has 32 bit address space,be represent as 192.168.1.32.

Broadcast address

Broadcast means that the same data is simultaneously sent (broadcast communication) to all of the nodes in one network. The address is allocated particularly to broadcast called "Broadcast address". Ordinarily in "Broadcast address", all bits use 1 IP address "255.255.255.255".

Multicast address

Contrary to the broadcast that send data to all nodes, a special address is used to send data to a specific group only,is called "Multicast address".

2. 1. 4 MAC (Media Access Control) address

Contrary to a logical address "IP address", a physical address specify to an installed hardware in order to identify network devices such as LAN card is called "MAC address". "MAC address" has 48-bit address space and to be notated 12-34-56-78-9A-BC or 12:34:56:78:9A:BC.

2. 1. 5 Port number

In network communication, a number identifies a program of communication partners is called "Port number". The node that communicate through TCP/IP has IP address that corresponds to the address inside the network, but in order to communicate with more than one node at the same time, we use port number in the range from 0 to 65535 as auxiliary address.

2. 1. 6 Big endian and little endian

The way multibyte numerical data is stored in memory is called "Endian". "Big endian" refers to the way that store the most significant byte in the sequence. "Little endian" refers to the way that store the least significant byte in the sequence

It is determined that the header information is transmitted by "big endian" through TCP/IP.

2. 1. 7 Packet

The Unit of data transceiver is called "packet". The packet includes 2 kinds of information. One contains actual stored data (data area) and the other contains the information used to manage as the information of source or destination of that data, error checking information (header area).

2. 1. 8 Host and node

Host refers to the computer that communicates on the network . And the connection points in a network such as server, client, hub, router, access point etc. are called "node".

2. 1. 9 Address Resolution Protocol（ARP）

A protocol used to translate the physical address (MAC address) from logical address (In case of TCP/IP,that is IP address) is called “ARP”.

2. 1. 10 Internet Protocol（IP）

The protocol which executes the communication between nodes or node and gateway is called “IP (IP protocol)”. “IP (IP protocol)” is an very important protocol of the upper layer. The role of “IP” is to transfer data to the destination through the router based on the IP address without however ensuring their delivery, thus, ensuring the reliability of data is upper layer’s responsibility

“IP address” mentioned above is placed in the header of this “IP protocol”.

2. 1. 11 Internet Control Message Protocol（ICMP）

A protocol provides the function that is to notify errors occurred in IP network communication and verify the state of network status is called “ICMP”. There are echo request and echo reply messages are called Ping which most well-known.

2. 1. 12 Internet Group Management Protocol（IGMP）

The protocol executes IP Multicast is called “IGMP”. We can usually send the same data to many different hosts efficiently.

2. 1. 13 User Datagram Protocol（UDP）

A protocol provides the connectionless mode datagram communication service is called “UDP”. IP does not have interface with application. “UDP” is the protocol which helps to use that function from application. As a result, there is no way to notify that packets have arrived to the partner and the order of arrived packets may be changed so UDP does not esure the reliability of data .

2. 1. 14 Transmission Control Protocol（TCP）

A protocol which provides connection mode stream communication service is called “TCP”. “TCP” is known as upper layer of IP protocol, which provides a reliable communication as flow control, retransmission, error correction and sequence control.

2. 1. 15 Dynamic Host Configuration Protocol（DHCP）

When connecting to a network, a protocol which assigns automatically the necessary information such as IP address is called “DHCP”.　To use “DHCP”, we have to prepare DHCP server and on server side, it’s necessary to prepare some IP addresses for DHCP client in advance (Address pool).

2. 1. 16 Hyper Text Transfer Protocol（HTTP）

A protocol used to transfer the contents such as HTML file of homepage or website is called “HTTP”. “HTTP” not only can transfer HTML file but also can send binary data which are displayed on WEB browser such as JPEG, GIF, PNG, ZIP file.

2. 1. 17 File Transfer Protocol（FTP）

We call the protocol which transfer files between hosts is “FTP”.

2. 1. 18 Domain Name System（DNS）

A name resolution mechanism which can exchange host name into IP address or IP address into host name (domain) is called DNS. In case of using “DNS”, it is possible to look up the host name based on IP address or look up IP address from the host name.

2. 1. 19 Socket

An endpoint for communication which applications use for conmmunicating TCP/IP is called “socket”. “Socket” is constructed by IP address and port number. The applications, through specifying the socket to establish a connection, can transceive data without caring about any details of communication procedure. There are variety of sockets depending on the protocol used in communication side. TCP socket uses TCP protocol to communicate data and UDP socket uses UDP protocol to communicate data. In μNet3, we use ID number to identify the socket which becomes an operational objective. The application utilize ID number to invoke socket API.

2. 1. 20 Blocking and non-blocking

When calling some function,if it does not return until the acction has completed, that is called “Blocking mode” and if it returns immediately without waiting for its completion, that is called “Non-blocking mode”.

For instance, in the socket API of μNe3, the task calling the rcv_soc function in “Blocking mode” is placed in the waiting status until that action completes (until data can be received). Calling the rcv_soc function in “Non-blocking mode” will return immediately with an E_WBLK error code and the completion of that action (EV_RCV_SOC) is notified to callback function.

By default, the μNet3 sockets are in “Blocking mode”, and in order to switch to “Non-blocking mode”, we have to use cfg_soc function and set up registration of callback function and callback event flag.

2. 1. 21 Callback function

The function used for notifying the status of protocol stack to the apllications asynchronously is called “Callback function”.

2. 1. 22 Task context

All API functions of μNet3 must be called from Task context.

Do not call the system call which is in status of wating for tasks such as slp_tsk from network callback function. Besides, do not call all API functions of μNet3 from network callback function.

2. 1. 23 Resource

The resource used in a program is called “Resource”. There include tasks, semaphores are called “Kernel objects” and memory.

※ Please make reference to μC3 User Guide on details of “Kernel object” such as tasks, semaphores.

2. 1. 24 MTU

In communication network, MTU(Maximum Transfer Unit) is a value indicating the maximum amount of data that can be transferred by one-time transfer. Moreover, MTU also shows the maximum data size of the frame in data link layer. In addition, the minimum value indicated by MTU is 68 bytes.

Specifying the maximum data size depends on the protocol used in data link layer and in Ethernet interface gerenally uses 1500 bytes.

2. 1. 25 MSS

MSS (Maximum Segment Size) indicates the maximum data size of TCP packet. Therefore, MSS value can be calculated by the following formula.

MSS = MTU– (IP Header size+ TCP Header size (normally 40 bytes))

In case of Ethernet interface, the value of MSS is generally 1460 bytes.

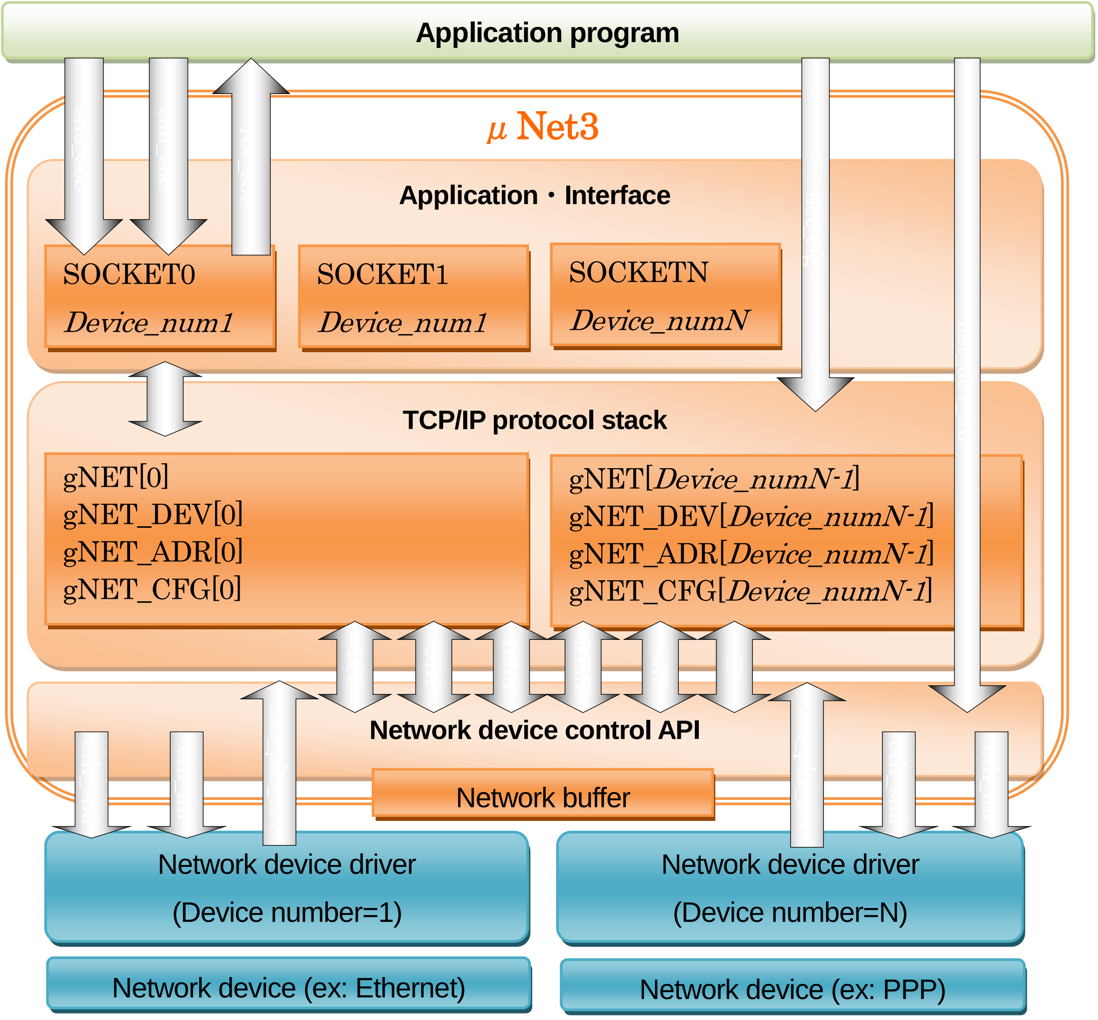
2. 1. 26 IP reassembly - fragment

The maximum size of an IP packet is 64K bytes. However, in order that MTU of communication interface becomes a smaller value than the original, it’s necessary that IP module must devide IP packet into smaller pieces to send. This processing is called “IP fragmentation” and divided IP packet is called “IP fragment”.

Moreover, IP module of receiver side needs to combine the divided “IP fragment” and we call this process is “IP reassembly”.

2. 2 Architecture of Network system

2. 2. 1 Block diagram of network system



Block diagram of network system

- **Application program**

  The user application program which is used for network communication. It includes application protocols such as DHCP, FTP, Telnet, HTTP etc.

- **Application Interface**

  Providing the interface (API) to utilize various network services such as transmission / reception of data or establishing connection to remote host.

  In case of normal application, we have to specify socket ID and device number before using Application Interface.

- **TCP/IP protocol stack**

  This program handles the network protocols such as TCP, UDP, ICMP, IGMP, IP and ARP.

- **Network device control API**

  In network system, maybe there exists various network devices. Every device needs a device driver. And the network device control API absorbs the difference between these devices, provides interface in order to access unifiedly. Using device number from application program before accessing to the devices.

- **Network device driver**

  A program that control network device. The content integration is different depending on the device.

  ※In μNet3, it is provided standard Ethernet and PPP driver device.

- **Network device**

  The hardware that execute the transmission and reception of actual nerwork data. This refers to Ethernet, PPP(RS- 232), WLAN etc.

- **Others**

  μNet3 using the below μC3 Kernel objects :

Object	**Object ID**	**Usage**
Task	ID_NET_MAIN_TSK	μNet3 startup task
Task	ID_TCP_TIM_TSK	μNet3 time management task
Task	ID_ETH_SND_TSK	Ether driver send task
Task	ID_ETH_RCV_TSK	Ether driver receive task
Task	ID_ETH_CTL_TSK	Ether driver control task
Semaphore	ID_TCP_SEM	μNet3 resource control semaphore
Event flag	ID_ETH_RCV_FLG	Ether driver event flag
Event flag	ID_ETH_SND_FLG	Ether driver event flag
Mailbox	ID_ETH_SND_MBX	Ether driver mail box
Mailbox	ID_ETH_RCV_MBX	Ether driver mail box
Fixed-length memory pool	ID_TCP_MPF	Network buffer area

2. 3 Directory and file organization

These files as below are included in μNet3.

Header file

/Network/Inc

net_cfg.h	Default configuration macro of TCP/IP protocol stack
net_hdr.h	Definition of the necessary informations to use TCP/IP protocol stack ※ **Please include this header file in source file of applications.**

Source file

/Network/Src

net_ini.c	Initialization module
net_buf.c	Network buffer management API
net_arp.c	ARP protocol module
net_ip4.c	IPv4 protocol module
net_icmp.c	ICMP protocol module
net_igmp.c	IGMPv2 protocol module
net_ipr.c	IP reassembly module
net_udp.c	UDP protocol module
net_tcp.c	TCP protocol module
net_tim.c	Protocol stack Timer module
net_soc.c	Socket API
net_dev	Device driver interface

Library files

This folder stores the library that has already built the TCP/IP protocol stack in various processor mode, and the project files are used to build.

/Network/Lib/<**CPU**>

uNet3**xxxx**_b_.a

uNet3**xxxx**_l_.a

CPU depends on architecture name of CPU

xxxx depends on processor mode or processor name of CPU.

'_b_' expresses big endian, '_l_' expresses little endian.

Application protocol source file

/Network/NetApp

dhcp_client.h	DHCP client macro, prototype, definition etc.
dhcp_client.c	DHCP client source code
ftp_server.h	FTP server macro, prototype, definition etc.
ftp_server.c	FTP sercer source code
http_server.h	HTTP server macro, prototype, definition etc.
http_server.c	HTTP server source code
dns_client.h	DNS client macro, prototype, definition etc.
dns_client.c	DNS client source code
ping_client.h	ICMP echo request macro, prototype, definition etc.
ping_client.c	ICMP echo request (ping) source code.
sntp_client.h	SNTP client macro, prototype, definition etc.
sntp_client.c	SNTP client macro source code.
net_strlib.h	String library function definition.
net_strlib.c	String library function source code.

/Network/NetApp/ext

dhcp_client.h	DHCP client extended version macro, prototype, definition etc.
dhcp_client.c	DHCP extended version client source code

Sample source file

/Network/sample

DDR_TEMPLATE_NET.c	μNet3 network device driver template code
DDR_LOOPBACK_NET.c	Loopback device driver.

Chapter 3: Overview functions ofμNet3

3. 1 Protocol stack

3. 1. 1 IP module

The IP module only receives and handles the arrived packets which has destination IP address matches with the IP address of local host.　Other packets are not handled.

IP Option

μNet3 supports router warning option of internal IGMP in IP option only. IP options which do not support will be ignored. .

TTL（Time to Live）

Default value of TTL in μNet3 is set DEF_IP4_TTL(64）. This value may be changed by using net_cfg(). In case of using net_cfg() to change the value of TTL, TTL value of all sockets are changed. In case that we want to change TTL value of each socket, please use cfg_soc().

TOS（Type Of Service）

In μNet3, TOS is set DEF_IP4_TOS (0).

Broadcast

Maybe receive broadcast or not depending on using net_cfg(). The initial value is set that ready to receive. Always can transmit broadcast. The broadcast setting is effective for all sockets but we can not set up whether receive broadcast by socket unit .

Regarding to transceive broadcast, please use UDP socket.

Multicast

In order to allow multicast reception, we use net_cfg() and register at the address of the multicast group which join to. Multicast group address may be registered by DEF_NET_MGR_MAX (8）.

Always can send multicast. The multicast setting is effective for all sockets but we can not set up whether receive multicast by socket unit.

TTL used for transffering multicast is set DEF_IP4_MCAST_TTL(１). This value can also be changed by using net_cfg().

Do not support multicast loopback.

Regarding to transceive of multicast, please use UDP socket.

MTU

In μNet3, DEF_PATH_MTU (1500 byte) is set as default value of MTU. This value can be configured by the configurator.

IP reassembly / fragment

In uNet3, maximum size of IP packet is 1500 byte as default (This value is related to the value of network buffer). In order to increase the size of IP packet to maximum,we need to enlarge the network buffer. For example, in case that transceive 2048byte of UDP data, we need to increase the value of network buffer larger than the value is canculated from this formula (control header size (100 bytes) + IP header size (20bytes) + UDP header size (8bytes) + 2048).

The default value of IP reassembly process timeout is DEF_IP4_IPR_TMO(10 seconds). If the reassembly process can not complete within this timeout, the reassembly process is cancelled, the ICMP error message (type 11: packet discarded by time excess) is sent to remote host.

The default number of times of the IP reassembly process is set DEF_NET_IPR_MAX(2). DEF_NET_IPR_MAX value expresses a value which host can execute IP reassembly process at the same time.

IGMP

In μNet3, the timeout until the “report (reply)” message is sent to “query (group inquiry)” (from router) is set by DEF_IGMP_REP_TMO (10 seconds)

μNet3 supports IGMPv2 and also supports IGMPv1 compatible function.

In case of getting query of IGMPv1, it will be changed into IGMPv1 mode and then processed. After that, within a certain time period, if there is no IGMPv1 message, it will be back to IGMPv2 mode.Timeout for returning from IGMPv 1 to IGMPv 2 is set by DEF_IGMP_V1_TMO (400 seconds).

ICMP

μNet3 supports messages of “echo response”, “echo request”, “time excess”.

3．1．2　　ARP module

（1） Resolve ip address

μNet3 will manage the mapping of IP address of host and physical address (MAC address). The administration table (conversion table) of this mapping is called ARP cache. ARP cache size is set by DEF_NET_ARP_MAX (8).

When sending IP packet to network, in case that there exists a compatible IP address which refers to ARP cache, it will send a packet to the destination that is the physical address has been recorded there. In case that there is no existing IP address, IP packet will be stored temporarily in queue, then, send broadcast ARP request packets. After receiving ARP response packets from remote host, record a received physical address in ARP cache newly. Then, remove IP packet from queue, send the packet to the newly acquired physical address.

Besides, ARP entry information is held in the cache table for a maximum of ARP_CLR_TMO (20 minutes）.

（2） Address conflict detection

According RFC5227, μNet3 will check whether Ipv4 address is non-duplicative in the same link. This feature is performed by API is called from application, when LAN interface boot up or link status changes.

After setting the IP address of the interface, the other host had set the same IP address, then the detected the conflict, μNet3 will notify the application.

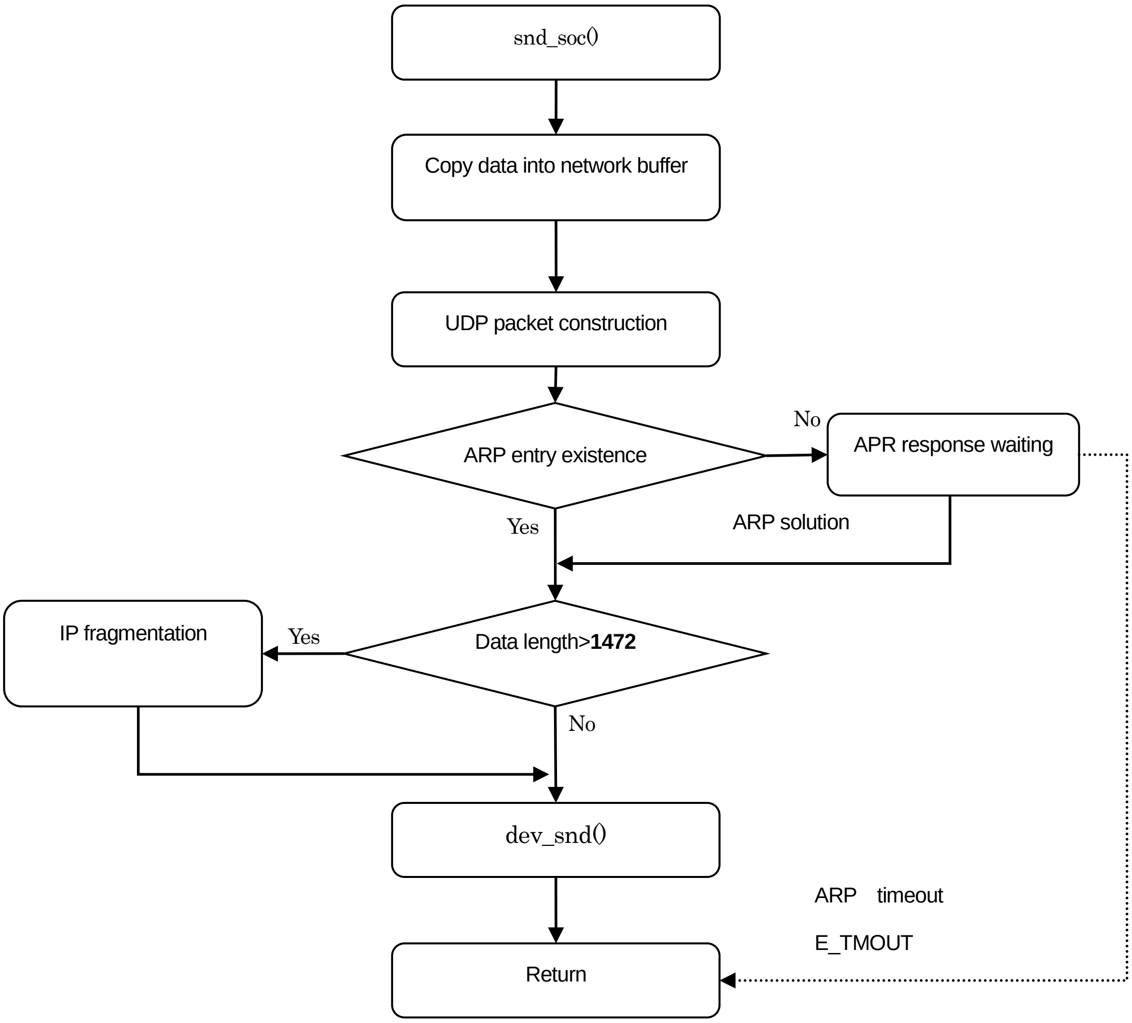
"ARP Probe" can detect whether the IP address that you will use is not already in use. Other host did not respond to **"ARP Probe"**(IP address conflict is not), μNet3 notify the other hosts that to use this IP address from now by sending the **"ARP Announce".**

3．1．3　　UDP module

UDP executes the transceiver of data without connecting to remote host.

（1） Sending data

Before sending data, we should use con_soc and accosiate a socket with a source address (IP address, port number). After that, we use snd_soc() to send data. The flow snd_soc（） processing is described in the diagram as below.



The flow of snd_soc processing of UDP socket

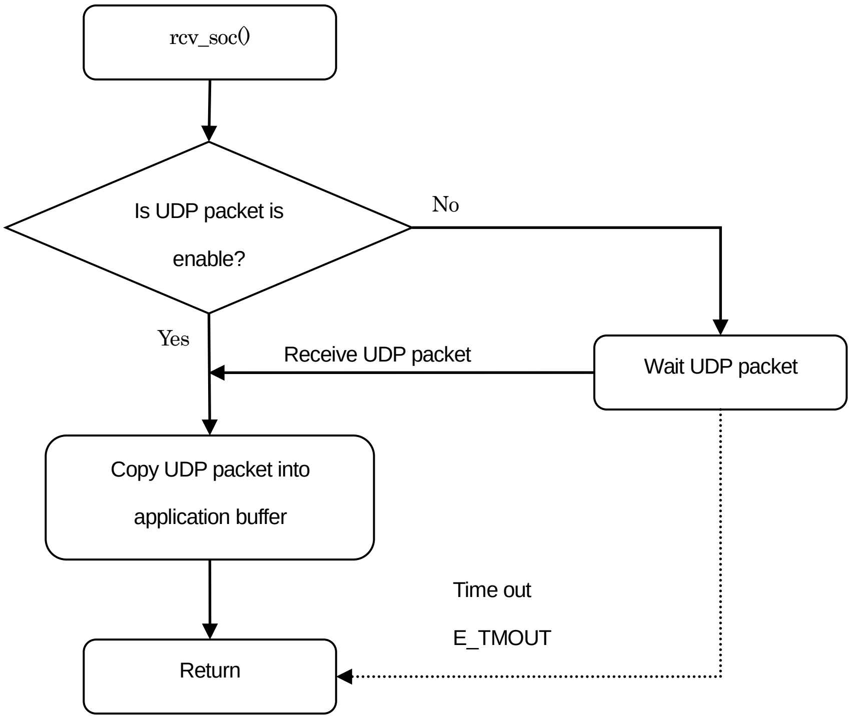
① The application data is copied into network buffer, adding UDP header such as port number,IP address of remote host,then construct UDP packet.

② In case that cannot resolve MAC address of remote host by ARP protocol, it will return E_TMOUT error.

③ In default, the maximum size of transmission data is set 1472 bytes (DEF_PATH_MTU (1500 bytes) – IP header size– UDP header size). In case of sending data with larger size than this, we need to set network buffer size. Regarding the details, please refer to the item of IP reassembly/fragment

（2）Data reception

Data reception is executed by using rcv_soc(). The flow of rcv_soc （） processing is described in the diagram as below.



The flow of rcv_soc processing of UDP socket

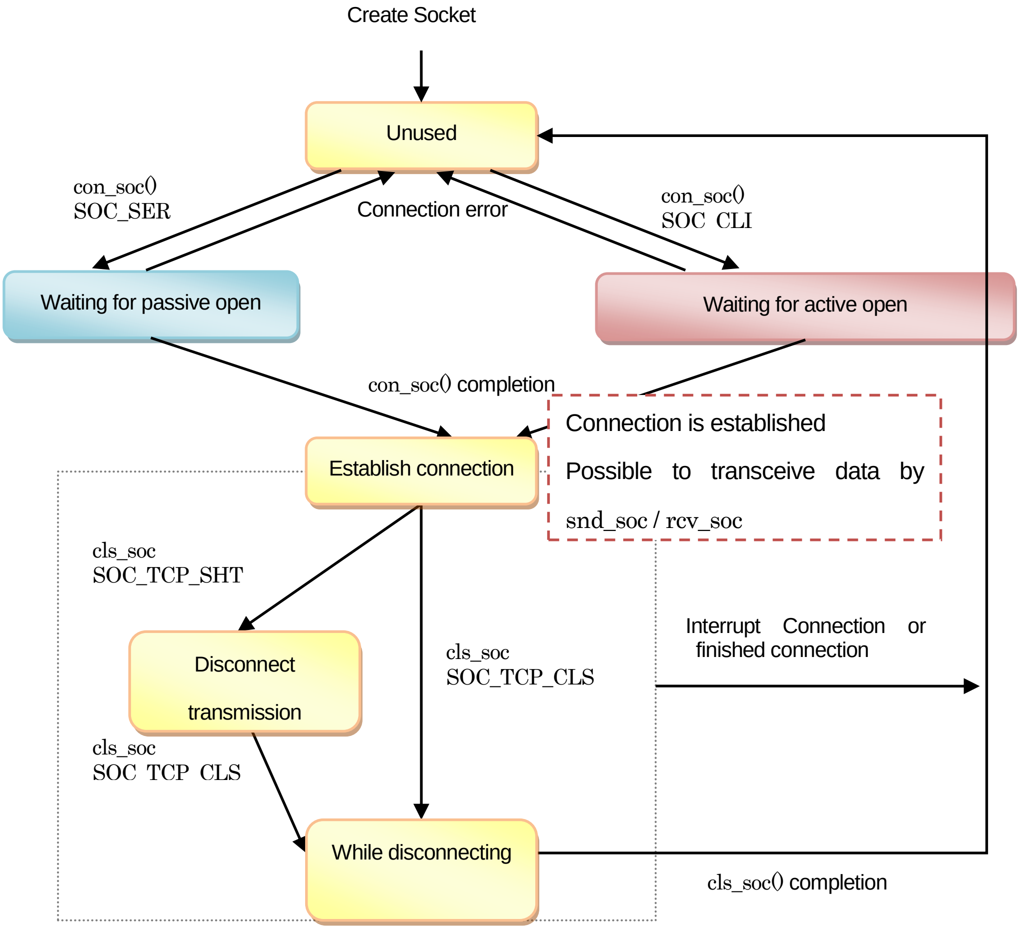
① If UDP packet has not been received yet, enter a state of wating for UPD packet reception. At that time, if it exceeds timeout of receiving socket, it will return E_TMOUT.

② If received packet size is smaller than requested data size, copy into application buffer. In case that received packet with bigger size, just copy the request size into application buffer. Remaining part will be ignored.

③ In default, maximum size of reception data is set 1472 bytes (DEF_PATH_MTU (1500 bytes) – IP header size– UDP header size). In case of receiving data with larger size than this, we need to set network buffer size. Regarding the details, please refer to the item of IP reassembly/fragment.

3. 1. 4 TCP module

TCP is different from UDP. TCP is connection mode, so it can allocate sending party and channels before transceiving data. TCP sequence is described in the diagram as below.



TCP sequence

(1) Establishing connection

There are two modes of TCP connection, active and passive connection. Active connection that requires connect to remote host by itself. On the contrary, passive connection that wait for the connection from remote host.

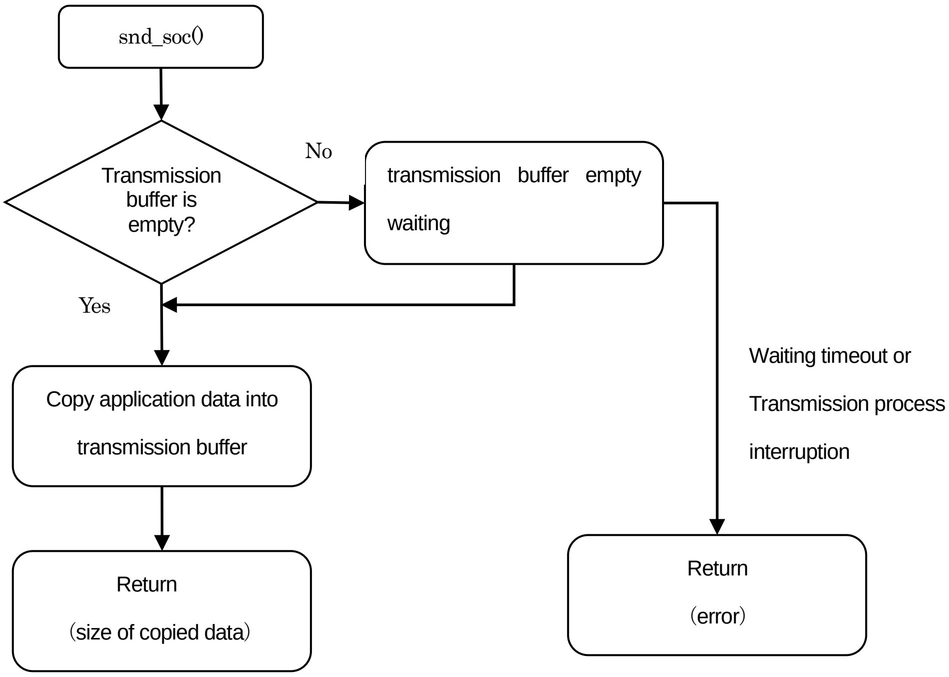
Use con_soc() to connect, and need to specify active connection by SOC_CLI and passive connection by SOC_SER.

(2) Connection completion

In order to disconnect the connection, we use cls_soc(). Specify SOC_TCP_CLS in order to disconnect all the connection completely,and SOC_TCP_SHT to dissconect the transmit direction only .

(3) Data transmission

Use snd_soc() to send data.The flow of snd_soc () processing is described as below.


snd_soc()
Transmission buffer is empty?
No
transmission buffer empty waiting
Yes
Copy application data into transmission buffer
Waiting timeout or Transmission process interruption
Return (size of copied data)
Return (error)

TCP socket–Flow of snd_soc processing

① Copy data of application into TCP transmission buffer. If copy is successful, TCP protocol will send data. If remote host received data, all data in TCP transmission buffer will be clear.

TCP transmission buffer

It is necessary to specify transmission buffer size when create TCP socket. Buffer size has a range from 4 bytes to 32 kilobytes and aligns to 2 power 2.

(4) Data reception

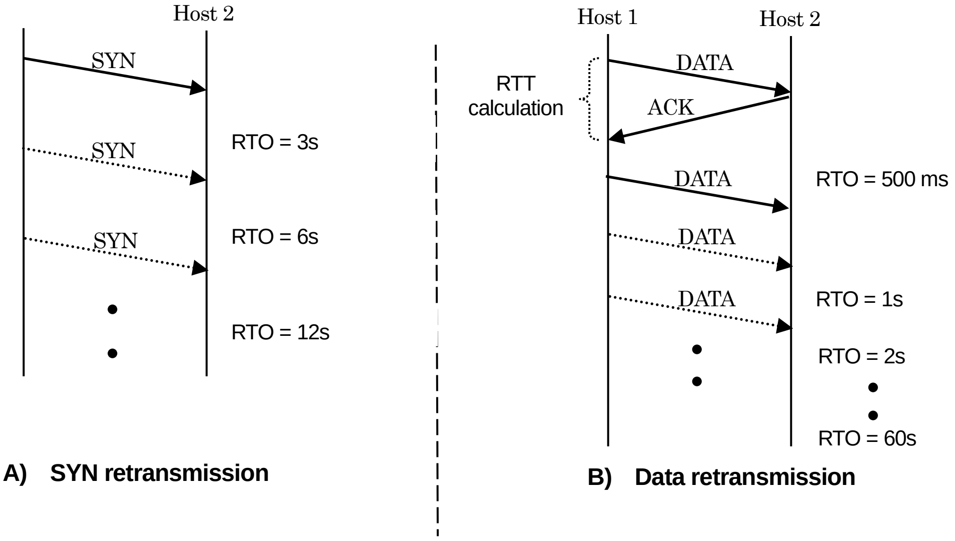
Use rcv_soc() to send data. Received TCP packet firstly will be registered at TCP reception buffer. When rcv_soc() is called, it will be copied from TCP reception buffer into application buffer.

TCP reception buffer (Window buffer)

It is necessary to specify reception buffer size when create TCP socket. Buffer size has a range from 4 bytes to 32 kilobytes and aligns to 2 power 2.

(5) Retransmission timeout

Timer sequence of resending is described in the diagram as below.



A) SYN retransmission

B) Data retransmission

An example of retransmission timer

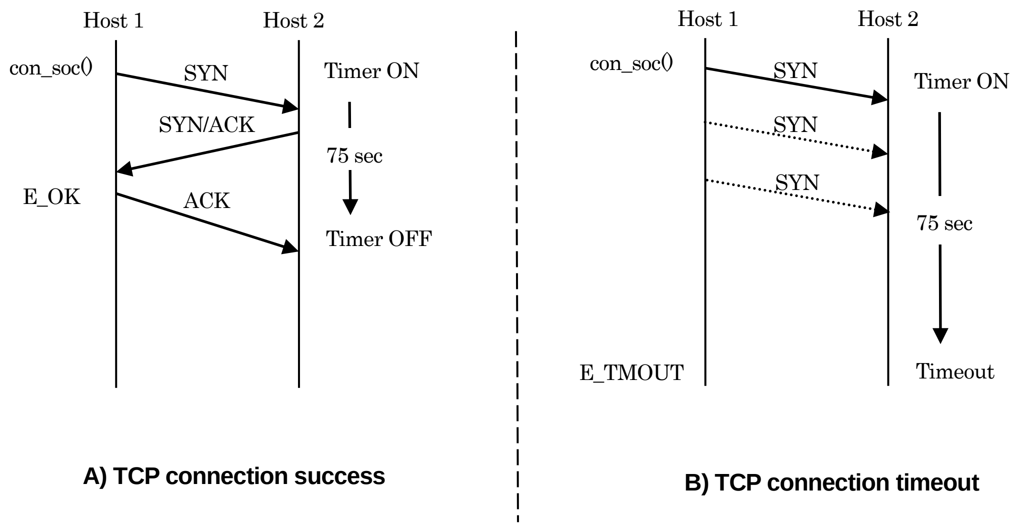
In TCP, if there are not response of ACK packet within a certain time for any reason,segment without response will be sent again. The waiting time until retransmission action is executed is called “RTO” (Retransmission Time Out）. Initial value of RTO is called “RTT” (Round Trip Time）, is “4 times＋α” of “Time that packet makes round trip to the other). RTO value is increased twice everytime resending action is done.

When restransmit SYN like the above A diagram, it uses DEF_TCP_RTO_INI (3 seconds) due to RTT value is not set. In the above B diagram of data retransmission, it calculates RTT value based on the previous successful transmission, that’s 500 milliseconds.

RTO scope is set from DEF_TCP_RTO_MIN (500 ms) to DEF_TCP_RTO_MAX (60 s).

（6）Connection timeout

Connection timer sequence is described in the below diagram.



An example of connection timeout

When call con_soc(), if this timer completes from starting up to three-way handshake timed out, it will return E_OK (**A**）. If finish timeout, it will return E_TMOUT（**B**）.

Timeout value of connection process (3-way handshake) is set DEF_TCP_CON_TMO (75 seconds）.

※When create TCP socket, it can specify blocking timeout used in connection. If this value runs out of time, connection process will be interrupted immediately and con_soc() will return E_TMOUT.

（7）Transmission timeout

Transmission timeout is set DEF_TCP_SND_TMO (64 seconds). While communicating data, if there is no response from the partner even though passes DEF_TCP_SND_TMO,the connection will be disconnected.

（8）Disconnection timeout

Timeout of disconnection process is set DEF_TCP_CLS_TMO (64 seconds). If cls_soc() does not complete at DEF_TCP_CLS_TMO, connection will be forcibly disconnected and cls_soc() will return E_TMOUT.

※When create TCP socket, it can specify blocking timeout used in connection. If this value runs out of time, connection process will be interrupted immediately and cls_soc() will return E_TMOUT.

（9）Delay ACK timeout

Delay ACK timeout is set DEF_TCP_ACK_TMO (200 milliseconds).

（10）TCP congestion control

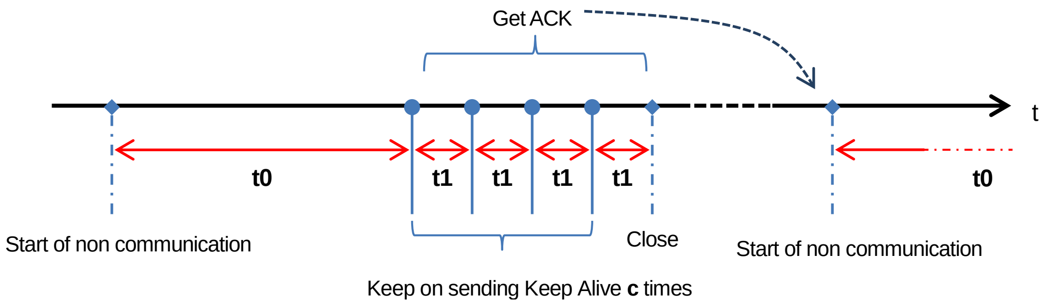
μNet3 supports fast retransmit and fast recovery. Number of duplicate ACK is set DEF_TCP_DUP_CNT (4).

（11）Maximum Segment Size（MSS）

MSS is set DEF_TCP_MSS (1460 bytes）.

（12）Keep Alive

μNet3 supports TCP Keep Ailve.


Get ACK
t
t0
t1
t1
t1
t1
t0
Start of non communication
Close
Start of non communication
Keep on sending Keep Alive **c** times

t0: The time of activation Keep-Avlie
t1: The interval of transmission Keep-Avlie
c: The number of transmission Keep-Avlie

Operation TCP Keep Alive

If Keep Alive feature is enabled (c > 0), after t0 seconds in non-communication state, start the trainsmission of Keep Alive packet to the destination host.

Until it is transmitted c times, or get ACK from destination, μNet3 will continue to send the Keep Alive packets at intervals of t1 seconds.

If no response is obtained in c times Keep Alive packet, then μNet3 close TCP connection.

It does not disconnect TCP connection automatically if Keep Alive feature is disable (c = 0)

3．2　Network device driver

In μNet3, it provides common interface of device driver which can correspond to all kinds of network device driver. Device driver is created in accordance with this interface may be used in μNet3.

Concretely, because protocol stack will access to device driver through **T_NET_DEV structure**, so the informations as device name, device number, the functions of device driver must be registered in **T_NET_DEV structure** in advance. Protocol stack specify and access to the device which is registered in **T_NET_DEV** by device number.

Regarding details of network device driver, please refer to “uNet3Ethernet driver interface” guide

3．2．1　Device structure

```
typedef struct t_net_dev {
        UB          name[8];   /* Device name*/
        UH          num;       /* Device number */
        UH          type;      /* Device type */
        UH          sts;       /*Reserve*/
        UH          flg;       /*Reserve */
        FP          ini;       /*Pointer to dev_ini function*/
        FP          cls;       /* Pointer to dev_cls function*/
        FP          ctl;       /* Pointer to dev_ctl function*/
        FP          ref;       /* Pointer to dev_ref function*/
        FP          out;       /* Pointer to dev_snd fucntion*/
        FP          cbk;       /* Pointer to dev_cbk fucntion*/
        UW          *tag;      /* Reserve*/
        union       cfg;       /* MAC address */
        UH          hhdrsz;    /* Device header size */
        UH          hhdrofs;   /*Position writing network buffer*/
} T_NET_DEV;
```

（1）Device number

Set unique number to specify device. Protocol stack will use this number to access to the device . **Device number should be numbered consecutively from 1.**

（2）Device name

Set the name in order to specify the device. The length of device name should be under 8 bytes long.

For example: eth0, eth1 etc.

(3) Device type

Set type of network device. There are some as below.

Device type	Meaning
NET_DEV_TYPE_ETH	Ethernet device
NET_DEV_TYPE_PPP	PPP device

(4) Function of driver device

Device driver needs to support the below functions. These functions are called from appropriate protocol stack.

Prototype	Description	Requirement
ER dev_ini(UH dev_num)	Device initialization	Require
ER dev_cls(UH dev_num)	Device release	No require
ER dev_snd(UH dev_num, T_NET_BUF *pkt)	Send packet to network	Require
ER dev_ctl(UH dev_num, UH opt, VP val)	Device control	No require
ER dev_ref(UH dev_num, UH opt, VP val)	Device status acquisition	No require
void dev_cbk(UH dev_num, UH opt, VP val)	Notify event from device (callback function)	No require

(5) MAC address

Set unique value to specify hardware.

```
union {
        struct {
             UB    mac[6];   /* MAC address */
        }eth;
} cfg;
```

3．2．2　　Interface

dev_ini　　　　　Device initialization

【Format】

ER ercd =dev_ini(UH dev_num);

【Parameter】

UH	dev_num	Device number

【Return value】

ER	ercd	Successful completion（E_OK）or error code

【Error code】

E_ID	Device number is wrong
E_OBJ	Already initialized
E_PAR	Illegal value set in T_NET_DEV
<0	Other errors (implementation dependent)

【Explanation】

Initialize device. This function is called to initialize device from protocol stack. Before calling this function,it is necessary to register device information in T_NET_DEV.

dev_cls Device release

【Format】

ER ercd =dev_cls(UH dev_num);

【Parameter】

UH	dev_num	Device number

【Return value】

ER	ercd	Successful completion (E_OK) or error code

【Error code】

E_ID	Device number is wrong
E_OBJ	Already released

【Explanation】

Release device.

dev_ctl Device control

【Format】

```
ER ercd = dev_ctl(UH dev_num, UH opt, VP val);
```

【Parameter】

UH	dev_num	Device number
UH	opt	Control code
VP	val	Value to be set

【Return value】

ER	ercd	Successful completion (E_OK） or error code

【Error code】

E_ID	Device number is wrong
E_PAR	Illegal parameter
E_OBJ	Already released

【Explanation】

The operation of this function is implementation dependent.

dev_ref Device status acquisition

【Format】

ER ercd = dev_ref(UH dev_num, UH opt, VP val);

【Parameter】

UH	dev_num	Device number
UH	opt	Status code
VP	val	Acquire value

【Return value】

ER	ercd	Successful completion (E_OK）or error code

【Eror code】

E_ID	Device number is wrong
E_PAR	Illegal parameter
E_OBJ	Already released

【Expalnation】

The operation of this function is implementation dependent.

dev_snd Packet transmission

【Format】

```
ER ercd =dev_snd(UH dev_num, T_NET_BUF *pkt);
```

【Parameter】

UH	dev_num	Device number
T_NET_BUF	*pkt	Pointer to network buffer

【Return value】

ER	ercd	Successful completion (E_OK） or error code

【Error code】

E_WBLK	Packet is registered in queue (not error)
E_ID	Device number is wrong
E_PAR	Illegal parameter
E_TMOUT	Packet transmission timed out
E_OBJ	Device status was wrong already

【Explanation】

This function transmits packet to Ethernet.

An example of integration

```
ER dev_snd(UH dev_num, T_NET_BUF *pkt)
{
    /* Copy to Ethernet frame (IP/TCP/UDP/Payload) */
    memcpy(txframe,   pkt->hdr,   pkt->hdr_len);
    /* Transmit to network */
    xmit_frame(txframe);
    return E_OK;
}
```

In the above example, the process of protocol stack is blocked by device driver. The next example shows the example that using queue and no blocking.

Non-blocking example

```
ER dev_snd(UH dev_num, T_NET_BUF *pkt)
{
    queue_tx(pkt)；        /* register packet in queue */
    return E_WBLK；  /* Non-blocking */
}


void queue_tx_task(void)
{
    dequeue_tx(pkt); /* Removing packet from queue */
    /* Copy to Ethernet frame (IP/TCP/UDP/Payload) */
    memcpy(txframe,   pkt->hdr,   pkt->hdr_len);
    xmit_frame(txframe)；  /* Transmit to network */
    if (transmission timeout) {
        pkt->ercd = E_TMOUT；    /* Set time out */
    }
    net_buf_ret(pkt);
}
```

In dev_snd transmission process is not executed, packet will register in queue and return E_WBLK. Actual packet transmission process is excuted by another task and release of network buffer is also executed there too.

dev_cbk Device event notification

【Format】

```
void dev_cbk(UH dev_num, UH opt, VP val);
```

【Parameter】

UH	dev_num	Device number
UH	opt	Event code
UH	val	Event value

【Return value】

None

【Error code】

None

【Explanation】

This function is to notify an event to the application from device driver. This function is implementation dependent.

3. 2. 3 Packet routing

To send a packet to the upper protocol stack from device driver, it uses the following API.

※This API can not be used from Applications.

net_pkt_rcv **Sending packet to protocol stack**

【Format】

void net_pkt_rcv(T_NET_BUF *pkt);

【Parameter】

T_NET_BUF	*pkt	Pointer to network buffer

【Return value】

no

【Error code】

None

【Explanation】

This function is to send packet to the upper protocol. The below example shows the example for sending packet to upper protocol stack from device driver.

Example

```
/* Network buffer allocation */
T_NET_BUF *pkt;
net_buf_get(&pkt, len, TMO);

/* Set received Ethernet header to network buffer */
pkt->hdr = pkt->buf + 2;
pkt->hdr_len = ETH_HDR_SZ;
memcpy(pkt->hdr, rx_frame, pkt->hdr_len);

/* Set received IP payload to network buffer */
pkt->dat= pkt->hdr + pkt->hdr_len;
pkt->dat_len = rx_frame_len – pkt->hdr_len;
memcpy(pkt->dat, rx_frame + pkt->hdr_len, pkt->dat_len);

/* Device information setting*/
```

```
pkt->dev = dev;

/* Transfer network buffer to protocol stack */
net_pkt_rcv(pkt);
```

Release of network buffer is executed by net_pkt_rcv(). net_pkt_rcv() must be called from task context.

3. 2. 4 Loopback Interface

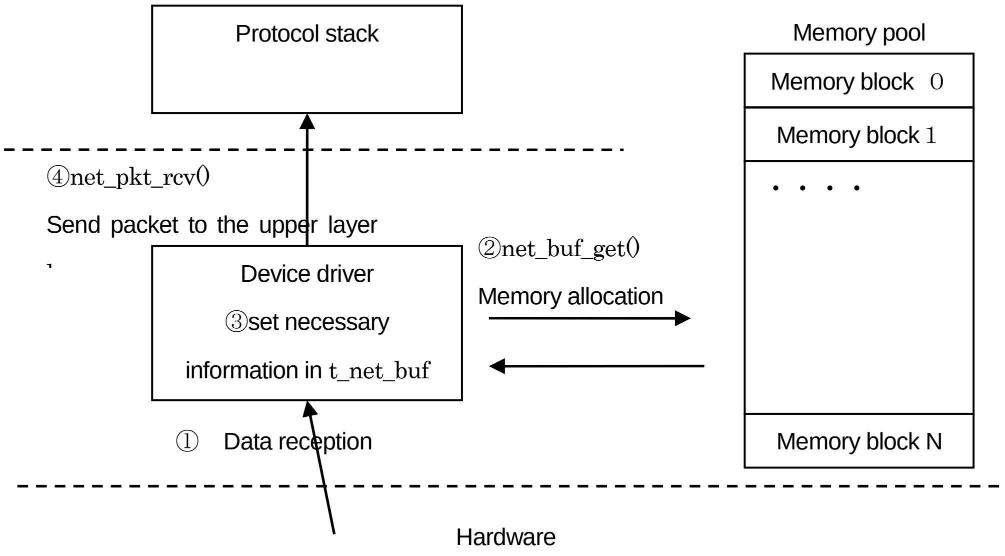
μNet3 provides a loopback interface to fold back in device driver layer. By using DDR_LOOPBACK_NET.c, a packet sent from interface comes back to μNet3.

Unlike general loopback interface represented 127.0.0.1, static MAC address and IP address are set in μNet3. In order to use loopback interface, choose "Loopback" as device type when you register interface with uNet3/Compact configurator.

To receive a packet from the loopback interface, packet's destination must be IP address assigned to the loopback interface. ARP performed even if for using loopback interface.

3．3　Memory management

In protocol stack, it uses network buffer in memory management. By using network buffer, it can allocate the empty block of memory actively. The following diagram shows an example of memory allocation. First, device driver which receives data from hardware will use API network buffer, then allocate memory (net_buf_get). Next, it will set necessary information in allocated memory and then send packet to the upper layer protocol stack (net_pkt_rcv).



Memory allocation diagram

3. 3. 1 Network buffer

In μNet3/Compact, block size uses maximum 8 memory pool with 1472byte fixed length. Network buffer provides the mechanism that allocate or free memory from this memory pool .

Network buffer organization（T_NET_BUF）

```
typedef struct t_net_buf {
            UW              *next;      /*Reserve */
            ID              mpfid;      /* ID memory pool */
            T_NET           *net;       /* Network interface */
            T_NET_DEV       *dev;       /* Network device */
            T_NET_SOC       *soc;       /* Socket */
            ER              ercd;       /* Error code */
            UH              flg;        /* Flag used to control protocol stack */
            UH              seq;        /* Fragment sequence*/
            UH              dat_len;    /* Data size of packet */
            UH              hdr_len;    /* Header size of packet */
            UB              *dat;       /* Showing data position in packet (buf) */
            UB              *hdr;       /* Showing header position in packet (buf) */
            UB              buf[];      /* Actual packet*/
} T_NET_BUF ;
```

The uNet3 uses **T_NET_BUF** to transceive packet between protocol and device driver or the protocols.

In TCP/IP, the actual packet datas are stored in “**buf” ,‘*dat’, ‘*hdr’,’hdr_len**’,‘**dat_len’** are used to access to that.

3. 3. 2　API network buffer

※This API network buffer can not be used from application.

net_buf_get　　　Network buffer allocation

【Format】

```
ER ercd =net_buf_get(T_NET_BUF **buf, UH len, TMO tmo);
```

【Parameter】

T_NET_BUF	**buf	Address of buffer that allocate memory
UH	len	Number of allocating bytes
UH	tmo	Timeout specification

【Return value】

ER	ercd	Successful　completion (E_OK）　or error code

【Error code】

E_PAR	Set wrong parameter value
E_NOMEM	Unable to allocate memory
E_TMOUT	Timeout

【Explanation】

Allocate memory from memory pool. Allocated buffer address returns to buf.

net_buf_ret **Network buffer release**

【Format】

voidnet_buf_ret(T_NET_BUF *buf) ;

【Parameter】

T_NET_BUF	*buf	Address of buffer that free memory

【Return value】

None

【Error code】

None

【Explanation】

Give back memory to the memory pool. If the socket is associated with network buffer, notify the free memory event to the socket.

3．4　Memory processing I / O

Comparison and writing process of contiguous memory occurred in the protocol are able to defined by user, so that it does not depend on device or compilation environment.

For devices with features DMA, Processing memory copy can use the DMA transfer instead of the memcpy() of standard library.

(※ This function is limited to version 2.0 or later of the μNet3.)

3．4．1　Memory processing I / O

※memory I/O API must be defined in application always. (since μNet3 ver 2.0)

net_memset　　Fill block of memory

【Format】

VP net_memset(VP d, int c, UINT n);

【Parameter】

VP	d	Pointer to the block of memory to fill
int	c	Value to be set
UINT	n	Number of bytes to be set

【Return value】

VP	d	Pointer to the block of memory to fill

【Explanation】

If the memory settings are successful, please return the destination pointer that is specified in the argument.

net_memcpy Copy bytes in memory

【Format】

VP net_memcpy(VP d, VP s, UINT n);

【Parameter】

VP	d	Pointer to the destination of memory
VP	s	Pointer to the source of data
UINT	n	Number of bytes to copy

【Return value】

VP	d	Pointer to the destination of memory

【Explanation】

If the memory copies are successful, please return the destination pointer that is specified in the argument.

net_memcmp Compare two blocks of memory

【Format】

int net_memcmp(VP d, VP s, UINT n);

【Parameter】

VP	d	pointer to blocks of memory1
VP	s	pointer to blocks of memory2
UINT	n	Number of bytes to compare

【Return value】

int		Comparison result

【Explanation】

Please return 0 if the same value in the specified number of bytes. Otherwise, please return the non-zero.

Chapter 4: Configuration

4. 1 Configuration of μNet3/Compact for Version 1.xx

In case of using μNet3/Compact, input the configuration information which becomes parameter of the objects are determined in system design into the configurator and enable to generate the skeleton code which is the template of necessary source file and application program.

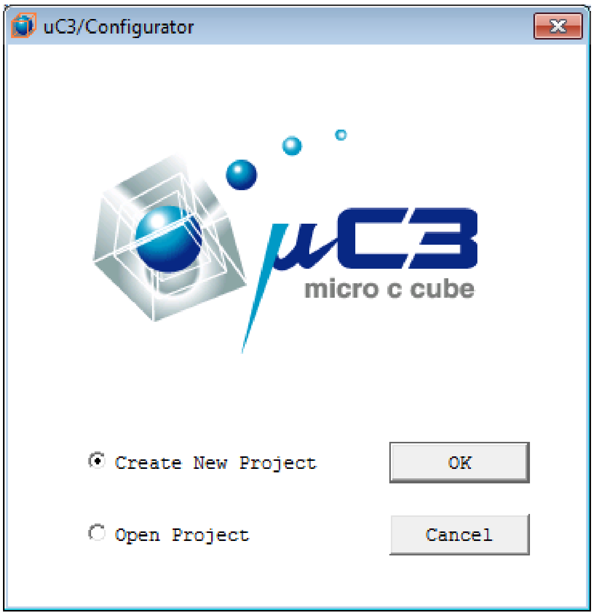
This chapter explains the configuration of TCP/IP protocol stack. Regarding to the configuration related to kernel and device, please refer to "μC3/Compact Users Guide".

4. 1. 1 Starting up configurator

Double click on "uC3conf.exe" to start up

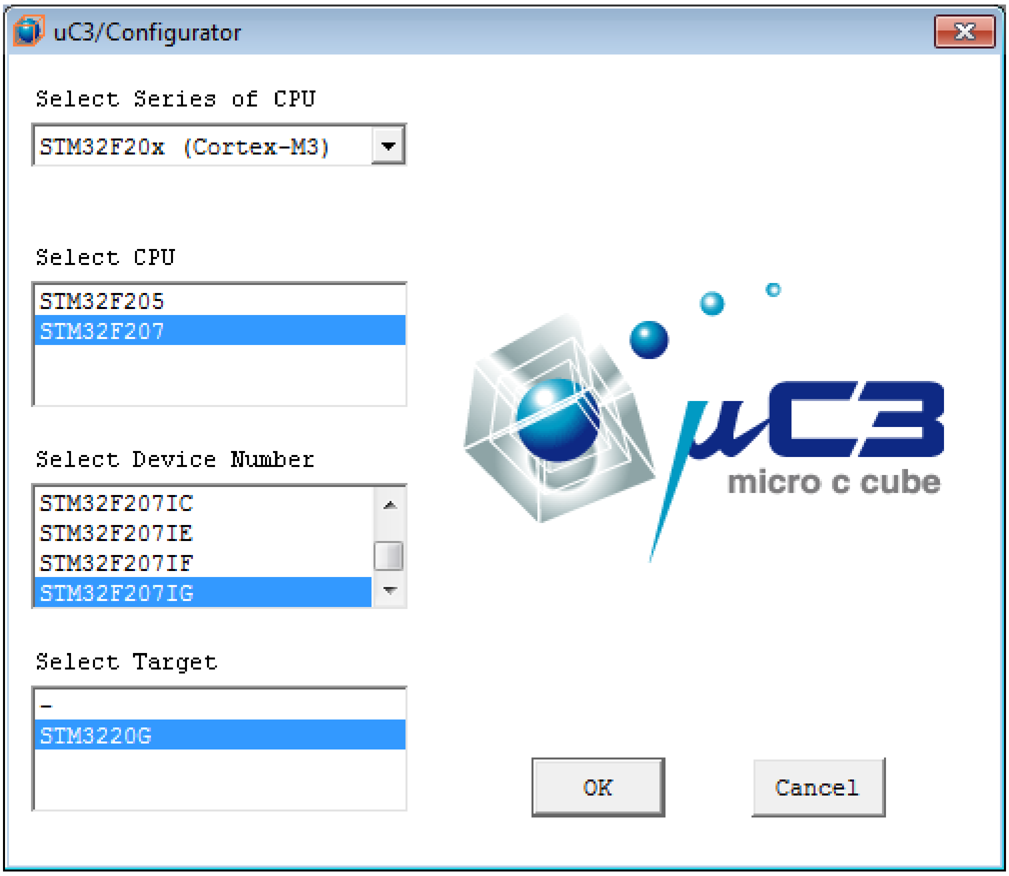
A. In case of create a new project

After choosing "Create new project", click "OK" and then go to"Select CPU ".



CPU selection

After selecting a series and part number of CPU from the list, click “OK” and go to “Main screen”.



uC3/Configurator

Select Series of CPU: STM32F20x (Cortex-M3)

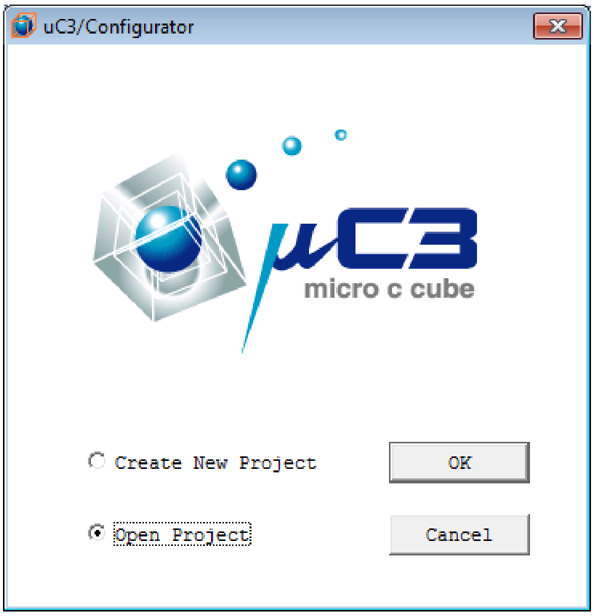
Select CPU: STM32F205, STM32F207 (selected)

Select Device Number: STM32F207IC, STM32F207IE, STM32F207IF, STM32F207IG (selected)

Select Target: -, STM3220G (selected)

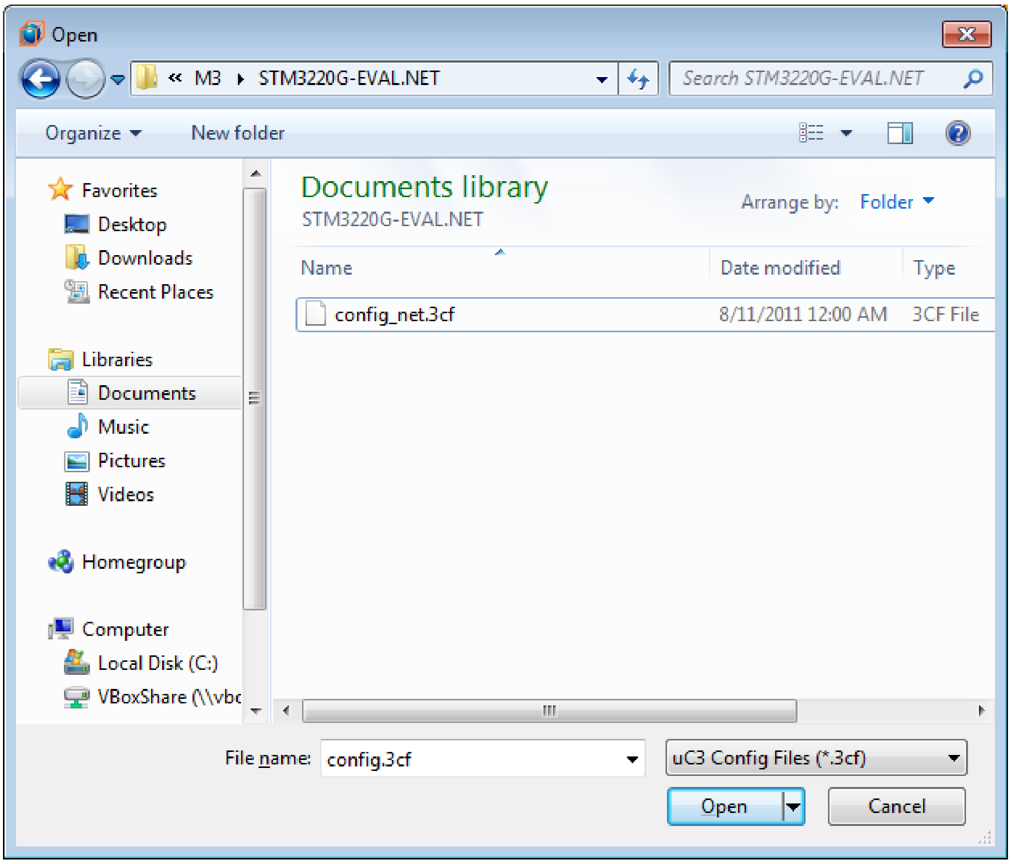
B. In case of opening the existing project

After choosing “Open existing project”, click “OK” and go to “Open file”.



Open file

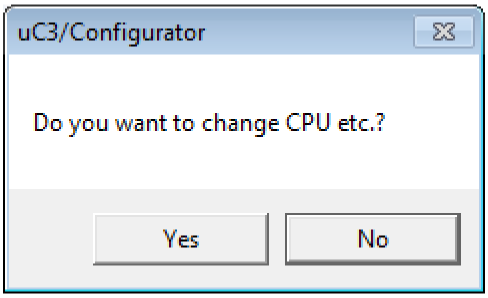
After choosing the saved project file (file extension.3cf) , click “Open” and go to “Main screen”.



Name	Date modified	Type
config_net.3cf	8/11/2011 12:00 AM	3CF File

File name: config.3cf

uC3 Config Files (*.3cf)



uC3/Configurator

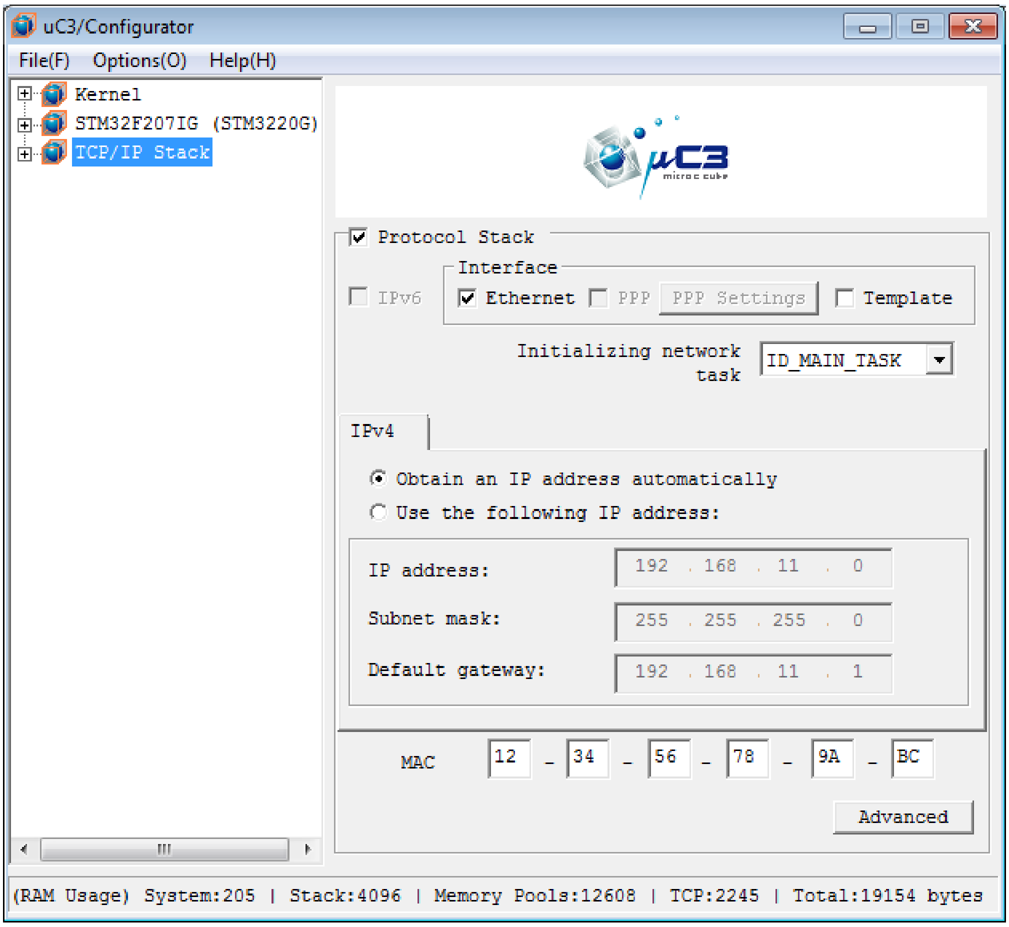
Do you want to change CPU etc.?

When the dialog of "Do you change CPU?" appears on screen, click "No".

C. Main screen

After starting up, the main screen is able to browse and edit the project. It can switch the configuration screen of each object by clicking on the object name in the tree view.

Here, there are configuration of dependent parts of processor, TCP/IP protocol stack and kernel,.



☑ Protocol Stack

Interface

□ IPv6 ☑ Ethernet □ PPP PPP Settings □ Template

Initializing network task: ID_MAIN_TASK

IPv4

● Obtain an IP address automatically

○ Use the following IP address:

IP address: 192 . 168 . 11 . 0

Subnet mask: 255 . 255 . 255 . 0

Default gateway: 192 . 168 . 11 . 1

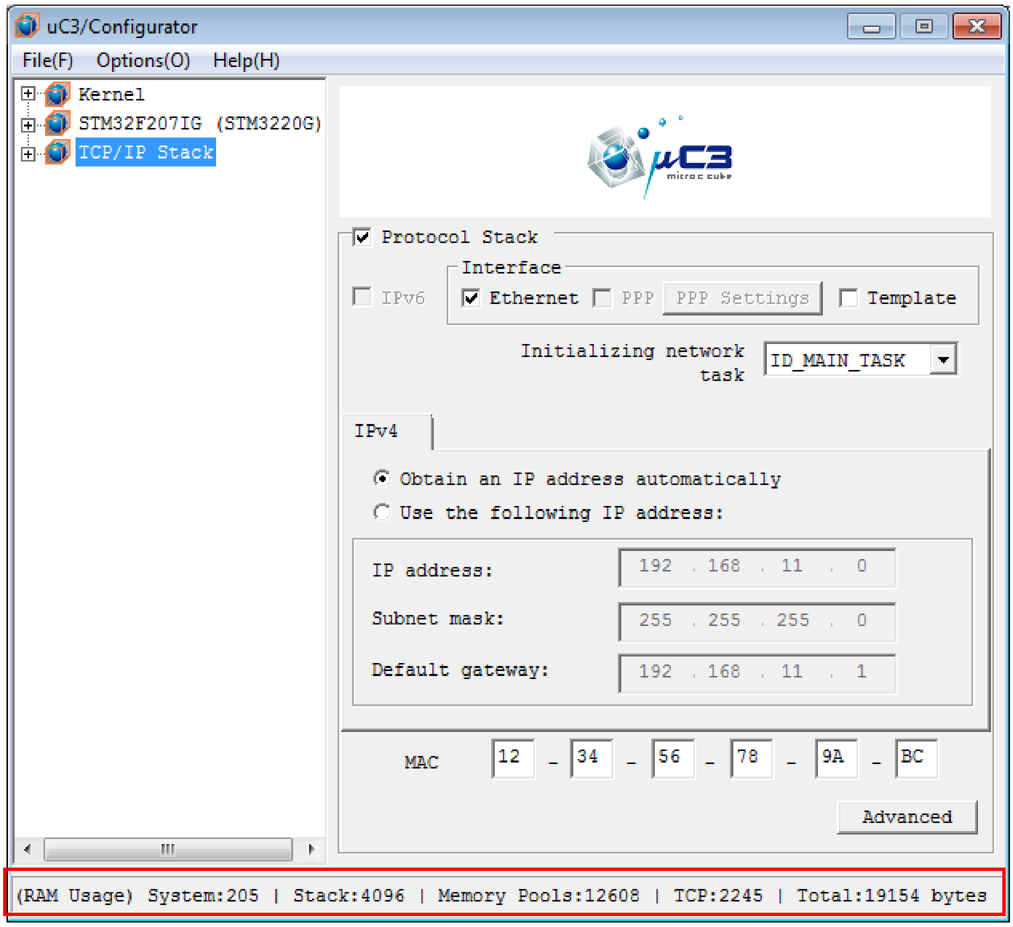
MAC 12 _ 34 _ 56 _ 78 _ 9A _ BC

(RAM Usage) System:205 | Stack:4096 | Memory Pools:12608 | TCP:2245 | Total:19154 bytes

4. 1. 2 Setup TCP/IP protocol stack

In TCP/IP protocol stack, there are the whole configuration of TCP/IP protocol stack and the configuration of TCP socket, UDP socket, application. On the screen of configuration of TCP socket and UDP socket, one socket will correspond to one tab.

RAM memory usage in system is always displayed at the status bar in the bottom. The following is the example of configuration screen.



☑ Protocol Stack

Interface

□ IPv6 ☑ Ethernet □ PPP □ Template

Initializing network task: ID_MAIN_TASK

● Obtain an IP address automatically

○ Use the following IP address:

IP address: 192 . 168 . 11 . 0

Subnet mask: 255 . 255 . 255 . 0

Default gateway: 192 . 168 . 11 . 1

MAC 12 - 34 - 56 - 78 - 9A - BC

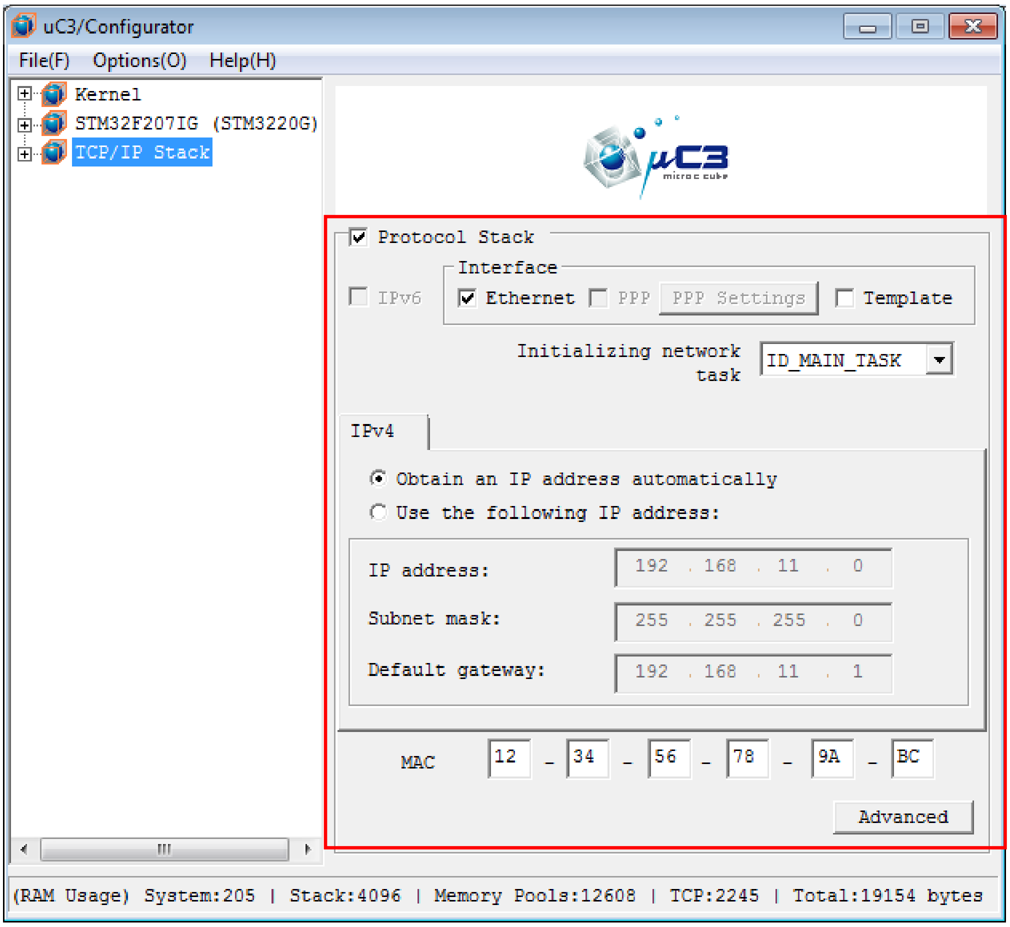
(RAM Usage) System:205 | Stack:4096 | Memory Pools:12608 | TCP:2245 | Total:19154 bytes

RAM usage

Items	Content
System	Memory is used by kernel itself , object management areas and data queue buffers
Stacks	System stack, stack of each task, common stack
Memory pool	Memory pool area of fixed-length memory pool
TCP/IP	Usage amount of sockets used by TCP/IP protocol stack

4. 1. 3 General configuration

If click on TCP/IP stack on the tree view,the general configuration screen of TCP/IP protocol stack will appears.



☑ Protocol Stack

Interface

□ IPv6 ☑ Ethernet □ PPP □ Template

Initializing network task: ID_MAIN_TASK

IPv4

● Obtain an IP address automatically

○ Use the following IP address:

IP address: 192 . 168 . 11 . 0

Subnet mask: 255 . 255 . 255 . 0

Default gateway: 192 . 168 . 11 . 1

MAC 12 - 34 - 56 - 78 - 9A - BC

(RAM Usage) System:205 | Stack:4096 | Memory Pools:12608 | TCP:2245 | Total:19154 bytes

Protocol stack

Specify whether to use protol stack in the check box. If check “ON”, the protocol stack is available. If “OFF”, it is unavailable.

※If check “OFF”, be careful that all the contents which has been setup until now will be clear off.

Interface

Choose interface want to use. In order to using Ethernet interface, check Ethernet and set to use Ethernet interface.

Besides, in case of integrating original network device, by checking Template, it can be used the empty network device driver integrated by the interface with μNet3.

Possible to choose the interface selected here when create a socket.

Network initializing task

Choose task that initialize in the network. Possible to remove the specifications here. In case that removed the specification, it is necessary to add the processing of network initialization “net_setup()”. Please set stack size of the task initilize network over 300 bytes.

Host setting-IP address

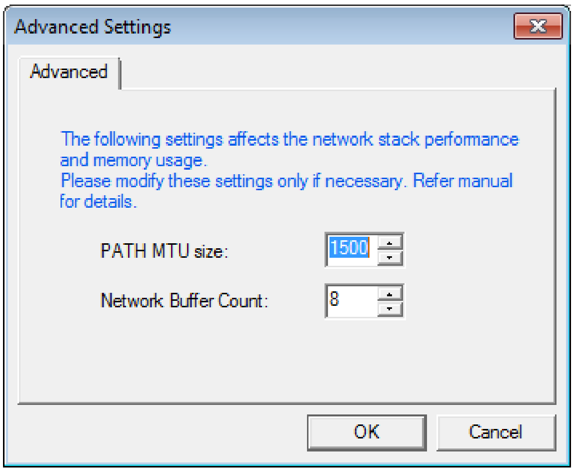
Please specify IP address of host. If select “Get IP address automatically”, use DHCP to set IP address automatically. At this time, the UDP socket used for DHCP will be added automatically. In case of selecting “Use the next IP address”, please specify the fixed IP address. The setup value is available for Ethernet interface.

Host setting-MAC address

Please specify MAC address for host. Input by the octet units. The seting value is available for Ethernet interface.

“Advanced setting” button

Display the screen that specify “PATH MTU Size” and “Number of network buffer” on.



The following settings affects the network stack performance and memory usage.
Please modify these settings only if necessary. Refer manual for details.

PATH MTU size: 1500

Network Buffer Count: 8

PATH MTU Size

Please specify PATH MTU. Please refer to “Chapter 2 μNet3/Compact Basic concepts” and “Chapter 3　Function overview of μNet3/Compact” in this document to execute the setup.

【Addition】

Setup of PATH MTU size is mapped automatically to “memory block size” of fixed-length memory pool ”ID_TCP_MPF” .

Memory block size is calculated as below.

Memory block size = PATH_MTU + size of management informasions

Number of network buffers

Please specify the number of network buffers. This numerical value specifies the number of memory blocks of network buffer used by TCP/IP protocol stack. Please refer to “3. 3　Network buffer”of “Chapter 3 Function overview of μNet3/Compact” in this document to execute the setup.

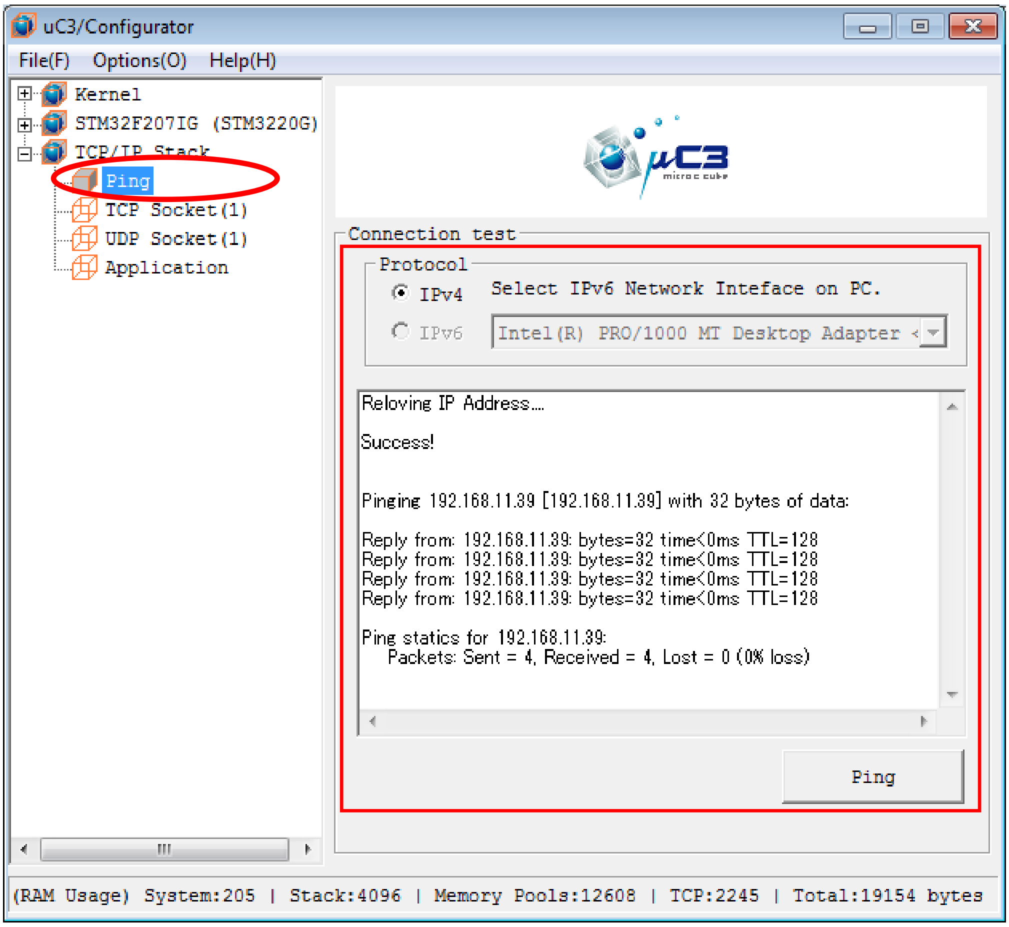
【Addition】

Setup of the number of network buffer is mapped automatically to “Number of memory blocks” of fixed-length memory pool ID_TCP_MPF”.

4. 1. 4 Communication test

If click on communication test on the tree view, communication test screen will appear.

Here, it doesn’t execute the configuration but execute communication test for the target.



```
Reloving IP Address....

Success!


Pinging 192.168.11.39 [192.168.11.39] with 32 bytes of data:

Reply from: 192.168.11.39: bytes=32 time<0ms TTL=128
Reply from: 192.168.11.39: bytes=32 time<0ms TTL=128
Reply from: 192.168.11.39: bytes=32 time<0ms TTL=128
Reply from: 192.168.11.39: bytes=32 time<0ms TTL=128

Ping statics for 192.168.11.39:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
```

(RAM Usage) System:205 | Stack:4096 | Memory Pools:12608 | TCP:2245 | Total:19154 bytes

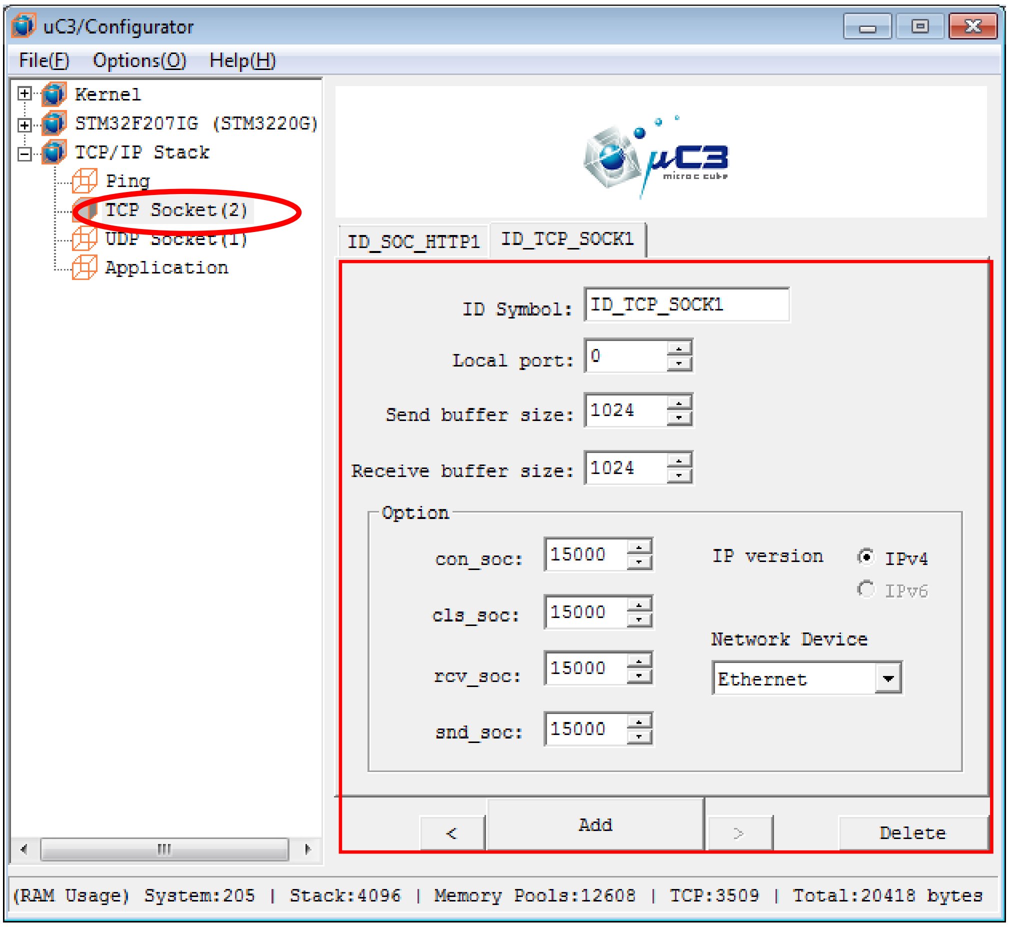
“Run communication test” button

The uC3/Configurator executes communication test through PING in accordance with host IP address specified in 4. 3. To execute communication test, it needs to integrate the program into target .The result of communication test will appear on the screen.

※In case PC has been installed virus security software, even though the target‘s program is working properly, communication test may fail. At that time, the it will execute the communication test after disable all the firewall settings of virus security software

4. 1. 5 Configuration of TCP socket

If click on TCP socket on the tree view, the configuration screen of TCP socket will appear.



ID Symbol: ID_TCP_SOCK1

Local port: 0

Send buffer size: 1024

Receive buffer size: 1024

con_soc: 15000

cls_soc: 15000

rcv_soc: 15000

snd_soc: 15000

IP version ● IPv4 ○ IPv6

Network Device: Ethernet

(RAM Usage) System:205 | Stack:4096 | Memory Pools:12608 | TCP:3509 | Total:20418 bytes

Defined name of ID

Please specify the arbitrary defined name which expresses the ID number of socket. This defined name is defined macro in net_id.h.

Local port number

Please specify port number of socket.

Transmission buffer size

Please specify transmission buffer size of socket.

Reception buffer size

Please specify reception buffer size of socket.

Timeout of con_soc

Please specify timeout period of API con_soc by the unit of millisecond (ms). If specify －1, con_soc will not return unless it completes successfully or occurs an error. This setting is available only for blocking mode.

Timeout of cls_soc

Please specify timeout period of API cls_soc by the unit of millisecond (ms). If specify －1, cls_soc will not return unless it completes successfully or occurs an error. This setting is available only for blocking mode.

Timeout of rcv_soc

Please specify timeout period of API rcv_soc by the unit of millisecond (ms). If specify －1, rcv_soc will not return unless it completes successfully or occurs an error. This setting is available only for blocking mode.

Timeout of snd_soc

Please specify timeout period of API snd_soc by the unit of millisecond (ms). If specify －1, snd_soc will not return unless it completes successfully or occurs an error. This setting is available only for blocking mode.

Selection of network device

Choose network device that is checked in interface. Socket should be necessary to associate with one network device. If choose nothing, do not specify network device when create socket. Regarding to the operation in this case, please refer to 5.4 Socket API.

“Add” button

Add new socket and correspondence tab of the socket.

“Delete” button

Delete socket of tab which is selected currently.

「←」button

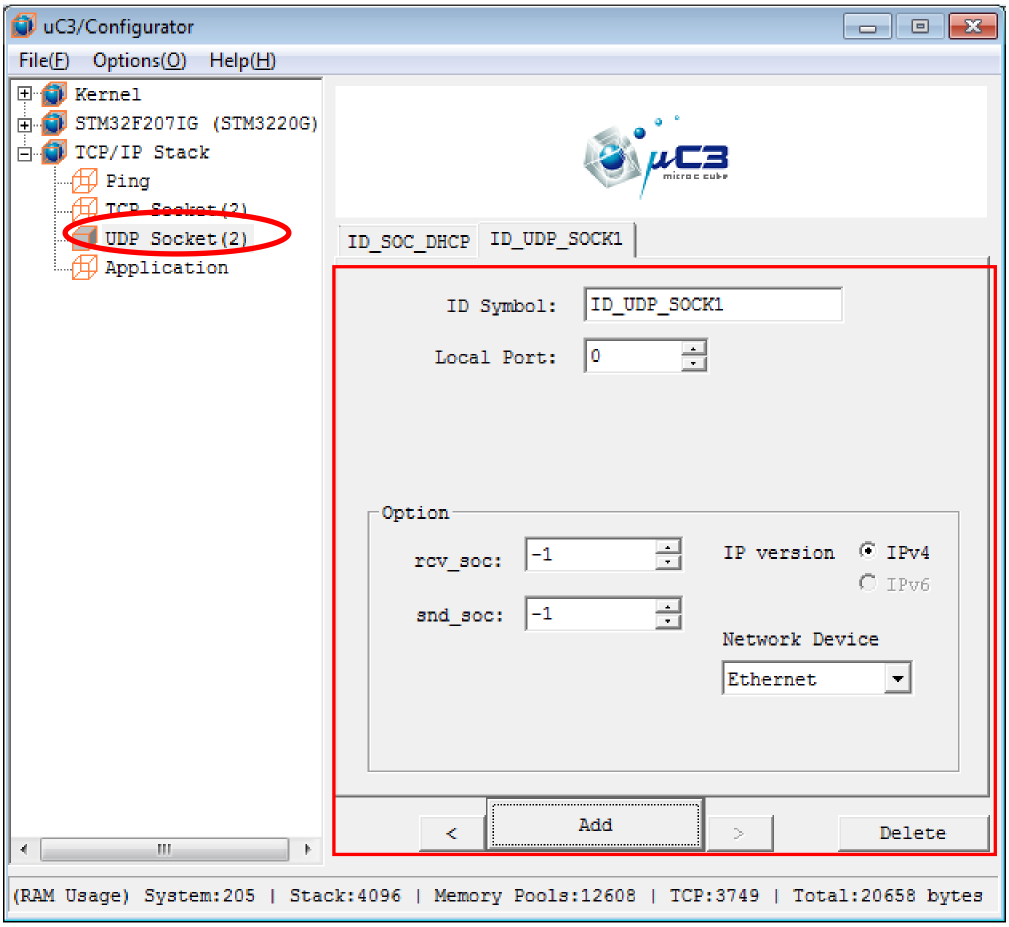
Move currently selected tab to the left.

「→」button

Move currently selected tab to the right.

4. 1. 6 Configuration of UDP socket

If click on UDP socket on the tree view, the configuration screen of UDP socket is displayed.



ID Symbol: ID_UDP_SOCK1

Local Port: 0

Option

rcv_soc: -1

snd_soc: -1

IP version ● IPv4 ○ IPv6

Network Device: Ethernet

(RAM Usage) System:205 | Stack:4096 | Memory Pools:12608 | TCP:3749 | Total:20658 bytes

Defined name of ID

Please specify the arbitrary defined name which expresses the ID number of socket. This defined name is defined macro in net_id.h.

Local port number

Please specify port number of socket.

Timeout of rcv_soc

Please specify timeout period of API rcv_soc by the unit of millisecond (ms). If specify −1, rcv_soc will not return unless it completes successfully or occurs an error. This setting is available only for blocking mode.

Timeout of snd_soc

Please specify timeout period of API snd_soc by the unit of millisecond (ms). If specify －1, snd_soc will not return unless it completes successfully or occurs an error. This setting is available only for blocking mode.

Selection of network device

Choose network device that is checked in interface. Socket should be necessary to associate with one network device. If choose nothing, do not specify network device when create socket. Regarding to the operation in this case, please refer to 5.4 Socket API.

“Add” button

Add new socket and correspondence tab of the socket.

“Delete” button

Delete socket of tab which is selected currently.

「←」button

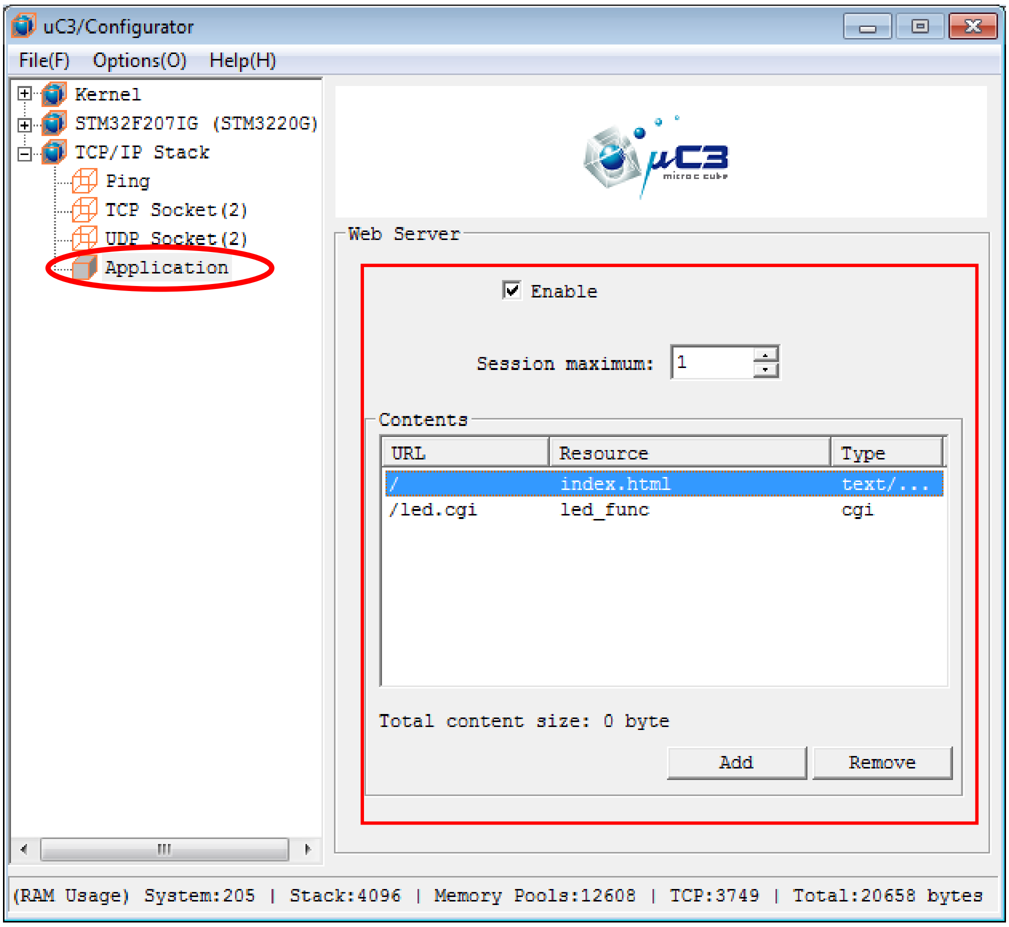
Move currently selected tab to the left.

「→」button

Move currently selected tab to the right.

4. 1. 7 Application configuration

If click on application on the tree view, the screen of application configuration will appear. By executing this configuration, it can be created WEB server application easily.



☑ Enable

Session maximum: 1

URL	Resource	Type
/	index.html	text/...
/led.cgi	led_func	cgi

Total content size: 0 byte

(RAM Usage) System:205 | Stack:4096 | Memory Pools:12608 | TCP:3749 | Total:20658 bytes

Using WEB server or not

Please specify whether to use WEB server. If check box is “ON”, WEB server is enabled. If “OFF”, it is disable. If WEB server is available, TCP socket used for HTTP will be added automatically.

Maximum number of session

Please specify the maximum of session which connect to WEB server. Maximum value to specify is 2.

Content list

Now registered contents are displayed. Maximum number of content to be registered is 50. If double click the content of the content list which want to change on the left side of the mouse, the content can be changed.

Total size of content

Now total size of registered content is displayed. Please do not let this size exceed the maximum size of ROM.

“Add” button

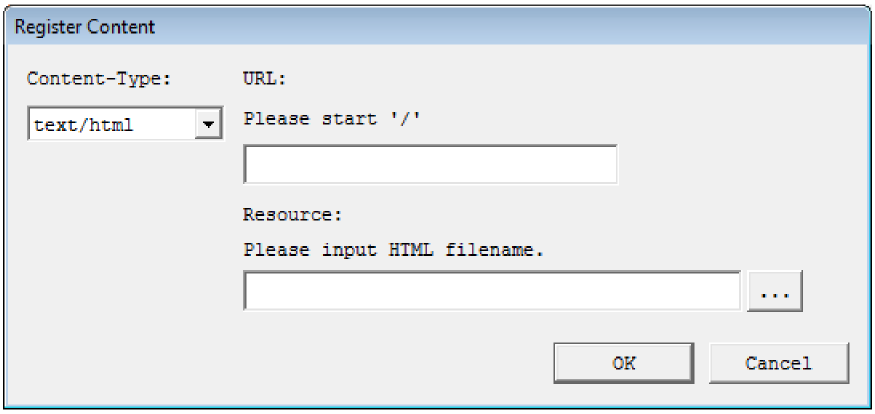
Add new content.

“Delete” button

Delete content which is selected now.

Addition and modification to content

If click on “Add” button or double click on registered contents registered in the list, the below registration screen will appear.



Register Content

Content-Type: text/html

URL:
Please start '/'

Resource:
Please input HTML filename.

Content-Type

Please specify content type (internet media type) to register. Content type will be one of the following options.

text/html

image/gif

image/jpeg

cgi

URL

Please specify URL of content. Please start to input URL by ‘/’. it can be input maximum 12 characters.

(Input example)

In case of text/html /index.html
In case of cgi /function.cgi （script name of CGI）

Resource

Please specify content resource.

- In case that content type is not cgi: please specify an actual file in accordance with specified Content-Type. Or if click on 「…」 button, the “File selection” screen will appear and then it can specify file there.
- In case content type is cgi : please specify the name of function execute CGI script. Specified function name is outputted to main.c. The follow characters are not included in the function name.

  Prohibition’s characters : " `{}*@;+:*,.#$%&'¥"!?~^=|/¥<>()"

OK button

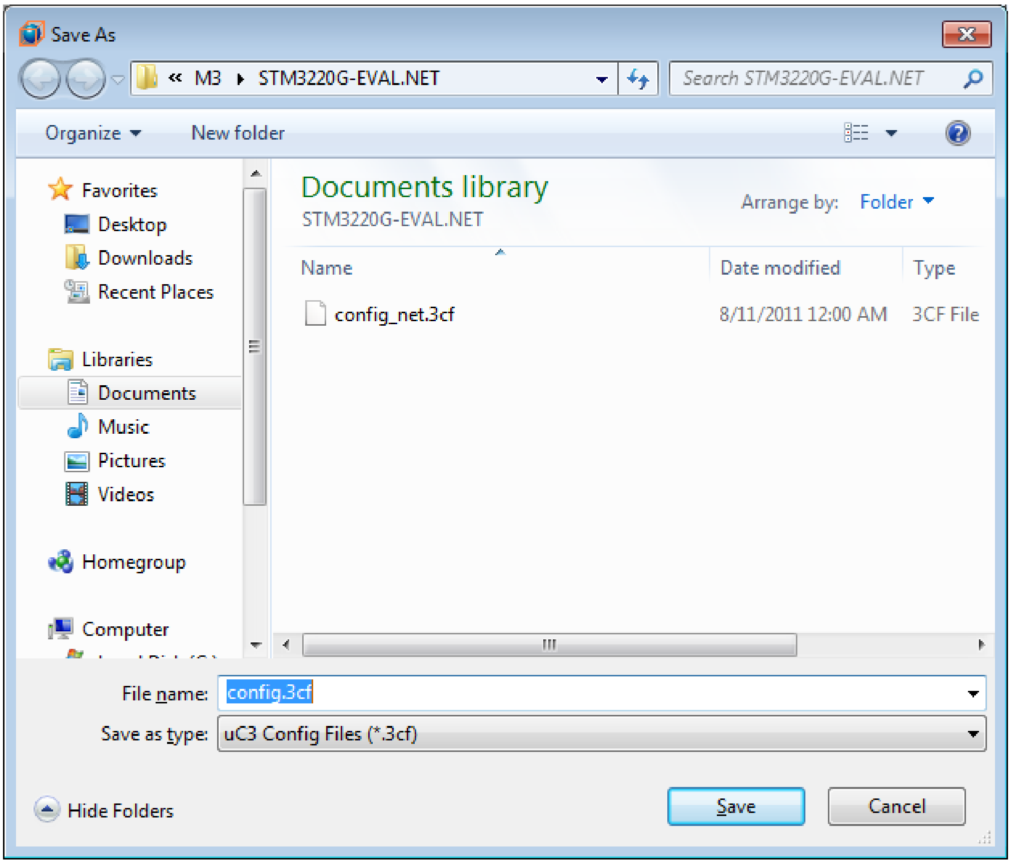
Register content.

Cancel button

Close the screen without register the content

4. 1. 8 Saving project file

By「File」→「Save…(S)」, open 「"Save as" screen」, indicate a destination folder to save that project file and then click OK.



Name	Date modified	Type
config_net.3cf	8/11/2011 12:00 AM	3CF File

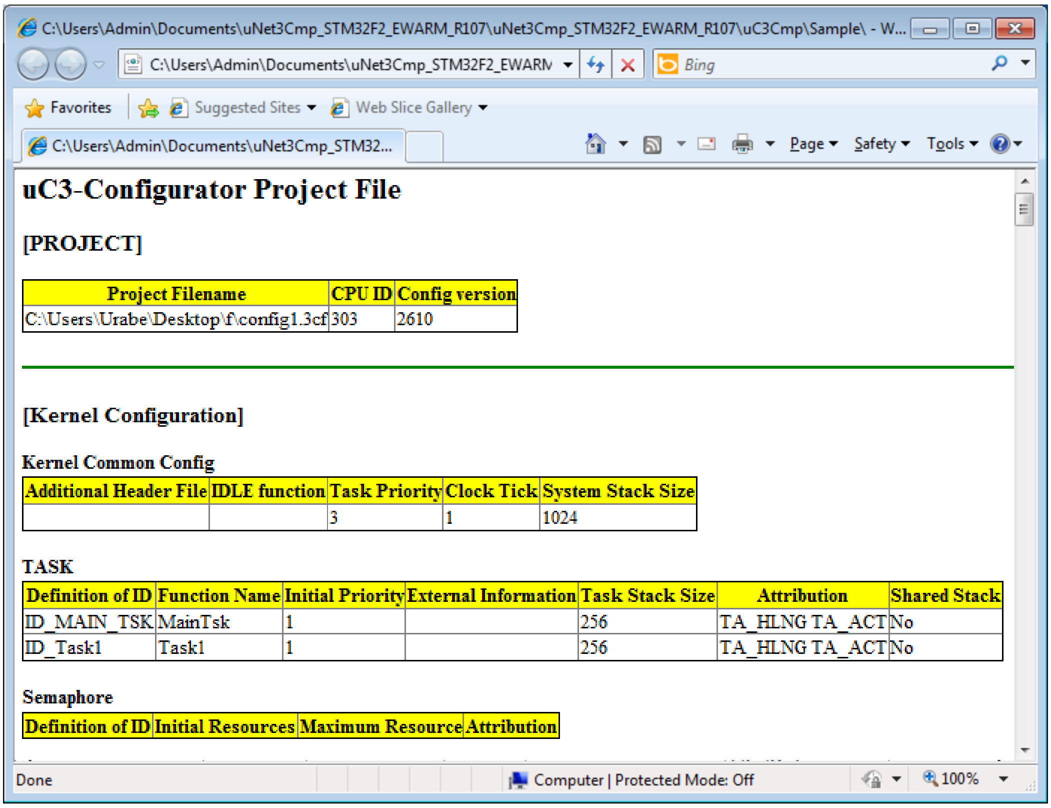
File name: config.3cf

Save as type: uC3 Config Files (*.3cf)

Saved file include the project file (default config..3cf) and file with file extension「xml」.

By browsing to open this file, you can confirm the configuration information.

This file is shown for the list of configuration.


C:\Users\Admin\Documents\uNet3Cmp_STM32F2_EWARM_R107\uNet3Cmp_STM32F2_EWARM_R107\uC3Cmp\Sample\ - W...
C:\Users\Admin\Documents\uNet3Cmp_STM32F2_EWARM
Bing
Favorites
Suggested Sites
Web Slice Gallery
C:\Users\Admin\Documents\uNet3Cmp_STM32...
Page
Safety
Tools
Done
Computer | Protected Mode: Off
100%

uC3-Configurator Project File

[PROJECT]

Project Filename	CPU ID	Config version
C:\Users\Urabe\Desktop\f\config1.3cf	303	2610

[Kernel Configuration]

Kernel Common Config

Additional Header File	IDLE function	Task Priority	Clock Tick	System Stack Size
		3	1	1024

TASK

Definition of ID	Function Name	Initial Priority	External Information	Task Stack Size	Attribution	Shared Stack
ID_MAIN_TSK	MainTsk	1		256	TA_HLNG TA_ACT	No
ID_Task1	Task1	1		256	TA_HLNG TA_ACT	No

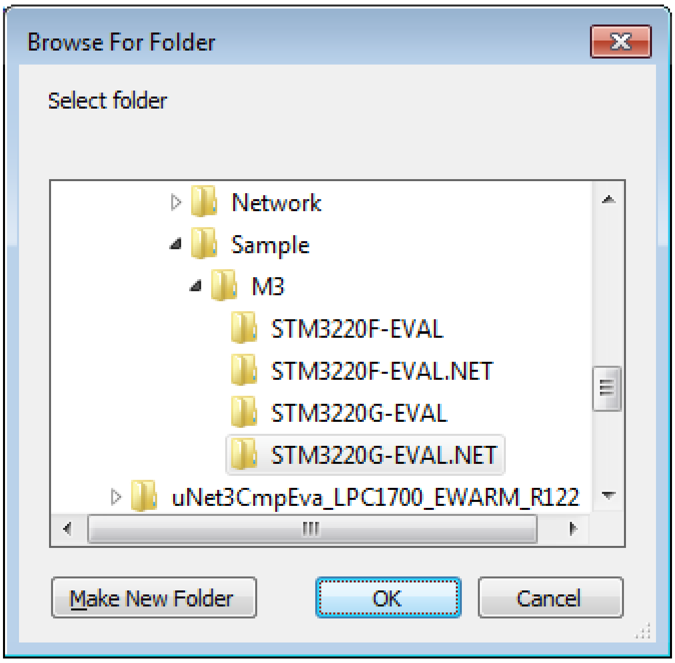
Semaphore

Definition of ID	Initial Resources	Maximum Resource	Attribution

This XML information is only Japanese in current version.

4. 1. 9 Source generation

By「File」→「Source generation…(G)」, open “Folder reference screen”, specify the arbitrary folder to develop the generating file and then click OK.



In case that skeleton code main.c exists already, there will appear a confirm message window to prevent the accidental erasure by overwriting the edited application..

【Recommendation】

In order to prevent the erasure by overwriting skeleton code, it is recommended to create an application program which is used as template, without edit skeleton code directly.

A. Created File

File	Content
net_cfg.c	Configuration code socket definition, IP address definition, MAC address definition of protocol stack
net_id.h	Header file of ID socket definition
net_hdr.h	Header file of protocol stack
main.c	main(),initial setting function, skeleton code of task or handler
Protocol stack library	Library that collected API group of protocol stack.
Application protocol source file	Code that collected API group of HTTP, DHCP, DNS, FTP protocol

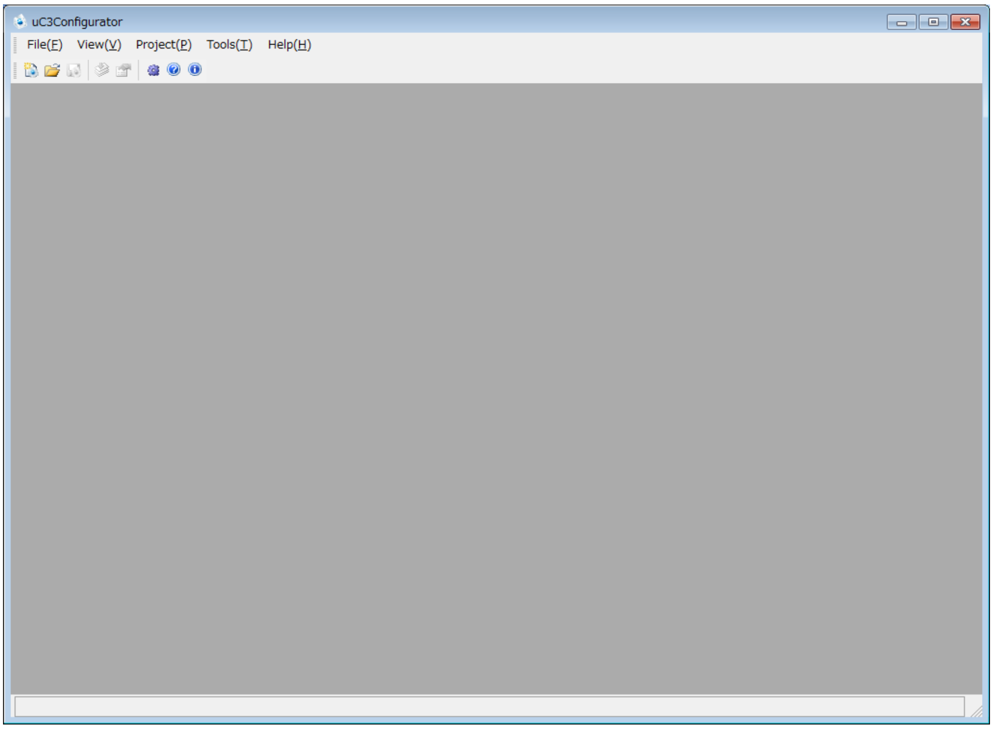
※The files relating to kernel, processor other than those above are also created, but it is not explaned in this document. Please make appropriate reference **to "μC3/Compact Users Guide".**

These created files are different by configuration, processor or device.

4. 2 Configuration of μNet3/Compact for Version 2.xx

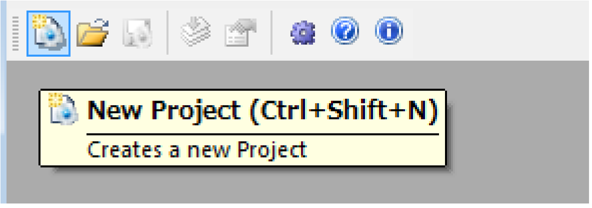
4. 2. 1 Starting up configurator

Double click on “uC3conf.exe” to start up



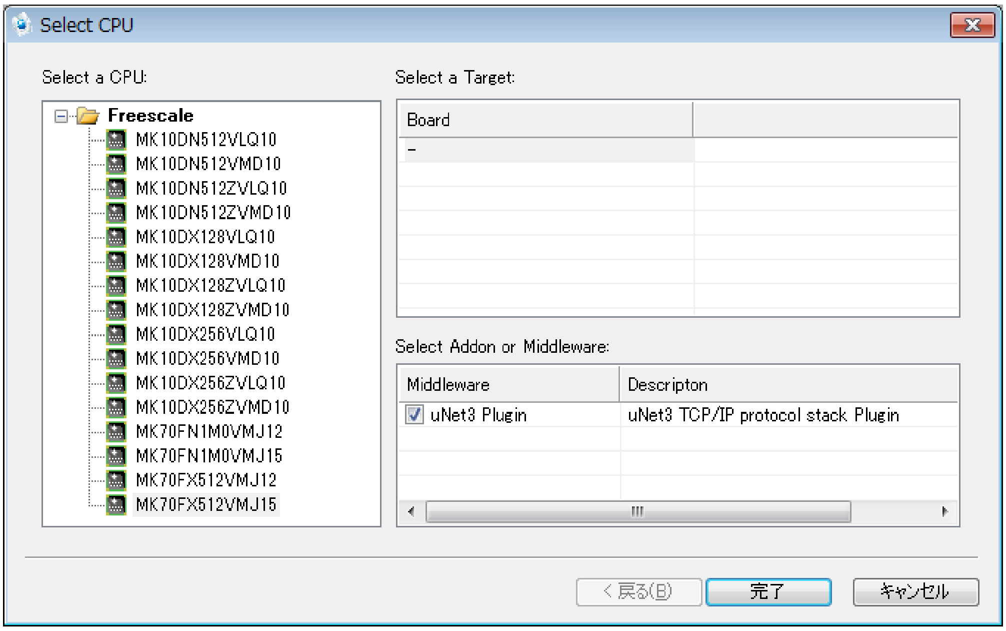
A. In case of create a new project

From the Configurator toolbar, click “New Project” and go to “Select CPU”



Select CPU

After selecting CPU vendors, CPU, serial number, target in List, click “OK” and go to “Main screen” .



Select CPU

Select a CPU:

- Freescale
  - MK10DN512VLQ10
  - MK10DN512VMD10
  - MK10DN512ZVLQ10
  - MK10DN512ZVMD10
  - MK10DX128VLQ10
  - MK10DX128VMD10
  - MK10DX128ZVLQ10
  - MK10DX128ZVMD10
  - MK10DX256VLQ10
  - MK10DX256VMD10
  - MK10DX256ZVLQ10
  - MK10DX256ZVMD10
  - MK70FN1M0VMJ12
  - MK70FN1M0VMJ15
  - MK70FX512VMJ12
  - MK70FX512VMJ15

Select a Target:

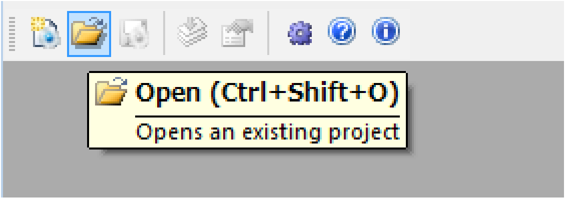
Board	
–	

Select Addon or Middleware:

Middleware	Descripton
☑ uNet3 Plugin	uNet3 TCP/IP protocol stack Plugin

B. In case of opening the existing project

From the Configurator toolbar, click “Open” and go to “Open file” .

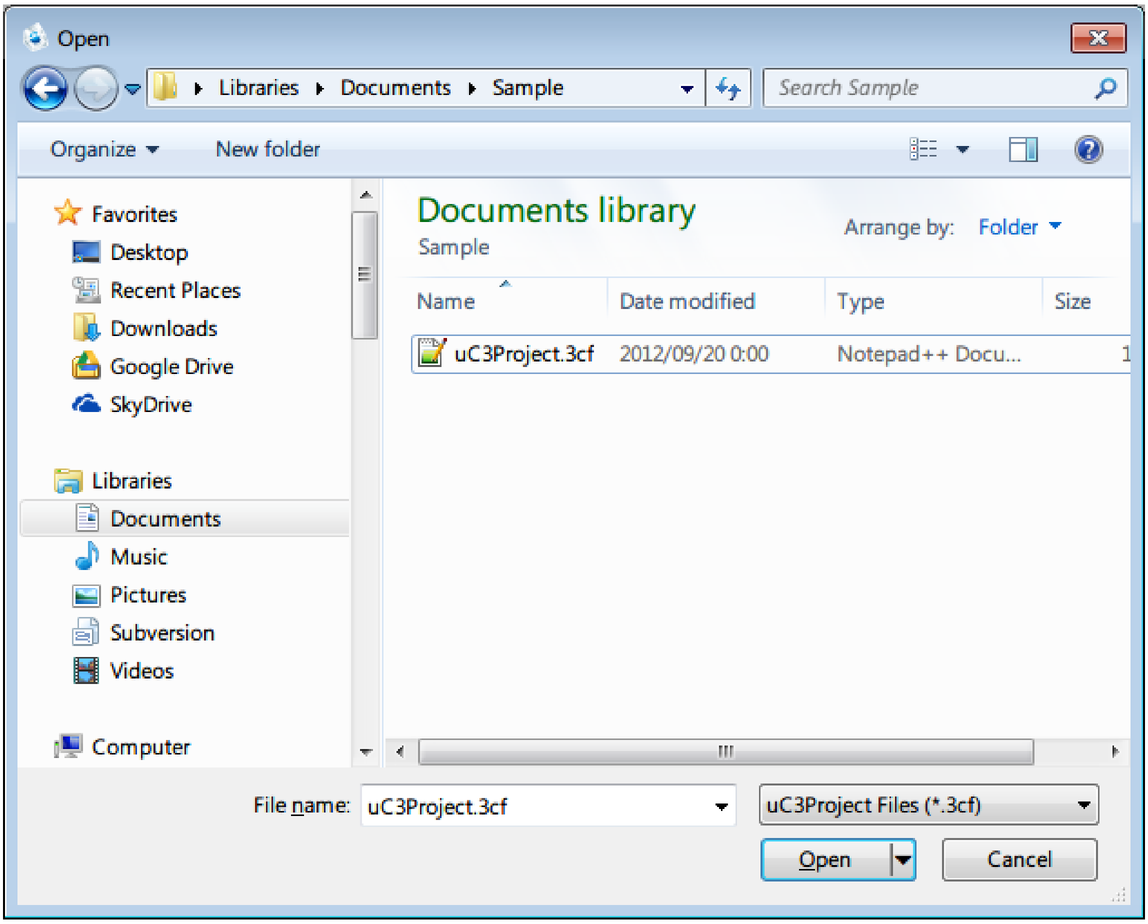


Open (Ctrl+Shift+O)

Opens an existing project

Open file

After selecting a saved project file (extension.3cf), click “Open” and go to “Main screen” .



Name	Date modified	Type	Size
uC3Project.3cf	2012/09/20 0:00	Notepad++ Docu...	1

File name: uC3Project.3cf

uC3Project Files (*.3cf)

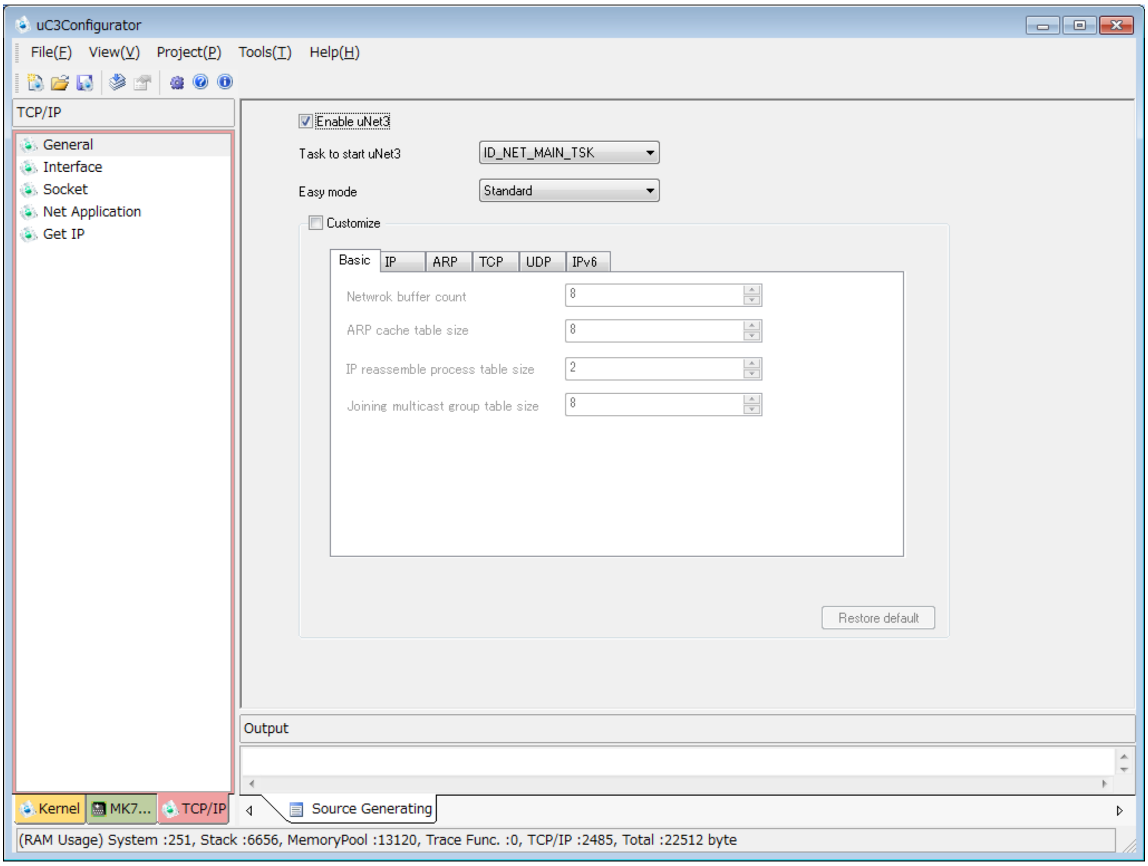
Open

*:Project file created by "Ver.2.x configurator" is read only kernel configuration.

(CPU configuration is not read)

C. Main screen

After starting up, it will go to the main screen where it is possible to refer or edit project. There is a menu screen to the left of the main screen. By clicking to each Object of Menu Screen, it will switch to each Object Configutation screen. Here, there is configuration of TCP/IP.



☑ Enable uNet3

Task to start uNet3: ID_NET_MAIN_TSK

Easy mode: Standard

□ Customize

Netwrok buffer count: 8

ARP cache table size: 8

IP reassemble process table size: 2

Joining multicast group table size: 8

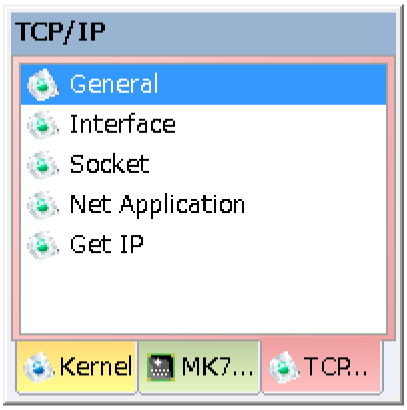
(RAM Usage) System :251, Stack :6656, MemoryPool :13120, Trace Func. :0, TCP/IP :2485, Total :22512 byte

4. 2. 2 uNet3 General configuration

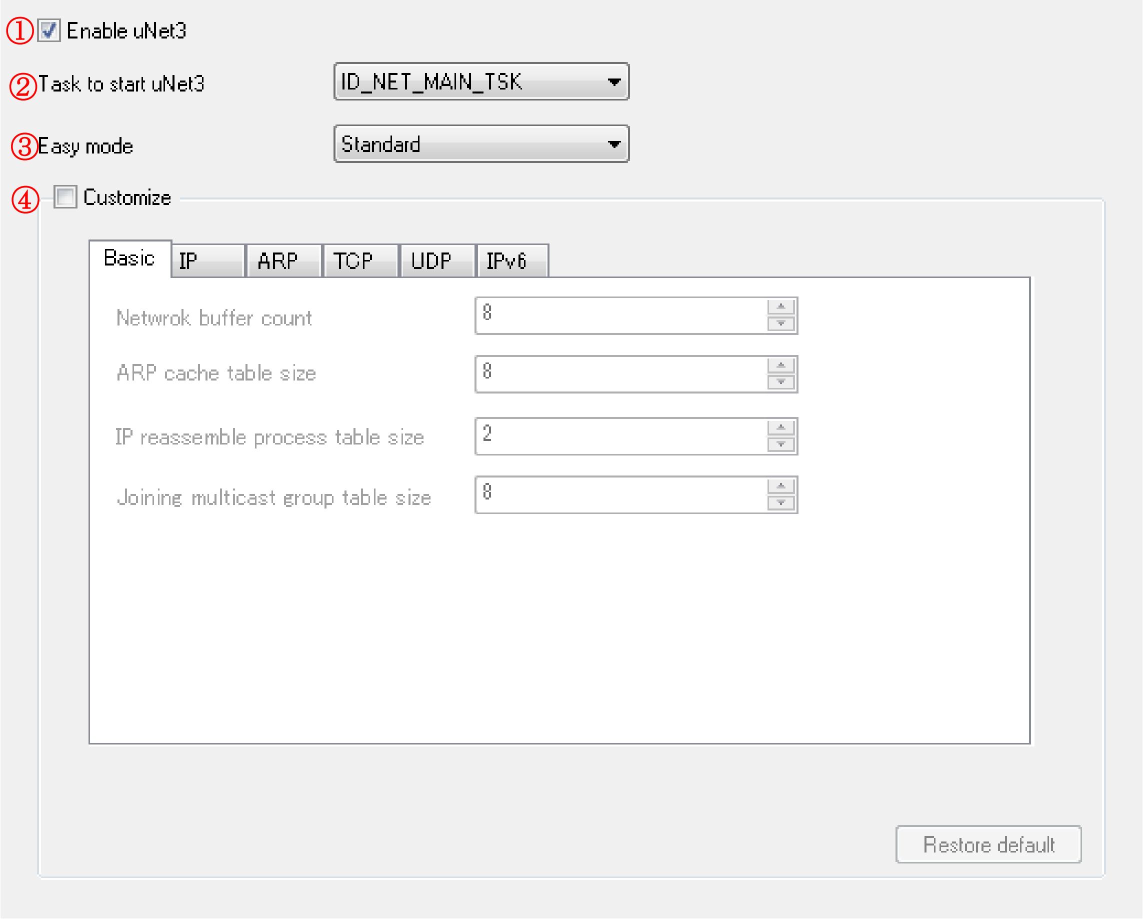
In TCP/IP protocol stack, it is possible to configure with Interface, Socket and Network Applications.

If click on TCP/IP General on the menu window, the general configuration screen of TCP/IP protocol stack will appears.

Menu Screen



Configuration Screen



① ☑ Enable uNet3

②Task to start uNet3: ID_NET_MAIN_TSK

③Easy mode: Standard

④ □ Customize

Netwrok buffer count: 8

ARP cache table size: 8

IP reassemble process table size: 2

Joining multicast group table size: 8

① Enable μNet3

Specify whether to use protol stack in the check box. If check “ON”, the protocol stack is available. If checked “OFF”, it is unavailable.

※If checked “OFF”, be careful that all the contents which has been setup until now will be clear off.

② Task to start μNet3

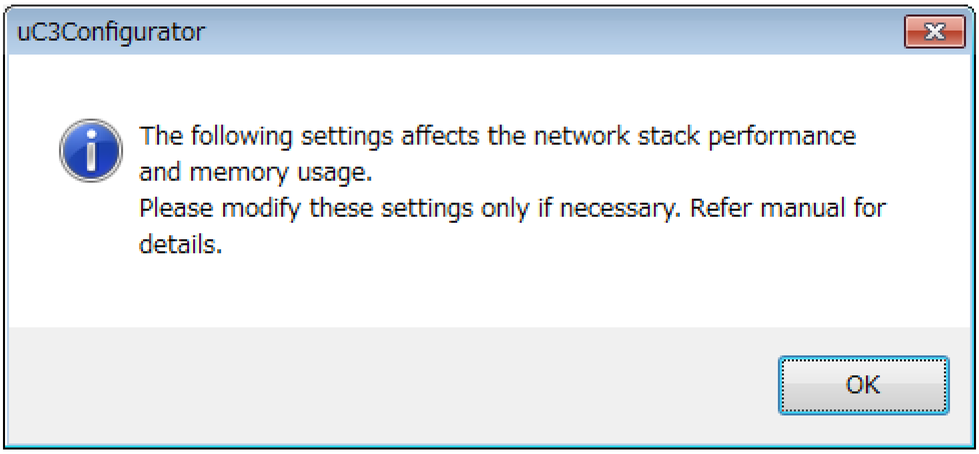
Choose task to start μNet3. It is possible to remove the specifications here. In case that removed the specification, it is necessary to add the processing of network initialization “net_setup()”. Please set stack size of the task initilize network over 300 bytes.

③ Easy mode

It is possible to set collectively table of various size according to meet the requirements of the system. If it is selected “Standard”, then table size is the default value.

④ Customize

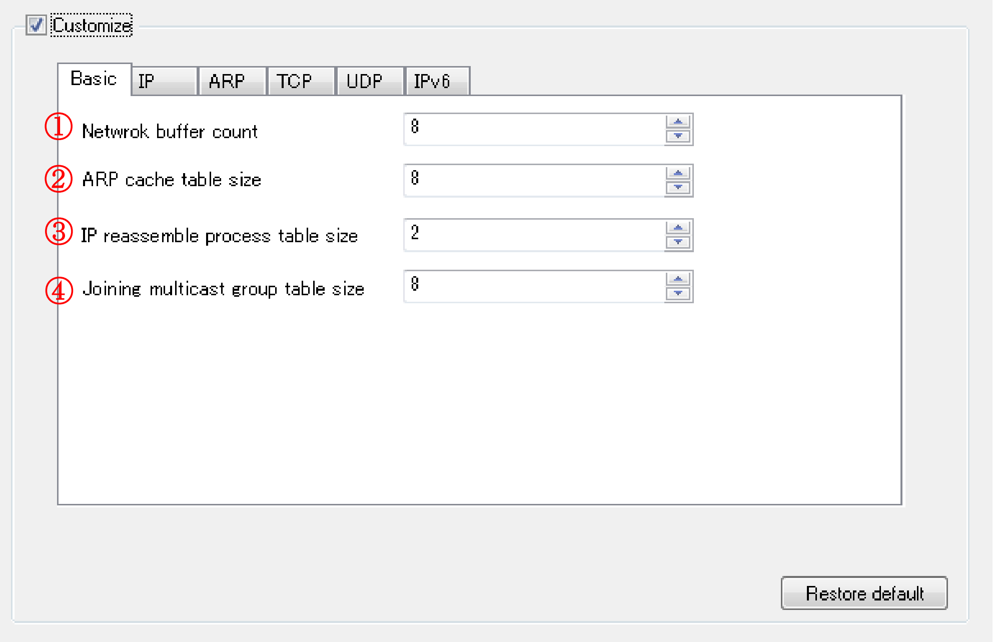
It is possible to set various details of the protocol provided by the μNet3.



uC3Configurator

The following settings affects the network stack performance and memory usage.
Please modify these settings only if necessary. Refer manual for details.

【Basic】



☑ Customize

① Netwrok buffer count 8

② ARP cache table size 8

③ IP reassemble process table size 2

④ Joining multicast group table size 8

① Network buffer counts

Please specify the number of network buffers. This numerical value specifies the number of memory blocks of network buffer used by TCP/IP protocol stack. Please refer to "3. 3 Network buffer"of "Chapter 3 Function overview of μNet3/Compact" in this document to execute the setup.

② ARP cache table size

Please specify the ARP cache table size.

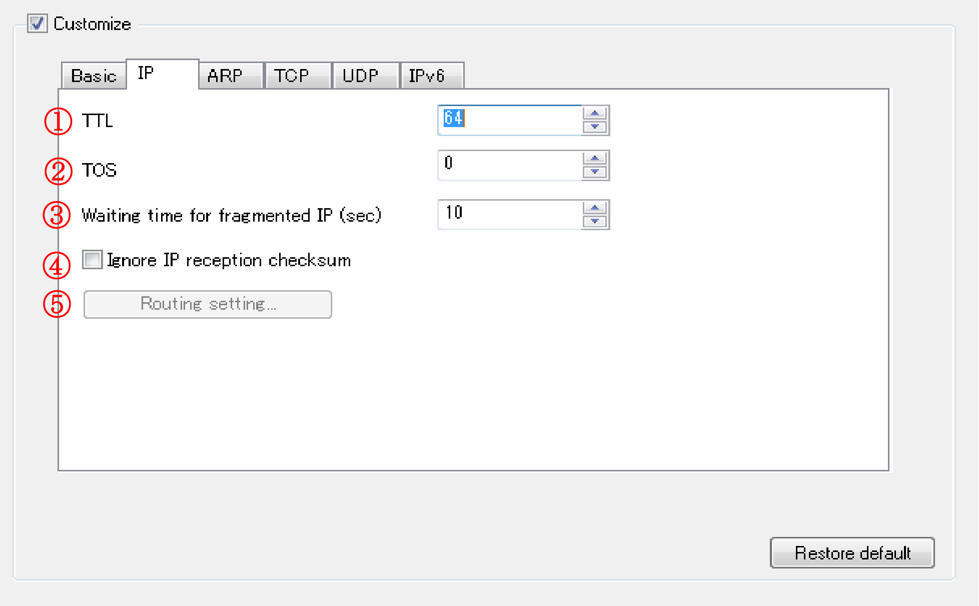
③ IP reassemble process table size

Please specify the number of queue that is used of IP reassemble process.

④ Joining multicast group table size

Please specify the maximum number of joining multicast groups.

【IP Setting】



☑ Customize

① TTL 64

② TOS 0

③ Waiting time for fragmented IP (sec) 10

④ □ Ignore IP reception checksum

⑤

① **TTL**

Please set the TTL value of IP header to be sent. Please use cfg_soc() if it is need to set in sockets.

② **TOS**

Please set the TOS value of IP header to be sent. Please use cfg_soc() if it is need to set in sockets.

③ **Waiting time for fragmented IP**

Please set the time to wait for the rest of the IP fragment packets, in processing IP reassembly.

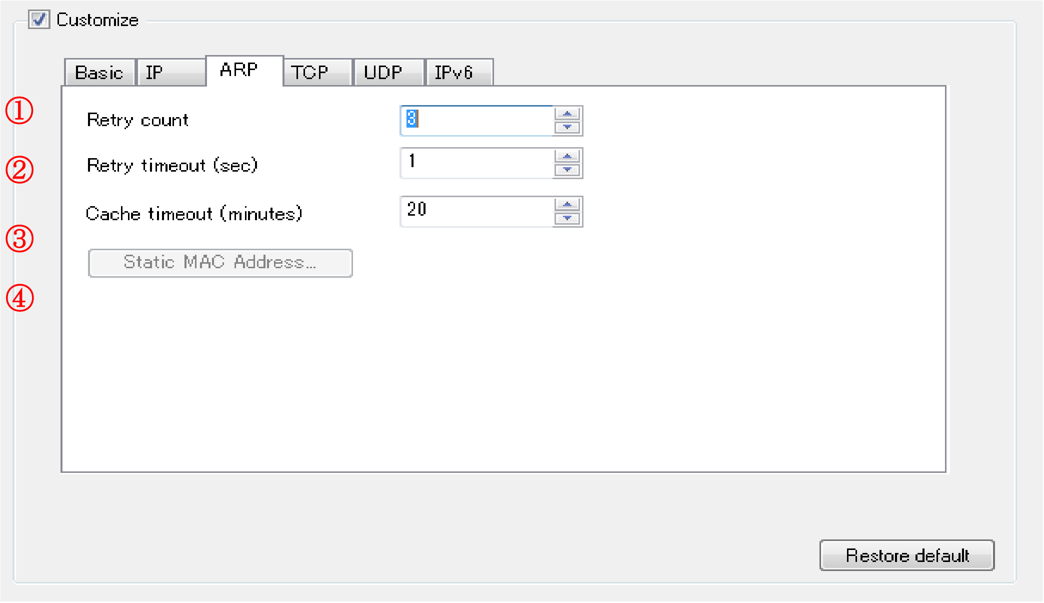
④ **Ignore IP reception checksum**

µNet3 will not verify the checksum value of the received IP packet.

⑤ **Routing setting**

This feature is not available yet.

【ARP setting】



☑ Customize

Basic | IP | ARP | TCP | UDP | IPv6

① Retry count: 3

② Retry timeout (sec): 1

③ Cache timeout (minutes): 20

④ Static MAC Address...

① Retry count

Please set the number of times to retry sending ARP until it receives response.

② Retry timeout

Please set the interval to send ARP retries.

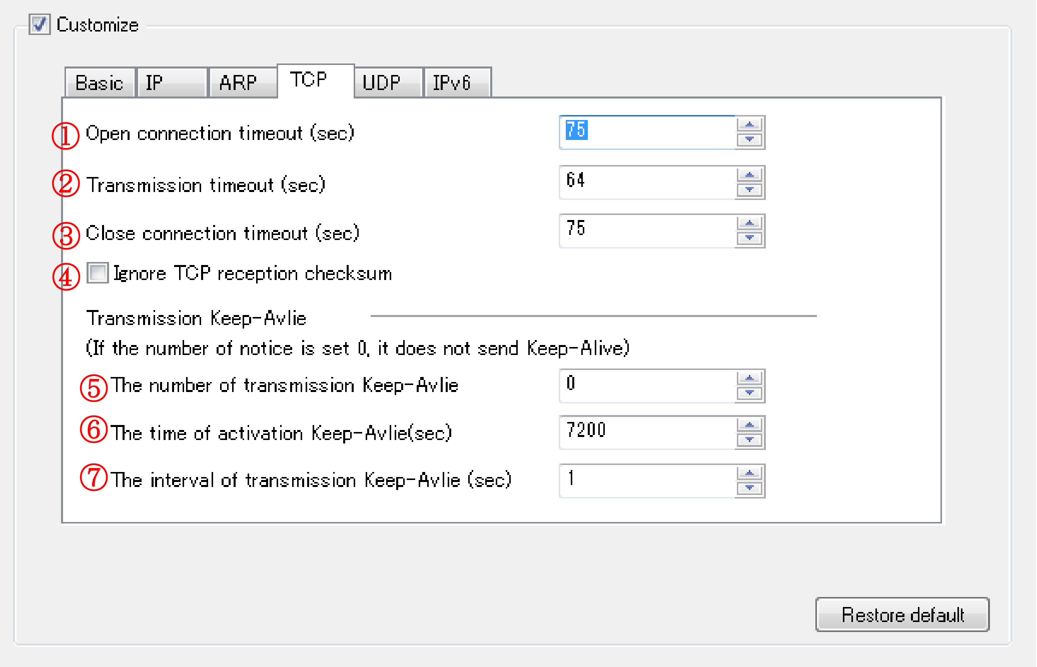
③ Cache timeout

Please set the time to keep ARP cache.

④ Static MAC Address

This feature is not available yet.

【TCP Setting】



☑ Customize

① Open connection timeout (sec) 75

② Transmission timeout (sec) 64

③ Close connection timeout (sec) 75

④ □ Ignore TCP reception checksum

Transmission Keep-Avlie
(If the number of notice is set 0, it does not send Keep-Alive)

⑤ The number of transmission Keep-Avlie 0

⑥ The time of activation Keep-Avlie(sec) 7200

⑦ The interval of transmission Keep-Avlie (sec) 1

① **Open connection timeout**

Please set the maximum amount of time waiting TCP active open complete.

② **Transmission timeout**

Please set the maximum amount of time waiting ACK .

③ **Close connection timeout**

Please set the maximum amount of time waiting TCP close session complete.

④ **Ignore TCP reception checksum**

µNet3 will not verify the checksum value of the received TCP packet.

⑤ **The number of transmission Keep-Alive**

Please set the number of times of Keep-Alive transmissions to be sent before disconnect TCP session. (Keep-Alive does not start if this value is 0)

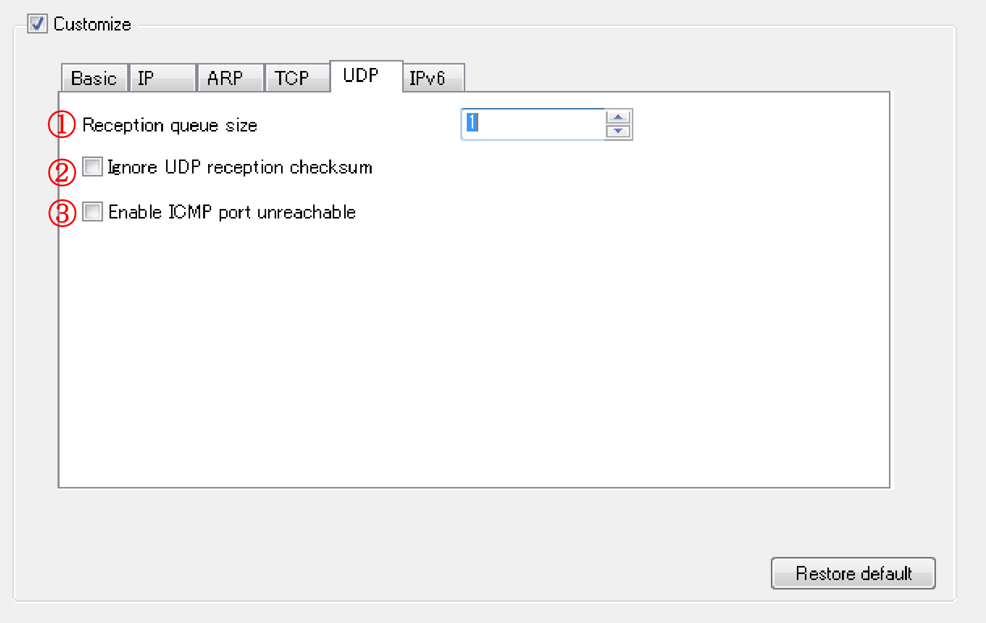
⑥ **The time of activation Keep-Alvie**

Please set the time to send the Keep-Alive first non-communication period has begun.

⑦ **The interval of transmission Keep-Alive**

Please set the time interval between the transmissions of Keep-Alive.

【UDP Setting】



☑ Customize

① Reception queue size: 1

② ☐ Ignore UDP reception checksum

③ ☐ Enable ICMP port unreachable

① Reception queue size

Please set the number of received UDP packets queued.

② Ignore UDP reception checksum

µNet3 will not verify the checksum value of the received UDP packet.

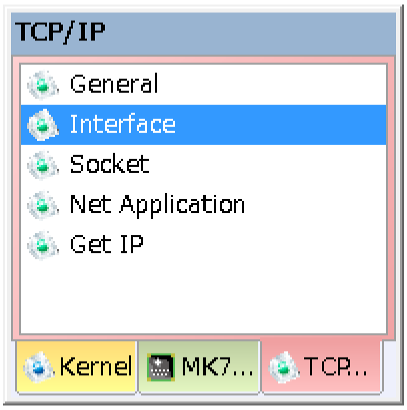
③ Enable ICMP port unreachable

µNet3 will send ICMP (Port unreachable) when receiving a packet to an unused port.

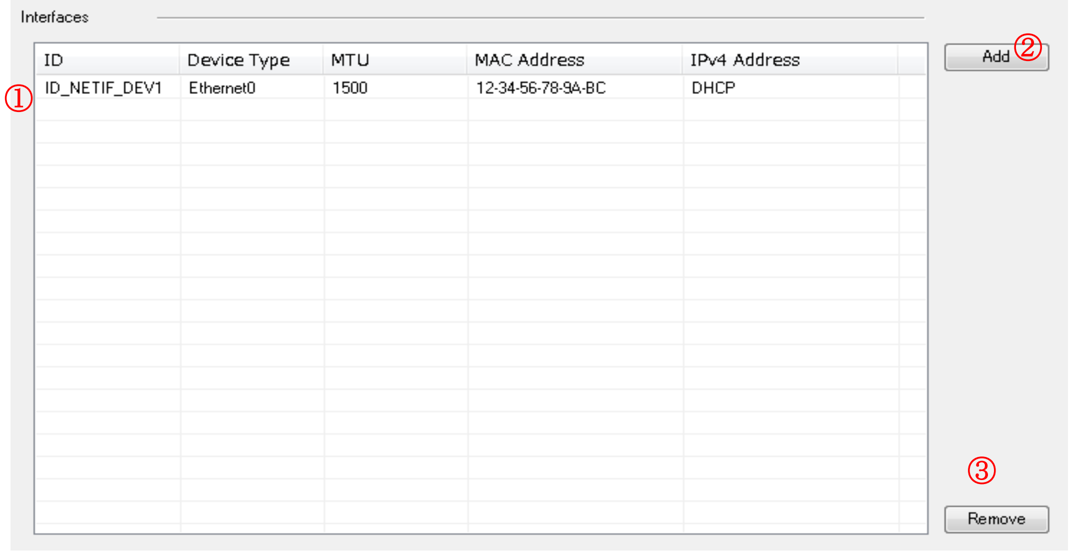
4. 2. 4 Interface configuration

If click on Interface on the menu window, the device interface screen will appears such as Ethernet or PPP setting.

Menu Screen



Configuration Screen



ID	Device Type	MTU	MAC Address	IPv4 Address
ID_NETIF_DEV1	Ethernet0	1500	12-34-56-78-9A-BC	DHCP

① List of Interfaces

List of interfaces that are currently set will be displayed. By double-clicking on the interface in the list it displays editing screen.

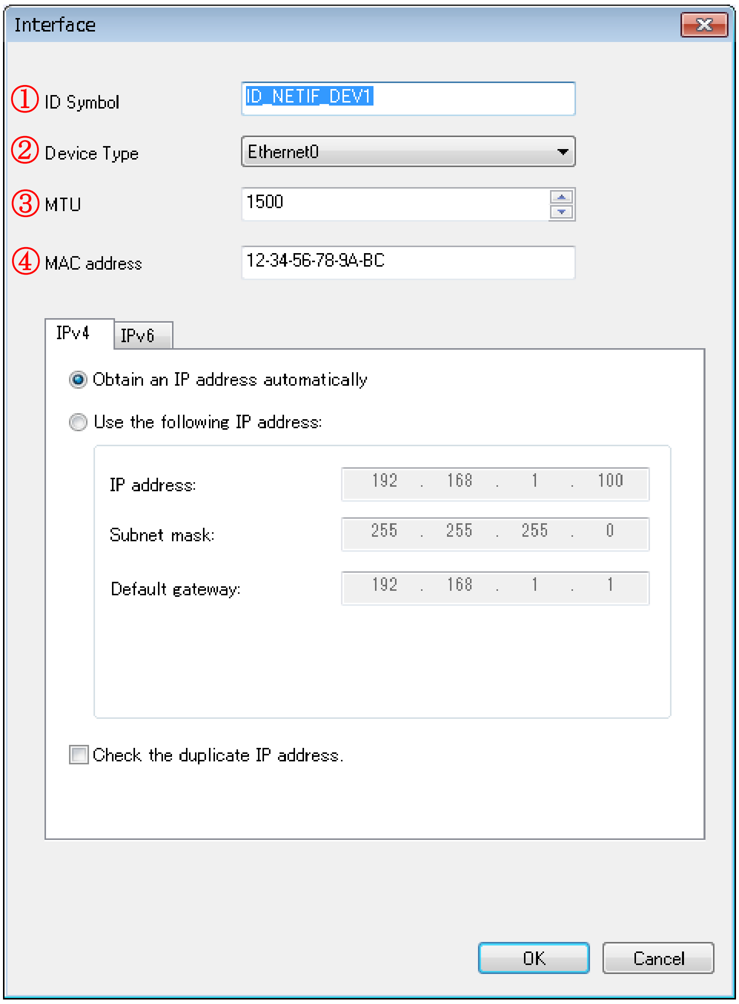
② ADD

Displays editing screen of the interface to add new ones.

③ Remove

Remove the selected interface.

【Interface】



Interface

① ID Symbol: ID_NETIF_DEV1

② Device Type: Ethernet0

③ MTU: 1500

④ MAC address: 12-34-56-78-9A-BC

● Obtain an IP address automatically

○ Use the following IP address:

IP address: 192 . 168 . 1 . 100

Subnet mask: 255 . 255 . 255 . 0

Default gateway: 192 . 168 . 1 . 1

□ Check the duplicate IP address.

① ID Symbol

Please specify the arbitrary defined name which expresses the ID number of interface. This defined name is defined macro in net_id.h.

② Device Type

Please select device type (Link Type) which is supported by the chip. If it is selected LOOPBACK you can use the network device driver to receive packet sent to wrap at the driver level

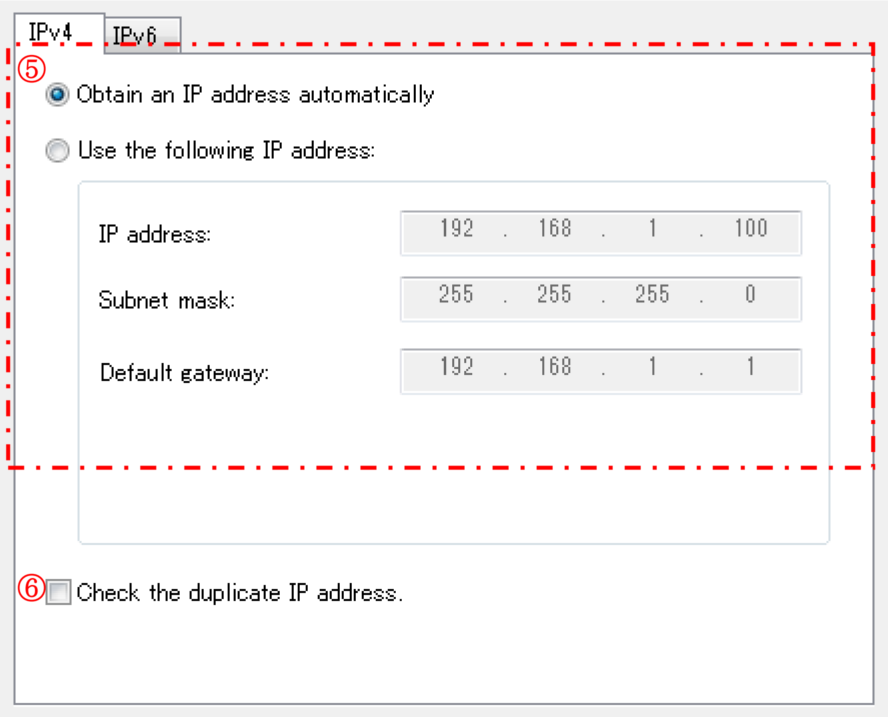
③ MTU

Please specify PATH MTU. Please refer to “Chapter 2 μNet3/Compact Basic concepts” and “Chapter 3 Function overview of μNet3/Compact” in this document to execute the setup.

④ MAC address

Please set MAC address of host. Input by the octet units. The seting value is available for Ethernet interface.

【Interface IPv4】



● Obtain an IP address automatically

○ Use the following IP address:

IP address: 192 . 168 . 1 . 100

Subnet mask: 255 . 255 . 255 . 0

Default gateway: 192 . 168 . 1 . 1

□ Check the duplicate IP address.

⑤ IP Address

Please specify IP address of host. If select "Get IP address automatically", use DHCP to set IP address automatically. At this time, the UDP socket used for DHCP will be added automatically. In case of selecting "Use the next IP address", please specify the fixed IP address. The setup value is available for Ethernet interface.

⑥ Check the duplicate IP address

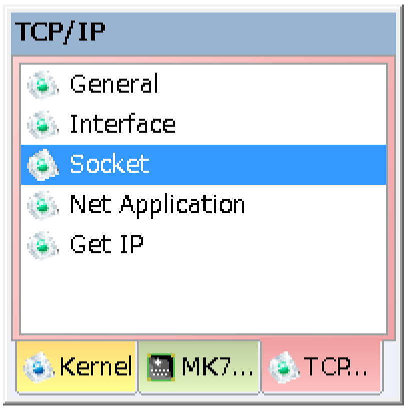
µNet3 will detect the IP address set to the host is not present on the same network by sending ARP.

Also in running, if other host sent the ARP using same IP address, µNet3 will detect it and notify the application through the callback function.

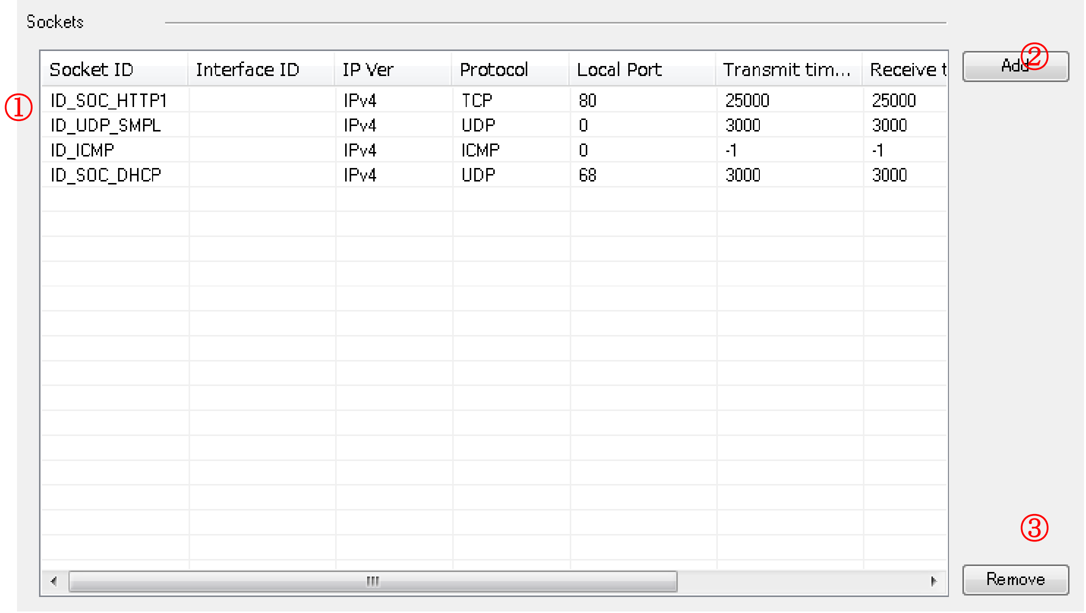
4. 2. 5 Socket configuration

If click on Socket on the menu window, the configuration screen of TCP and UDP socket will appear.

Menu Screen



Configuration Screen



Socket ID	Interface ID	IP Ver	Protocol	Local Port	Transmit tim...	Receive t
ID_SOC_HTTP1		IPv4	TCP	80	25000	25000
ID_UDP_SMPL		IPv4	UDP	0	3000	3000
ID_ICMP		IPv4	ICMP	0	-1	-1
ID_SOC_DHCP		IPv4	UDP	68	3000	3000

① List of Socket

List of sockets that are currently set will be displayed. By double-clicking on the socket in the list it displays editing screen.

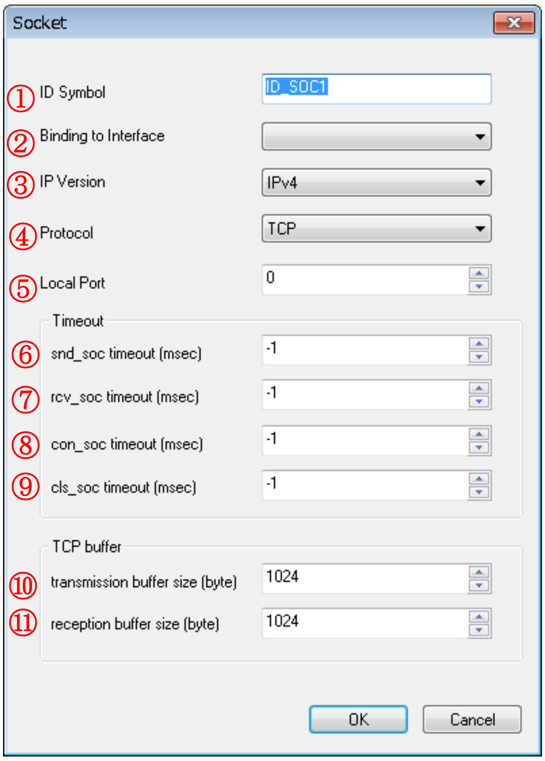
② ADD

Displays editing screen of the socket to add new one.

③ Remove

Remove the selected socket.

【Socket】



Socket

① ID Symbol: ID_SOC1

② Binding to Interface:

③ IP Version: IPv4

④ Protocol: TCP

⑤ Local Port: 0

Timeout

⑥ snd_soc timeout (msec): -1

⑦ rcv_soc timeout (msec): -1

⑧ con_soc timeout (msec): -1

⑨ cls_soc timeout (msec): -1

TCP buffer

⑩ transmission buffer size (byte): 1024

⑪ reception buffer size (byte): 1024

①ID Symbol

Please specify the arbitrary defined name which expresses the ID number of socket. This defined name is defined macro in net_id.h.

② Binding to interface

Please choose interface that is added interface configuration. Socket should be necessary to associate with one network interface. If choose nothing, do not specify network devie when create socket. Regarding to the operation in this case, please refer to 5.4 Socket API.

③ IP version

Please choose ip version of socket IPv4 or IPv6 (only μNet3 IPv6 package can choose IPv6)

④ Protocol

Please choose protocol of socket.

⑤ Local port

Please specify local port number of socket.

⑥ snd_soc timeout

Please specify timeout period of API snd_soc by the unit of millisecond (ms). If specify －1, snd_soc will not return unless it completes successfully or occurs an error. This setting is available only for blocking mode.

⑦ rcv_soc timeout

Please specify timeout period of API rcv_soc by the unit of millisecond (ms). If specify －1, rcv_soc will not return unless it completes successfully or occurs an error. This setting is available only for blocking mode.

⑧ con_soc timeout

Please specify timeout period of API con_soc by the unit of millisecond (ms). If specify －1, con_soc will not return unless it completes successfully or occurs an error. This setting is available only for blocking mode at TCP socket.

⑨ cls_soc timeout

Please specify timeout period of API cls_soc by the unit of millisecond (ms). If specify －1, cls_soc will not return unless it completes successfully or occurs an error. This setting is available only for blocking mode at TCP socket.

⑩ Transmission buffer size

Please specify transmission buffer size of socket.

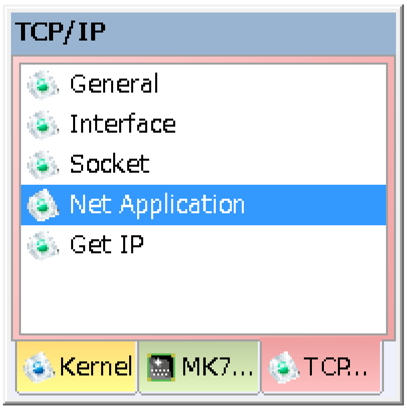
⑪ Reception buffer size

Please specify reception buffer size of socket.

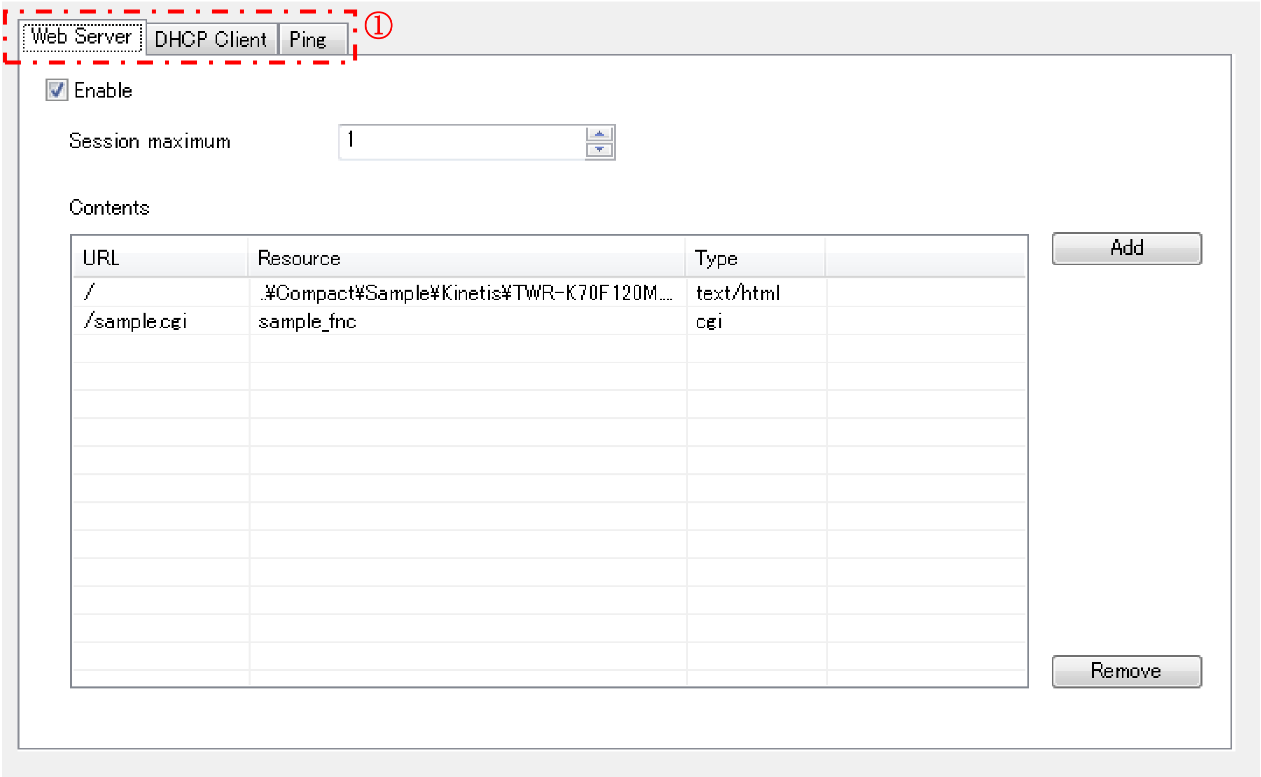
4. 2. 6 Configuration of network application

If click on Net Application on the menu window, the configuration screen of network application provided by µNet3 will appear.

Menu Screen



Configuration Screen



☑ Enable

Session maximum 1

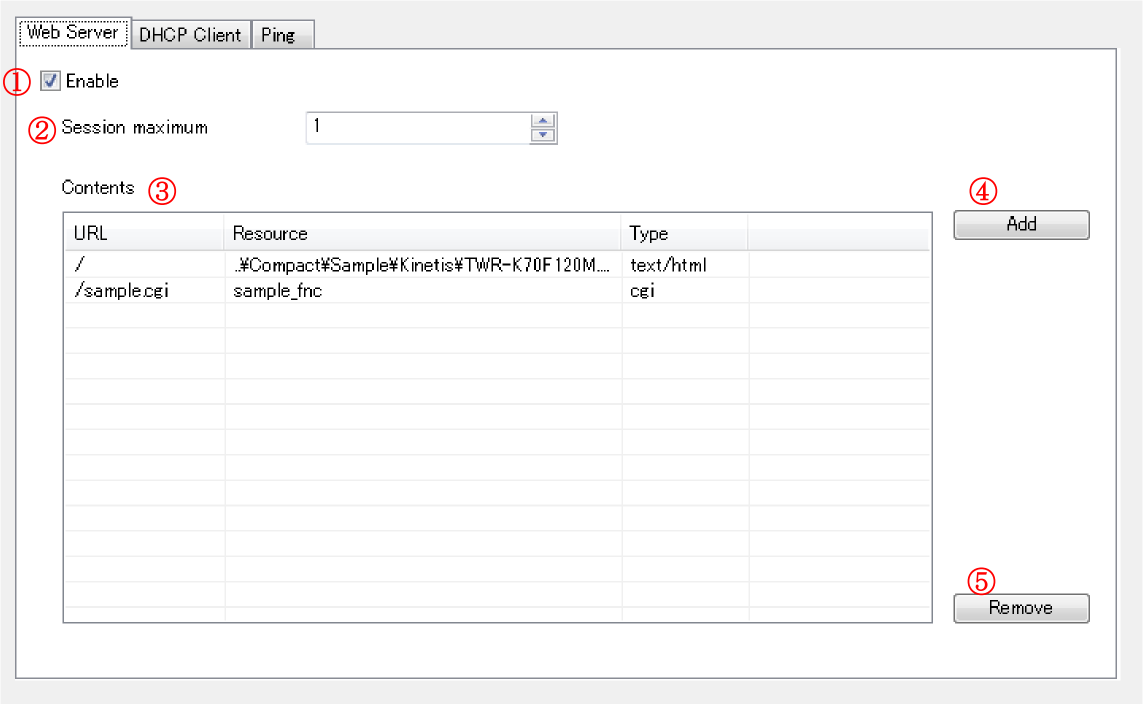
Contents

URL	Resource	Type
/	.¥Compact¥Sample¥Kinetis¥TWR-K70F120M...	text/html
/sample.cgi	sample_fnc	cgi

① Tab of application

Do the configuration for each application.

【Web Server】



☑ Enable

Session maximum: 1

Contents

URL	Resource	Type
/	.¥Compact¥Sample¥Kinetis¥TWR-K70F120M...	text/html
/sample.cgi	sample_fnc	cgi

① Enable

Please specify whether to use WEB server. If check box is “ON”, WEB server is enabled. If “OFF”, it is disable. If WEB server is available, TCP socket used for HTTP will be added automatically.

② Session maximum

Please specify the maximum of session which connect to WEB server.

③ Contents

Now registered contents are displayed. Maximum number of content to be registered is 50. If double click the content of the content list which want to change on the left side of the mouse, the content can be changed.

④ Add

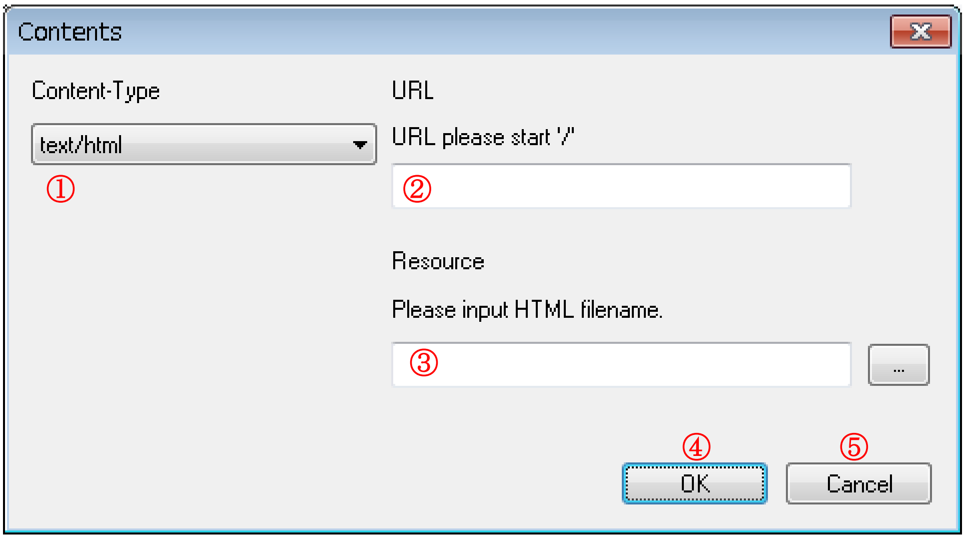
Add new content.

⑤ Remove

Delete content which is selected now.

Addition and modification to content

If click on “Add” button or double click on registered contents registered in the list, the below registration screen will appear.



① Content-Type

Please specify content type (internet media type) to register. Content type will be one of the following options.

text/html

image/gif

image/jpeg

cgi

② URL

Please specify URL of content. Please start to input URL by ‘/’.

(Input example)

In case of text/html　　/index.html

In case of cgi　　/function.cgi　(script name of CGI）

③ Resource

Please specify content resource.

- In case that content type is not cgi: please specify an actual file in accordance with specified Content-Type. Or if click on 「…」 button, the “File selection” screen will appear and then it can specify file there.
- In case content type is cgi : please specify the name of function execute CGI script. Specified function name is outputted to main.c. The follow characters are not included in the

function name.

Prohibition’s characters :   **" ` { } * @ ; + : * , . # $ % & ' ¥ " ! ? ~ ^ = | / ¥ < > ( ) "**

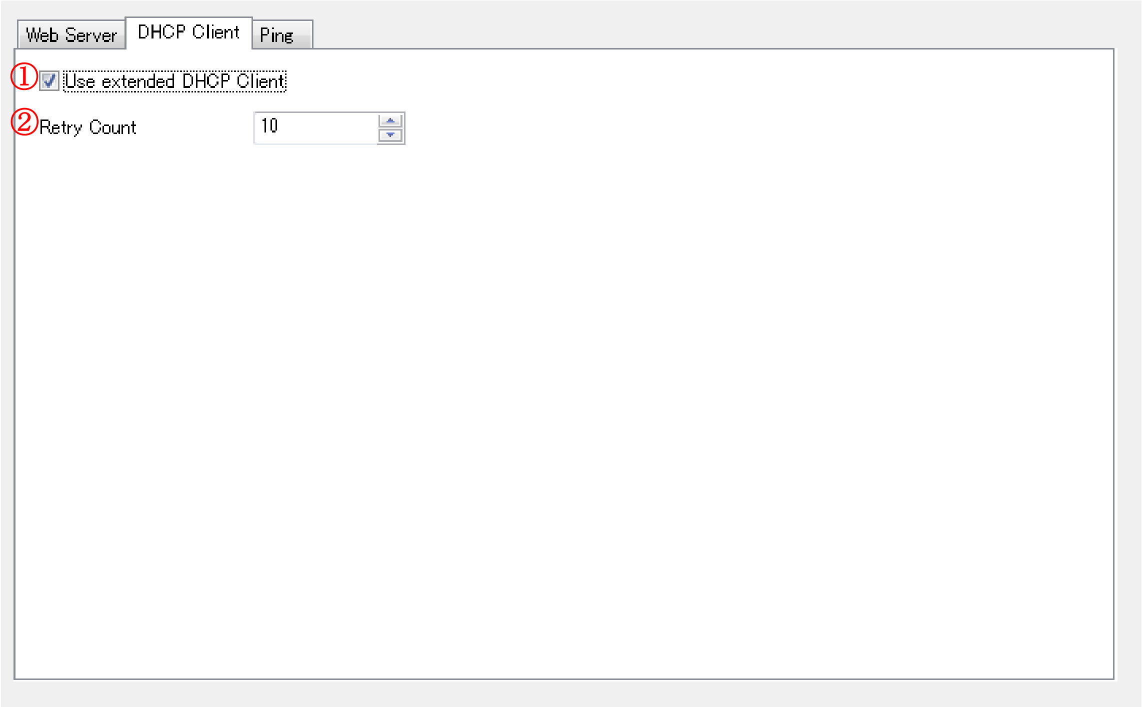
④ **OK**

Register content.

⑤ **Cancel**

Close the screen without register the content

【DHCP Client】



☑ Use extended DHCP Client

Retry Count 10

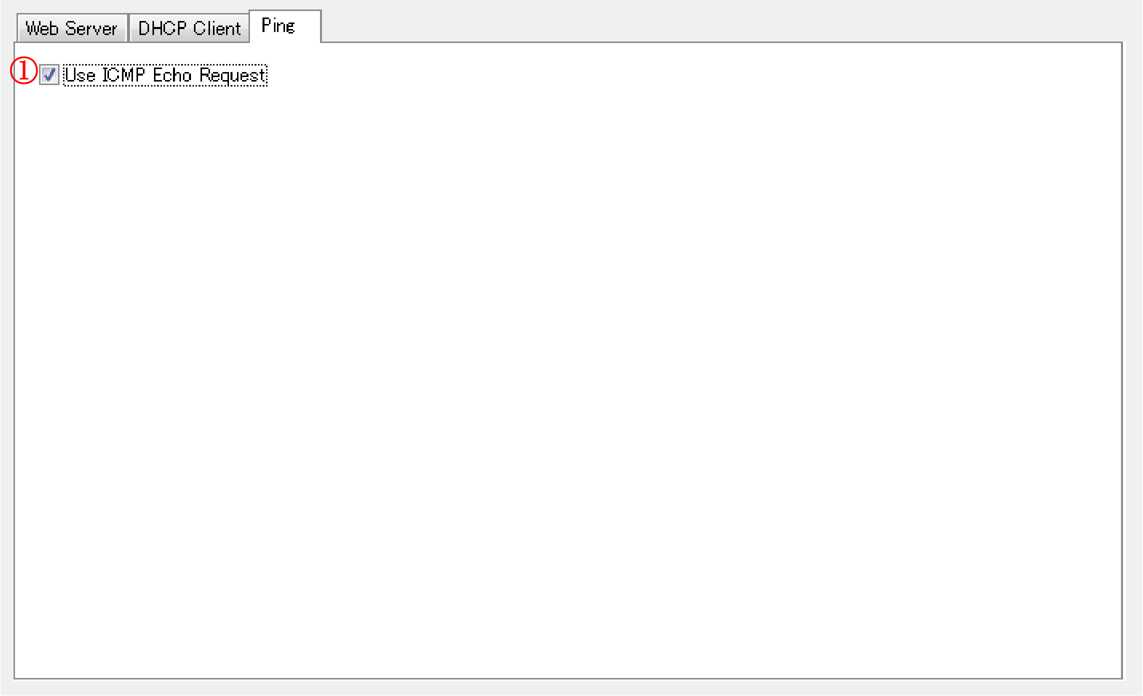
① **Use extended DHCP Client**

Use the extended version of DHCP client. About extensions, Pease refer "DHCPlient Extended Version"network application.

② **Retry Count**

Please set the number of retries, if the DHCP client has timeout.

【Ping】



☑ Use ICMP Echo Request

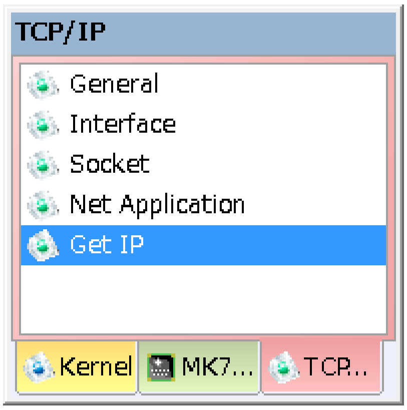
① Use ICMP Echo Request

Create a socket for ICMP echo request (ping)

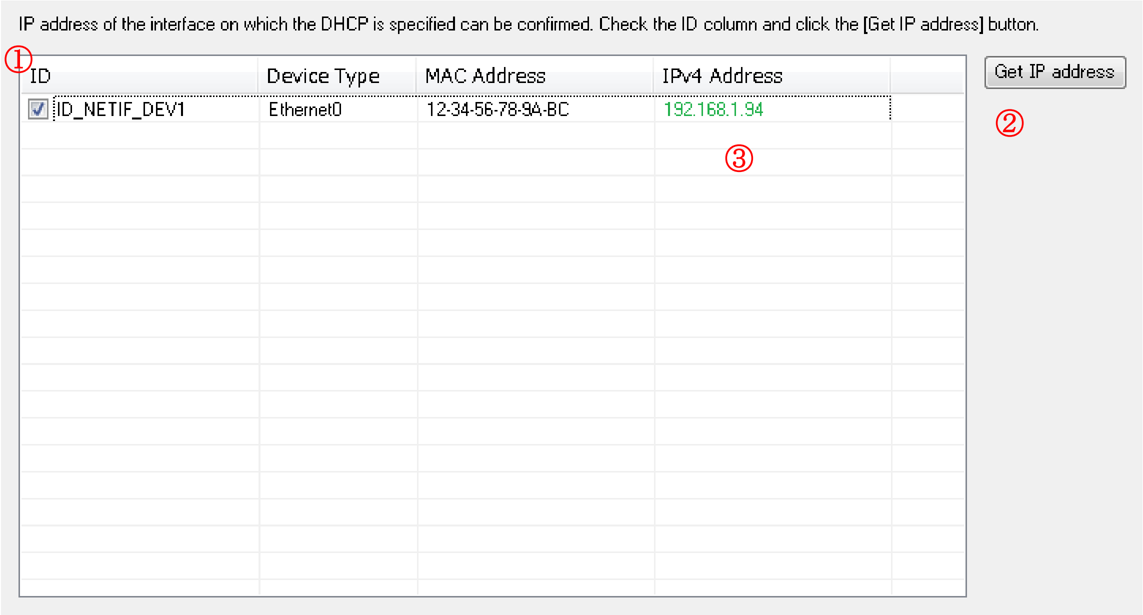
4. 2. 7 Get IP from target

If click on Get IP on the menu window, It is possible to get ip address from target which is enabled state of DHCP client.

Menu Screen



Configuration Screen



IP address of the interface on which the DHCP is specified can be confirmed. Check the ID column and click the [Get IP address] button.

ID	Device Type	MAC Address	IPv4 Address
☑ ID_NETIF_DEV1	Ethernet0	12-34-56-78-9A-BC	192.168.1.94

① List of interface

The interfaces checked will be subject of IP address acquisition.

② Get IP address

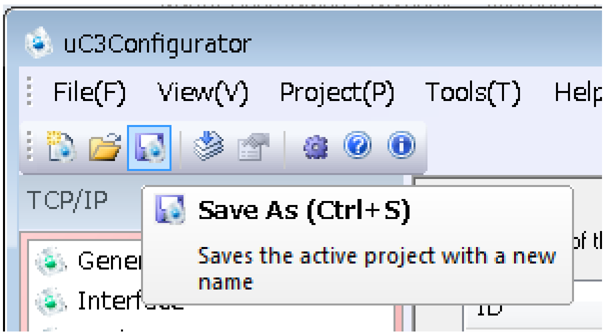
Get the IP address from target.

③ Output screen

Show the result of retrieving IP address.

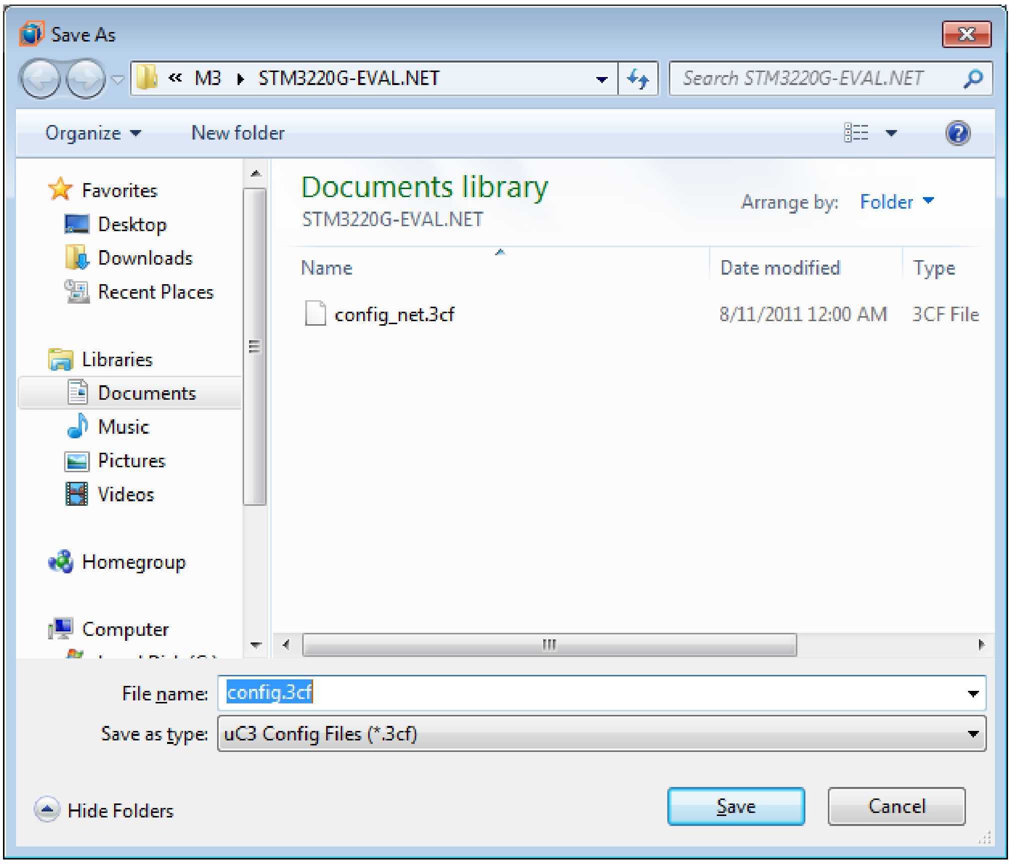
4. 2. 8 Saving project file

From the Configurator toolbar, click “Save As”, open “name and save screen” specify saving folder for project file and click “OK” .



Save As (Ctrl+S)

Saves the active project with a new name



Name	Date modified	Type
config_net.3cf	8/11/2011 12:00 AM	3CF File

File name: config.3cf

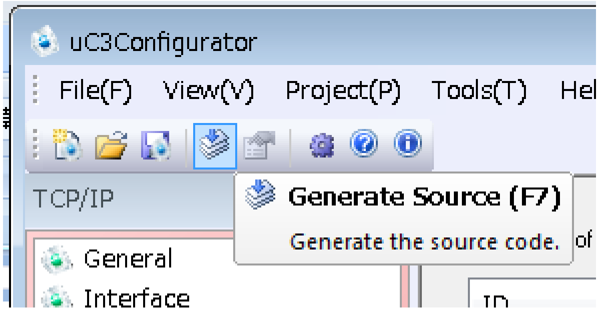
Save as type: uC3 Config Files (*.3cf)

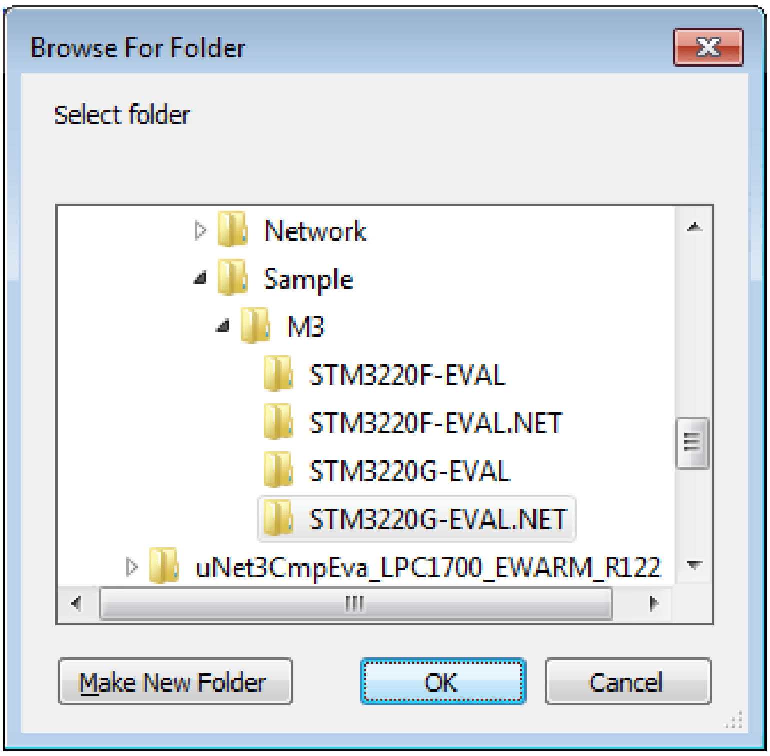
Regarding to the saved file, the file that changed project file (default uC3Project.3cf)and extension to “xml” would be saved.

By opening this file by browser, it is possible to confirm configuration information.

4. 2. 9 Generate source

From the Configurator toolbar, click “Generate Source”, open “screen of referring folder” , specify optional folder which deploy to create file and click “OK” .


uC3Configurator
File(F) View(V) Project(P) Tools(T) Hel
TCP/IP
General
Interface
Generate Source (F7)
Generate the source code.
of
ID



In case there is already skeleton code main.c existing, previous “main.c” is backed up as “main.bak”.

【Recommendation】

In order to prevent skeleton code from being overwritten and deleted, it is recommended not to directly edit to skeleton code but using template to create application program.

4．3　Configuration of μNet3/Standard

In case of using μNet3/Standard, by defining parameters such as IP address and transmission buffer size in net_cfg.c, it will execute configuration of μNet3. net_cfg.c in sample folder is used as template, please change each parameter in accordance with system design.

4．3．1　Configuration list

The following is configurable parameter list.

Configuration definition	Default	Configuration content
CFG_NET_DEV_MAX	1	Number of data link device
CFG_NET_SOC_MAX	10	Maximum number of using socket
CFG_NET_TCP_MAX	5	Maximum number of using TCP socket (※1) (※3)
CFG_NET_ARP_MAX	8	Number of ARP entries
CFG_NET_MGR_MAX	8	Number of multicast entries
CFG_NET_IPR_MAX	2	Number of IP reassembly queue
CFG_NET_BUF_SZ	1576	Network buffer size (※2)
CFG_NET_BUF_CNT	8	Number of network buffers
CFG_NET_BUF_OFFSET	2	Position of writing network buffer data (※2)
CFG_PATH_MTU	1500	MTU size (※2)、(※4)
CFG_ARP_RET_CNT	3	Number of times of ARP retry (※4)
CFG_ARP_RET_TMO	1(sec)	Timeout of ARP retry (※4)
CFG_ARP_CLR_TMO	20(minu)	Clear timeout of ARP cache (※4)
CFG_ARP_PRB_WAI	1(sec)	Time of waiting to start proving
CFG_ARP_PRB_NUM	3	Number of proving.
CFG_ARP_PRB_MIN	1(sec)	Minimum time to send next proving.
CFG_ARP_PRB_MAX	2(sec)	Maximum time to send next proving.
CFG_ARP_ANC_WAI	2(sec)	
CFG_ARP_ANC_NUM	2	
CFG_ARP_ANC_INT	2(sec)	
CFG_IP4_TTL	64	TTL value of IP header (※4)
CFG_IP4_TOS	0	TOS value of IP header (※4)
CFG_IP4_IPR_TMO	10(sec)	Waiting time for IP reassemble packet(※4)
CFG_IP4_MCAST_TTL	1	TTL of IP header(multicast packet)(※4)
CFG_IGMP_V1_TMO	400(sec)	IGMPV1 timeout(※4)
CFG_IGMP_REP_TMO	10(sec)	IGMP report timeout(※4)

Configuration definition	Default	Configuration content
CFG_TCP_MSS	1460	MSS(※4)
CFG_TCP_RTO_INI	3(sec)	Initial value of TCP retry timeout (※4)
CFG_TCP_RTO_MIN	500(msec)	Minimum value of TCP retry timeout(※4)
CFG_TCP_RTO_MAX	60(sec)	Maximum value of TCP retry timeout(※4)
CFG_TCP_RTO_MAX	60(sec)	Maximum value of TCP retry timeout(※4)
CFG_TCP_SND_WND	1024	Transmission buffer size(※3),(※4)
CFG_TCP_RCV_WND	1024	Reception buffer size (Window size)(※4)
CFG_TCP_DUP_CNT	4	Duplicate ACK number of retry beginning (※4)
CFG_TCP_CON_TMO	75(sec)	SYN timeout(※4)
CFG_TCP_SND_TMO	64(sec)	Transmission timeout (※4)
CFG_TCP_CLS_TMO	75(sec)	FIN timeout(※4)
CFG_TCP_CLW_TMO	20(sec)	Close Wait timeout (※4)
CFG_TCP_ACK_TMO	200(msec)	ACK timeout (※4)
CFG_TCP_KPA_CNT	0	KeepAlive notification number before disconnecting
CFG_TCP_KPA_INT	1(sec)	Notification interval after the start of the KeepAlive
CFG_TCP_KPA_TMO	7200(sec)	Non-communication time until the start of the KeepAlive
CFG_PKT_RCV_QUE	1	Queuing number of reception packet (※4)
CFG_PKT_CTL_FLG	0	Flag to disable checksum for reception packet

※1 It is necessary under CFG_NET_SOC_MAX. Besides, difference part with CFG_NET_SOC_MAX becomes maximum limit number of non-TCP socket.

※2 Network buffer size must be bigger than combined size between MTU, data link header size (14 bytes in case of Ethernet) and data writing position in buffer size of network management structure (44 bytes).

※3 Transmission buffer of TCP uses global variable UB gTCP_SND_BUF[ ] in common without regard to the using device. gTCP_SND_BUF[ ] determines this size by CFG_TCP_SND_WND×CFG_NET_TCP_MAX. If the uNet3 uses various devices in TCP transmission buffer size, it needs to set gTCP_SND_BUF[ ] in accordance with that maximum value.

※4 It's possible to set using device units. We set device number -1 as index in gNET_CFG[ ].

4. 3. 2 IP address

Set up IP address. Because every network device needs an IP address, please register CFG_NET_DEV_MAX part for IP address.

The following is the example for set up of IP address:192.168.1.10, gateway : 192.168.1.1, subnet mask: 255.255.255.0.

```
T_NET_ADR gNET_ADR[] = {
{   /*   for Device 1    */
       0x0,                  /* Necessarily specify 0 */
       0x0,                  /* Necessarily specify 0*/
0xC0A8000A,       /*Setting IP address 192.168.1.10 */
0xC0A80001,       /* gateway 192.168.1.1        */
0xFFFFFF00,       /* subnet mask 255.255.255.0 */
   }
};
```

4. 3. 3 Device Driver

Set device driver. Please register CFG_NET_DEV_MAX part for device driver.

```
T_NET_DEV gNET_DEV[] = {
   { ..}
}
```

Please refer to **3. 2 Network device driver** for more details.

4. 3. 4 Information table of protocol stack

Set global variable of protocol stack as below.

```
const VP net_inftbl[] = {
   0,                                    /* Necessarily specify 0*/
   (VP)gNET_SOC,           /* Set NULL in case of not using socket */
   (VP)gNET_TCP,           /* Set NULL in case of not using socket*/
   (VP)gNET_IPR,           /* Set NULL in case of not using IP reassembly function */
   (VP)gNET_MGR,           /* Set NULL in case of not using IGMP*/
   (VP)gTCP_SND_BUF,    /* Set NULLin case of not using TCP socket*/
};
```

4. 3. 5 µC3 resource

The µNet3 uses the below kernel objects.

c_net_tsk	task	Use in timer event of TCP/IP
c_net_sem	Semaphore	Use in TCP exclusive control
c_net_mpf	Memory pool	Use in network buffer

Chapter 5: Description of application programming interface

5. 1 Initialization of protocol stack

To use TCP/IP protocol stack,the initialization of protocol stack and initialization of network device are needed. Basically initialization is as follows:

Example of initialization code

```
/* Initialization of protocol stack */
        ercd     =     net_ini();
if (ercd != E_OK) {
         return ercd;
        }
/* Network device initialization (device number N) */
        ercd     =     net_dev_ini(N);
If (ercd != E_OK) {
return ercd;
        }
```

Initialization code is executed by net_setup function of net_cfg.c.

5. 2 Network Interface API

net_ini Initialization of TCP/IP protocol stack

【Format】

ER ercd = net_ini(void);

【Parameter】

no

【Return value】

ER	ercd	Successful completion（E_OK）or error code

【Error code】

< 0	Initialization failure

【Explanation】

Initialize the resource to be used by protocol stack. Kernel objects (tasks, memory pools, semaphores) to be used by protocol stack are also created and initialized simultaneously. Besides, the initial value is set in global variable used in protocol stack.

In case of using protocol stack, it needs to issue this API prior to any other API.

net_cfg Parameter setting of network interface

【Format】

ER ercd = net_cfg(UH num, UH opt, VP val);

【Parameter】

UH	num	Device number
UH	opt	Parameter code
VP	val	Value to set

【Return value】

ER	ercd	Successful completion（E_OK）or error code

【Error code】

E_NOSPT	Wrong parameter code
E_ID	Wrong device number
E_NOMEM	Too many multicast table

【Explanation】

Set up forIP address and subnet mask, broadcast address, multicast and others.

Setting sample
net_cfg(1, NET_BCAST_RCV, (VP)1);　/* enable broadcast reception */

Parameter code	Data type	Meaning
NET_IP4_CFG	T_NET_ADR	Set IP address, subnet mask, gateway. Please hand pointer of T_NET_ADR to val.
NET_IP4_TTL	UB	Set TTL (Time to Live） Default is set 64.
NET_BCAST_RCV	UB	Set whether to receive broadcast or not. Set 1 to receive and 0 to not receive.
NET_MCAST_JOIN	UW	Register IP address of multicast group to join
NET_MCAST_DROP	UW	Set IP address of multicast group to drop
NET_MCAST_TTL	UB	Set TTL to be used in multicast transmission
NET_ACD_CBK	Callback function pointer	In this field, set the callback function to notify that it has detected an IP address conflict during operation. Notification feature is enabled by this setting of conflict detection.

net_ref　　　Reference parameter for network interface

【Format】

ER ercd = net_ref(UH num, UH opt, VP val);

【Parameter】

UH	num	Device number
UH	opt	Parameter code
VP	val	Pointer to buffer of the value to get

【Return value】

ER	ercd	Successful completion（E_OK）or error code

【Error code】

E_NOSPT	Wrong parameter code
E_ID	Wrong device number

【Explanation】

Verify the basic setting of IP address and subnet mask, broadcast address and others.

Setting sample

```
UB bcast;
net_ref(1, NET_BCAST_RCV, (VP)&bcast);   /* reception status of broadcast */
```

Parameter code	Data type	Meaning
NET_IP4_CFG	T_NET_ADR	Get IP address, subnet mask, gate-way. Please hand pointer of T_NET_ADR to val
NET_IP4_TTL	UB	Get TTL（Time to Live）.
NET_BCAST_RCV	UB	Get the receive status of broadcast
NET_MCAST_TTL	UB	Get TTL broadcast transmission

net_acd Detection IP Address Confliction

【API】

ER ercd = net_acd(UH dev_num, T_NET_ACD *acd);

【Parameter】

UH	dev_num	Deviec Number
T_NET_ACD	*acd	Address Conflict Information

【Return Value】

ER	ercd	Success (E_OK) or Error Code

【Error Code】

E_ID	Illegal Device Number
E_PAR	Illegal Parameter
E_OBJ	Call Duplicate or Call host IP undefined
E_TMOUT	Time out ARP transmit
E_SYS	Detect IP address conflict
E_OK	No Detect IP address conflict

【DESCRIPTION】

This API is done in the device specified by dev_num, IP address conflict detection.

If it detects a conflict for the IP address, MAC address of the other party is stored in the conflict information of the argument.

If you want to detect conflicts in asynchronous IP address separately, you will need to register a callback function () API net_cfg with this API.

Note : Function is recommended that a maximum of about 10 seconds, call a dedicated task to attempt the detection of address conflicts.

acd_cbk Callback IP address conflict detected.

【API】

ER acd_cbk(T_NET_ACD* acd);

【Return Value】

ER	ercd	Success (E_OK) or Error Code

【Parameter】

T_NET_ACD	*acd	Address Conflict Information

【DESCRIPTION】

This function is called when it detects an IP address conflict during operation. The conflict is stored what conflict information of the argument MAC address of the host that.
If the IP address for the conflict, continue to use the IP address in the host itself, please return the E_OK. Otherwise, please return the E_SYS.

Called on the task that received the ARP packet (task received Ethernet driver) callback function. The callback function should therefore be terminated immediately. There is no callback function that is called (running () net_acd) detection in IP address.

Use Casee

```
/* Callback function at the time of detecting address conflicts */
ER acd_detect(T_NET_ACD * acd)
{
    return E_OK;
}

/* Network Initialize Function */
ER net_setup(void)
{
    ER ercd;
    T_NET_ACD acd;

    ercd = net_ini();
    if (ercd != E_OK) {
        return ercd;
    }
    ercd = net_dev_ini(ID_DEVNUM_ETHER);
    if (ercd != E_OK) {
        return ercd;
    }

    /* Detect IP Address Conflict */
    ercd = net_acd(ID_DEVNUM_ETHER, &acd);
    if (ercd == E_OK) {
        /* No Information IP Address conflict */
        /* Callback function at the time of detecting IP address conflicts */
        net_cfg(ID_DEVNUM_ETHER, NET_ACD_CBK, (VP)acd_detect);
    }
    else if (ercd == E_SYS) {
        /* MAC address is conflict a host of acd.mac IP address */
    } else {
        /* Failed to detect IP address conflict */
    }
    return ercd;
}
```

5．3　Network Device Control API

The Network Device Control API provides interface to access unifiedly from application to device driver. For each device, it specifys a device number to access this API. Device number is the pecific number to identify the device.

net_dev_ini　　Network device initialization

【Format】

ER ercd = net_dev_ini(UH dev_num);

【Parameter】

UH	dev_num	Device number

【Return value】

ER	ercd	Successful completion（E_OK）or error code

【Error code】

< 0	Initialization failure

【Explanation】

Use dev_num to initialize a specific device. In fact, net_dev_ini uses dev_ini of driver device to initialize the device.

If it completes normally, it can handle the packet through that network device.

net_dev_cls Release network device

【Format】

ER ercd = net_dev_cls(UH dev_num);

【Parameter】

UH	dev_num	Device number

【Return value】

ER	ercd	Successful completion（E_OK）or error code

【Error code】

< 0	Release failure

【Explanation】

Release specific device by using dev_num. In fact, net_dev_cls will release device by using dev_cls of device driver.

net_dev_ctl Network device control

【Format】

ER ercd = net_dev_ctl(UH dev_num, UH opt, VP val);

【Parameter】

UH	dev_num	Device number
UH	opt	Control code
VP	val	Value to set

【Return value】

ER	ercd	Successful completion（E_OK）or error code

【Error code】

< 0	Failure

【Explanation】

Control the specific device by using dev_num. Because net_dev_ctl only calls dev_ctl of device driver, the actual actions depend much on intergration of driver device.

net_dev_sts Acquire the status of network device

【Format】

ER ercd = net_dev_sts(UH dev_num, UH opt, VP val);

【Parameter】

UH	dev_num	Device number
UH	opt	Status code
VP	val	Getting value

【Return value】

ER	ercd	Successful completion（E_OK）or error code

【Error code】

< 0	Failure

【Explanation】

Acquire the status of specific device by using dev_num. Because net_dev_sts only calls dev_ref of device driver, the detailed action depends on intergration of device driver.

5. 4 Socket API

Application uses socket API to exchange TCP/UDP data with remote host.

When creating socket or connecting, it's necessary to use device number to specify network device to connect. In case specify device number 0, it means that "Don't specify device", the interface selection action between socket and network device is different depending on transmission or reception. Besides, when creating socket, if it specify device number beside 0, it don't need to specify device number when connecting.

For example, in the system is constructed by N nertwork devices (N is more than 2) with the using socket APIs, show in the below it.

	Device number when creating (※1)	**Device number when connecting (※2)**	**Device in use**
Socket transmission action con_soc() of snd_soc() and TCP client (SYN transmission)	0	0	Device number 1(top)
	0	N	Device number N
	N	ANY	Device number N
Socket reception action con_soc() of rcv_soc() and TCP server (SYN reception)	0	0	Notified device (※3)
	0	N	Device number N
	N	ANY	Device number N

※1 In case of μNet3/Compact, when adding socket by the configurator, we specify network device. In case of Standard, it specify by host->num argument of con_soc() API.

※2 Specify by host->num argument of con_soc() API. In case of receiving through UDP socket, do not need to call con_soc() API.

※3 The socket which does not specify device number even when creating or connecting socket, if port number and protocol are matched, it can receive packet from any device. The socket in this case uses the device notified packet in subsequent operation.

cre_soc Socket generation

【Format】

ER ercd = cre_soc(UB proto, T_NODE *host);

【Parameter】

UH	proto	Protocol type
T_NODE	*host	Information of local host

【Return value】

ER	ercd	ID of generated socket（>0）or error code

【Error code】

E_NOMEM	Unable to generate socket (exceed maximum number of socket)
E_PAR	‘host’ is wrong
E_NOSPT	’proto’ is wrong

【T_NODE】

Specify local port number and device interface to use.

UH	port	Port number	Port number of local host. Specify the value from 1 to 65535 or PORT_ANY. In case that PORT_ANY is specified, it will determine port number by protocol stack.
UH	ver	IP version	Specify 0（use IP_VER4）
UB	num	Device number	Specify device number of the device it want to use
UW	ipa	IP address	Specify 0（use local IP address）

【proto】

Protocol type of socket to create

IP_PROTO_TCP	TCP socket
IP_PROTO_UDP	UDP socket

【Explanation】

This API creates the socket of specified protocol.

Example of creating TCP socket

```
T_NODE　host；
host.num = 1；
host.port = 7；
host.ver = IP_VER4；
host.ipa = INADDR_ANY；
cre_soc(IP_PROTO_TCP, &host)；
```

※Socket creating function is a function provided for Standard version only. In case of using Compact version, it is necessary to define socket by the configurator.

del_soc Delete socket

【Format】

```
ER ercd = del_soc(UH sid);
```

【Parameter】

UH	sid	ID is used to identify socket

【Return value】

ER	ercd	Successful completion (E_OK）or error code

【Error code】

E_ID	Wrong ID number
E_NOEXS	Socket does not exist (Socket has not been created yet)
E_OBJ	Status of socket is wrong

【Explanation】

This API deletes the specified ID socket. When delete TCP socket, please call cls_soc() in advance and close the socket.

※Socket deleting function is a function provided only for Standard version. In case of using Compact version, can not delete the socket dynamically.

con_soc Socket connection

【Format】

ER ercd = con_soc(UH sid, T_NODE *host, UB con_flg) ;

【Parameter】

UH	sid	ID is used to identify socket
T_NODE	*host	Information of remote host
UB	con_flg	Connection mode

【Return value】

ER	ercd	Succesful completion (E_OK） or error code

【Error code】

E_ID	Wrong ID number
E_NOEXS	Socket does not exist (Socket has not been created yet)
E_PAR	Host or con_flg is wrong
E_OBJ	Socket status is wrong (for example, calling this API for the socket which has already connected）
E_TMOUT	Connection process is out of time
E_WBLK	Processed by non-blocking mode
E_CLS	Refuse the connection from remote host
E_RLWAI	Connection process is interrupted
E_QOVR	con_soc() is already executing

【T_NODE】

Specify remote host and device interface to use.

UH	port	Port number	Port number of remote host （from 1 to 65535）
UH	ver	IP version	Specify 0
UB	num	Device number	Device number of the device that wants to use
UW	ipa	IP address	IP address of remote host

【con_flg】

Specify the waiting for connection (server) or the acctive connection (client).

Usually specify 0 for UDP socket..

SOC_CLI	Connect to remote host (active connection)
SOC_SER	Wait for connection (passive connection)

【Explanation】

This API has different behavior depending on using protocol.

In the course of TCP, establish the connection to remote host, in case of UDP, accociate the socket with the destination of data transmission.

An example for the connection of TCP server socket

```
T_NODE　remote = {0};                    /* clear by 0 */
con_soc(ID, &remote, SOC_SER);
```

An example for the connection of TCP client socket

```
T_NODE　remote;
remote.port = 100;                      /* Port number of remote host */
remote.ver = IP_VER4;
remote.num = 1;                         /*Specify device number to use */
remote.ipa = ip_aton(“192.168.11.1”); /*　IP address of remote host */
con_soc(ID, &remote, SOC_CLI);
```

cls_soc Socket disconnetion

【Format】

ER ercd = cls_soc(UH sid, UB cls_flg);

【Parameter】

UH	sid	ID is used to identify socket
UB	cls_flg	Disconnetion mode

【Return value】

ER	ercd	Successful completion（E_OK）or error code

【Error code】

E_ID	Wrong ID number
E_NOEXS	Socket does not exist (Socket has not been created yet)
E_PAR	‘cls_flg is wrong
E_OBJ	Socket status is wrong（Such as when calling this API in the status of disconnection）
E_TMOUT	Disconnection process is out of time
E_WBLK	Processed by non-blocking mode
E_CLS	Forced ternimation of the connection from remote host
E_RLWAI	Disconnection process is interrupted
E_QOVR	cls_soc() is executing already

【cls_flg】

This parameter is only valid for TCP socket

SOC_TCP_CLS	Disconnect socket (End the connection)
SOC_TCP_SHT	Disable transmission process only. Reception is possible. (In case that want to stop the connection completely after using SOC_TCP_SHT to discontinue the transmission process only, it need to use SOC_TCP_CLS to disconnect completely.）

【Explanation】

This API has different behavior depending on using protocol.
In case of TCP, stop the connection to remote host, in case of UDP, clear the information of the destination or the source of data associated with socket. (After that, it can not send UDP data).

cfg_soc　Set parameter of socket

【Format】

```
ER ercd = cfg_soc(UH sid, UB code, VP val) ;
```

【Parameter】

UH	sid	ID is used to identify socket
UB	code	Parameter code
VP	val	Value to set

【Return value】

ER	ercd	Successful completion（E_OK）or error code

【Error code】

E_ID	Wrong ID number
E_NOEXS	Socket does not exist (Socket has not beed created yet)
E_NOSPT	Wrong parameter code
E_PAR	Wrong parameter value
E_OBJ	Status of socket is wrong

【Explanation】

This API can set parameter as below. Please cast and then hand the setting value to VP type.

Example of setting

```
UB ttl = 32;
cfg_soc(ID, SOC_IP_TTL, (VP)ttl);
```

Parameter code	Data type	Meaning
SOC_TMO_CON	TMO	Calling timeout of con_soc
SOC_TMO_CLS	TMO	Calling timeout of cls_soc
SOC_TMO_SND	TMO	Calling timeout of snd_soc
SOC_TMO_RCV	TMO	Calling timeout of rcv_soc
SOC_IP_TTL	UB	Set TTL of IP header（Time to Live）
SOC_IP_TOS	UB	Set TOS of IP header (Type of Server）
SOC_CBK_HND	Pointer for function	Register callback function
SOC_CBK_FLG	UH	Set bit pattern of callback event flag (Value to set is as below)
SOC_PRT_LOCAL	UH	Change Local Port Number

Callback event flag bit	Meaning
EV_SOC_CON	Set con_soc() in non-blocking mode （only TCP socket）
EV_SOC_CLS	Set cls_soc() in non-blocking mode （only TCP socket）
EV_SOC_SND	Set snd_soc() in non-blocking mode
EV_SOC_RCV	Set rcv_soc() in non-blocking mode

Regarding callback event flag bit, it can set a multiple bit. In case of setting it multiple, set by OR. An example of setting is as below.

Ex: ercd = cfg_soc(ID socket, SOC_CBK_FLG, (VP)(EV_SOC_CON|EV_SOC_SND|EV_SOC_RCV|EV_SOC_CLS));

Socket event set in non-blocking disables socket timeout of that event.

When enable callback event flag bit, it is necessary to register callback function in SOC_CBK_HND. Regarding callback function, please refer to the following.

soc_cbt Callback function

【Format】

UW soc_cbt(UH sid, UH event, ER ercd);

【Parameter】

UH	sid	ID is used to identify socket
UH	event	Callback event flag bit
ER	ercd	Error code

This callback function is called out from TCP/IP stack. However, in case of execution API socket of non-blocking mode, if API process is necessary to enter the waiting sate, it will return E_WBLK value without enter. This time, it will be notified from TCP/IP stack that the process of callback function has completed.

Call back event flag bit (event)	**Error code (ercd)**	**Meaning**
EV_SOC_CON	E_OK	con_soc() process completes normally
	< 0	con_soc() process completes with error. Regarding error content of this time, please refer to error code of con_soc().
EV_SOC_CLS	E_OK	cls_soc() process completes normally
	< 0	cls_soc() process complete with error. Regarding error content of this time, please refer to error code of cls_soc().
EV_SOC_SND	> 0	UDP socket： snd_soc()process completes normally TCP socket： In case of TCP transmission buffer is available, it will show the available size by 'ercd' value. Again, snd_soc() is called and then it can copy transmission data into TCP transmission buffer.
	<= 0	snd_soc() process completes normally. Regarding error content of this time, please refer to error code of snd_soc().
EV_SOC_RCV	> 0	UDP socket： There is receipt data in UDP socket. Shows receipt data size by 'ercd' value. Again, it can call rcv_soc() to receive data. TCP socket： There exists receipt data in TCP socket. Shows receipt data size by 'ercd' value. Again, it can call rcv_soc() to receive data.
	<= 0	rcv_soc() process completes normally. Regarding error content of this time, please refer to error code of rcv_soc().

※Prohibit tocall all API functions of μNet3 from callback function. (Please think callback unction the same as interrupting handler and use it).

ref_soc Refer parameter of socket

【Format】

ER ercd = ref_soc(UH sid, UB code, VP val) ;

【Parameter】

UH	Sid	ID is used to identify socket
UB	Code	Parameter code
VP	val	Pointer for buffer of the value to get

【Return value】

ER	ercd	Successful completion (E_OK）or error code

【Error code】

E_ID	Wrong ID number
E_NOEXS	Socket does not exist (socket has not been created yet)
E_NOSPT	Wrong parameter code
E_PAR	Wrong parameter value (in case that val is NULL)
E_OBJ	Socket status is wrong (Can not refer to socket)

【Explanation】

Refer the parameters as below. Please cast and then hand the setting value to VP type.

Example of getting remote host information

```
T_NODE remote;
ref_soc(ID, SOC_IP_REMOTE, (VP)&remote);
```

Parameter code	Data type	Meaning
SOC_TMO_CON	TMO	Calling timeout of con_soc
SOC_TMO_CLS	TMO	Calling timeout of cls_soc
SOC_TMO_SND	TMO	Calling timeout of snd_soc
SOC_TMO_RCV	TMO	Calling timeout of rcv_soc
SOC_IP_LOCAL	T_NODE	Getting Port number and IP address of local host
SOC_IP_REMOTE	T_NODE	Getting port number and IP address of remote host
SOC_IP_TTL	UB	Getting TTL（Time to Live）
SOC_IP_TOS	UB	Getting TOS (Type Of Service)
SOC_RCV_PKT_INF	T_RCV_PKT_INF	Getting newest information of packet received by socket (in case of TCP, it’s unable to get)
SOC_PRT_LOCAL	UH	Reference Loca Port Number

For the socket having both multicast address and unicast address, to know IP address of the last received packet, please refer to the below.

Example of getting received IP address

```
T_RCV_PKT_INF rcv_pkt_inf;
ref_soc(ID, SOC_RCV_PKT_INF, (VP)&rcv_pkt_inf);
if(rcv_pkt_inf.dst_ipa == MULTICASTADDRESS) {
    /* received by multicast address */
}
```

abt_soc Abort the socket process

【Format】

ER ercd = abt_soc(UH sid, UB code);

【Parameter】

UH	sid	ID is used to identify socket
UB	code	Control code

【Return value】

ER	ercd	Successful completion (E_OK) or error code

【Error code】

E_ID	Wrong ID number
E_NOEXS	Socket does not exist (socket has not been created yet)
E_NOSPT	Wrong control code
E_OBJ	Socket status is wrong

【Explanation】

This API can cancel waiting status of con_soc, cls_soc, snd_soc, rcv_soc. Cancelled API returns E_RLWAI .

Control code	Meaning
SOC_ABT_CON	Discontinuation of con_soc() process
SOC_ABT_CLS	Discontinuation of cls_soc() process
SOC_ABT_SND	Discontinuation of snd_soc() process
SOC_ABT_RCV	Discontinuation of rcv_soc() process
SOC_ABT_ALL	Process discontinuation of all sockets

snd_soc Data transmission

【Format】

ER ercd = snd_soc(UH sid, VP data, UH len);

【Parameter】

UH	sid	ID is used to identify socket
VP	data	Pointer to the transmit data
UH	len	Size of the transmit data

【Return value】

ER	ercd	Actual transmitted data size (>0) or error code

【Error code】

E_ID	Wrong ID number
E_NOEXS	Socket does not exist (socket has not been created yet)
E_PAR	Wrong transmitted data or data size for transmission is not specified.
E_OBJ	Socket status is wrong
E_TMOUT	Transmission process is out of time
E_WBLK	Processed by non-blocking mode
E_CLS	Forced termination of the connection from remote host
E_RLWAI	Transmission process is interrupted
E_NOMEM	Memory is not enough
E_QOVR	snd_soc() is executing already
EV_ADDR	Unknown default gateway

【Explanation】

This API transmits data to remote host. When the process succeeds, it will return the actual transmitted data size. Besides that case, it will return error code.

In case of TCP socket, this API will copy data into protocol stack inside, and return that copied size. (Returned data size is less than len specified by argument）. Please refer to“3．1．4 TCP module” for details.

In case of UDP socket, data will be transmitted to network and return that transmitted size. Please refer to “3．1．3 UDP module” for details.

rcv_soc Data reception

【Format】

ER ercd = rcv_soc(UH sid, VP data, UH len);

【Parameter】

UH	sid	ID is used to identify socket
VP	data	Pointer to receipt data
UH	len	Receipt data size

【Return value】

ER	ercd	Actual received data size (>0) or error code

【Error code】

E_ID	Wrong ID number
E_NOEXS	Socket does not exist (socket has not been created yet)
E_PAR	Wrong receipt data or receipt data size is not specified.
E_OBJ	Socket status is wrong
E_TMOUT	Receipt process timed out
E_WBLK	Processed by non-blocking mode
E_CLS	Forced termination of the connection from remote host
E_RLWAI	Reception process is interrupted
E_QOVR	rcv_soc() is executing already
0	The connection is disconnected

【Explanation】

This API receive data which is sent from remote host.

In case of TCP, the maximum receivable size to receive is “Reception buffer size” specified by the configurator. Please refer to “3.1.4 TCP module” for details.

In case of UDP, the maximum receivable size to receive is 1472 bytes (MTU default – IP header size – UDP header size). Please refer to “3.1.3 UDP module” for details.

5．5 Other API

htons Convert 16 bit value to network byte order

【Format】

UH htons(UH val);

【Parameter】

UH	Val	16 bit value host byte order

【Return value】

UH	16 bit value to network byte order

htonl Convert 32 bit value to network byte order

【Format】

UW htonl(UW val);

【Parameter】

UW	val	32 bit value host byte order

【Return value】

UW	32 bit value network byte order

ntohs Convert 16 bit value to host byte order

【Format】

UH ntohs(UH val);

【Parameter】

UH	val	16-bit value network byte order

【Return value】

UH	16-bit value host byte order

ntohl Convert 32 bit value to host byte order

【Format】

UWntohl(UW val);

【Parameter】

UW	val	32 bit value network byte order

【Return value】

UW	32 bit value host byte order

ip_aton	Convert an IPv4 address string in dot-notation to 32 bit value

【Format】

UW ip_aton(const char *str);

【Parameter】

char *	str	Pointer to IPv4 address string in dot-notation

【Return value】

UW	> 0	Successful completion (32 bit value after converting)

【Error code】

0	Wrong IP address is specified

ip_ntoa	Convert 32-bit value IPv4 address to IPv4 address string in dot-notation

【Format】

void ip_ntoa(const char *str, UW ipaddr);

【Parameter】

char *	str	Pointer that accepted IP address string after converting
UW	ipaddr	32-bit value IP address

【Return value】

None

【Explanation】

If the process completes successfully, the character string will be set in str, but str will be NULL if the error occurs.

ip_byte2n Convert IPv4 address array to 32 bit value

【Format】

UW ip_byte2n(UB *ip_array);

【Parameter】

UB *	ip_array	Pointer to byte value array of IP address

【Return value】

UW	> 0	Successful completion (32-bit value after converting）

【Error code】

0	Wrong IP address is specified

ip_n2byte Convert 32 bit value IPv4 address to array

【Format】

void ip_n2byte(UB *ip_arry, UW ip);

【Parameter】

UB *	ip_arry	Pointer to byte value array of IP address
UW	ip	32-bit value IP address

【Return value】

None

【Explanation】

After completing successfullyl, value is set in asip_array. In case of error, ip_array turns into NULL.

Chapter 6: Network application

6．1 DHCP client

DHCP client obtains IP address information which is used in network from DHCP server. Acquired IP address is assigned to the host.

USE THIS FUNCTION OF RENEW, RELEASE, DECLINE, FEATURES INFORM. FOR DHCP EXTENDED VERSION, SEE THE EXTENDED VERSION 6.5 DHCP CLIENT.。

（1）Host address information

```
typedef struct t_host_addr {
        UW      ipaddr;              /* IP address */
        UW      subnet;              /* Subnet Mask */
        UW      gateway;             /* Gateway*/
        UW      dhcp;                /* DHCP Server address */
        UW      dns[2];              /* DNS Address */
        UW      lease;               /* Lease period of the DHCP address*/
        UW      t1;                  /* Renewal period of DHCP address*/
        UW      t2;                  /* Rebind period of DHCP address */
        UB      mac[6];              /* MAC address */
        UB      dev_num;             /* Device address */
        UB      state;               /* DHCP Cliente status*/
        UH      socid;               /* ID of UDP socket*/
} T_HOST_ADDR ;
```

This structure is used as an argument of DHCP client API. **Device numbe**r and **ID of UDP socket** have to be set by user application. The remaining parameters are set by the response from a DHCP server.

ID of UDP socket

In DHCP client, use UDP socket. UDP socket must be created by the following parameter. (In Compact version, the socket is defined by configurator in advance, but in Standard version, it is created in DHCP client applications)

Protocol	ID	Port	Transmission timeout	Reception timeout
UDP	ID_SOC_DHCP	68	3 seconds	3 seconds

Device number

In device number, specify network device used by DHCP client. If specify '0 ', default network device is used.

6. 1. 1 DHCP client API

dhcp_client **Starting DHCP Client**

【Format】

ER ercd = dhcp_client(T_HOST_ADDR *addr);

【Parameter】

T_HOST_ADDR	*addr	host address information

【Return value】

ER	ercd	Successful completion（E_OK）or error code

【Error code】

E_PAR	*addr is NULL or socid is specified
E_OBJ	Socket status is wrong (socket has not been created yet)
E_TMOUT	Response from DHCP server delays. Or DHCP server does not exist

【Explanation】

This API acquires an IP address, a subnet mask, a gateway address from a DHCP server and assigns them to host. Maybe E_TMOUT error occurs due to the construction of using network. At that time, we recommend that you try to retry until it succeeds.

This API will also initiate a new DHCP session. When you call the API, you will start to send operation DISCOVER always expect receive OFFER, send REQUEST, the reception of the ACK that is.

The expiration of an IP address which is acquired from a DHCP server is specified 'lease (lease period).

As follows: DHCP client to do lease before the lease period expires.

DHCP client example （Exclusive task）

```
void dhcp_tsk(VP_INT exinf)
{
    ER ercd;
    T_HOST_ADDR dhcp_addr = {0};
    UB status = DHCP_STS_INIT;

    dhcp_addr.socid = ID_SOC_DHCP;
    dhcp_addr.dev_num = ID_DEVNUM_ETHER;

    for (;;) {
        ercd = dhcp_client(&dhcp_addr);
        if (ercd == E_OK) {
            /* BOUND period */
            dly_tsk(dhcp_addr.t1*1000);
            /* RENEWING period */
            status = DHCP_STS_RENEWING;
            continue;
        }
        if (status == DHCP_STS_RENEWING) {
            /* REBINDING period */
            dly_tsk((dhcp_addr.t2-dhcp_addr.t1)*1000);
            status = DHCP_STS_INIT;
            continue;
        }
        /* INIT period */
        dly_tsk(1000);
    }
}
```

If it is need to update the lease on the DHCPREQUEST message, please use the extended version of DHCP CLient.

6．2 FTP Server

FTP server enables to download and upload files to the remote host.

（2）FTP Server Control Information

```
typedef struct t_ftp_server {
        UB      dev_num;            /* Device Number*/
        UH      ctl_socid;          /* ID Socket used for Command */
        UH      data_socid;         /* ID Socket used for data*/
        UB      *fs_file;           /* Store buffer */
        UW      fs_maxsz;           /* Store buffer size */
} T_FTP_SERVER ;
```

Set necessary information in this structure and then transfer as argument of FTP Server API.

Device Number

In device number, specify network device used in FTP server. In case of specifying “0”, default network device will be used (Please set 0 normally)

TCP socket

FTP server requires two TCP sockets for commands and data. TCP socket should be created in the following parameters. (For compact version, please define a socket by configurator in advance. In Standard version, it will be created in the FTP server application)

Socket used for command:

ID	Protocol	Port	Timeout				Buffer size	
			send	receive	connect	interrupt	send	receive
ID_SOC_FTP_CTL	TCP	21	5s	15s	-1	5s	1024	1024

Socket used for data：

ID	Protocol	Port	Timeout				Buffer size	
			send	receive	connect	interrupt	send	receive
ID_SOC_FTP_DATA	TCP	20	5s	15s	5s	5s	1024	1024

FTP file saving

In this FTP server, because it does not support file system, received files are stored in memory. Please set memory address of storing location in **store buffe**r. In addition, please set size of memory of storing location in **Store buffer size**.

6. 2. 1　FTP Server API

ftp_server　　**Start up FTP Server**

【Format】

```
ER ercd = ftp_server(T_FTP_SERVER *ftp);
```

【Parameter】

T_FTP_SERVER	* ftp	FTP server control information

【Return value】

ER	ercd	Successful completion（E_OK）or error code

【Error code】

E_PAR	Wrong parameter is specified （*fpt is NULL. ctl_socid or data_socid is not specified *fs_file is NULL fs_maxsz is zero)

【Explanation】

This API initializes FTP server, accepts and processes requests from FTP clients. Because this API becomes blocking calling, pleas use specific task to call it.

FTP server example

```
T_FTP_SERVER ftpd;
UB ftp_buf[1024];

/* FTP server task */
void FtpServerTask(VP_INT exinf)
{
    memset((char*)&ftpd, 0, sizeof(ftpd));
    ftpd.ctl_socid = ID_SOC_FTP_CTL;
    ftpd.data_socid = ID_SOC_FTP_DATA;
    ftpd.fs_file = ftp_buf;
    ftpd.fs_maxsz = 1024;
    ftp_server(&ftpd);
    ext_tsk();
}
```

6. 2. 2　Restriction terms

- Supported command includes login, put, get, quit.
- Do not support file system. Therefore, impossible to create directory structure and handle multiple files.

6．3　　HTTP server

HTTP server transmits content to HTTP client (internet browser) statically and dinamically .

（1）HTTP content information

```
typedef struct t_http_file {
        const char    *path;                        /* URL */
        const char    *ctype;                       /* Content type */
        const char    *file;                        /* Content */
        Int           len;                          /* Content size*/
                                                    /* HTTP callback function
        void(*cbk)(T_HTTP_SERVER *http);            or
                                                    CGI handler */
} T_HTTP_FILE ;
```

Register content to be used in HTTP server in this structure.

URL

Show URL of content. For example, if there is request from client to that URL, corresponding content will be sent to client.

Impossible to specify NULL in URL. Besides, URL starts by ’/ as usual.

Content type

Specify Content-Type of text/html etc. In case of dynamic content, specify NULL.

Content

Specify actual content. In case of dynamic content, specify NULL.

Content size

Specify size of content. In case of dynamic content, specify 0.

Callback function or CGI handler

When it’s dynamic content, specify pointer of the function called from HTTP server. In case of static content, specify NULL.

(2) HTTP server control information

```
typedef struct t_http_server {
      UW                  sbufsz;        /* transmission buffer size */
      UW                  rbufsz;        /* reception buffer size*/
      UW                  txlen;         /*internal data */
      UW                  rxlen;         /* internal data*/
      UW                  rdlen;         /* internal data*/
      UW                  len;           /* internal data*/
      UB                  *rbuf;         /* transmission buffer*/
      UB                  *sbuf;         /* reception buffer*/
      UB                  *req;          /* internal data*/
      UH                  Port;          /* Listening port number*/
      UH                  SocketID;      /*   ID socket */
      T_HTTP_HEADER       hdr;           /* HTTP client request */
      UB                  NetChannel;    /* Device number*/
      UB                  ver;           /* Version of IP*/
} T_HTTP_SERVER;
```

This structure is used as argument of HTTP server API. ID socket needs to be set by user application.

Device number

For device number, specify network device to use in HTTP server. In case of specifying “0”, default network device is used. (Please set “0” normally).

ID socket

In HTTP server, use TCP socket. TCP socket needs to be created by the following parameter. (For Compact versiom, socket is defined by configurator in advance. In Standard version, it will be created in HTTP server application).

ID	protocol	port	timeout				Buffer size	
			send	receive	connect	intterupt	send	receive
ID_SOC_HTTP	TCP	80	25s	25s	25s	25s	1024	1024

Transmit buffer Receive buffer

In the HTTP server uses the network protocol stack buffers for each send and receive packets.

And receive buffers (transmit), if you (for example, you want to send content to a larger network buffers, for example) for reasons such as content size, you want to use your own buffer application in which the value of the buffer own buffer size (send) and receive set. You will not get the network buffer on the HTTP server in that case.

Own area set can not be shared with other processes HTTP server also.

6．3．1　　HTTP server API

http_server　　　　　　Start up HTTP server

【Format】

ER ercd = http_server(T_HTTP_SERVER *http);

【Parameter】

T_HTTP_SERV	*http	HTTP server control information

【Return value】

ER	ercd	Successful completion（E_OK）or error code

【Error code】

E_PAR	Wrong parameter is specified （*http is NULL. SocketID is not specified.）

【Explanation】

This API initializes HTTP session, then accepts and processes request from HTTP client. In case the URL which is requested from client exists in content table (T_HTTP_FILE), send that content to client, if not, send HTTP error message”404 File not found”. In case content is dynamic (cbk is not NULL), call that callback function.

Because this API becomes blocking call, please use a specific task to call it.

If NULL, the receive buffer of the control information of the argument is HTTP server uses the network buffer.

If NULL, the transmit buffer of the control information of the argument is HTTP server uses the network buffer.

CgiGetParam CGI argument analysis

【Format】

```
void CgiGetParam(char *msg, int clen, char *cgi_var[], char *cgi_val[], int *cgi_cnt);
```

【Parameter】

char	*msg	CGI argument
int	clen	CGI argument size
char	*cgi_var[]	Analysed CGI argument
char	*cgi_val[]	Value of analysed CGI argument
int	*cgi_cnt	Number of articles of the analysed CGI argument

【Return value】

None

【Error code】

None

【Explanation】

This API analyses query string to be constructed in “field value” group. For example, analysis result in case of query string is given as “name1=value1&name2=value2” as below.

```
cgi_cnt = 2;

cgi_var[0] = “name1”;
cgi_var[1] = “name2”;

cgi_val[0] = “value1”;
cgi_val[1] = “value2”;
```

HttpSendText Transmission of text content

【Format】

```
ER ercd = HttpSendText(T_HTTP_SERVER *http, char *str, int len);
```

【Parameter】

T_HTTP_SERVER	*http	HTTP server control information
char	*str	String to transmit
int	len	length of string to transmit

【Return value】

ER	ercd	Successful completion only (E_OK)

【Error code】

None

【Explanation】

This API transmits dynamic content. Please call this API only from HTTP callback function.

Example

```
char page1[] = “<html><body>Welcome to this web server </body></html>”;

void Http_Callback(T_HTTP_SERVER *http)
{
   HttpSendText(http, page1, sizeof(page1));
}
```

HttpSendFile Send Attached File

【API】

ER HttpSendFile(T_HTTP_SERVER*http, char*str, int len, char*name, char *type);

【Parameter】

T_HTTP_SERVER	*http	HTTP server information
char	*str	File to be send
int	len	Length to be send
char	*name	Filename
char	*type	Content-Type value or strings of HTTP header

【Return Value】

ER	ercd	Success (E_OK)

【Error Code】

None

【DESCRIPTION】

This API sends the dynamic content. Please only be called from this API function callback HTTP.

Send file attachments in API: This is sent in (Content-Disposition attachment).

Use Case

```
char file[1024];

void Http_Callback(T_HTTP_SERVER *http)
{
    int len;
                              :
/* Specify the “file” of contents, the size set to “len” */
                              :
    HttpSendFile(http, file , len, “FILE NAME”, “text/plain”);
}
```

HttpSendImage Send Image Content

【API】

ER ercd = HttpSendImage(T_HTTP_SERVER *http, char *str, int len);

【Parameter】

T_HTTP_SERVER	*http	HTTP server information
char	*str	Image data to be send
int	len	Length to be send

【Return Value】

ER	ercd	Success (E_OK)

【Error Code】

None

【DESCRIPTION】

This API sends the dynamic content. Please only be called from this API function callback HTTP.

6. 3. 2 HTTP server sample

```
/* Definition of content */
const char index_html[] =
                          "<html>¥
                          <title> uNet3 HTTP Server </title>¥
                          <body>¥
                          <h1>Hello World!</h1>¥
                          </body>¥
                          </html>";

/* Initialization of content list */
T_HTTP_FILE const content_list[] =
{
    {"/", "text/html", index_html, sizeof(index_html), NULL},
    {"", NULL, NULL, 0, NULL} /* terminal */
};

/* Starting HTTP session */
static T_HTTP_SERVER http_server1;
void httpd_tsk1(VP_INT exinf)
{
    /* Initialize the content list global pointer */
    gHTTP_FILE = (T_HTTP_FILE*)content_list;

memset((char* )&http_server1, 0, sizeof(http_server1));
    http_server1.SocketID = ID_SOC_HTTP1;

    http_server(&http_server1);
}
```

6. 4 DNS client

In DNS client, use UDP socket. UDP socket will be created by the below parameter.

ID	protocol	port	timeout	
			send	receive
ID_SOC_DNS	UDP	0	5s	5s

6. 4. 1 DNS client API

dns_get_ipaddr **Acquire IP address from host name**

【Format】

ER ercd = dns_get_ipaddr(ID socid, UW dns_server, char *name, UW *ipaddr);

【Parameter】

ID	socid	UDP socket ID
UW	dns_server	IP address of DNS server
char	*name	Host name
UW	*ipaddr	IP address to acquire

【Return value】

ER	ercd	Successful completion（E_OK）or error code

【Error code】

E_PAR	Wrong parameter is specified
E_TMOUT	No response from DNS server
E_NOMEM	Memory error
E_OBJ	Unable to resolve IP address from host name

Example of use

```
UW ip;
ER ercd;
UW dns_server = ip_aton(“192.168.11.1”);

dns_get_ipaddr(ID_SOC_DNS, dns_server, “www.eforce.co.jp”, &ip);
```

## dns_get_name		Acquire host name from IP address

【Format】

ER ercd = dns_get_name(ID socid, UW dns_server, char *name, UW *ipaddr);

【Parameter】

ID	socid	UDP socket ID
UW	dns_server	IP address of DNS
char	*name	host name to acquire
UW	*ipaddr	IP address

【Return value】

ER	ercd	Successful completion（E_OK）or error code

【Error code】

E_PAR	Wrong parameter is specified
E_TMOUT	no response from DNS server
E_NOMEM	Memory error
E_OBJ	Unable to acquire host name from IP address

Example of use

```
UW ip = ip_aton(“192.168.11.30”);
ER ercd;
char host_name[256];
UW dns_server = ip_aton(“192.168.11.1”);

dns_get_name(ID_SOC_DNS, dns_server, host_name, &ip);
```

6．5　　DHCP Client

For the existing DHCP client, holds the lease on the resources of the state, such as IP, DHCP client extended version information (RENEW), release (RELEASE), denial (DECLINE), a restart (REBOOT), an extension of these has been enhanced to provide the function get (INFORM).

(1) DHCP Client Information

```
typedef struct t_dhcp_client {
        T_DHCP_CTL      ctl             /* Control Informatio */
        UW              ipaddr;         /* IPaddress*/
        UW              subnet;         /* Subnet Mask */
        UW              gateway;        /* Gateway Address */
        UW              dhcp;           /* DHCPserver address */
        UW              dns[2];         /* DNSAddress */
        UW              lease;          /* Release time for DHCP Addless */
        UW              t1;             /* Renewal DHCP address */
        UW              t2;             /* Rebind time of DHCP address.*/
        UB              mac[6];         /* MAC addresss */
        UB              dev_num;        /* Device number */
        UB              state           /* Status of DHCP Clients */
        UH              socid;          /* UDP with socket ID */
        UB              arpchk;         /* Duplicate IP check */
} T_DHCP_CLIENT ;
```

This structure is intended to be used as an argument to the DHCP client API, is an extension of the host address information structure. The same manner as described above, UDP socket ID number and device must be set by the user application. Please refer to the DHCP client is the value to be set.

If you set the "ARP_CHECK_ON" to check whether or not duplicate IP, you do the duplicate check using the ACD feature for the IP, which is leased from the DHCP server.

This structure is used, even when I update the IP address that you set IP address at the time of acquisition. We can not change the DHCP client state and control information in the application for that.

6．5．1 DHCP Client Extended API

dhcp_bind Get DHCP Lease Information

【API】

ER ercd = dhcp_bind(T_DHCP_CLIENT *dhcp);

【Parameter】

T_DHCP_CLIENT	*dhcp	DHCP Client Information

【Return Value】

ER	ercd	Success (E_OK) or Error Code

【Error Code】

E_PAR	DHCP is NULL or No specified socid (Compact only)
E_OBJ	Incorrect Socket status (No create the socket)
E_SYS	Address conflict another host when the IP address is assigned.
E_TMOUT	Response is delay from DHCP server or the DHCP server doesn't exist.

【DESCRIPTION】

This API provides the same functionality as the () API dhcp_client traditional.
To verify the IP address that you get that you do not have overlap with other hosts, we set up a check for duplicate IP ARP_CHECK_ON the presence of DHCP client information of the argument. If a duplicate IP address is detected at this time, to send a message to the DHCP server DHCP_DECLINE, API will return the E_SYS.

dhcp_renew Renewal DHCP Lease

【API】

ER ercd = dhcp_renew(T_DHCP_CLIENT *dhcp);

【Parameter】

T_DHCP_CLIENT	*dhcp	DHCP Client Information

【Return Value】

ER	ercd	Success(E_OK) or Error Code

【Error Code】

E_PAR	DHCP is NULL or No specified socid (Compact only)
E_OBJ	Incorrect DHCP client, or request denied by DHCP server.
E_SYS	Address conflict another host when the IP address is assigned.
E_TMOUT	Response is delay from DHCP server or the DHCP server doesn't exist.

【DESCRIPTION】

This API to extend the validity period of IP address obtained from the DHCP server. The argument specifies the DHCP client information obtained by dhcp_bind ().

This API should be called (t1) within the validity period. Lifetime is measured by the application using the timer or control task.

This feature also includes the ability RENEW REBIND. The difference between the two is only to send broadcast messages to send unicast REQUEST. If you can not receive the ACK after sending a REQUEST message to the DHCP server at the beginning, we perform a broadcast transmission immediately.

To verify the IP address that the extension does not have an overlap with other hosts, we set up a check for duplicate IP ARP_CHECK_ON the presence of DHCP client information of the argument. If a duplicate IP address is detected at this time, to send a message to the DHCP server DHCP_DECLINE, API will return the E_SYS.

dhcp_reboot DHCP Client Reboot

【API】

ER ercd = dhcp_reboot(T_DHCP_CLIENT *dhcp);

【Parameter】

T_DHCP_CLIENT	*dhcp	DHCP Client Information

【Return Value】

ER	ercd	Success(E_OK) or, Error Code

【Error Code】

E_PAR	DHCP is NULL or No specified socid (Compact only) or there is no address available.
E_OBJ	Illegal DHCP client information or DHCP server has refused the request.
E_SYS	The assigned IP address conflict with another host.
E_TMOUT	Delayed response from the DHCP server. Or DHCP server does not exist.

【DESCRIPTION】

This API is reusing the IP resource to which the client was previously used, API is used to verify its legitimacy in DHCP server. If you do not, such as when you remove and insert the LAN cable or if the LAN interfaces in the pause is activated again, and security that are part of the same network before and after, the DHCP server for the IP resource that has been previously used, for example notice.
The argument specifies the DHCP CLIENT INFORMATION dhcp_bind acquired by ().

If an ACK is not received after sending REQUEST message, API This is an error, or if it receives a DHCPNAK.

To verify the IP address that you notice that you do not have overlap with other hosts, we set the duplicate IP ARP_CHECK_ON to check whether or not the argument of the DHCP CLIENT INFORMATION. If a duplicate IP address is detected at this time, to send a message to the DHCP server DHCP_DECLINE, API will return the E_SYS.

dhcp_release Release DHCP Lease Information

【API】

ER ercd = dhcp_release(T_DHCP_CLIENT *dhcp);

【Parameter】

T_DHCP_CLIENT	*dhcp	DHCP CLIENT INFORMATION

【Return Value】

ER	ercd	Success (E_OK) or Error Code

【Error Code】

E_PAR	DHCP is NULL or No specified socid (Compact only)
E_OBJ	Illegal DHCP client information
E_TMOUT	Timeout send DHCPRELEASE mesage

【DESCRIPTION】

This API will notify the DHCP server to release the resources when it no longer want to use the IP address obtained from the DHCP server.

The argument specifies the DHCP information obtained dhcp_bind in ().

dhcp_inform Get DHCP Options

【API】

ER ercd = dhcp_inform(T_DHCP_CLIENT *dhcp);

【Parameter】

T_DHCP_CLIENT	*dhcp	DHCP CLIENT INFORMATION

【Return Value】

ER	ercd	Success (E_OK) or Error Code

【Error Code】

E_PAR	DHCP is NULL or No specified sockid (only Compact). Or, there isn’t avairablre of the address。
E_OBJ	No specified the IP address b Host。
E_TMOUT	Delay DHCP server response. Or no exsistance DHC server.

【DESCRIPTION】

This API to get the information other than the IP address from the DHCP server. Set the static IP address, for example, the address of the DNS server is used, for example, if you want to get from the DHCP server.

To the argument set DHCP CLIENT INFORMATION (for μNet3/Compact) only socket ID and device number of the interface.

6．5．2　DHCP Client information Extended

DHCP Client Extended Information

```
void dhcp_client_tsk(VP_INT exinf)
{
    ER ercd;
    FLGPTN ptn;
    T_DHCP_CLIENT dhcp_client = {0};

    dhcp_client.dev_num = ID_DEVNUM_ETHER;
    dhcp_client.socid = ID_SOC_DHCP;

    whil (1) {
        ercd = dhcp_bind(&dhcp_client);

        while (ercd == E_OK) {
            /* Set Time event setup t1 */

            wai_flg(ID_DHCP_FLG, 0xFFFF, TWF_ORW, &ptn);

            /* t1 timeout */
            if (ptn & T1_EVENT) {
                ercd = dhcp_renew(&dhcp_client);
            }
            /* re-boot */
            else if (ptn & REBOOT_EVENT) {
                ercd = dhcp_reboot(&dhcp_client);
            }
        }
        dly_tsk(1000);
    }
    dhcp_release(&dhcp_client);
}
```

6．6　Ping Client

For any destination, Ping client sends the ICMP echo request. We found that in the IP address of the communication is possible if there is an echo response from the other party.
In addition, Ping client sends and receives using the ICMP socket. If μNet3/Compact use the socket ID "ID_ICMP" that is reserved by check instead of defining the ICMP socket, to "use the request ICMP Echo" from the net Configurator application.

6．6．1　Ping Client API

ping_client　　**ICMP Echo (Transmite Request and Receive Response)**

【API】

ER ping_client(T_PING_CLIENT *ping_client);

【Parameter】

T_PING_CLIENT	*ping_client	Ping transmit information

【Return Value】

ER	Ercd	Success（E_OK）or Error Code

【Error Code】

E_PAR	Specified the incorrect Parameter
E_TMOUT	No responde from remote or failed address resolver
E_NOMEM	Memory Errror
E_OBJ	Incorrect ping transmit information

【DESCRIPTION】

This API sends a ping to the IP address that you set in the argument. I continue to wait for a response from the other party and then I get a timeout that is specified in the argument to expire. If you get a response I will return E_OK.
This API is limited to IPv4.

Use Case

```
ER ping_send(void)
{
        T_PING_CLIENT ping = {0};
        ER ercd;

        ping.sid = ID_ICMP;
        ping.devnum = ID_DEVNUM_ETHER;
        ping.tmo = 1000; /* Timeout 1 second*/
        ercd = ping_client(&ping);
        if (ercd == E_OK) {
                /* ping success */
        }
        return ercd;
}
```

6．7　SNTP Client

SNTP client get the (number of seconds starting from 1/1/1900) time from NTP (NTP) server time on the network using the NTP packet.

6．7．1　SNTP Client API

sntp_client　　Get NTP time

【API】

```
ER sntp_client(T_SNTP_CLIENT *sntp_client, UW *sec, UW *msec);
```

【Parameter】

T_SNTP_CLIENT	*sntp_client	Information of SNTP client
UW	*sec	MTP Time (second)
UW	*msec	NTP Time(32-bit fixed-point representation below the decimal)

【Return Value】

ER	ercd	Success（E_OK）or　Error Code

【Error Code】

E_PAR	Specified the illegal parameter
E_TMOUT	No response from remote or failed address resplver
E_NOMEM	Memory Error
E_OBJ	Incorrect Information of SNTP client

【DESCRIPTION】

This API gets the time from NTP NTP server you set up in the argument. To set the NTP server, specify the IPv4 address and port number.

In the SNTP client uses the UDP socket. If uNet3/Compact argument should be set to the socket ID available.

This API returns the E_OK if you can successfully get the time NTP.

NTP time is shown in the sec and msec arguments at this time. Because you are starting from 1/1/1900, NTP time and the conversion to Unix time (JST) UTC must be calculated by the caller.

Use Case

```
ER sntp_time(void)
{
        T_SNTP_CLIENT sntp = {0};
        UW sec, msec;
        ER ercd;

        sntp.sid = ID_UDP;
        sntp.devnum = ID_DEVNUM_ETHER;

        ercd = sntp_client(&sntp, &sec, &msec);
        if (ercd == E_OK) {

                /* Convert to Unix Time */
                sec -= 2208988800;

                /* Accuracy for ms */
                msec >>= 16;
                msec *= 1000;
                msec >>= 16;
        }
        return ercd;
}
```

6．8　String Library

μNet3 system provides a standard library of String so that it is not dependent on the compiler. Network applications you can use to provide these functions.

net_strncasecmp　　Compare String (case-insensitive letter)

【API】

W net_strncasecmp(const char *str1, const char *str2, W len);

【Parameter】

const char *	str1	String to be compared
const char *	str2	String to be compared
W	len	Length of compare

【Return Value】

W	Result

【DESCRIPTION】

The results were compared with the character code, I will return 0 if str1 = str2 then positive value, you will return a negative value if str1 <str2 if str1> str2.

I arrived at the end of either string number of characters until it reaches the comparison is to be compared. By this function equate the case of the letters.

net_strcmp String Compare

【API】

W net_strcmp(const char *str1, const char *str2);

【Parameter】

const char *	str1	String to be compared
const char *	str2	String to be compared

【Return Value】

W		Result of Compare

【DESCRIPTION】

The results were compared with the character code, I will return 0 if str1 = str2. positive value, you will return a negative value if str1 <str2 if str1> str2.

Reach the end of the string until one of them to be compared

net_strcpy String Copy

【API】

char* net_strcpy(char *str1, const char *str2);

【Parameter】

char *	str1	Address of copy destination string
const char *	str2	Address of copy source string

【Return Value】

char*	str1	Address of copy destination string

【DESCRIPTION】

This API is to copy of the srt2 to the end of str1 (NULL) .

net_strlen Get String Length

【API】

UW net_strlen(const char *str);

【Parameter】

char *	str	String

【Return Value】

UW	String length

【DESCRIPTION】

Gets the number of characters up to (NULL) end of str. (NULL is not included)

net_strcat String concatenation

【API】

char* net_strcat(char *str1, const char *str2);

【Parameter】

char *	str1	Address of destination string
const char *	str2	Address of source string

【Return Value】

char*	str1	Address of destination string

【DESCRIPTION】

Copy to the end of the str2 starting at (NULL) coupling the end of the destination string str1.

net_strchr　　　　　　Serch Character

【API】

char* net_strchr(const char *str, int ch);

【Parameter】

const char *	str	Serch target character
int	ch	Serch character

【Return Value】

char*	str	In the subject string, address search string appears.
		If the search string does not appear NULL

Appendix

7．1　Packet format

（1）T_NODE

Information of communication endpoint

```
typedef struct t_node {
        UH          port;          /* Port number of socket */
                                   /* IP version*/
        UB          ver;
                                   /* Necessarily specify IP_VER4 */
        UB          num;           /* Device number*/
        UW          ipa;           /* IP address */
} T_NODE;
```

（2）T_NET_ADR

Information of network address

```
typedef struct t_net_adr {
                                   /* IP- Version*/
        UB          ver;
                                   /* Necessarily specify IP_VER4*/
        UB          mode;          /* Reserve*/
        UW          ipaddr;        /* IP Address*/
        UW          gateway;       /* Gateway*/
        UW          mask;          /* Subnet mask*/
} T_NET_ADR;
```

（3）T_NET_DEV

The information of the network device

```
typedef struct t_net_dev {
        UB          name[8];     /* Device name */
        UH          num;         /* Device number */
        UH          type;        /* Device type */
        UH          sts;         /*Reserve */
        UH          flg;         /* Reserve */
        FP          ini;         /* Pointer to dev_ini function*/
        FP          cls;         /*Pointer to dev_cls function*/
        FP          ctl;         /* Pointer to dev_ctl function*/
        FP          ref;         /*Pointer to dev_ref function*/
```

```
            FP          out;        /*Pointer to dev_snd function*/
            FP          cbk;        /*Pointer to dev_cbk function*/
            UW          *tag;       /*Reserve */
            union {                 /* MAC address */
              struct {
                    UB  mac[6];
              }eth;
            } cfg;
            UH          hhdrsz;     /* Device header size */
            UH          hhdrofs;    /* Position of writing network buffer*/
        } T_NET_DEV;
```

（4） T_NET_BUF

Information of network buffer

```
typedef struct t_net_buf {
        UW              *next;      /* Reserve */
        ID              mpfid;      /* Memory pool ID */
        T_NET           *net;       /* Network interface */
        T_NET_DEV       *dev;       /* Network device */
        T_NET_SOC       *soc;       /* Socket*/
        ER              ercd;       /* Error code */
        UH              flg;        /* Broadcast multicast flag*/
        UH              seq;        /* Fragment sequence */
        UH              dat_len;    /* Data size of packet */
        UH              hdr_len;    /* Header size of packet*/
        UB              *dat;       /* Indicate data position of packet (buf) */
        UB              *hdr;       /* Indicate header position of packet (buf) */
        UB              buf[];      /* Actual packet */
} T_NET_BUF ;
```

（5） T_HOST_ADDR

Information of host address

```
typedef struct t_host_addr {
        UW      ipaddr;             /* IP address*/
        UW      subnet;             /* Subnet mask */
        UW      gateway;            /* Gateway */
        UW      dhcp;               /* DHCPserver address */
```

```
          UW      dns[2];                   /* DNS address */
          UW      lease;                    /* Lease period of DHCP address */
          UW      t1;                       /* Renewal period of DHCP address*/
          UW      t2;                       /* Rebind period of DHCP address */
          UB      mac[6];                   /* MAC address */
          UB      dev_num;                  /* Device number */
          UH      socid;                    /* UDP socket ID */
    } T_HOST_ADDR ;
```

（6）T_FTP_SERVER

FTP server control information

```
    typedef struct t_ftp_server {
          UB      dev_num;                  /* Device number */
          UH      ctl_socid;                /* SocketID for command */
          UH      data_socid;               /* SocketID for data */
          UB      *fs_file;                 /* Store buffer */
          UW      fs_maxsz;                 /* Store buffer size */
    } T_FTP_SERVER ;
```

（7）T_HTTP_FILE

HTTP Content Information

```
    typedef struct t_http_file {
          const char    path[12];                    /* URL */
          const char    ctype[12];                   /* Content type*/
          const char    *file;                       /* Content */
          int           len;                         /* Content size*/
                                                     /* HTTP callback function
          void(*cbk)(T_HTTP_SERVER *http);           or
                                                     CGI handler */
    } T_HTTP_FILE ;
```

（8）T_HTTP_SERVER

HTTP Server control information

```
     typedef struct t_http_server {
        UW                  sbufsz;          /* Transmission buffer size */
        UW                  rbufsz;          /* Reception buffer size */
        UW                  txlen;           /* Internal data*/
        UW                  rxlen;           /* Internal data*/
        UW              rdlen;          /* Internal data*/
        UW              len;            /* Internal data*/
        UB              *rbuf;          /* Transmission buffer*/
        UB              *sbuf;          /* Reception buffer */
        UB              *req;           /* Internal data*/
        UH              Port;           /* Listerning port number*/
        UH              SocketID;       /* Socket ID */
        T_HTTP_HEADER   hdr;            /* HTTP client request */
        UB              NetChannel;     /* Device number */
        UB              ver             /* IP version */
    } T_HTTP_SERVER;
```

（9）T_RCV_PKT_INF

Reception packet information

```
typedef struct t_rcv_pkt_inf{
        UW      src_ipa;            /* Source IP address of packet*/
        UW      dst_ipa;            /* Destination IP address of packet*/
        UH      src_port;           /* Source port number of packet*/
        UH      dst_port;           /* Destination port number of packet*/
        UB      ttl;                /* IP header TTL of packet*/
        UB      tos;                /* IP header TOS of packet*/
        UB      ver;                /* IP header version of packet*/
        UB      num;                /* Reception device number of packet/
} T_RCV_PKT_INF;
```

（10）T_DHCP_CLIENT

DHCP Client information

```
typedef struct t_dhcp_client {
    T_DHCP_CTL      ctl             /* Internal data*/
    UW              ipaddr;         /* IP address*/
    UW              subnet;         /* Subnet mask */
    UW              gateway;        /* Gateway */
    UW              dhcp;           /* DHCPserver address */
    UW              dns[2];         /* DNS address */
    UW              lease;          /* Lease period of DHCP address */
    UW              t1;             /* Renewal period of DHCP address*/
```

```
        UW          t2;         /* Rebind period of DHCP address */
        UB          mac[6];     /* MAC address */
        UB          dev_num;    /* Device number */
        UB          state;      /* DHCP client status */
        UH          socid;      /* UDP socket ID */
        UB          arpchk      /* APR check */
    } T_DHCP_CLIENT;
```

（11）T_PING_CLIENT

Ping Client Information

```
    typedef struct  t_ping_client {
        ID          sid;        /* ICMP Socket ID */
        UW          ipa;        /* Destination IP Address */
        TMO         tmo;        /* Response Time out */
        UH          devnum;     /* Device number */
        UH          len;        /* Packet Size */
    } T_PING_CLIENT;
```

（12）T_SNTP_CLIENT

SNTP Client Information

```
    typedef struct  t_sntp_client {
        ID          sid;        /* UDP Socket ID */
        UW          ipa;        /* NTP server IP address */
        TMO         tmo;        /* Response Time out */
        UH          devnum;     /* Device numbr */
        UH          port;       /* NTP server port number */
        UB          ipv;        /* IP vrsion */
    } T_SNTP_CLIENT;
```

7．2　Constant and Macro

（1）IP Address

ADDR_ANY	IP address 0
IP_VER4	IP version 4

（2）Port Number

PORT_ANY	Port number 0

（3）IP protocol

IP_PROTO_TCP	TCP protocol
IP_PROTO_UDP	UDP protocol
IP_PROTO_ICMP	ICMP protocol

（4）Network interface control

NET_IP4_CFG	Configure and verify IP Address, Subnet mask
NET_IP4_TTL	Configure and vefiry TTL
NET_BCAST_RCV	Configure and verify reception of broadcast
NET_MCAST_JOIN	Join in multicast group
NET_MCAST_DROP	Drop from multicast Group
NET_MCAST_TTL	Configure TTL used in multicast transmission

（5）Parameter of socket

SOC_IP_TTL	Configure and verify TTL of Socket
SOC_IP_TOS	Configure and verify TOS of Socket
SOC_TMO_SND	Configure and verify blocking time-out of snd_soc
SOC_TMO_RCV	Configure and verify blocking time-out of rcv_soc
SOC_TMO_CON	Configure and verify blocking time-out of con_soc
SOC_TMO_CLS	Configure and verify blocking time-out of cls_soc
SOC_IP_LOCAL	Get port number and IP address of local host
SOC_IP_REMOTE	Get port number and IP address of remote host
SOC_CBK_HND	Register callback function
SOC_CBK_FLG	Specify callback event
SOC_RCV_PKT_INF	Get information of reception packet

（6）Connection mode of socket

SOC_CLI	Connect to remote host (active connection)
SOC_SER	Wait for connection (passive connection)

（7）Termination mode of socket

SOC_TCP_CLS	Disconnect socket. (Terminate connection)
SOC_TCP_SHT	Disable only the transmission process. Reception is possible

（8）Interruption mode of socket

SOC_ABT_CON	Abort con_soc()
SOC_ABT_CLS	Abort cls_soc()
SOC_ABT_SND	Abort snd_soc()
SOC_ABT_RCV	Abort rcv_soc()
SOC_ABT_ALL	Abort all the processes of socket

（9）Callback Event

EV_SOC_CON	Enable con_soc() to be non-blocking mode
EV_SOC_CLS	Enable cls_soc() to be non-blocking mode
EV_SOC_SND	Enable snd_soc() to be non-blocking mode
EV_SOC_RCV	Enable rcv_soc() to be non-blocking mode

7．3　Error Code List

E_NOSPT	-9	Unsupported function
E_PAR	-17	Parameter error
E_ID	-18	Illegal ID number
E_NOMEM	-33	Insufficient memory
E_OBJ	-41	Object status error
E_NOEXS	-42	Uncreated object
E_QOVR	-43	Queuing overflow
E_RLWAI	-49	Forced cancellation of wait state
E_TMOUT	-50	Polling failure or time-out
E_CLS	-52	Change status of wating object
E_WBLK	-57	Non-blocking acceptance
E_BOVR	-58	Buffer overflow
EV_ADDR	-98	Unknown default gateway

7．4 API List

API Name	
A）Network Interface	
net_ini	Initialize TCP / IP protocol stack
net_cfg	Configure parameters of network interface
net_ref	Refer parameters of network interface
net_acd	Detection IP Address Confiliction
B) Network Device Control	
net_dev_ini	Initialize network device
net_dev_cls	Release Network Device
net_dev_ctl	Control network device
net_dev_sts	Get status of network device
C) Socket	
cre_soc	Create socket (Standard version only)
del_soc	Delete a socket (Standard version only)
con_soc	Socket connection
cls_soc	Socket interruption
snd_soc	Send data
rcv_soc	Receive data
cfg_soc	Configure parameter of socket
ref_soc	Refer parameter of socket
abt_soc	Abort process of socket
D) Network Application	
dhcp_client	Start DHCP Client
ftp_server	Start FTP Server
http_server	Start HTTP server
CgiGetParam	Analyze CGI argument
HttpSendText	Send text content
HttpSendFile	Send Attached File
HttpSendImage	Send Image Content
dns_get_ipaddr	Get IP address from host name
dns_get_name	Get host name from IP address
dhcp_bind	Get DHCP Lease Inforamation

API Name	
dhcp_renew	Renewal DHCP lease information
dhcp_reboot	Reboot DHCP client
dhcp_release	Release DHCP lease information
dhcp_inform	Get DHCP option
ping_client	ICMP Echo request and response
sntp_client	Get NTP time
E) Others	
ip_aton	Convert IPv4 address string in dot notation to 32-bit value
ip_ntoa	Convert 32-bit value IPv4 address to IPv4 address string in dot notation
ip_byte2n	Convert IPv4 addresse array to 32 bit value
ip_n2byte	Convert 32-bit value IPv4 addresse to array
htons	Convert 16-bit value to network byte order
ntohs	Convert 16-bit value to host byte order
htonl	Convert 32-bit value to network byte order
ntohl	Convert 32- bit value to host byte order

Index

μNet3 Users Guide

2009	June	1st Edition
2010	May	2nd Edition
2010	December	3rd Edition
2011	May	4th Edition
2012	July	5th Edition
2012	October	6th Edition
2013	January	7th Edition
2014	February	8th Edition
2014	April	9th Edition

eForce Co., Ltd.　　　http://www.eforce.co.jp/

7F Gembay Building,5-4 Nihonbashi Tomizawa-cho,Tyo-ku,Tokyo Japan,

103-0006

TEL +81-3-5614-6918　FAX +81-3-5614-6919

Contact us :　info@eforce.co.jp