To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

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

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

R8C Family, H8/300H Tiny Series, H8/300H Series and M16C/Tiny Series

M3S-T2-Tiny: Ultracompact TCP/IP Protocol Stack Software

Introduction

This document explains the outline of ultracompact TCP/IP protocol stack software library .

Target device

R8C Family, H8/300H Tiny Series, H8/300H Series and M16C/Tiny Series

Contents

1. Overview

The Tiny TCP/IP library M3S-T2-xxx (hereafter referred to as “the T2") is the network software library for the following series and groups of microcomputers. This manual provides the common information necessary to create application programs using the T2. The MCU-dependent information (e.g., development environment) is supplied in the respective Release Notes.

M3S-T2-Tiny : R8C Family, H8/300H Tiny Series, H8/300H Series and M16C/Tiny Series
M3S-T2-30 : M16C/62 Group
M3S-T2-308 : M32C/83 Group
* The xxx in the description below denotes the series and MCU names.

Fig 1 shows the position of the T2 in a software group associated with network middleware.

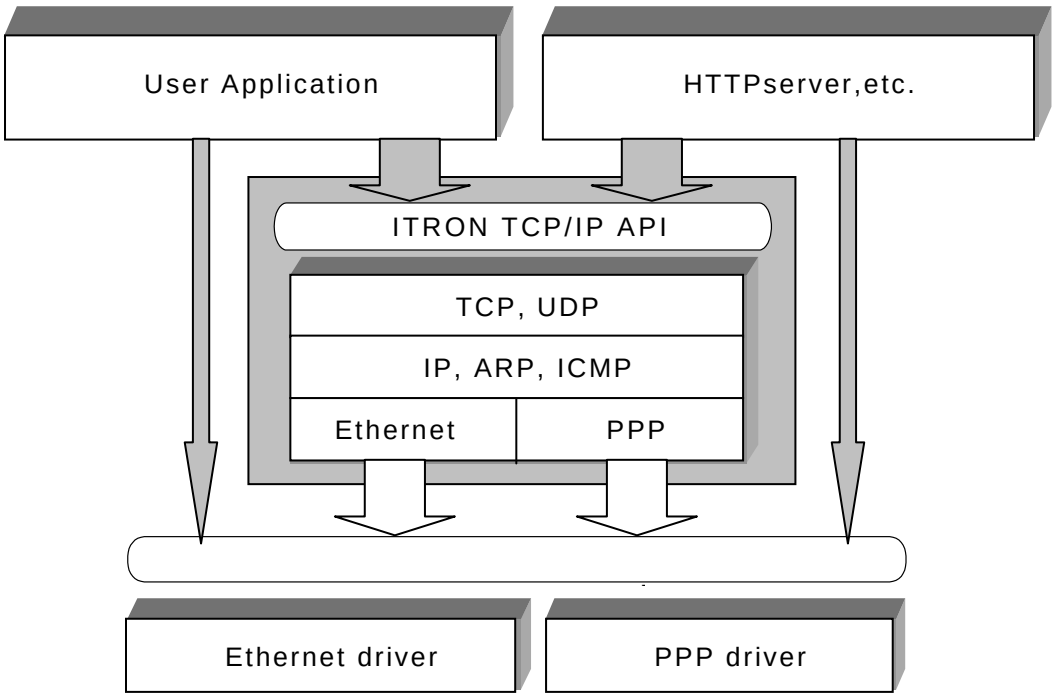

Fig 1. Position of the T2 in network software structure

- Fig 1 shows the position of the T2 in a general structure of the software that comprises network middleware. Shown in the gray background at center is the T2.
  The arrows represent function calls.
- The T2 performs protocol processing between the Ethernet or PPP driver and an application program as it sends or receives data to and from the network.
- Included as protocol processing in the T2 are TCP, UDP, IP, ICMP, ARP and PPP.
  To perform TCP or UDP communication from a user application or a higher-level protocol program, functions must be called from within the library by using the API (compliant with ITRON TCP/IP API) that is stipulated in the T2. IP, ICMP, ARP and PPP are the lower-level protocols below TCP and UDP, so that in no case will the functions to process these protocols be called directly from a user application.
- The T2 supports Ethernet and PPP as the data link and physical layers. Of these, part of the library that depends on the user hardware such as LAN controller or serial I/O is separated as a driver from the library body.
  To create drivers matched to the hardware used, refer to specifications for the driver API that are written in the user’s manual for the driver concerned.
  * The user’s manuals for drivers are included with the product package separately from this manual.
  When using the T2 to create a program, make sure that the driver’s initialization and termination functions are called directly from the program. (In Figure 1, these function calls are shown by the gray arrows.) Specifications of these API functions are described in the T2 manual, not just in the user’s manual for the driver.

- This product package comes with sample programs for drivers. Refer to these sample programs when creating your original driver.
- The T2 library does not use the real-time OS.

2. Outline of the T2

2.1 Product information

The T2 consists of library files, sample drivers and sample application.

Table 1. Directory Structure for the T2

	Description
T2 Library	
T2_ether_xxx.lib T2_ppp_xxx.lib	T2 Library file (for Ethernet) T2 Library file (for PPP)
itcpip.h	Header file for T2
config_tcpudp.c.tpl	Configuration file template for T2
Sample Driver	
eth_drv.c eth_drv.h	Source file for the Ethernet driver Header file for the Ethernet driver
ppp_drv.c ppp_drv.h	Source file for the PPP driver Header file for the PPP driver
Sample Application Program	
main.c config_tcpudp.c intprg.c	Sample program Configuration file Cyclic activation program of TCP/IP processing

2.1.1 T2 Library

- Library files (binary file)
  These files contain API functions and various protocol processing programs. Use these files along with the application program after linking.

Use T2_ether_xxx.lib when using Ethernet or T2_ppp_xxx.lib when using PPP.

- Header files (itcpip.h)
  The T2 API prototype declarations, macro definitions, etc. are written in this file. For application programs that call T2 APIs, include this header file.

- Configuration file template (config_tcpudp.c.tpl)
  This template is provided for the file to define the necessary data for communication such as the T2 communication end points and reception points. Although this file is referred to in the T2 as the configuration file, it actually is comprised of a C source file that contains the above data definition, and must therefore be compiled and linked along with the user application program.
  Because this template contains all of the necessary data definitions, rewrite the content of each data according to the user application.

* This file is not intended for direct use, it must always be modified before use.

2.1.2 Sample Driver

Contained in this directory are the Ethernet and PPP sample drivers that can be used with the T2.

The Ethernet driver handles device-dependent processing such as Ethernet frame transmission/reception processing. This driver supports RTL8019AS, the Ethernet controller available from RealTek.

The PPP driver performs device-dependent processing such as modem control, dialing-up and serial I/O data transmission/reception.

2.1.3 Sample Application Program

Sample sources for application programs (which use Telenet) are included.

2.2 Outline Library Specifications

Outline specifications of the T2 are shown in Table 2. This library supports the TCP, UDP, IP, ICMP, ARP and PPP protocols. The Application Program Interface (API) is compliant with ITRON TCP/IP API Specification.

Table 2. Outlines Specifications by Protocol

Protocol	Item	Specification
TCP	API	Compliant with ITRON TCP/IP API Specification
	Non-blocking call	Not supported (Only blocking calls are supported)
	Callback feature	Not supported
	Maximum size	TCP data segment length (data length): 1,480 (1,460) bytes
	TCP header option	Only MSS is supported Reception: Header options except MSS are ignored Transmission: No header options except MSS are accepted
UDP	API	Compliant with ITRON TCP/IP API Specification
	Non-blocking call	Supported
	Callback feature	Supported
	Queuing	Not supported
	Maximum size *	UDP datagram length (data length): 1,480 (1,472) bytes
IP	Version	IPv4 (version 4) only
	Fragment	Not supported Reception: Fragmented datagrams are discarded Transmission: Datagrams cannot be fragmented
	Header option	Not supported Reception: Datagrams that include header options are discarded Transmission: No header options are accepted
	Maximum size *	IP datagram (data length): 1,500 (1,480) bytes
ICMP	Message type	Only echo responses are supported Reception: Only echo request Other messages discarded Transmission: Only echo response
ARP	Cache entry	depend on User definition
	Cache retention time	Approx. 10 minutes
PPP	Server/client	Only client is supported, server is unsupported
	Maximum size *	PPP frame (data length): 1,504 (1,500) bytes
	Authentication method	PAP
	Compression option	Compression-related options are not supported Setup requests for compressing the protocol field, address and control field and TCP/IP header are rejected

* The maximum sizes of TCP, UDP, IP, and PPP depend on the transmission/reception buffers sizes of the driver.

3. T2 API Specification

Table 3 lists the APIs that are supported by the T2. The TCP and UDP APIs each are compliant with ITRON TCP/IP Specification. The number of APIs that can executed at the same time in the T2 is one for TCP and UDP each.

Table 3. List of T2 APIs

T2 APIs		C Language API
TCP	Wait for Connection Request (Passive Open)	tcp_acp_cep()
	Request Connection (Active Open)	tcp_con_cep()
	Terminate data transmission	tcp_sht_cep()
	Close communication end point	tcp_cls_cep()
	Transmit data	tcp_snd_dat()
	Receive data	tcp_rcv_dat()
UDP	Transmit packet	udp_snd_dat()
	Receive packet	udp_rcv_dat()
Initialization	Get work memory size used	tcpudp_get_ramsize()
	Open T2 library	tcpudp_open()
Cyclic process	TCP/IP protocol process	_process_tcpip()
Termination processing	Close T2 library	tcpudp_close()

* Refer to release note for the stack size of each APIs.

The following describes the data structures and macro definitions used in APIs. This is followed by a detailed description of each API, which is given in the format shown below.

[(API format)]

Shows the API format.

[(Parameters)]

Shows the meaning of parameters to the API and limitations on acceptable values.

[(Returns and error codes)]

Shows the type of value or error code returned by the API and the conditions under which an error occurs.

[(Description)]

Shows the functionality and behavior of each API and the precautions to be observed when using API.

3.1 Data Structures and Macro Definitions

The data structures and macro definitions used in the T2 APIs are defined in the header file itcpip.h. Also found in itcpip.h are the API prototype declarations.

(1) Data structures

[(Data type)]

```
typedef char            B;
typedef unsigned char   UB;
typedef short           H;
typedef unsigned short  UH;
typedef long            W;
typedef unsigned long   UW;
typedef void            (*FP)();
typedef void far        *VP;
typedef H               INT;
typedef H               ID;
typedef H               TMO;
typedef H               ER;
typedef H               ATR;
typedef INT             FN;
```

[(IP address and port number)]

```
typedef   struct   t_ipv4ep {
    UW              ipaddr;             /* IP address */
    UH              portno;             /* Port number */
} T_IPV4EP;
```

[(TCP reception point)]

```
typedef   struct   t_tcp_crep {
    ATR             repatr;             /* Attribute of TCP reception point */
    T_IPV4EP        myaddr;             /* Local IP address and port number */
} T_TCP_CREP;
```

[(TCP communication end point)]

```
typedef   struct   t_tcp_ccep {
    AT              cepatr;             /* Attribute of TCP communication end point */
    VP              sbuf;               /* Top address of transmit window buffer */
    INT             sbufsz;             /* Size of transmit window buffer */
    VP              rbuf;               /* Top address of receive window buffer */
    INT             rbufsz;             /* Size of receive window buffer */
    FP              (*callback)(ID cepid, FN fncd , VP p_parblk);   /* Callback routine */
} T_TCP_CCEP;
```

[(UDP communication end point)]

```
typedef   struct   t_udp_ccep {
    ATR             cepatr;             /* Attribute of UDP communication end point */
    T_IPV4EP        myaddr;             /* Local IP address and port number */
    FP              (*callback)(ID cepid, FN fncd , VP p_parblk);   /* Callback routine */
} T_UDP_CCEP;
```

[(T2 library environment variable)]

```
typedef struct      {
  UB                ipaddr[4];          /* Local IP address */
  UB                maskaddr[4];              /* Subnet mask */
  UB                gwaddr[4];                /* Gateway address */
} TCPUDP_ENV;
```

(2)Macro definitions

[(Macros used in APIs)]

Table 4. Functional Codes and Event Codes Used in APIs

Name	Value	Meaning
TFN_UDP_SND_DAT	-0x0223	Transmit UDP data
TFN_UDP_RCV_DAT	-0x0224	Receive UDP data
TEV_UDP_RCV_DAT	0x221	UDP packet received
TMO_POL	0	Polling
TMO_FEVR	-1	Waiting forever
TMO_NBLK	-2	Non-blocking call

Table 5. Special IP Addresses and Port Numbers

Name	Value	Meaning
IPV4_ADDRANY	0	IP address specification omitted
TCP_PORTANY	0	TCP port number specification omitted
NADR	-1	Invalid address

[(Error codes)]

Table 6. Error Codes Used in APIs

Name	Value	Meaning
E_OK	0	Terminated normally
E_OBJ	-63	Object status error
E_QOVR	-73	Queuing overflow
E_WBLK	-83	Non-blocking call accept
E_TMOUT	-85	Timeout
E_CLS	-87	Failed to connect
E_BOVR	-89	Buffer overflow

3.2 TCP wait for connection request (Passive Open) tcp_acp_cep()

[(API Format)]

ER tcp_acp_cep(ID cepid, ID repid, T_IPV4EP *p_dstaddr, TMO tmout)

[(Parameters)]

ID	cepid	Specify the ID of a TCP communication end point (Only “1”)
ID	repid	Specify the ID of a TCP reception point (Only “1”)
T_IPV4EP	*p_dstaddr	Get the remote IP address and port number Get the IP address and port number of the remote node that requested a connection
TMO	tmout	Specify a timeout Positive value: Timeout period for waiting until a connection is completed The unit of time is 10 ms TMO_FEVR: Keeps waiting until a connection is completed (waiting forever)

[(Returns and error codes)]

E_OK	Terminated normally (connection established)
E_OBJ	Object status error (communication end point ID in use is specified)
E_TMOUT	Time out (time set in tmout expired)

[(Description)]

This API waits for a connection request for the TCP reception point repid. When a connection request is received, the API establishes a connection by using the TCP communication end point cepid and stores the IP address and port number of the remote node which requested a connection in the area indicated by the parameter p_dstaddr before returning that information to the task.

The API is in a wait state until a connection is established. A timeout period may be specified for this wait state. An error code E_TMOUT is returned if a connection cannot be established within the specified time.

When a connection is established, the return value E_OK is returned. If a connection is not established, one of the above error codes other than E_OK is returned depending on the cause of error. The cause of error is indicated in ( ) for each error code.

Specify “1” for the ID number, because only one TCP reception point and communication end point can exist in the T2. When id the specified value is other than “1”, no errors are returned and processing is continued assuming the ID number is “1”.

3.3 TCP request connection (Active Open)) tcp_con_cep()

[(API format)]

```
ER  tcp_con_cep( ID cepid, T_IPV4EP *p_myaddr, T_IPV4EP *p_dstaddr, TMO tmout )
```

[(Parameters)]

ID	cepid	Specify the ID of a TCP communication end point (Only "1")
T_IPV4EP	*p_myaddr	Specify the local IP address and port number
T_IPV4EP	*p_dstaddr	Specify the remote IP address and port number
TMO	tmout	Specify a timeout Positive value: Timeout period for waiting until a connection is ompleted The unit of time is 10 ms TMO_FEVR: Keeps waiting until a connection is completed (waiting forever)

[(Returns and error codes)]

E_OK	Terminated normally (connection established)
E_OBJ	Object status error (communication end point ID in use or local port number in use is specified)
E_TMOUT	Time out (time set in tmout expired)
E_CLS	Failed to connect (connection rejected)

[(Description)]

This API requests a connection to the IP address and port number of the other node to which to connect by using the TCP communication end point cepid, and keeps waiting until a connection is completed. Although the API remains waiting until a connection is established, a timeout period may be specified for this wait state. An error code E_TMOUT is returned if a connection cannot be established within the specified time.

If the connection request is canceled due to time out or if the connection request is rejected for reasons that for example, a port number unsupported by the remote node is specified, the communication end point cepid returns to an "unused" state.

When this API is called, whether specified communication end point is using is checked first.

The local IP address that is set in the variable tcpudp_env is assumed for the local IP address.

If any value other than 0 is specified for the local port number, this value is set. On the other hand, if TCP_PORTANY (0) is specified for the local port number, the API assigns a port number from within the range of numbers 1,024 to 5,000 in the T2.

If NADR(–1) is specified for the local IP address and port number p_myaddr, the API assigns the IP address that is set in the variable tcpudp_env for the local IP address, and for the local port number, it assigns a port number from within the range of numbers 1,024 to 5,000 in the T2.

When a connection is established, the return value E_OK is returned. If a connection is not established, one of the above error codes other than E_OK is returned depending on the cause of error. The cause of error is indicated in ( ) for each error code.

Specify "1" for the ID number, because only one communication end point can exist in the T2. When id the specified value is other than "1", no errors are returned and processing is continued assuming the ID number is "1".

3.4 TCP Terminate data transmission tcp_sht_cep()

[(API format)]

ER tcp_sht_cep(ID cepid)

[(Parameters)]

ID	cepid	Specify the ID of a TCP communication end point (Only “1”)

[(Returns and error codes)]

E_OK	Terminated normally (data transmission completed)
E_OBJ	Object status error (specified communication end point is unconnected)

[(Description)]

This API terminates data transmission as a preparatory procedure before closing a connection to the TCP communication end point cepid. More specifically, it sends FIN upon receiving ACK for the transmitted data. Because this API only prepares for disconnection processing, it does not enter a wait state.

After this API is called, no data can be transmitted to the TCP communication end point cepid. If transmission is attempted, an error code E_OBJ is returned. Data can be received though.

When a data transmission terminating procedure is completed, a value E_OK is returned. If failed to terminate data transmission, one of the above error codes other than E_OK is returned depending on the cause of error. The cause of error is indicated in ( ) for each error code.

Specify “1” for the ID number, because only one communication end point can exist in the T2. When id the specified value is other than “1”, no errors are returned and processing is continued assuming the ID number is “1”.

3.5 TCP Close communication end point tcp_cls_cep()

[(API format)]

ER tcp_cls_cep(ID cepid, TMO tmout)

[(Parameters)]

ID	cepid	Specify the ID of a TCP communication end point (Only "1")
TMO	tmout	Specify a timeout Positive value: Timeout period for waiting until a connection is closed The unit of time is 10 ms TMO_FEVR: Keeps waiting until a connection is closed (waiting forever)

[(Returns and error codes)]

E_OK Terminated normally (connection closed normally)
E_OBJ Object status error (specified communication end point is unconnected)
E_TMOUT Time out (time set in tmout expired / connection is forcibly closed)

[(Description)]

This API disconnect for the TCP communication end point cepid. After this API is called, the data transmitted from the other node is discarded.

If the connection closing processing is canceled due to time out, RST is sent from the TCP communication end point specified in this API to forcibly close the connection. In this case, because the connection is not closed normally, an error code E_TMOUT is returned.

This API waits until the communication end point enters an "unused" state regardless of whether disconnected normally or forcibly before it returns, the TCP communication end point cepid can be used immediately after returning from the API. The communication end point does not enter an "unused" state unless a connection is completely closed. According to TCP/IP standards, the communication end point remains in a TIME_WAIT state for some time which normally is several minutes until a connection is completely closed. The TIME_WAIT time, 2MSL, can be set in the T2 configuration file.

When a connection is closed normally, a value E_OK is returned. If forcibly closed, an error code E_TMOUT is returned. When one of these codes is returned, the specified TCP communication end point is in an "unused" state because a connection has been completely closed. If the TCP communication end point specified in this API does not connect, E_OBJ is returned.

Specify "1" for the ID number, because only one communication end point can exist in the T2. When id the specified value is other than "1", no errors are returned and processing is continued assuming the ID number is "1".

[(Supplements)]

In ITRON specifications, if data transmission is not completed yet, the API waits until the transmission finishes, and disconnects by sending FIN. In the T2, because multiple APIs cannot be executed at the same time, data transmission is already completed by the time this API is issued. Therefore, the API will not wait until FIN segment is sent.

3.6 Transmit TCP data tcp_snd_dat()

[(API format)]

ER tcp_snd_dat(ID cepid, VP data, INT len, TMO tmout)

[(Parameters)]

ID	cepid	Specify the ID of a TCP communication end point (Only “1”)
VP	data	Specify the top address of transmit data The top address of the data to be sent by the user
INT	len	Specify the length of transmit data Positive value
TMO	tmout	Specify a timeout Positive value: Timeout period for waiting until it finishes sending The unit of time is 10 ms TMO_FEVR: Keeps waiting until it finishes sending (waiting forever)

[(Returns and error codes)]

Positive value	Terminated normally (transmitted data size in bytes = value of len)
E_OBJ	Object status error (specified communication end point is unconnected or it has transmission completed)
E_TMOUT	Time out (time set in tmout expired)
E_CLS	Connection closed by remote node (forcibly closed by transmission/reception RST)

[(Description)]

This API transmits data from the TCP communication end point cepid. When transmitted normally, it returns the transmitted data size.

The T2 differs from ITRON TCP Specification in several points for reasons of increased RAM usage efficiency and transmission speed. In this library, the area in which transmit data is stored (hereafter called the user transmit buffer) serves as the transmit window defined in ITRON TCP Specification. Similarly, the transmit window size equals the transmit data length, and the size varies with the usage condition of this API.

In ITRON Specification, the task returns from the API when it finished copying data from the user transmit buffer to the transmit window. In this API, however, the task returns from the API when it received ACK for the data it transmitted. More specifically, if a TCP-level segment division occurs for reasons of the MSS or the receive window size of the other node, the task returns from the API only when ACK is received for all of the transmitted data, and not for the first divided data transmitted.

When data has been transmitted normally, the transmitted data size (= value of len) is returned. If failed to transmit, one of the above error codes is returned depending on the cause of error. The cause of error is indicated in ( ) for each error code.

Specify “1” for the ID number, because only one communication end point can exist in the T2. When id the specified value is other than “1”, no errors are returned and processing is continued assuming the ID number is “1”.

[(Supplements)]

In ITRON specifications, the size of transmit data is determined by the size of free space in the transmit window. Therefore, the value returned by the API when it normally terminated does not always match the third parameter len. This requires caution when the API is ported to the protocol stack compliant with ITRON TCP/IP API specifications other than the T2.

3.7 Receive TCP data tcp_rcv_dat()

[(API format)]

ER tcp_rcv_dat(ID cepid, VP data, INT len, TMO tmout)

[(Parameters)]

ID	cepid	Specify the ID of a TCP communication end point (Only “1”)
VP	data	Specify the top address of area in which received data will be stored The top address of the buffer reserved by the user for storing the received data
INT	len	Maximum size in which to store the received data Positive value
TMO	tmout	Specify a timeout Positive value: Timeout period for waiting until it finishes receiving The unit of time is 10 ms TMO_FEVR: Keeps waiting until it finishes receiving (waiting forever)

[(Returns and error codes)]

Positive value	Terminated normally (received data size in bytes)
0	Data terminated (connection closed normally. All data is received until disconnected)
E_OBJ	Object status error (specified communication end point is unconnected)
E_TMOUT	Time out (time set in tmout expired)
E_CLS	Connection closed and the receive window is empty (forcibly closed by reception RST)

[(Description)]

This API receives data from the TCP communication end point cepid.

The data transmitted from the other node of communication is stored in the receive window. This API copies data from the receive window to the user area indicated by the parameter data (hereafter referred to as “retrieving data”) and then returns. If the receive window is empty, the API is kept waiting until data is received.

If the data size stored in the receive window is smaller than the data size len to be received, data is retrieved from the receive window until it is emptied and the retrieved data size is returned.

When the connection is closed normally by the remote node and all of data is retrieved from the receive window so that no data exists in it, value 0 is returned from the API.

When data has been received normally, the received data size is returned. If failed to receive, one of the above error codes is returned depending on the cause of error. The cause of error is indicated in ( ) for each error code.

Specify “1” for the ID number, because only one communication end point can exist in the T2. When id the specified value is other than “1”, no errors are returned and processing is continued assuming the ID number is “1”.

3.8 Transmit UDP data udp_snd_dat()

[(API format)]

ER udp_snd_dat(ID cepid, T_IPV4EP *p_dstaddr, VP data, INT len, TMO tmout)

[(Parameters)]

ID	cepid	Specify the ID of a UDP communication end point (Only "1")
T_IPV4EP	*p_dstaddr	Specify the remote IP address and port number
VP	data	Specify the top address of transmit data The top address of the data to be sent by the user
INT	len	Specify the length of transmit data no fewer than 0, nor more than 1,472
TMO	tmout	Specify a timeout Positive value: Timeout period for waiting until it finishes sending The unit of time is 10 ms TMO_FEVR: Keeps waiting until it finishes sending (waiting forever)

[(Returns and error codes)]

0, positive value	Terminated normally (transmitted data size in bytes = value of len)
E_PAR	Parameter error (tmout is invalid)
E_QOVR	Queuing overflow (total transmit/receive count that can be queued is exceeded)
E_TMOUT	Time out (time set in tmout expired)
E_WBLK	Non-blocking call accepted

[(Description)]

This API sends UDP datagram from the UDP communication end point cepid after specifying the remote IP address and the port number. The API returns when the UDP datagram is stored in the transmit buffer. The transmit buffer referred to here is not the one reserved internally in the T2, but is the internal transmit buffer of the Ethernet controller (for Ethernet) or the transmit buffer reserved by the serial driver (for PPP). When data is stored in the transmit buffer, the size of the transmitted data (= value of the fourth parameter len) is returned. If the API failed to send, one of the above error codes is returned, with the cause of the error indicated in ( ).

In this API, a unicast address or multicast address can be specified for the remote IP address before sending data. No data can be sent whose destination is specified by a broadcast address. If a broadcast address is specified, the behavior and the returned value of the API are indeterminate.

In the T2, the maximum value of the UDP datagram size that can be sent at a time (It's referred to as N.) depends on the size of the transmit buffer. If M bytes of Ethernet / PPP frame data can be transmitted with the driver interface function lan_write() / ppp_write(), N is as follows.

N = M – Size of Frame header (F) – Minimum size of IP header (I) – Size of UDP header (U)

M <= 1514 (for Ethernet), M <= 1504 (for PPP)

F = 14 (for Ethernet), F = 4 (for PPP)

I = 20

U = 8

Because IP fragments are not supported in the T2, if the data larger than N bytes needs to be sent, the data must be divided to smaller than or equal to N bytes. If a data size larger than N bytes is specified, the behavior and the returned value of the API are indeterminate.

If TMO_FEVR is specified for the timeout specification tmout, the API is kept waiting until data is stored in the transmit buffer. If a positive value is specified for the timeout specification tmout, the API is kept waiting until the specified time is reached. If no data is stored in the transmit buffer by the specified time, the error code E_TMOUT is returned. If data is stored in the transmit buffer by the specified time, the size of the stored data is returned.

If TMO_NBLK is specified for the timeout specification tmout, a non-blocking call is assumed and this API is not kept waiting. When a transmit request is accepted, non-blocking call accepted E_WBLK is returned. In this case, a callback function is called at the time data is stored in the transmit buffer. The callback function has the UDP communication end point ID, function code TFN_UDP_SND_DAT, and the pointer to error code passed to it as arguments. The size of the stored data is indicated in the error code.

The following period, it is assumed that UDP transmit processing is underway. Therefore, do not rewrite the transmit data during this processing period.

* Within a period from when this API is blocking-called to when the API returns.
* For a while until the callback function (with function code TFN_UDP_SND_DAT) is called after the non-blocking call.

In the T2, except when udp_rcv_dat() is called by polling specification (TMO_POL) in the callback function called by the event code TEV_UDP_RCV_DAT, multiple UDP functions (udp_rcv_dat() and udp_snd_dat()) cannot be issued at the same time. If this API is issued while any UDP function is being processed, the error code E_QOVR is returned.

Specify “1” for the ID number, because only one communication end point can exist in the T2. When if the specified value is other than “1”, no errors are returned and processing is continued assuming the ID number is “1”.

3.9 Receive UDP data udp_rcv_dat()

[(API format)]

```
ER  udp_rcv_dat( ID cepid, T_IPV4EP *p_dstaddr, VP data, INT len, TMO tmout )
```

[(Parameters)]

ID	cepid	Specify the ID of a UDP communication end point (Only "1")
T_IPV4EP	*p_dstaddr	Get the remote IP address and port number Get the IP address and port number of the remote node that transmitted data
VP	data	Specify the top address of area in which to store the received data The top address of the buffer reserved by the user for storing the received data
INT	len	Specify the maximum size in which to store the received data no fewer than 0, nor more than 1,472
TMO	tmout	Specify a timeout Positive value: Timeout period for waiting until it finishes receiving The unit of time is 10 ms TMO_FEVR: Keeps waiting until it finishes receiving (waiting forever) TMO_NBLK: Non-blocking call TMO_POL: Polling using in callback function for the event code TEV_UDP_RCV_DAT

[(Returns and error codes)]

0, positive value	Terminated normally (received data size in bytes)
E_PAR	Parameter error (tmout is invalid)
E_QOVR	Queuing overflow (total transmit/receive count that can be queued is exceeded)
E_TMOUT	Time out (time set in tmout expired)
E_WBLK	Non-blocking call accepted
E_BOVER	Buffer overflow (large data received that exceeds the received data storage area)

[(Description)]

This API receives UDP datagram from the UDP communication end point cepid and gets the remote IP address and port number. When data has been received, the size of the received data is returned. If the API failed to receive, one of the above error codes is returned, with the cause of the error indicated in ( ).

In the T2, the maximum value of the UDP datagram size that can be received at a time (It's referred to as N.) depends on the size of the receive buffer of the driver. If M bytes of Ethernet / PPP frame data can be transmitted with the driver interface function lan_read() / ppp_read(), N is as follows.

N = M – Size of Frame header (F) – Minimum size of IP header (I) – Size of UDP header (U)

M <= 1514 (for Ethernet), M <= 1504 (for PPP)

F = 14 (for Ethernet), F = 4 (for PPP)

I = 20

U = 8

Because IP fragments are not supported in the T2, if the data larger than N bytes is received, the data is discarded.

If TMO_FEVR is specified for the timeout specification tmout, the API is kept waiting until UDP datagram is received. If a positive value is specified for the timeout specification tmout, the API is kept waiting until the specified time is reached. If no UDP datagram is received by the specified time, the error code E_TMOUT is returned.

If TMO_NBLK is specified for the timeout specification tmout, a non-blocking call is assumed and this API is not kept waiting. When a receive request is accepted, the non-blocking call-accepted E_WBLK is returned. In this case, a callback function is called at the time the received data is stored in the user's receive buffer. The callback function has the UDP communication end point ID, function code TFN_UDP_RCV_DAT, and the pointer to error code passed to it as arguments. The size of the received data is indicated in the error code, and the received data is stored in the user area data that is specified in the API in which the non-blocking call was invoked.

The following period, it is assumed that the API is waiting for the UDP data to receive and that UDP receive processing is underway.

* Within a period from when this API is blocking-called to when the API is exited.

* For a while until the callback function (with function code TFN_UDP_RCV_DAT) is invoked after the non-blocking call.

If UDP datagram is received while the API is waiting for the UDP data to receive, the received UDP datagram is copied to the received data area data beginning with the first part byte of data. If the size of the received data is equal to or less than the specified data size len, all amounts of the received data is copied. If the size of the received data is larger than the specified data size len, the received data is copied for up to an amount equal to the specified data size len and the rest is discarded. The error code returned in the former case is the size of the received data, and that in the latter case is the buffer overflow E_BOVR.

If UDP datagram is received while the API is not waiting for the UDP data to receive, a callback function for the event code TEV_UDP_RCV_DAT is called. The callback function has the UDP communication end point ID, event code TEV_UDP_RCV_DAT, and the pointer to error code (size of the received data) passed to it as arguments. The size of received data is indicated in the error code. Data can be read out only when this API that specifies polling TMO_POL for the timeout specification tmout is called in the callback function. If data is not read out by polling specification, the receive buffer of the driver is freed after exiting the callback function.

In the T2, except when udp_rcv_dat() is called by polling specification (TMO_POL) in the callback function called by the event code TEV_UDP_RCV_DAT, multiple UDP functions (udp_rcv_dat() and udp_snd_dat()) cannot be issued at the same time. If this API is issued while any UDP function is being processed, the error code E_QOVR is returned.

In this API, it is possible to receive UDP datagrams whose remote IP address is a unicast address or multicast addresses (224.0.0.0 – 239.255.255.255). However, since IGMP is not supported in the T2, routers cannot be notified of participation in multicast communication. No UDP datagrams can be received that have broadcast addresses set in their remote IP address.

Specify “1” for the ID number, because only one communication end point can exist in the T2. When if the specified value is other than “1”, no errors are returned and processing is continued assuming the ID number is “1”.

[(Supplements)]

When using a non-blocking call, note that depending on the call timing, a callback function for the event code TEV_UDP_RCV_DAT may be called before that. In this case, a callback function for the function code TFN_UDP_RCV_DAT is called upon receiving the next UDP data.

3.10 UDP callback function callback()

[(API format)]

ER callback(ID cepid, FN fncd, VP p_parblk)

[(Parameters)]

ID	cepid	The ID of a UDP communication end point (Only "1")
FN	fncd	Event code
VP	data	The address of parameter block defined for each event

[(Returns and error codes)]

A value of type ER is returned, which is not referred on the library side, however.

[(Description)]

The UDP callback function is called when the process of a non-blocking call has completed and when the UDP data is received in a sate where no udp_rcv_dat() is pending.

The processing of the UDP callback function consists in each notification, one that is created by the user. The argument fncd indicates the type of notification. Kinds of events and the timing with which the callback function is called are shown in Table 7.

Table 7. Callback Function Invocation Timing and Arguments

Timing	Argument fncd	Argument p_parblk	Remark
When the UDP data transmission is completed after calling udp_snd_dat (,, s_buf,, TMO_NBLK)	TFN_UDP_SND_DAT	Transmitted data size (equivalent to the returned value of a blocking call)	Called after sending the user's transmit data s_buf.
When the UDP data reception is completed after calling udp_rcv_dat (, d_addr, r_buf,, TMO_NBLK)	TFN_UDP_RCV_DAT	Received data size or E_BOVR (equivalent to the returned value of a blocking call)	Called when the received data has been copied to the user's data receive area r_buf. The remote IP address and port number are stored in the region pointed by d_addr.
When UDP data is received while not waiting for UDP data to receive	TEV_UDP_RCV_DAT	Received data size	The received data is copied to buf when udp_rcv_dat (,, buf,, TMO_POL) is called in the callback function. The returned value is either the received data size or E_BOVR.

The arguments to the callback function can only be read, and cannot be written to.

Although specifications of the callback function dictate that it returns a value of type ER, the returned value is not inspected on the T2 library side.

When using the UDP callback function, in UDP communication end point definitions of the T2 configuration file, be sure to set the address to the created callback function in the address of callback routine.

3.11 Get work memory size used tcpudp_get_ramsize()

[(API format)]

```
W    tcpudp_get_ramsize( void )
```

[(Parameters)]

None

[(Returns and error codes)]

The size of the work area used by the T2 (in bytes)

[(Description)]

This API gets the size of the work area used by the T2 (RAM size) which is returned as its return value. The work area is a memory area used for the TCP receive window and other purposes, which needs to be reserved in a program and initialized with parameters to the initialization function of API, tcpudp_open(). The top address of the work area is aligned to 4 byte boundaries.

EX) When this API's return value is 100, the work area used by the T2 is aligned as follows:

```
UW    work[100/4];
```

This API can be used for the following purposes:

1) To reserve a work area using a static array

Although this API calculates the size of the work area used by the T2 according to the contents set in the TCP/IP configuration file, the user will have difficulty calculating it in advance. Therefore, when the contents of the T2 configuration file have been determined, rebuild the program and execute this API in a debugger to examine its returned value. Then reserve as much memory array for the work area as the size indicated by the returned value.

Note, however, that if the T2 configuration file is altered, the necessary size of the work area changes. In that case, recalculate the size of the work area by using this API. Furthermore, as a processing to check for errors, always be sure to call this API in the initialization processing and compare the returned value with the final size of the work area determined by the user. If they do not match, branch to error handling. That way, errors can be found at the debugging or test stages.

2) To reserve a work area from dynamic memory

Call this API in the initialization step of the application program to calculate the size of the work area. Reserve the calculated size of the work area from dynamic memory and pass the reserved area to the initialization function tcpudp_open() to have the work area initialized with it.

3.12 Open T2 library tcpudp_open()

[(API format)]

```
ER    tcpudp_open( UW *work )
```

[(Parameters)]

B	*work	Address of the work area used by the T2

[(Returns and error codes)]

E_OK	Terminated normally
Negative value	Failed to initialize

[(Description)]

This API initializes the T2 library. In the library initialization processing, the API performs memory allocation and initialization of the internal managed area, as well as invokes the TCP/IP cyclic processing each task used by the library.

Specify the top address of the work area used by the T2 for work. The necessary size of the work area may be obtained from the returned value of tcpudp_get_ramsize().

Call this API before lan_open() when it is used in combination with Ethernet or before ppp_start() when used in combination with PPP too.

3.13 TCP/IP processing _process_tcpip()

[(API format)]

```
void    _process_tcpip( void )
```

[(Parameters)]

None

[(Returns and error codes)]

None

[(Description)]

This API processes the TCP/IP protocol of the T2 library.

Make sure this API is called at intervals of less than 10 ms (for example, by using a Timer interrupt).

Since the TCP/IP protocol of the T2 manages time in 10 ms units, if the invocation interval exceeds 10 ms, one or more of the following problems may be incurred:

* The timeout specified in an API does not occur within the designated time.
* Retransmission operations are not performed at designated intervals.
* The zero window probe is not performed at designated intervals.
* The timeout period of 2MSL for disconnecting does not conform to the designated time.

3.14 Close T2 library tcpudp_close()

[(API format)]

ER tcpudp_close(void)

[(Parameters)]

None

[(Returns and error codes)]

E_OK	Terminated normally
Negative value	Failed to close

[(Description)]

This API closes the T2 library. In the library closing processing, the API terminates the TCP/IP cyclic processing used by the library.

Before calling this API, be sure to disconnect all of the TCP and UDP communication end points and enter an "unused" state.

Before calling this API, call lan_close() when it is used in combination with Ethernet or ppp_stop() when used in combination with PPP.

The work area specified by the library open function tcpudp_open() is deallocated after executing this API, so that the work area is free to use in the application program.

3.15 T2 Limitations and Usage Precautions

The T2 limitations and usage precautions are described in (1) to (17) below.

(1) The cancel pending process functions tcp_can_cep() and udp_can_cep().
(2) The data transmit and receive functions that use copy-saving APIs tcp_get_buf(), tcp_snd_buf(), tcp_rcv_buf() and tcp_rel_buf(), are not supported.
(3) The emergency data transmit and receive functions tcp_snd_oob() and tcp_rcv_oob() are not supported.
(4) The option setting and acquisition functions tcp_set_opt(), tcp_get_opt(), udp_set_opt() and udp_get_opt() are not supported.
(5) In TCP APIs, multicast address, broadcast address and loopback address cannot be specified for the remote IP address.
(6) In UDP APIs, broadcast address and loopback address cannot be specified for the remote IP address.
(7) The callback feature is not supported in TCP. The address of callback routine specified in TCP communication end point definition have no effect.
(8) The timeout specification that can be accepted in TCP APIs are only waiting forever (TMO_FEVR) and positive values. If a non-blocking call (TMO_NBLK) or polling (TMO_POL) is specified for the timeout specification, the behavior is indeterminate.
(9) Polling (TMO_POL) that is specified to be the UDP receive function udp_rcv_dat() is accepted in only the callback function called by the event code TEV_UDP_RCV_DAT. If polling (TMO_POL) is specified in other than the callback function, a parameter error (E_PAR) is returned.
(10) Since IGMP is not supported, routers cannot be requested for transfer of IP datagrams whose remote IP addresses are multicast addresses.
(11) The number of APIs that can be executed at the same time in the T2 is one for TCP and UDP each. If multiple APIs are executed at the same time, the behavior is indeterminate.
(12) For TCP, the T2 only supports the MSS in header option. All options but MSS are ignored when receiving TCP segments.
(13) For IP, the T2 does not support IP options and fragments. If the received IP datagram contains IP options or has been fragmented, the datagram is discarded.
(14) For ICMP, the T2 only supports receiving an echo request and transmitting an echo response in return for that. If the received packet contains any other ICMP messages, the packet is discarded.
(15) For PPP, the T2 does not support compression-related options (i.e., protocol field compression, address and control field compression and TCP/IP header compression). If a request is received that requires setting compression-related options, Configure-Rejects of setting is transmitted in response.
(16) Calling APIs in an interrupt handler does not recommend.
(17) Before calling the function tcpudp_close(), be sure to close all of the communication end points and enter an “unused” state. If tcpudp_close() is called while communication is on, the program may behave erratically.

Website and Support

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

Inquiries
http://www.renesas.com/inquiry
csc@renesas.com

Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Apr.24.09	—	First edition (M3S-T2-Tiny Ver.1.01 Release02E)

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life

   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.