

Renesas microcomputer

TCP/IP for Embedded system

R20UW0032EJ0105

Rev.1.05

Apr 01, 2014

M3S-T4-Tiny: Ethernet Driver Interface Specification

Introduction

This document explains the direction for uses of the M3S-T4-Tiny Ethernet driver interface.

Target Device

Renesas microcomputer

Contents

1. Overview	2
2. Preconditions	2
3. Internal Configuration of the Ethernet Driver	3
4. Function Specifications	4
5. Detailed Description of Driver Functions	4
5.1 lan_open.....	5
5.2 lan_close.....	5
5.3 lan_read.....	6
5.4 rcv_buff_release.....	6
5.5 lan_write.....	7
5.6 lan_reset.....	7
5.7 tcp_api_slp.....	8
5.8 tcp_api_wup.....	8
5.9 udp_api_slp	9
5.10 udp_api_wup	9
5.11 dis_int.....	10
5.12 ena_int.....	10
5.13 tcpudp_act_cyc	11
5.14 tcpudp_get_time	11
5.15 lan_inthdr	12
5.16 report_error.....	13

1. Overview

This specification describes the Ethernet driver interface specification for the Tiny TCP/IP library “M3S-T4-Tiny” (called the T4). Although the T4 library supports Ethernet-based communication, the part of whose depending on LAN controller specifications is separated from the main library to a driver unit, and it makes possible you to customize that part of library as necessary.

In this specification, it describes the specifications of the functions that you may need to use when developing Ethernet drivers according to the specifications of your LAN controller.

2. Preconditions

- The library is useful for only one Ethernet interface (not including loopback).
- Packets in Ethernet format are sent and received using the Ethernet procedure.
- The transmit/received packet data are assumed to be those of Ethernet packets except CRC.
- The transmit packet data is separated into the header and the data parts. The header part is stored in a global variable and the data is stored in a 1-byte integer type (char) array before being passed to the driver.
- The received packet data for one octet is stored in the receive buffer in network byte order (big endian), when the data is transferred by Ethernet in order.
- The maximum length of the received packet data is limited to 1,520 octets (allowed range of Ethernet).
- When the receive buffer length in the driver is shorter than 1,520 octets, it is the maximum length of the received packet data.
- The receive buffer is controlled by the driver, whose pointer is passed to the global variable. The number of buffers can be defined by the users.
- The receive buffers are assumed to be located at addresses aligned with two-byte boundaries.

3. Internal Configuration of the Ethernet Driver

The relationship between protocol processing section and the driver in T4 library is shown in Fig 1. The driver interface is outlined in Table 1 (detailed in Section 5).

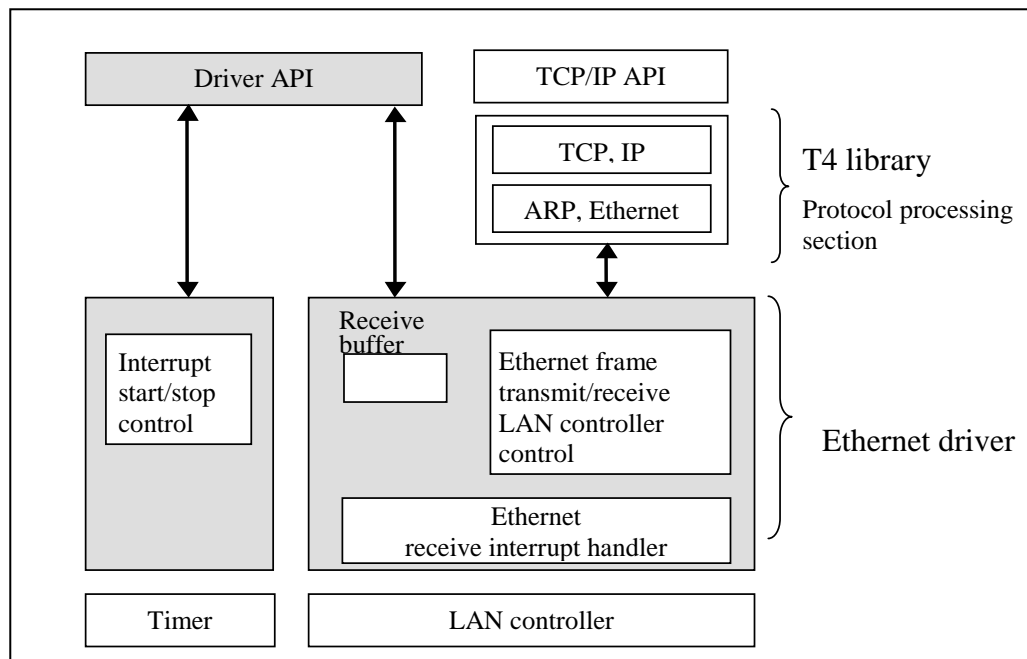


Fig 1. Block diagram of the Ethernet Driver

Table 1. List of Driver Interface

Function Name	Description
ER lan_open(void)	Initialize and start LAN controller
ER lan_close(void)	Deactivate the LAN controller
H lan_read(UB, B**)	Receive data
H rcv_buff_release(UB)	Release the receive buffer of the driver
H lan_write(UB, B*, H, B*, H)	Send data
void lan_reset(UB)	Reset LAN controller
void tcp_api_slp(ID)	Wait for completion of API
void tcp_api_wup(ID)	Cancel the wait state of the API completion
void udp_api_slp(ID)	Wait for completion of API
void udp_api_wup(ID)	Cancel the wait state of the API completion
void ena_int(void)	Enable interrupt used T4 cyclic
void dis_int(void)	Disable interrupt used T4 cyclic
void tcpudp_act_cyc(UB)	Control cyclic activation of TCP/IP processing function
UH tcpudp_get_time(void)	Get time information
void lan_inthdr(void)	Interrupt handler
void report_error (UB, H , UB*)	Report error

The driver uses the functions listed in Table 1, as the interface to initialize, transmit and receive the data and operate other task.

4. Function Specifications

4.1 Global Variables

- (1) Ethernet address
UB _myethaddr[6]

This variable is stored in a MAC address of a LAN controller. It is possible to be set by users in configuration file of T4. When this variable is set all 0s by user, a MAC address is read from ROM and is set to a LAN controller.

5. Detailed Description of Driver Functions

Prototype

Shows the API format.

Explanation

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

Arguments

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

Return value

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

5.1 lan_open

Prototype

```
ER lan_open( void )
```

Explanation

This function initializes the Ethernet controller to make it useful for other driver functions.

It also initializes the receive buffers. If the global variable (`_myethaddr`) is all 0s, the Ethernet address stored in EEPROM is set to the Ethernet controller and also copied to `_myethaddr`.

If the global variable (`_myethaddr`) is not all 0s, its value is set to the Ethernet controller.

Called by the user program.

Arguments

Argument	Type	Explanation
none		

Return value

Type	Explanation
0	normal
Negative value	Error (could not be activated)

5.2 lan_close

Prototype

```
ER lan_close ( void )
```

Explanation

This function deactivates operation of the Ethernet controller.

Called by the user program.

Arguments

Argument	Type	Explanation
none		

Return value

Type	Explanation
E_OK	Normal
-1	Error

5.3 lan_read

Prototype

H lan_read(UB lan_port_no , B **buf)

Explanation

Store the receive buffer pointer to the parameter pointer (**buf) indicated by LAN port channel parameter.
And, return the status corresponds receive status.

Used in a protocol processing section.

Arguments

Argument	Type	Explanation
lan_port_no	UB	Channel number
buf	B **	Pointer to be stored in the receive buffer area.

Return value

Type	Explanation
0 or greater	Size of the received packet
-1	Error (could not be activated)
-2	Controller is inactive
-5	Ethernet controller is operating erratically, or Ethernet controller needs to be reset
-6	Received packet CRC error

5.4 rcv_buff_release

Prototype

H rcv_buff_release(UB lan_port_no)

Explanation

This function release the receive buffer (specified in lan_read parameter) using in T4 indicated LAN port number parameter.

If the timing that reception buffer release permission (rcv_buff_release()) is returned is not defined, please control a reception buffer.

Used in a protocol processing section.

Arguments

Argument	Type	Explanation
lan_port_no	UB	LAN port number corresponds to release buffer

Return value

Type	Explanation
0	completed

5.5 lan_write

Prototype

```
H lan_write( UB lan_port_no, B *hdr, H dlen, B *buf, H dlen )
```

Explanation

This function writes the contents of the header and data areas passed by the parameters to the transmit buffer of the Ethernet controller for one packet before sending a packet.

Used in a protocol processing section.

Arguments

Argument	Type	Explanation
lan_port_no	UB	LAN port number to send
hdr	B*	Pointer to the header area to send one
dlen	H	Length of the header to send
buf	B*	Pointer to the data area to send
dlen	H	Length of the data to send

Return value

Type	Explanation
0	Transmission is succeeded.
-5	Transmission is failed.

5.6 lan_reset

Prototype

```
void lan_reset( UB lan_port_no )
```

Explanation

This function resets the Ethernet controller as shown in the following step. This operation does not involve initializing the receive buffers and other variables.

- (1) Deactivates the Ethernet controller (by using `lan_close()`).
- (2) Set the registers, etc. of the Ethernet controller again.
- (3) Restart the Ethernet controller.

Used in a protocol processing section.

Arguments

Argument	Type	Explanation
lan_port_no	UB	LAN port number to reset

Return value

Type	Explanation
none	

5.7 tcp_api_slp

Prototype

```
void tcp_api_slp( ID cepid )
```

Explanation

In the T4, it determines whether or not the issued API has been completed, after issuing each API.

This function is called every time it is checked. The intervals of each check can be altered by user definition, and can be switched to another task until the API is completed.

When using the μ ITRON OS, for example, it is possible to switch to another task by calling `tslp_tsk()` or `dly_tsk()` in this function. The CPU can be used effectively.

Furthermore, if the MCU supports the wait mode (in which the CPU clock is deactivated until an interrupt is generated), it is possible to reduce the electricity consumption in the chip by the wait mode in this function.

If this function is empty (no processing), the completed check will be performed at short intervals, but it is not a problem for a function.

Used in protocol processing section

Arguments

Argument	Type	Explanation
cepid	ID	The CEPID of start waiting API

Return value

Type	Explanation
none	

5.8 tcp_api_wup

Prototype

```
void tcp_api_wup( ID cepid)
```

Explanation

This function is called when the issued TCP API has been completed, and it cancels the wait state by the function `tcp_api_slp()` that waits for completion of API.

When using the μ ITRON OS, for example, it is possible to wait until the API is completed by calling the system call `slp_tsk()` in `tcp_api_slp()`, and to cancel the wait in API completion by calling the system call `iwup_tsk()` in this function. The CEPID of completed API is set to argument. User can know which task should be waking up by this ID.

If the wait mode is entered into in the function `tcp_api_slp()`, the wait in `tcp_api_slp()` is not always needed to cancel by this function so that the wait is canceled by an interrupt. If the wait of the function `tcp_api_slp()` is canceled by an interrupt or other factors, this function can be empty (no processing) without causing any problem.

Used in protocol processing section.

Arguments

Argument	Type	Explanation
cepid	ID	The CEPID of completed API

Return value

Type	Explanation
none	

5.9 udp_api_slp

Prototype

```
void udp_api_slp( ID cepid )
```

Explanation

In the T4, it determines whether or not the issued API has been completed, after issuing each API.

Other explanation is same as tcp_api_slp().

Arguments

Argument	Type	Explanation
cepid	ID	The CEPID of start waiting API

Return value

Type	Explanation
none	

5.10 udp_api_wup

Prototype

```
void udp_api_wup( ID cepid)
```

Explanation

This function is called when the issued API has been completed, and it cancels the wait state by the function `api_slp()` that waits for completion of API.

Other explanation is same as tcp_api_wup().

Arguments

Argument	Type	Explanation
cepid	ID	The CEPID of completed API

Return value

Type	Explanation
none	

5.11 dis_int

Prototype

```
void dis_int( void )
```

Explanation

This function is called when the cancel API (tcp_can_cep / udp_can_cep) is called. This function makes T4 cycle (timer interrupt) disabled.

Used in protocol processing section.

Arguments

Argument	Type	Explanation
none		

Return value

Type	Explanation
none	

5.12 ena_int

Prototype

```
void dis_int( void )
```

Explanation

This function is called when the cancel API (tcp_can_cep / udp_can_cep) is called. This function makes T4 cycle (timer interrupt) enabled.

Used in protocol processing section.

Arguments

Argument	Type	Explanation
none		

Return value

Type	Explanation
none	

5.13 tcpudp_act_cyc

Prototype

```
void tcpudp_act_cyc( UB cycact )
```

Explanation

This function controls cyclic activation of TCP/IP processing function `_process_tcpip()` corresponding to parameter `cycact`. The interval of cyclic activations of `_process_tcpip()` must be set to 10 ms or less.

Used in protocol processing section.

Precautions

- The interval of cyclic activations of the TCP/IP processing function `_process_tcpip()` is set to 10 ms or less.
- This function is called from the open T4 library function `tcpudp_open()` and the close T4 library function `tcpudp_close()`.

Arguments

Argument	Type	Explanation
cycact	UB	Set to start or stop cyclic activation of TCP/IP processing function. 0: Stop cyclic activation of TCP/IP processing function. 1: Start cyclic activation of TCP/IP processing function.

Return value

Type	Explanation
none	

5.14 tcpudp_get_time

Prototype

```
UH tcpudp_get_time( void )
```

Explanation

This function returns the current time. The accuracy of current time is 10 ms, using integer division, rounding down. Current time is 0 when system starts. Current time is incremented each 10ms. Current time returns 0 when this value would overflow.

Used in protocol processing section.

Arguments

Argument	Type	Explanation
none		

Return value

Type	Explanation
	The current time

5.15 lan_inthdr

Prototype

```
void lan_inthdr( void )
```

Explanation

This function is called by an interrupt signal from the Ethernet controller.

Used in the INT1 interrupt handler.

Arguments

Argument	Type	Explanation
none		

Return value

Type	Explanation
none	

5.16 report_error

Prototype

```
void report_error( UB lan_port_no , H error_code, UB *buf);
```

Explanation

This function notifies error information occurred in T4 Library receiving process using function argument, and notifies pointer that indicates error packet data area.

User can make process corresponding error code in this function.

In case error occurred in same timing, error code will be output bigger code. (near 0)

Arguments

Argument	Type	Explanation
lan_port_no	UB	Specify the port that has error occurred
lan_port_no	UB	Specify the port that has error occurred
error_code	H	-1 : receive length error receiving data length is out of 60-1514.
		-2 : network layer error receiving data is not IP packet or ARP packet
		-3 : transport layer error receiving data is not TCP or UDP or ICMP packet. The received packet reach to the transport layer, and, IP address is matched with T4 included.
		-21 : ARP message error 1 Destination IP address is mismatch in ARP header.
		-22 : ARP message error 2 Data error in ARP header. hardware type 0x0001 : Ethernet upper layer protocol type 0x0800 : IP hardware address length 0x06 : MAC address protocol address length 0x04 : IP address
		-41 : IP header error 1 Destination IP address is mismatch in IP header
		-42 : IP header error 2 Source IP address is multicast or broadcast address.
		-43 : IP header error 3 Source IP address is loopback address.
		-44 : IP header error 4 IP version is not "4"

Argument	Type	Explanation
error_code	H	-44 : IP header error 4 IP version is not "4"
		-45 : IP header error 5 Including IP header options.
		-46 : IP header error 6 Incorrect IP checksum
		-47 : IP header error 7 Incorrect IP header length
		-48 : IP header error 8 Incorrect IP address
		-49 : IP header error 9 Including IP fragment flag.
		-61 : TCP header error 1 Any TCP connections are not established. And, all TCP communication endpoint has been established, the additional connection is coming. Example1: Remote host specifies the port 80, but T4 listen the port 20 only, this error code will occur. Example2: Remote host specifies the port 80, but T4 does not listen the any port, this error code will occur. Example3: T4 uses 5 TCP communication endpoint, all these are connected and, additional connection is coming, this error code will occur.
		-62 : TCP header error 2 Incorrect TCP checksum
		-81 : UDP header error 1 Incorrect UDP checksum
		-82 : UDP header error 2 UDP checksum is zero and variable "udp_enable_zerochecksum" is set value excepting 0.
		-83 : UDP header error 3 Incorrect UDP port number
		-101 : ICMP header error 1 Incorrect ICMP type (excepting echo request 0x08)
buf	UB *	pointer indicating error packet header.

Return value

Type	Explanation
none	

Website and Support

Renesas Electronics Website

<http://www.renesas.com/>

Inquiries

<http://www.renesas.com/contact/>

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.05	Apr.01.14		lan_read(B**) -> lan_read(UB, B**) rcv_buff_release(void) -> rcv_buff_release(UB) lan_write(B*, H, B*, H) -> lan_write(UB, B*, H, B*, H) lan_reset(void) -> lan_reset(UB)
1.04	Jun.21.13	-	Updated document template.
1.03	Apr.01.12	8-10	Change spec. api_slp(void) -> tcp_api_slp(ID), udp_api_slp(ID) api_wup(ID) -> tcp_api_wup(ID), udp_api_slp(ID) Correct typo.
		5	lan_read() argument "Buf" -> "buf"
1.02	Aug.23.11	13	Added report_error() function.
1.01	Jan 06.11	9	Change api_wup() prototype.
1.00	Oct.07.10	-	First edition issued

General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins

Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

- The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

The state of the product is undefined at the moment when power is supplied.

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

3. Prohibition of Access to Reserved Addresses

Access to reserved addresses is prohibited.

- The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

- When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

- The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation 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.
 2. 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.
 3. 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.
 4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
 5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). 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 for which it is not intended. 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 which the product is not intended by Renesas Electronics.
 6. 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.
 7. 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 systems manufactured by you.
 8. 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.
 9. 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. You should not use Renesas Electronics products or 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. When exporting the Renesas Electronics 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.
 10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
 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.



SALES OFFICES

Renesas Electronics Corporation

<http://www.renesas.com>

Refer to "<http://www.renesas.com/>" for the latest and detailed information.

Renesas Electronics America Inc.

2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited

1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited

Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K.
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH

Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.

Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.

Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited

Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.

13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.

80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.

Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.

12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141