User's Manual

TCP/IP for Embedded system M3S-T4-Tiny: Ethernet Driver Interface Specification

Renesas Microcomputer

Renesas Electronics
www.renesas.com

Rev.1.09 2021.04

Notice

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Contact information

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

Trademarks

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

How to Use This Manual

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

1. Organization

This manual can be broadly divided into the following units.

1. Overview
2. Preconditions
3. Internal Configuration of the Ethernet Driver Interface
4. Function Specifications
5. Detailed Description of Driver Functions

Table of Contents

1. Overview

This manual 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 manual, 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 preconditions are as follows.

1. Packets in Ethernet format are sent and received using the Ethernet procedure.

2. The transmit/received packet data are assumed to be those of Ethernet packets except CRC.

3. 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.

4. 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.

5. 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.

6. 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.

3. Internal Configuration of the Ethernet Driver Interface

The relationship between protocol processing section and Ethernet driver in T4 library is shown in **Figure 3.1**. Ethernet driver interface is outlined in **Table 3.1** (detailed in Section 5).

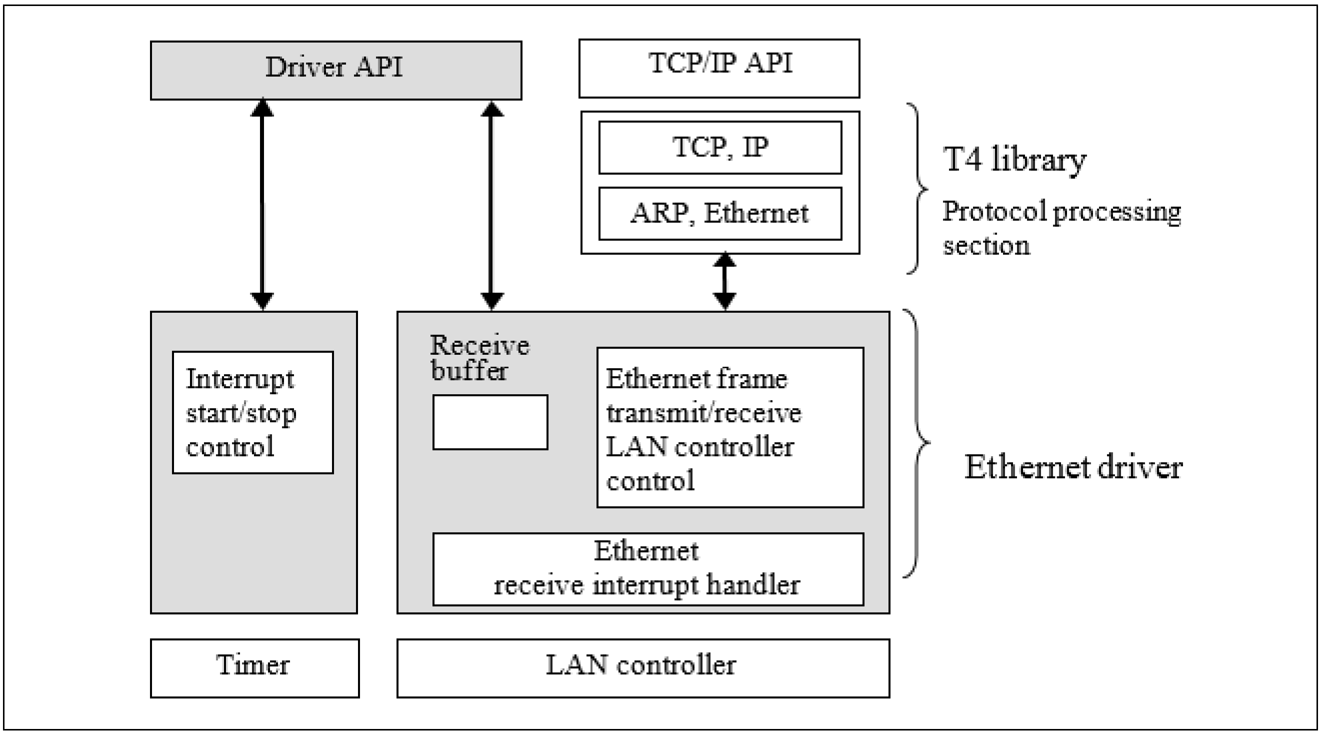
Driver API
TCP/IP API
TCP, IP
ARP, Ethernet
T4 library
Protocol processing section
Interrupt start/stop control
Receive buffer
Ethernet frame transmit/receive LAN controller control
Ethernet receive interrupt handler
Ethernet driver
Timer
LAN controller

Figure 3.1 Block diagram of the Ethernet Driver

The driver uses the functions listed in **Table 3.1**, as the interface to initialize, transmit and receive the data and operate other task.

Table 3.1 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
void get_random_number(UB *, UW)	Random number acquisition
void get_hash_value(UB, UB*, UW, UB**, UW*)	Get hash value
void register_callback_linklayer(callback_from_system_t)	Register the callback function called when Ethernet link layer connected/disconnected.
H lan_check_link(UB)	Check the link status of the Ethernet layer.

4. Function Specifications

4.1 Global Variables

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

Each API details are shown as following format.

< Format >

Shows the API format.

< Explanation >

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

< Argument >

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

< Format >

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.

< Argument >

Argument	I/O	Type	Explanation
none	-	-	-

< Return Value >

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

5.2 lan_close

< Format >

ER lan_close (void)

< Explanation >

This function deactivates operation of the Ethernet controller.

Called by the user program.

< Argument >

Argument	I/O	Type	Explanation
none	-	-	-

< Return Value >

Type	Explanation
0	Normal
-1	Error

5.3 lan_read

< Format >

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.

< Argument >

Argument	I/O	Type	Explanation
lan_port_no	Input	UB	Channel number
buf	Output	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

< Format >

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.

< Argument >

Argument	I/O	Type	Explanation
lan_port_no	Input	UB	LAN port number corresponds to release buffer

< Return Value >

Type	Explanation
0	completed

5.5 lan_write

< Format >

H lan_write(UB lan_port_no, B *header, H header_len, B *data , H data_len)

< 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.

< Argument >

Argument	I/O	Type	Explanation
lan_port_no	Input	UB	LAN port number to send
header	Input	B*	Pointer to the header area to send one
header_len	Input	H	Length of the header to send
data	Input	B*	Pointer to the data area to send
data_len	Input	H	Length of the data to send

< Return Value >

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

5.6 lan_reset

< Format >

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.

< Argument >

Argument	I/O	Type	Explanation
lan_port_no	Input	UB	LAN port number to reset

< Return Value >

Type	Explanation
none	-

5.7 tcp_api_slp

< Format >

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 μITRON 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.

< Argument >

Argument	I/O	Type	Explanation
cepid	Output	ID	The CEPID of start waiting API

< Return Value >

Type	Explanation
none	-

5.8 tcp_api_wup

< Format >

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 µITRON 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 wakening 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.

< Argument >

Argument	I/O	Type	Explanation
cepid	Output	ID	The CEPID of completed API

< Return Value >

Type	Explanation
none	-

5.9 udp_api_slp

< Format >

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().

< Argument >

Argument	I/O	Type	Explanation
cepid	Output	ID	The CEPID of start waiting API

< Return Value >

Type	Explanation
none	-

5.10 udp_api_wup

< Format >

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 udp_api_slp() that waits for completion of API.

Other explanation is same as tcp_api_wup().

< Argument >

Argument	I/O	Type	Explanation
cepid	Output	ID	The CEPID of completed API

< Return Value >

Type	Explanation
none	-

5.11 ena_int

< Format >

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.

< Argument >

Argument	I/O	Type	Explanation
none	-	-	-

< Return Value >

Type	Explanation
none	-

5.12 dis_int

< Format >

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.

< Argument >

Argument	I/O	Type	Explanation
none	-	-	-

< Return Value >

Type	Explanation
none	-

5.13 tcpudp_act_cyc

< Format >

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.

< Argument >

Argument	I/O	Type	Explanation
cycact	Input	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

< Format >

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.

< Argument >

Argument	I/O	Type	Explanation
none	-	-	-

< Return Value >

Type	Explanation
UH	The current time

5.15 lan_inthdr

< Format >

void lan_inthdr(void)

< Explanation >

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

Used in the INT1 interrupt handler.

< Argument >

Argument	I/O	Type	Explanation
none	-	-	-

< Return Value >

Type	Explanation
none	-

5.16 get_random_number

< Format >

void get_random_number (UB *data, UW len)

< Explanation >

Set the random number to the data pointer with the length that specified data length.

< Argument >

Argument	I/O	Type	Explanation
data	Output	UB *	Data pointer to write the random value.
len	Input	UW	Byte length for needed random value.

< Return Value >

Type	Explanation
none	-

5.17 get_hash_value

< Format >

void get_hash_value(UB lan_port_no, UB * message, UW message_len, UB **hash, UW *hash_len)

< Explanation >

For the specified LAN port number, output the hash value (message digest) and its byte length calculated form the specified message and its byte to the specified pointers. Any hash algorithm is OK. MD5 is recommended in RFC6528, but MD5 is not generally recommended and SHA256 is often used instead.

< Argument >

Argument	I/O	Type	Explanation
lan_port_no	Input	UB	LAN port number corresponds to get hash value.
message	Input	UB *	Message.
message_len	Input	UW	Byte length for message.
hash	Output	UB **	Hash value.
hash_len	Output	UW *	Byte length for above hash value.

< Return Value >

Type	Explanation
none	-

5.18 report_error

< Format >

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

< Explanation >

This function notifies the Error information that is occurred in the T4 Library to the Ethernet Driver from lan_read(). And notifies pointer that indicates error packet data area to the Ethernet Driver.

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)

< Argument >

Argument	I/O	Type	Explanation
lan_port_no	Output	UB *	Specifiy the port that has error occured
error_code	Output	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 The receiving data that comes from lan_read() is IP packet, but not TCP packet or UDP packet or ICMP,IGMP packet. IP address is matched with T4 IP address, the packet reaches at transport layer.
			-21 : ARP message error 1 The receiving data that comes from lan_read() is ARP packet, but destination IP address in ARP message is unmatched.
			-22 : ARP message error 2 The receiving data that comes from lan_read() is ARP packet, but the data error in ARP message. - 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 The receiving data that comes from lan_read() is IP packet, but destination IP address in IP header is unmatched.

			-42 : IP header error 2 The receiving data that comes from lan_read() is IP packet, but source IP address in IP header is multicast address or broadcast address.
			-43 : IP header error 3 The receiving data that comes from lan_read() is IP packet, but source IP address in IP header is loopback address.
			-44 : IP header error 4 The receiving data that comes from lan_read() is IP packet, but IP version is not “4” in IP header.
			-45 : IP header error 5 The receiving data that comes from lan_read() is IP packet, but including IP header options.
			-46 : IP header error 6 The receiving data that comes from lan_read() is IP packet, but incorrect IP checksum
			-47 : IP header error 7 The receiving data that comes from lan_read() is IP packet, but IP data length included in IP header is larger than lan_read() return value(receive data length), or smaller than IP header minimum length(20 byte).
			-48 : IP header error 8 The receiving data that comes from lan_read() is IP packet, but source IP address included in the IP header is network address(ex: xxx.xxx.xxx.0/24) or broadcast address(ex: xxx.xxx.xxx.255/24).
			-49 : IP header error 9 The receiving data that comes from lan_read() is IP packet, but IP fragment flag is ON in IP header.
			-61 : TCP header error 1 The receiving data that comes from lan_read() is TCP/IP packet, but the port number included in the TCP header does not match the communication endpoint that status is “established” or “listen”. 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 The receiving data that comes from lan_read() is TCP/IP packet, but incorrect TCP checksum
			-81 : UDP header error 1 The receiving data that comes from lan_read() is UDP/IP packet, but incorrect UDP checksum
			-82 : UDP header error 2 The receiving data that comes from lan_read() is UDP/IP packet, but UDP checksum is zero and variable “udp_enable_zerochecksum” is set value excepting 0.
			-83 : UDP header error 3 The receiving data that comes from lan_read() is UDP/IP packet, but incorrect UDP port number
			-101 : ICMP header error 1 The receiving data that comes from lan_read() is ICMP/IP packet, but incorrect ICMP type (excepting echo request 0x08)
			-121 : IGMP header abnormality 1 The receiving data that comes from lan_read() is ICMP/IP packet, checksum of IGMP header was abnormal
			-122 : IGMP header abnormality 2 The receiving data that comes from lan_read() is ICMP/IP packet, if the IGMP type is other than IGMPv2Report (0x16) and IGMPv1Report (0x12)
			-131 : Illegal DHCP process
			-132 : DHCP transmission timeout occurred
buf	Output	UB *	pointer indicating error packet header.

< Return Value >

Type	Explanation
none	-

5.19 register_callback_linklayer

< Format >

void register_callback_linklayer(callback_from_system_t call_fp)

< Explanation >

This function is the function that registers the callback function to notify event from link layer to T4. To receive the notification, please register the function pointer has type of callback_from_system_t. Callback function can be set only one function. If user registers several functions, always over-write.

The following is callback function.

typedef ER (*callback_from_system_t) (UB channel, UW eventid, VP param);

If function pointer is not registered, T4 will not notify of the status. The argument channel means LAN port number, eventid is kind of notification, param is the pointer (currently, in not use, therefore this value is zero) that provide parameter value to the user.

ETHER_EV_LINK_OFF : Disconnected the Ethernet layer.

ETHER_EV_LINK_ON : Connected the Ethernet layer.

ETHER_EV_COLLISION_IP : Detected IP address collision.

< Argument >

Argument	I/O	Type	Explanation
call_fp	Input	callback_from_system_t	Register the callback function

< Return Value >

Type	Explanation
none	-

5.20 lan_check_link

< Format >

H lan_check_link (UB lan_port_no)

< Explanation >

Check the link status of the Ethernet layer.

< Argument >

Argument	I/O	Type	Explanation
lan_port_no	Input	UB	Channel number

< Return Value >

Type	Explanation
0	Link Off
1	Link On

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Oct 07, 2010	—	First Edition issued
1.01	Jan 06, 2011	8	Change api_wup() prototype.
1.02	Aug 23, 2011	12	Added report_error() function.
1.03	Apr 01, 2012	8-9	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. lan_read() argument “Buf” -> “buf”
1.04	Jun 21, 2013	—	Updated document template.

1.05	Apr 01, 2014	6-16	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)

1.06	Aug 07, 2015	—	Update document template.
1.07	Dec 01, 2015	3,16	Added I/O information for each functions arguments. Added get_random_number() function.
1.08	Nov 30, 2016	8,30	Update document template. Added the information about register_callback_linklayer() function that is called L2 link layer will be Link Off or On. Added the information about lan_check_link() function.
1.09	Apr 01, 2019	9, 28	Added the information about get_hash_value() function that is called to get the hash value.

TCP/IP for Embedded system M3S-T4-Tiny:
Ethernet Driver Interface Specification

Publication Date: Rev.1.00 Oct 07, 2010
Rev.1.09 Apr 01, 2021

Published by: Renesas Electronics Corporation

TCP/IP for Embedded system M3S-T4-Tiny: Ethernet Driver Interface Specification

Renesas Electronics Corporation