RZ/T1 Group

Multi Port Ethernet Driver

R01AN3501EJ0110
Rev.1.10
Sep 1, 2017

Outline

This application note explains a sample program to be run with the Ethernet drivers that support evaluation board provided in the Renesas Starter Kit for RZ/T1 Group, which incorporates a microcontrollers of the RZ/T1 group.

Evaluations proceeds by issuing ping commands on a device connected to the evaluation board and confirming the responses from the board.

Target Devices

RZ/T1 Group

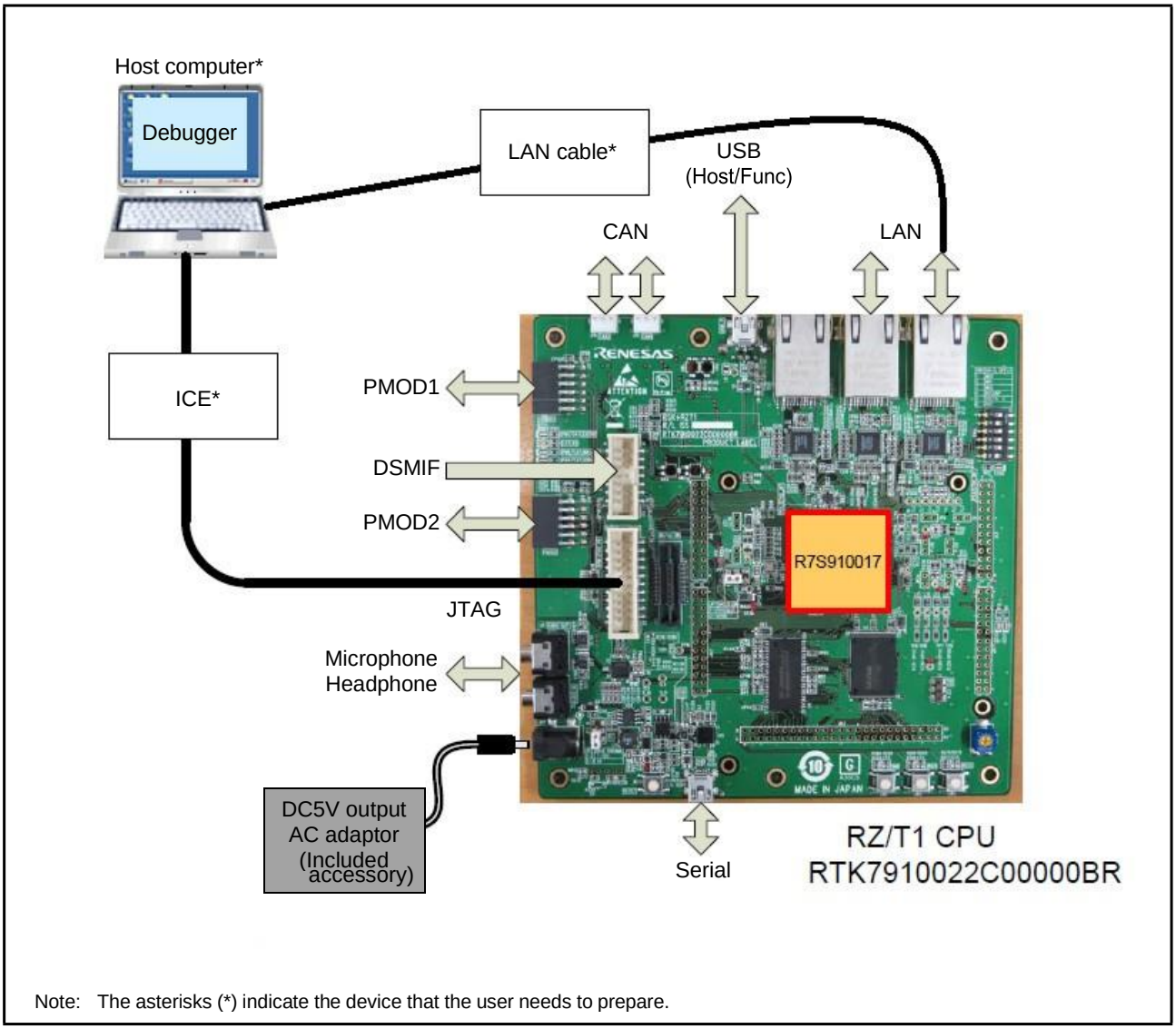
Table of Contents

1. Specifications

Peripheral devices used in this sample program and the applications are listed in Table 1.1. A connection diagram is illustrated in Figure 1.1.

Table 1.1 Peripheral Modules and Applications

Peripheral Module	Application
Cortex-R4F	Running the program.
Tightly-coupled memory (TCM)	Storing program code for execution by the Cortex-R4F and data for use in the programs
Ethernet MAC (ETHERC)	Transfers data through the Ethernet port.
Ethernet switch (ETHSW)	The Ethernet switch is also utilized.



Note: The asterisks (*) indicate the device that the user needs to prepare.

Figure 1.1 Connection Diagram

2. Operating Environment

The Ethernet driver explained in this application note is for the environment below.

Table 2.1 Operating Environment

Item	Description
Microcomputer	RZ/T1 R7S910017
Operating frequency	CPU clock (Cortex-R4F): 450 MHz
Operation voltage	Internal: 1.2 V, I/O: 3,3 V
Integrated development environment	IAR Embedded Workbench for ARM version 7.60.1
Emulator	IAR JATG emulator I-jet for ARM
Operating mode	SPI boot mode 16-bit-bus boot mode
Board	RZ/T1 Evaluation Board (RTK7910022C00000BR)

3. Reference Documents

The documents related to this application note are listed below.

- RZ/T1 Group User's Manual: Hardware (R01UH0483EJ)
- RZ/T1 Group Application Note: Initial Settings (R01AN2554EJ)

4. Hardware

4.1 Pins

Pins used in the sample program and their functions are listed here.

Table 4.1 Pins Used and Their Functions

Pin Symbol	Input/Output	Function
ETH0_TXC ETH1_TXC	Input	Inputs for the 10M or 100M transmission clock signals (2.5 MHz or 25 MHz).
ETH0_TXEN ETH1_TXEN	Output	Outputs for the transmission enable signal.
ETH0_TXER ETH1_TXER	Output	Outputs for the transmission error signal.
ETH0_TXD0 to 3 ETH1_TXD0 to 3	Output	Outputs for the transmission data signal.
ETH0_RXC ETH1_RXC	I/O	Receive clock I/O pins
ETH0_RXDV ETH1_RXDV	Input	Inputs for the receive data enable signal.
ETH0_RXER ETH1_RXER	Input	Inputs for the receive data error signal.
ETH0_RXD0 to 3 ETH1_RXD0 to 3	Input	Inputs for the receive data signal.
ETH0_CRS ETH1_CRS	Input	Inputs for the career sense signal.
ETH0_COL ETH1_COL	Input	Inputs for the collision detection signal.
ETH_MDC	Output	Output for the management interface clock.
ETH_MDIO	I/O	Management data signal I/O pin
PHYLINK0 PHYLINK1	Input	Inputs for the PHY Link signal.
ETHSWSECOUT	Output	Event output pin for Ether switch per second.
PHYRESETOUT#	Output	Outputs for the PHY RESET signal for Ether0 and Ether1.

4.2 Reference Circuit

The blocks involved in Ethernet communications are illustrated below.

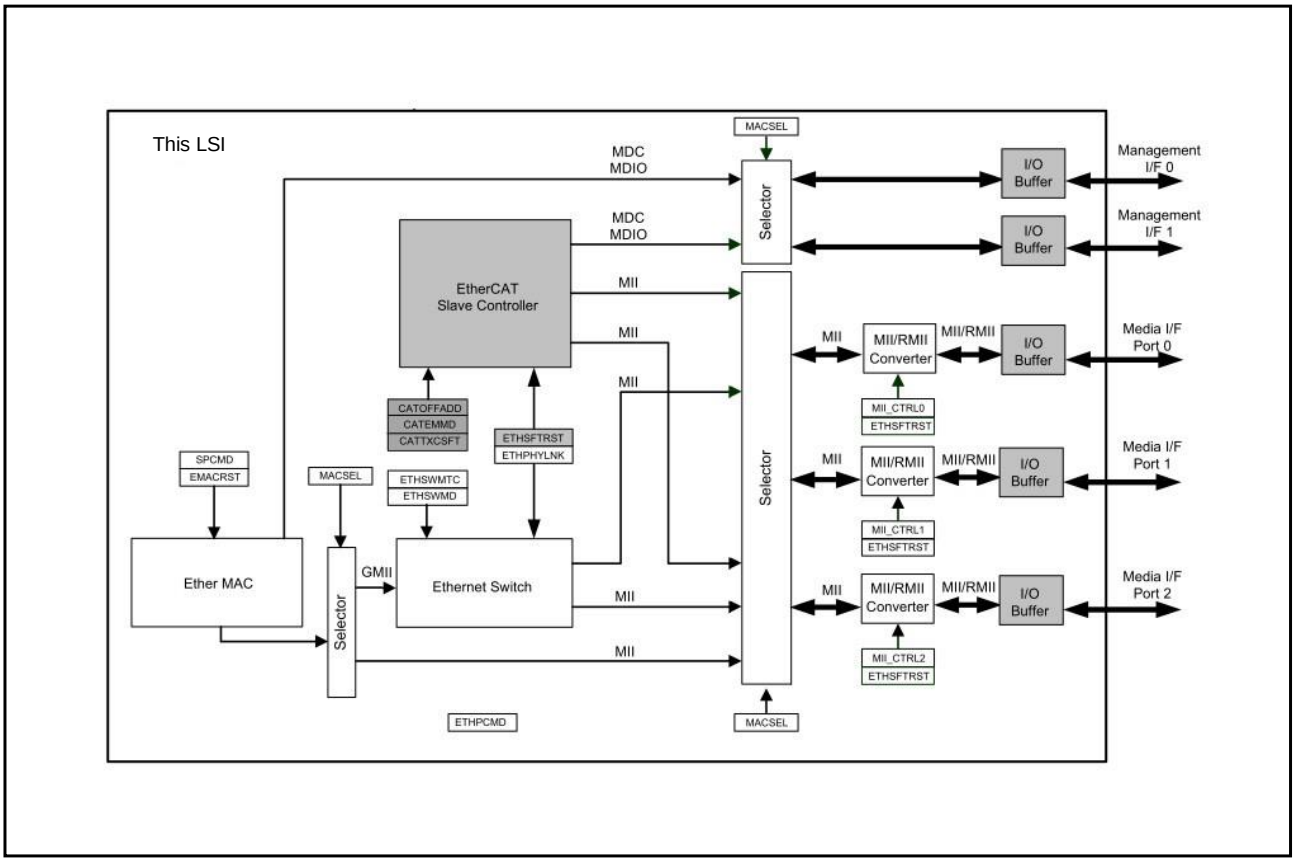


Figure 4.1 Diagram of Blocks Involved in Ethernet Communications

5. Software (Driver)

5.1 Outline of the Driver

This sample program handles Ethernet communications by issuing calls for driver functions to utilize the Ethernet MAC (ETHERC), Ethernet PHY (ETHPHY), and Ethernet switch (ETHSW) blocks.

5.2 Files

The drivers used in this sample program are listed below.

Table 5.1 Files Used in Sample Program

Filename	Outline
.. *1/inc/eth_hwfnc.h	The header file for defining hardware functions.
.. *1/inc/r_eth.h	The disclosed header file for the Ethernet-related drivers.
.. *1/inc/r_eth_mac.h	The header file for defining the ETHERC driver.
.. *1/inc/r_eth_phy.h	The header file for defining the ETHPHY driver.
.. *1/inc/r_eth_sw.h	The header file for defining the ETHSW driver.
.. *1/src/drv/eth/eth_hwfnc.c	The file used to mount the API for the hardware function driver
.. *1/src/drv/eth/r_eth_mac.c	The file used to mount the API for the ETHERC driver
.. *1/src/drv/eth/r_eth_phy.c	The file used to mount the API for the ETHPHY driver
.. *1/src/drv/eth/r_eth_sw.c	The file used to mount the API for the ETHSW driver

Note: ".. *1" represents the folder name of the sample program.

5.3 Major Data Types

The major data types used in this sample program are listed below. These integers are defined in the standard library.

Table 5.2 Major Data Type

Symbol	Description
int8_t	8-bit signed integer
int16_t	16-bit signed integer
int32_t	32-bit signed integer
int64_t	64-bit signed integer
uint8_t	8-bit unsigned integer
uint16_t	16-bit unsigned integer
uint32_t	32-bit unsigned integer
uint64_t	64-bit unsigned integer

5.4 Constants

The major constants are listed below.

Table 5.3 Constants

Constant Name	Value	Description	Definition File
USE_ETHSW	1	Ethernet switch, an optional function	r_eth.h
USE_ETHSW_MGTAG	1	Management tag function, an optional function of Ethernet switch	r_eth.h
MULTICAST_MODE_ENA	1	Receives multicast frames, an optional function	r_eth.h
PROMISCUOUS_MODE_ENA	1	Receives all frames, an optional function	r_eth.h
MAC0_TYPE_MASK	0x01	Masks the MAC address type	r_eth.h
MAC0_TYPE_UCAST	0x00	MAC address of the unicast frame	r_eth.h
MAC0_TYPE_MCAST	0x01	MAC address of the multicast frame	r_eth.h
ETH_HEADR_SIZE	14	Ethernet frame header size	r_eth.h
ETH_VLAN_HEADR_SIZE	18	Header size of the Ethernet frame with a VLAN tag	r_eth.h
ETH_EVT_PHYINTn	0x00000001 0x00000002 0x00000004	Ether PHYn event (n=0 to 2)	r_eth.h
ETH_EVT_RXDMACMP	0x00000100	Ether MACDMA reception completed	r_eth.h
ETH_EVT_RXDMAERR	0x00000200	Ether MACDMA reception error	r_eth.h
ETH_EVT_RXFIFOOVF	0x00000400	Ether RX-FIFO overflow	r_eth.h
ETH_EVT_TXDMACMP	0x00010000	Ether MACDMA transmission completed	r_eth.h
ETH_EVT_TXDMAERR	0x00020000	Ether MACDMA transmission error	r_eth.h
ETH_EVT_TXCMP	0x00100000	Ether transmission completed	r_eth.h
ETH_EVT_TXFIFOUDF	0x00200000	Ether TX-FIFO underflow	r_eth.h
ETH_EVT_TXFIFOERR	0x00400000	Ether TX-FOFO error	r_eth.h
ETH_EVT_PHY	ETH_EVT_PHYINT0 | ETH_EVT_PHYINT1 | ETH_EVT_PHYINT2	Ether PHY event	r_eth.h
ETH_EVT_TX	ETH_EVT_TXDMACMP | ETH_EVT_TXCMP | ETH_EVT_TXDMAERR | ETH_EVT_TXFIFOUDF | ETH_EVT_TXFIFOERR	Events related to Ether transmission	r_eth.h
ETH_TYPE_IPv4	0x0800	Ethernet Type: IPv4	r_eth.h
ETH_TYPE_ARP	0x0806	Ethernet Type: ARP	r_eth.h
ETH_PORT0	1	Specifies Ethernet port 0 as the port used for transmission and reception.	r_eth_mac.h
ETH_PORT1	2	Specifies Ethernet port 1 as the port used for transmission and reception.	r_eth_mac.h
ETH_PORTALL	(ETH_PORT0 | ETH_PORT1)	Specifies all Ethernet ports as the ports used for transmission and reception.	r_eth_mac.h
MACINFO_TBLEND	0xFF	Specifies the end point of the eth_macinfo_t structure.	r_eth_mac.h
MACINFO_UCAST_MAX	2	Specifies the maximum number of unicast MAC addresses that can be registered in the eth_macinfo_t structure.	r_eth_mac.h
htonl	ntohl	Converts the host byte order to 32-bit network byte order.	r_eth_mac.h

htons	ntohs	Converts the host byte order to 16-bit network byte order.	r_eth_mac.h
PHY_ADR0	1	Address of PHY0 for use in accessing PHY registers	r_eth_phy.h
PHY_ADR1	2	Address of PHY1 for use in accessing PHY registers	r_eth_phy.h
PHY_MAX_NUM	2	Specifies the number of PHY interfaces	r_eth_phy.h
PHY_LINK_MASK	0x80004000	Masks the link state of the PHY unit.	r_eth_phy.h
PHY_LINK_UP	0x00004000	The link with the PHY unit is up.	r_eth_phy.h
PHY_LINK_DOWN	0x00000000	The link with the PHY unit is down.	r_eth_phy.h
PHY_AUTONEGO_MASK	0x80002000	Masks the auto negotiation state of the PHY unit.	r_eth_phy.h
PHY_AUTONEGO_EN	0x00002000	Auto negotiation of the PHY unit is enabled.	r_eth_phy.h
PHY_AUTONEGO_DS	0x00000000	Auto negotiation of the PHY unit is disabled.	r_eth_phy.h
PHY_SPEED_MASK	0x80000030	Masks the transfer rate of the PHY unit.	r_eth_phy.h
PHY_SPEED_10M	0x00000000	Specifies 10BASE-T.	r_eth_phy.h
PHY_SPEED_100M	0x00000010	Specifies 100BASE-T.	r_eth_phy.h
PHY_SPEED_1G	0x00000020	Specifies 1000BASE-T.	r_eth_phy.h
PHY_DUPLEX_MASK	0x80000001	Masks the duplex type of the PHY unit.	r_eth_phy.h
PHY_DUPLEX_HALF	0x00000000	Uses the PHY unit in half-duplex mode.	r_eth_phy.h
PHY_DUPLEX_FULL	0x00000001	Uses the PHY unit in full-duplex mode	r_eth_phy.h

Table 5.3 Constants

LAN_MODE_MSK	(PHY_SPEED_MASK | PHY_DUPLEX_MASK)	Masks the configurations of the PHY unit.	r_eth_phy.h
LAN_10T_HD	(PHY_SPEED_10M | PHY_DUPLEX_HALF)	Specifies 10BASE-T with half-duplex.	r_eth_phy.h
LAN_10T_FD	(PHY_SPEED_10M | PHY_DUPLEX_FULL)	Specifies 10BASE-T with full-duplex.	r_eth_phy.h
LAN_100TX_HD	(PHY_SPEED_100M | PHY_DUPLEX_HALF)	Specifies 100BASE-TX with half-duplex.	r_eth_phy.h
LAN_100TX_FD	(PHY_SPEED_100M | PHY_DUPLEX_FULL)	Specifies 100BASE-TX with full-duplex.	r_eth_phy.h
LAN_1000T_HD	(PHY_SPEED_1G | PHY_DUPLEX_HALF)	Specifies 1000BASE-T with half-duplex.	r_eth_phy.h
LAN_1000T_FD	(PHY_SPEED_1G | PHY_DUPLEX_FULL)	Specifies 1000BASE-T with full-duplex.	r_eth_phy.h
PHY_REG_****	---	Used for the PHY standard register address	r_eth_phy.h
PHY_CONTROL_****	---	Assigns bits to the PHY mode control register.	r_eth_phy.h
PHY_STATUS_****	---	Assigns bits to the PHY mode status register.	r_eth_phy.h
PHY_ANADV_****	---	Assigns bits to the PHY auto-negotiation advertisement register.	r_eth_phy.h
PHY_ANLINK_****	---	Assigns bits to the PHY auto-negotiation link partner ability register.	r_eth_phy.h
ETHSW_LUT_D_PORT0	0	Specifies port 0 as the dynamic record in the lookup table.	r_eth_sw.h
ETHSW_LUT_D_PORT1	1	Specifies port 1 as the dynamic record in the lookup table.	r_eth_sw.h
ETHSW_LUT_D_PORT2	2	Specifies port 2 as the dynamic record in the lookup table.	r_eth_sw.h
ETHSW_LUT_S_PORT0	1	Specifies port 0 as the static record in the lookup table.	r_eth_sw.h
ETHSW_LUT_S_PORT1	2	Specifies port 1 as the static record in the lookup table.	r_eth_sw.h
ETHSW_LUT_S_PORT2	4	Specifies port 2 as the static record in the lookup table.	r_eth_sw.h
LRN_CHK_CNT	10	The maximum number of MAC addresses that can be sequentially read by the learning interface.	r_eth_sw.h

5.5 Configurations

Modify the configuration settings listed below as required.

Table 5.4 List of Configurations

Constant Name	Description	Definition File
USE_ETHSW	Selects use of the Ethernet switch, an optional function.	r_eth.h
USE_ETHSW_MGTAG	Selects use of the management tag function, an optional function of the Ethernet switch.	r_eth.h
MULTICAST_MODE_ENA	Selects the reception of multicast frames, an optional function.	r_eth.h
PROMISCUOUS_MODE_ENA	Selects the reception of all frames, an optional function.	r_eth.h
USE_NET_PORT2	Selects the Ethernet port to be used, an optional function.	r_eth.h
PHY_ADR0	Base address for use in access to the PHY registers for PHY0.	r_eth_phy.h
PHY_ADR1	Base address for use in access to the PHY registers for PHY1.	r_eth_phy.h

5.5.1 Details on Configurations

(1) USE_ETHSW

This constant is used to enable or disable the Ethernet switch function.

- Enabled (= 1):

This is the default setting. A two-port PHY interface is enabled so that network topologies such as the line and ring can be configured without involving an external switching hub. EtherCAT1 (J1) and EherCAT2 (J2) are respectively available as Ethernet ports 0 and 1 with this evaluation board.

- Disabled (= 0):

Only EtherCAT2 (J2) is available as a general-purpose Ethernet port with this evaluation board.

(2) USE_ETHSW_MGTAG

This constant is used to enable or disable the management tag function of Ethernet switch. This function is usable only when Ethernet switch is enabled (USE_ETHSW = 1).

- Enabled (=1):

This is the default setting. Management tags are added to the frames. With the management tags, acquisition of the information of the general-purpose Ethernet port through which received frames have passed and specification of the destination Ethernet ports of frames for transmission are possible.

- Disabled (=0):

Management tags are not added to the frames.

(3) MULTICAST_MODE_ENA

This constant is used to specify the receiving condition of multicast frames.

- Receives all multicast frames (=0).
- Receives the specified multicast frames only (=1): This is the default setting.

(4) PROMISCUOUS_MODE_ENA

This constant is used to specify the receiving condition of all Ethernet frames.

- Receives all Ethernet frames (=0).
- Receives only those frames which have passed address filtering (=1): This is the default setting.

(5) USE_NET_PORT2

This constant is used to switch Ethernet ports to be used.

- General-purpose Ethernet ports 0 and 1 used (= 0): Default
- General-purpose Ethernet port 2 used (= 1)

General-purpose Ethernet ports 0 and 1 are disabled, and the evaluation board EtherMAC(J7) is enabled as a general-purpose Ethernet port.

The PHY address is equal to Ethernet port 0.

(6) PHY_ADR

This constant is used to specify the addresses of the ETHPHY units. Set the same addresses as are configured in hardware for the individual PHY units, which are general-purpose Ethernet ports 0 (address 1) and 1 (address 2) on this evaluation board.

- PHY_ADR0:

This is the base address for access to registers of general-purpose Ethernet ports 0 and 2.

- PHY_ADR1:

This is the base address for access to registers of general-purpose Ethernet port 1.

5.6 Structures and Unions

A list of structures and unions is given below. Details are described in the subsequent tables.

Table 5.5 List of Structures and Unions

Type Definition	Description	Definition File
eth_frm_t structure	Structure for Ethernet frames	r_eth.h
eth_frmvlan_t structure	Structure for Ethernet frames with VLAN tags	r_eth.h
ip4_pkt_t structure	IPv4 packet structure	r_eth.h
eth_frminfo_t structure	Structure for the information from Ethernet frames	r_eth_mac.h
eth_macinfo_t structure	Structure for the information on MAC address	r_eth_mac.h
eth_txdesc_t structure	Structure for the descriptor which handles transmission	r_eth_mac.h

5.6.1 Details on the Structures and Unions

Table 5.6 eth_frm_t Structure

Member	Description
uint8_t dst[6]	Destination MAC address
uint8_t src[6]	Source MAC address
uint16_t type	Ethernet type
uint8_t data[2]	The initial part of the payload data

Table 5.7 eth_frmvlan_t Structure

Member	Description
uint8_t dst[6]	Destination MAC address
uint8_t src[6]	Source MAC address
uint32_t vlan	VLAN tag
uint16_t type	Ethernet type
uint8_t data[2]	The initial part of the payload data

Table 5.8 ip4_pkt_t Structure

Member	Description
uint8_t ver_ihl	Version & internet header length
uint8_t tos	Type of service
uint16_t tl	Total length
uint16_t ident	Identification
uint16_t vcf_fo	Flags & fragment offset
uint8_t ttl	Time tolLive
uint8_t prtcl	Protocol type
uint16_t hc	Header checksum
uint32_t sa	Source IP address
uint32_t da	Destination IP address
uint8_t data[2]	The initial part of the payload

Table 5.9 eth_frminfo_t Structure

Member	Description	
info	uint8_t BYTE	Frame information
	uint8_t vlan :1	Flag indicating whether the VLAN tag is valid or not. 0: VLAN tag is not valid 1: VLAN tag is valid
	uint8_t tcpchk:1	Flag indicating the results of calculation of checksums for the IPv4 and TCP/UDP headers by the software 0: The results of calculation of checksums are determined (checksums for the IPv4 and TCP/UDP headers are correct). 1: The results of calculation of checksums are not determined (checksum by the software is required). Note: This flag is enabled for receiving frames only.
	uint8_t :5	Reserved
uint8_t port		Transmission and reception ports For received frames, this is set to the value for the general-purpose Ethernet port number through which the frame has passed. For frames for transmission, set the value for the general-purpose Ethernet port number to which the fame is to be sent. Note: This parameter is only valid when the management tag function of the ETHSW is enabled.
uint32_t frm_len		Frame length
uint8_t *frm		A pointer to the buffer where the transmission and reception data are stored.

Table 5.10 eth_macinfo_t Structure

Member	Description
uint8_t mac[6]	MAC address

Table 5.11 eth_txdesc_t Structure

Member	Description
uint32_t addr	The address at which transmission starts from.
uint32_t len	The amount of transmission in bytes

5.7 Error Codes

This driver returns negative integers to indicate errors and zero or a positive integer to indicate normal execution.

5.8 Functions

A list of functions is given below.

Table 5.12 List of Functions

Function Name	Description	Scope	Definition File
R_ETH_Init	Initializes ETHERC.	Global	r_eth_mac.c
R_ETH_Rcv	Transmits Ethernet frames.	Global	r_eth_mac.c
R_ETH_Snd	Receives Ethernet frames.	Global	r_eth_mac.c
R_ETH_UpdateMode	Configures the operating mode of ETHERC.	Global	r_eth_mac.c
R_ETH_SetPhyreg	Makes configurations for the ETHPHY registers.	Global	r_eth_mac.c
R_ETH_GetPhyreg	Gets the information of the ETHPHY registers.	Global	r_eth_mac.c
R_ETH_memcpy	Copies memory for use in the Ethernet driver.	Global	r_eth_mac.c
R_ETH_GetEvent	Obtains events generated by Ethernet modules.	Global	r_eth_mac.c
R_ETH_ClrEvent	Clears events generated by Ethernet modules.	Global	r_eth_mac.c
ntohl	Converts to 32-bit byte order.	Global	r_eth_mac.c
ntohs	Converts to 16-bit byte order.	Global	r_eth_mac.c
dmac_memcpy	Handles copying by DMA within memory.	Local	r_eth_mac.c
eth_wait	Wait for a specified period of time.	Local	r_eth_mac.c
eth_int_init	Initializes Ethernet-related interrupts.	Local	r_eth_mac.c
ETHDMAIR_isr	Ether MACDMA reception completion interrupt handler	Local	r_eth_mac.c
ETHDRIE_isr	Ether MACDMA reception error interrupt handler	Local	r_eth_mac.c
ETHRFIV_isr	Ether RX-FIFO overflow interrupt handler	Local	r_eth_mac.c
ETHDMAIT_isr	Ether MACDMA transmission completion interrupt handler	Local	r_eth_mac.c
ETHDTIE_isr	Ether MACDMA transmission error interrupt handler	Local	r_eth_mac.c
ETHIT_isr	Ether transmission completion interrupt handler	Local	r_eth_mac.c
ETHTFIU_isr	Ether TX-FIFO underflow interrupt handler	Local	r_eth_mac.c
ETHTFIE_isr	Ether TX-FIFO error interrupt handler	Local	r_eth_mac.c
R_PHY_Init	Initializes ETHPHY.	Global	r_eth_phy.c
R_PHY_Reset	Resets ETHPHY.	Global	r_eth_phy.c
R_PHY_SetMode	Configures the operating mode of ETHPHY.	Global	r_eth_phy.c
R_PHY_GetMode	Gets the operating mode of ETHPHY.	Global	r_eth_phy.c
R_PHY_Link	ETHPHY link-up	Global	r_eth_phy.c
R_ETHSW_Init	Initialization of ETHSW.	Global	r_eth_sw.c
R_ETHSW_AddMaclut	Adds dynamic records to the address table of ETHSW.	Global	r_eth_sw.c
R_ETHSW_AddMaclutStatic	Adds static records to the address table of ETHSW.	Global	r_eth_sw.c
R_ETHSW_MacLearning	Learns the MAC address of ETHSW.	Global	r_eth_sw.c
ethsw_update_maclut	Updates the address table of ETHSW.	Local	r_eth_sw.c
ethsw_getCRC8	Gets a hash key by using a CRC-8 polynomial.	Local	r_eth_sw.c
ethsw_cap_timer	Captures the timer value for ETHSW.	Local	r_eth_sw

5.8.1 Details on the Functions Used with the Ethernet MAC

(1) R_ETH_Init

R_ETH_Init		
Synopsis	Initializes the Ethernet MAC module.	
Header	r_eth.h r_eth_sw.h eth_hwfnc.h	
Declaration	int32_t R_ETH_Init(eth_macinfo_t *macinfo)	
Argument	eth_macinfo_t *macinfo	: A pointer to the registered MAC address table.
Returned value	0	: Normal termination
	-1	: Parameter error
Execution condition	This function is only executed once prior to all other API functions of the drivers for the Ethernet system.	
Description	This function • checks whether the registered MAC address table is valid or not; • configures the Ethernet-related clocks and releases the Ethernet-related modules from the module-stopped states; • initializes the ETHERC; • releases the Ethernet-related modules from reset states; • sets up the hardware functions; • initializes the ETHSW and ETHPHY; • sets the MAC address; • secures the buffers for the transmission driver; • initializes Ethernet-related interrupts; and • permits the reception of Ethernet frames.	
Error condition	The number of unicast MAC addresses registered in the MAC address table exceeded the number specified in the constant MACINFO_UCAST_MAX.	
Supplementary note	None	

Usage example	`eth_macinfo_t macAddr_t[] = {`
	`{ 0x12, 0x34, 0x56, 0x78, 0x9A, 0xBC }, /* Unicast MAC addresses */`
	`{ 0x01, 0x80, 0xC2, 0x00, 0x00, 0x0E }, /* Multicast MAC addresses */`
	`{ MACINFO_TBLEND,MACINFO_TBLEND,MACINFO_TBLEND,`
	`MACINFO_TBLEND ,MACINFO_TBLEND ,MACINFO_TBLEND }`
	`};`
	
	`int32_t errcd;`
	
	`errcd = R_ETH_Init(macAddr_t);`	

(2) R_ETH_Rcv

R_ETH_Rcv		
Synopsis	Receives the Ethernet frames.	
Header	r_eth.h r_eth_sw.h eth_hwfnc.h	
Declaration	int32_t R_ETH_Rcv(eth_frminfo_t *frminfo)	
Argument	eth_frminfo_t *frminfo	: A pointer to the information of the reception frame.
Returned value	>0	: Size of the reception data
	0	: No reception data
	-1	: An invalid frame was received.
Execution condition	After the initialization function R_ETH_Init() is executed.	
Description	This function • checks the receive buffer information register so see if valid data have not been read; • checks the header of the received frames; • copies the received frame to the buffer pointed by the parameter "frminfo"; • obtains data in the received frame; and • releases the buffer which held reception frames.	
Error conditions	• An invalid frame being received. • A checksum error in the headers at the IP and TCP/UDP levels. • Copy from a reception frame failed. • Release of the buffer to store the reception frame failed.	
Supplementary note	Whether the checksums for the IPv4 and TCP/UDP headers by the hardware (the Ethernet accelerator) have been determined or not is included in the information of the reception frame, i.e. tcpchk. If the results have not been determined, checksums by the software are required.	

Usage example	`uint8_t rcv_buf[1514];`
	`eth_frminfo_t rxfrm = {0};`
	`int32_t errcd;`
	
	`rxfrm.frm = rcv_buf;`
	
	`errcd = R_ETH_Rcv(&rxfrm);`
	`if(0 < errcd){`
	`  /* There is a reception data */`
	`} else if( 0 > errcd){`
	`  /* An invalid frame was received */`
	`}`	

(3) R_ETH_Snd

R_ETH_Snd		
Synopsis	Transmits Ethernet frames.	
Header	r_eth.h r_eth_sw.h eth_hwfnc.h	
Declaration	int32_t R_ETH_Snd(eth_frminfo_t *frminfo)	
Argument	eth_frminfo_t *frminfo	: A pointer to the transmission Ethernet frame
Returned values	0	: Successful transmission
	-1	: Transmission failure
Execution condition	After the initialization function R_ETH_Init() is executed.	
Description	This function • copies the transmission frame from the buffer pointed by the parameter "frminfo" to the transmission buffer; • configures the information of the transmission frame controller; • configures the transmission frame descriptor; • starts up the DMA for transmission; • waits for completion of a frame transmission; and • acquires error statuses.	
Error conditions	• Errors in inter-buffer copy • Starting the DMA for transmission failed. • Transmission errors	
Supplementary note	Calculation for checksums for the IPv4 and TCP/UDP headers of packets other than fragment packets by the software is not required because hardware in the form of the Ethernet accelerator detects checksum errors.	
Usage example		

```
uint8_t         snd_buf[1514];
eth_frminfo_t txfrm = {0};
int32_t         errcd;
eth_frm_t       *ethfrm_tx;

txfrm.frm = snd_buf;

ethfrm_tx = (eth_frm_t *) txfrm.frm;

R_ETH_memcpy(ethfrm_tx->src,/* Source MAC Addr */,6);
R_ETH_memcpy(ethfrm_tx->dst,/* Destination MAC Addr */,6);
ethfrm_tx->type = htons(/* Ethernet Type */);
R_ETH_memcpy(ethfrm_tx->data,/* Payload Data */,/* Payload Data
Length*/);
#if(USE_ETHSW & USE_ETHSW_MGTAG)
/* Select the transfer port */
txfrm .port = ETH_PORT***;
#endif
txfrm .frm_len = /* Ethernet Frame Length */
errcd = R_ETH_Snd(&txfrm);
if(0 > errcd){
   /* Transmission error */
} else {
   /* Transmission completed */
}
```

(4) R_ETH_UpdateMode

R_ETH_UpdateMode		
Synopsis	Configures the operating mode of the Ethernet MAC.	
Header	r_eth.h r_eth_sw.h eth_hwfnc.h	
Declaration	int32_t R_ETH_UpdateMode(uint8_t port)	
Argument	uint8_t port	: The port to which the operating mode is to be applied. : ETH_PORT0 : ETH_PORT1
Returned values	0	: Successful transmission
	-1	: Transmission failure
Execution condition	After the initialization function R_ETH_Init() was executed.	
Description	This function configures the operating mode of the Ethernet MAC so that it supports the configuration of the Ethernet PHY unit specified in the argument "port", if the link with the unit is up.	
Error condition	The value specified in the argument port is outside the range.	
Supplementary note	None	

Usage example	`int32_t errcd;`
	
	`errcd = R_ETH_UpdateMode(ETH_PORT0);`
	`if(0 > errcd){`
	`  /* Configuration error */`
	`} else {`
	`  /* Configuration completed */`
	`}`	

(5) R_ETH_SetPhyreg

R_ETH_SetPhyreg		
Synopsis	Makes configurations of the Ethernet PHY registers.	
Header	r_eth.h r_eth_sw.h eth_hwfnc.h	
Declaration	void R_ETH_SetPhyreg(uint8_t phyadr, uint8_t regadr, uint16_t val)	
Arguments	uint8_t phyadr	: PHY address
	uint8_t regadr	: Address of the PHY register to which the configuration is to be applied.
	uint16_t val	: Value to be set to the PHY register.
Returned value	None	
Execution condition	After the initialization function R_ETH_Init() is executed.	
Description	This function sets the value specified in the argument "val" in the register corresponding to "regadr" of the PHY module specified by "phyadr".	
Error condition	None	
Supplementary note	None	

Usage example	`R_ETH_SetPhyreg(PHY_ADR0,`
	`  PHY_REG_CONTROL,`
	`  PHY_CONTROL_SPEED_100M);`	

(6) R_ETH_GetPhyreg

R_ETH_GetPhyreg	
Synopsis	Obtains data in the Ethernet PHY register.
Header	r_eth.h r_eth_sw.h eth_hwfnc.h
Declaration	uint16_t R_ETH_GetPhyreg(uint8_t phyadr, uint8_t regadr)
Arguments	uint8_t phyadr : PHY address uint8_t regadr : Address of the PHY register where the data is obtained from.
Returned value	The result of referring to the PHY register.
Execution condition	After the initialization function R_ETH_Init() is executed.
Description	This function reads the register corresponding to "regadr" of the PHY module specified by "phyadr" and returns the result.
Error condition	None
Supplementary note	None

Usage example	`uint16_t data;`
	
	`data = R_ETH_SetPhyreg(PHY_ADR0,PHY_REG_CONTROL);`

(7) R_ETH_memcpy

R_ETH_memcpy	
Synopsis	Copies the memory for the drivers in the Ethernet system.
Header	r_eth.h r_eth_sw.h eth_hwfnc.h
Declaration	int32_t R_ETH_memcpy(void *dst, const void *src, uint32_t n)
Arguments	void *dst : Address of the source of the data for copying in memory const void *src : Address of the destination of the data in memory uint32_t n : Size of the copy range
Returned value	0 : Normal termination
Execution condition	None
Description	This function calls the DMA copy function.
Error condition	None
Supplementary note	None
Usage example	`R_ETH_memcpy(/* Destination Addr */ , /* Source Addr */ , /* Copy Length */);`

(8) R_ETH_GetEvent

R_ETH_GetEvent		
Synopsis	Obtains events generated by Ethernet modules.	
Header	r_eth.h r_eth_sw.h eth_hwfnc.h	
Declaration	uint32_t R_ETH_GetEvent(void)	
Arguments	None	
Returned value	Normal cases	bit22: 1=ETH_EVT_TXFIFOERR bit21: 1=ETH_EVT_TXFIFOUDF bit20: 1=ETH_EVT_TXCMP bit17: 1=ETH_EVT_TXDMAERR bit16: 1=ETH_EVT_TXDMACMP bit10: 1=ETH_EVT_RXFIFOOVF bit9: 1=ETH_EVT_RXDMAERR bit8: 1=ETH_EVT_RXDMACMP
Execution condition	None.	
Description	This function obtains events generated by Ethernet modules.	
Error condition	None	
Supplementary note	None	

Usage example	`uint32_t EventFlag;`
	
	`EventFlag = R_ETH_GetEvent();`	

(9) R_ETH_ClrEvent

R_ETH_ClrEvent		
Synopsis	Clears events generated by Ethernet modules.	
Header	r_eth.h r_eth_sw.h eth_hwfnc.h	
Declaration	void R_ETH_ClrEvent(uint32_t clrflg)	
Arguments	uint32_t clrflg	: Clearable events : ETH_EVT_RXDMACMP : ETH_EVT_RXDMAERR : ETH_EVT_RXFIFOOVF : ETH_EVT_TXDMACMP : ETH_EVT_TXDMAERR : ETH_EVT_TXCMP : ETH_EVT_TXFIFOUDF : ETH_EVT_TXFIFOERR
Returned value	None.	
Execution condition	None.	
Description	This function clears events generated by Ethernet modules.	
Error condition	None	
Supplementary note	None	
Usage example	R_ETH_ClrEvent(ETH_EVT_TXDMACMP | ETH_EVT_TXCMP);	

(10) ntohl

ntohl	
Synopsis	Converts the 32-bit byte order.
Header	r_eth.h r_eth_sw.h eth_hwfnc.h
Declaration	uint32_t ntohl(uint32_t val)
Argument	uint32_t val : Value to which the byte order is to be converted.
Returned value	Conversion result
Execution condition	None
Description	This function converts 32 bits in the network byte order to that of the host device.
Error condition	None
Supplementary note	None

Usage example	`uint32_t data;`
	
	`data = ntohl(0x11223344);`

(11) ntohs

ntohl	
Synopsis	Converts the 16-bit byte order.
Header	r_eth.h r_eth_sw.h eth_hwfnc.h
Declaration	uint16_t ntohl(uint16_t val)
Argument	uint16_t val : Value to which the byte order is to be converted.
Returned value	Conversion result
Execution condition	None
Description	This function converts 16 bits in the network byte order to that of the host device.
Error condition	None
Supplementary note	None

Usage example	`uint16_t data;`
	
	`data = ntohs(0x1122);`

(12) dmac_memcpy

dmac_memcpy		
Synopsis	Performs a DMA copy.	
Header	r_eth.h r_eth_sw.h eth_hwfnc.h	
Declaration	static void *dmac_memcpy(void *dst, const void *src, uint32_t n)	
Arguments	void *dst	: Address of the source of the data for copying in memory
	const void *src	: Address of the destination of the data in memory
	uint32_t n	: Size of the copy range
Returned value	Copy destination address	
Execution condition	None	
Description	This function configures the DMAC register and performs DMA transmissions.	
Error condition	None	
Supplementary note	None	
Usage example	None	

(13) eth_wait

eth_wait		
Synopsis	Wait for a specified period of time.	
Header	r_eth.h r_eth_sw.h eth_hwfnc.h	
Declaration	static void eth_wait(uint32_t value)	
Arguments	uint32_t value	: Value for the wait time in microseconds
Returned value	None	
Execution condition	None	
Description	This function calculates the approximate number of cycles required for the wait process from the value specified for wait time and creates a loop so that a wait process without depending on a timer is achieved.	
Error condition	None	
Supplementary note	None	
Usage example	None	

(14) eth_int_init

eth_int_init	
Synopsis	Initializes Ethernet-related interrupts.
Header	r_eth.h r_eth_sw.h eth_hwfnc.h
Declaration	static void eth_int_init(void)
Arguments	None
Returned value	None
Execution condition	None
Description	This function registers the following Ethernet-related interrupt handlers: Ether MACDMA reception completion interrupt handler Ether MACDMA reception error interrupt handler Ether RX-FIFO overflow interrupt handler Ether MACDMA transmission completion interrupt handler Ether MACDMA transmission error interrupt handler Ether transmission completion interrupt handler Ether TX-FIFO underflow interrupt handler Ether TX-FIFO error interrupt handler
Error condition	None
Supplementary note	None
Usage example	None

(15) ETHDMAIR_isr

ETHDMAIR_isr	
Synopsis	Ether MACDMA reception completion interrupt handler.
Header	r_eth.h r_eth_sw.h eth_hwfnc.h
Declaration	void ETHDMAIR_isr(void)
Arguments	None
Returned value	None
Execution condition	None
Description	This function sets the Ether MACDMA reception completion event of the Ether MACDMA reception completion interrupt handler.
Error condition	None
Supplementary note	None
Usage example	None

(16) ETHDRIE_isr

ETHDRIE_isr	
Synopsis	Ether MACDMA reception error interrupt handler
Header	r_eth.h r_eth_sw.h eth_hwfnc.h
Declaration	void ETHDRIE_isr(void)
Arguments	None
Returned value	None
Execution condition	None
Description	This function sets the Ether MACDMA reception error event of the Ether MACDMA reception error interrupt handler.
Error condition	None
Supplementary note	None
Usage example	None

(17) ETHRFIV_isr

ETHRFIV_isr	
Synopsis	Ether RX-FIFO overflow interrupt handler
Header	r_eth.h r_eth_sw.h eth_hwfnc.h
Declaration	void ETHRFIV_isr(void)
Arguments	None
Returned value	None
Execution condition	None
Description	This function sets the Ether RX-FIFO overflow event of the Ether RX-FIFO overflow interrupt handler.
Error condition	None
Supplementary note	None
Usage example	None

(18) ETHDMAIT_isr

ETHDMAIT_isr	
Synopsis	Ether MACDMA transmission completion interrupt handler
Header	r_eth.h r_eth_sw.h eth_hwfnc.h
Declaration	void ETHDMAIT_isr(void)
Arguments	None
Returned value	None
Execution condition	None
Description	This function sets the Ether MACDMA transmission completion event of the Ether MACDMA transmission completion interrupt handler.
Error condition	None
Supplementary note	None
Usage example	None

(19) ETHDTIE_isr

ETHDTIE_isr	
Synopsis	Ether MACDMA transmission error interrupt handler
Header	r_eth.h r_eth_sw.h eth_hwfnc.h
Declaration	void ETHDTIE_isr(void)
Arguments	None
Returned value	None
Execution condition	None
Description	This function sets the Ether MACDMA transmission error event of the Ether MACDMA transmission error interrupt handler.
Error condition	None
Supplementary note	None
Usage example	None

(20) ETHIT_isr

ETHIT_isr	
Synopsis	Ether transmission completion interrupt handler
Header	r_eth.h r_eth_sw.h eth_hwfnc.h
Declaration	void ETHIT_isr(void)
Arguments	None
Returned value	None
Execution condition	None
Description	This function sets the Ether transmission completion interrupt event of the Ether transmission completion interrupt handler.
Error condition	None
Supplementary note	None
Usage example	None

(21) ETHTFIU_isr

ETHTFIU_isr	
Synopsis	Ether TX-FIFO underflow interrupt handler
Header	r_eth.h r_eth_sw.h eth_hwfnc.h
Declaration	void ETHTFIU_isr(void)
Arguments	None
Returned value	None
Execution condition	None
Description	This function sets the Ether TX-FIFO underflow event of the Ether TX-FIFO underflow interrupt handler.
Error condition	None
Supplementary note	None
Usage example	None

(22) ETHTFIE_isr

ETHTFIE_isr	
Synopsis	Ether TX-FIFO error interrupt handler
Header	r_eth.h r_eth_sw.h eth_hwfnc.h
Declaration	void ETHTFIE_isr(void)
Arguments	None
Returned value	None
Execution condition	None
Description	This function sets the Ether TX-FIFO error interrupt event of the Ether TX-FIFO error interrupt handler.
Error condition	None
Supplementary note	None
Usage example	None

5.8.2 Details on Ethernet PHY Functions

(1) R_PHY_Init

R_PHY_Init		
Synopsis	Initializes the Ethernet PHY.	
Header	r_eth.h	
Declaration	int32_t R_PHY_Init(uint8_t phyadr)	
Argument	uint8_t phyadr	: Address of the PHY unit to be initialized • PHY_ADR0 • PHY_ADR1
Returned values	0	: Normal termination
	-1	: Parameter error
Execution condition	This function is called while the initialization function R_ETH_Init() is executed.	
Description	This function • resets the ETHPHY; and • initializes the ETHPHY.	
Error condition	The value specified in the argument phyadr is outside the range.	
Supplementary note	None	

Usage example	`int32_t errcd;`
	
	`errcd = R_PHY_Init(PHY_ADR0);`	

(2) R_PHY_Reset

R_PHY_Reset		
Synopsis	Resets the Ethernet PHY module.	
Header	r_eth.h	
Declaration	int32_t R_PHY_Reset(uint8_t phyadr)	
Argument	uint8_t phyadr	: Address of the PHY unit to be reset • PHY_ADR0 • PHY_ADR1
Returned values	0	: Normal termination
	-1	: Parameter error
Execution condition	None	
Description	This function resets the specified PHY unit.	
Error condition	The value specified in the argument phyadr is outside the range.	
Supplementary note	None	

Usage example	`int32_t errcd;`
	
	`errcd = R_PHY_Reset(PHY_ADR0);`	

(3) R_PHY_SetMode

R_PHY_SetMode		
Synopsis	Configures the operating mode of the Ethernet PHY.	
Header	r_eth.h	
Declaration	int32_t R_PHY_SetMode(uint8_t phyadr, uint16_t mode , uint8_t nego)	
Arguments	uint8_t phyadr	: Address of the PHY unit to which the operating mode is to be applied. • PHY_ADR0 • PHY_ADR1
	uint16_t mode	: Either of the following operating modes • LAN_10T_HD • LAN_10T_FD • LAN_100TX_HD • LAN_100TX_FD
	uint8_t nego	: Auto-negotiation mode • 0 = Disables auto-negotiation • 1 = Enables auto-negotiation
Returned values	0	: Normal termination
	-1	: Parameter error
Execution condition	After the initialization function for the Ethernet MAC R_ETH_Init() is executed.	
Description	This function configures the operating mode considering the address of the PHY unit and the setting of the auto-negotiation mode.	
Error condition	The value specified in the argument phyadr is outside the range.	
Supplementary note	Access to the PHY registers are made by using the Ethernet MAC driver R_ETH_SetPhyreg().	

Usage example	`int32_t errcd;`
	
	`errcd = R_PHY_SetMode(PHY_ADR0, LAN_100TX_HD,0);`	

(4) R_PHY_GetMode

R_PHY_GetMode		
Synopsis	Obtains the operating mode of the Ethernet PHY.	
Header	r_eth.h	
Declaration	int32_t R_PHY_GetMode(uint8_t phyadr)	
Argument	uint8_t phyadr	: Address of the PHY unit from which the operating mode is obtained. • PHY_ADR0 • PHY_ADR1
Returned values	Normal cases	bit 31 : 0 = No error bit 14 : Link status • 0 = Link down • 1 = Link up bit 13 : Auto-negotiation mode status • 0 = Disabled • 1 = Enabled bit 2, 1 : Transfer rate • 01b = 10BASE-T • 10b = 100BASE-T bit 0 : Duplex mode • 0 = Half duplex mode • 1 = Full duplex mode
	-1	: Parameter error
Execution condition	After the initialization function for the Ethernet MAC R_ETH_Init() was executed.	
Description	This function obtains the information of the registers such as link state and the setting of auto-negotiation mode from the PHY address.	
Error condition	The value specified in the argument phyadr is outside the range.	
Supplementary note	Access to the PHY registers are made by using the Ethernet MAC driver R_ETH_SetPhyreg().	

Usage example	`int32_t status;`
	
	`status = R_PHY_GetMode(PHY_ADR0);`
	`if((status & PHY_LINK_MASK) == PHY_LINK_UP){`
	`  /* PHY Link-up */`
	`}else{`
	`  /* PHY Link-down */`
	`}`	

(5) R_PHY_Link

R_PHY_Link		
Synopsis	Checks the link-up state of the Ethernet PHY and updates the operating mode.	
Header	r_eth.h	
Declaration	int32_t R_PHY_Link(uint8_t phyadr)	
Argument	uint8_t phyadr	: Address of the PHY unit whose link-up state is checked. • PHY_ADR0 • PHY_ADR1
Returned values	Normal cases	bit 31 : 0 = No error bit 14 : Link status • 0 = Link down • 1 = Link up bit 13 : Auto-negotiation mode status • 0 = Disabled • 1 = Enabled bit 2, 1 : Transfer rate • 01b = 10BASE-T • 10b = 100BASE-T bit 0 : Duplex mode • 0 = Half duplex mode • 1 = Full duplex mode
	-1	: Parameter error
Execution condition	After the initialization function for the Ethernet MAC R_ETH_Init() was executed.	
Description	This function • obtains the operating mode of the Ethernet PHY unit of the specified address; • calls the R_ETH_UpdateMode() function of the driver for the Ethernet MAC to configure its operating mode if the obtained value varies from the previous result and the link with the PHY unit is up; and • returns the operating mode of the Ethernet PHY.	
Error condition	The value specified in the argument phyadr is outside the range.	
Supplementary note	Access to the PHY registers are made by using the Ethernet MAC driver R_ETH_SetPhyreg().	

Usage example	`int32_t status;`
	`uint8_t rcv_buf[1514];`
	`eth_frminfo_t rxfrm = {0};`
	`int32_t errcd;`
	
	`rxfrm.frm = rcv_buf;`
	
	`status = R_PHY_Link(PHY_ADR0);`
	`if((status & PHY_LINK_MASK) ==`
	`PHY_LINK_UP){ errcd = R_ETH_Rcv(&rxfrm);`
	`}`	

5.8.3 Details on the Functions Used with the Ethernet Switch

(1) R_ETHSW_Init

R_ETHSW_Init		
Synopsis	Initializes the Ethernet switch	
Header	r_eth.h r_eth_sw.h	
Declaration	int32_t R_ETHSW_Init(eth_macinfo_t *macinfo , uint8_t hub)	
Argument	eth_macinfo_t *macinfo	: A pointer to the registered MAC address table.
	uint8_t hub	: Switching hub • 0 = Disabled • 1 = Enabled
Returned values	0	: Normal termination
Execution condition	This function is called while the initialization function R_ETH_Init() is executed.	
Description	This function • initializes the functions of the Ethernet switch such as MAC, switches, and timers; and • makes configuration for the hub with the argument "hub".	
Error condition	None	
Supplementary note	None	

Usage example	`eth_macinfo_tmacAddr_t[] = {`
	`{ 0x12, 0x34, 0x56, 0x78, 0x9A, 0xBC },/* Unicast MAC addresses */`
	`{ 0x01, 0x80, 0xC2, 0x00, 0x00, 0x0E },/* Multicast MAC addresses */`
	`{ MACINFO_TBLEND,MACINFO_TBLEND,MACINFO_TBLEND,`
	`MACINFO_TBLEND ,MACINFO_TBLEND ,MACINFO_TBLEND }`
	`};`
	
	`int32_t errcd;`
	
	`errcd = R_ETHSW_Init(macAddr_t,1);`	

(2) R_ETHSW_AddMaclut

R_ETHSW_AddMaclut		
Synopsis	Adds dynamic MAC records to the address table of the Ethernet switch.	
Header	r_eth.h r_eth_sw.h	
Declaration	int32_t R_ETHSW_AddMaclut(uint8_t *mac , uint8_t port)	
Argument	uint8_t *mac	: A pointer to the MAC address
	uint8_t port	: Either of the following ETHSW dynamic port numbers • ETHSW_LUT_D_PORT0 = Port 0 • ETHSW_LUT_D_PORT1 = Port 1 • ETHSW_LUT_D_PORT2 = Port 2 (CPU port) Note: Specifying a combination of ports is NOT allowed.
Returned values	>=0	: The entry position in the address table (the hash key)
	-1	: Registration of the MAC record failed.
Execution condition	After the initialization function for the Ethernet MAC R_ETH_Init() was executed.	
Description	This function • creates a dynamic MAC record based on the arguments "mac" and "port"; and • obtains a hash key from the MAC address and adds it to the address table.	
Error condition	The address table has no more space to add a new MAC address.	
Supplementary note	None	

Usage example	`uint8_t mac [] = { 0x12, 0x34, 0x56, 0x78, 0x9A, 0xBC };`
	
	`int32_t hash;`
	
	`hash = R_ETHSW_AddMaclut(mac,ETHSW_LUT_D_PORT2);`	

(3) R_ETHSW_AddMaclutStatic

R_ETHSW_AddMaclutStatic		
Synopsis	Adds the static MAC records to the address table of the Ethernet switch.	
Header	r_eth.h r_eth_sw.h	
Declaration	int32_t R_ETHSW_AddMaclutStatic(uint8_t *mac , uint8_t port)	
Argument	uint8_t *mac	: A pointer to the MAC address
	uint8_t port	: Either of the following ETHSW static port number. • ETHSW_LUT_S_PORT0 = port 0 • ETHSW_LUT_S_PORT1 = port 1 • ETHSW_LUT_S_PORT2 = port 2 Note: Specifying a combination of ports is possible.
Returned values	>=0	: The entry position in the address table (the hash key)
	-1	: Registration of the MAC record failed.
Execution condition	After the initialization function for the Ethernet MAC R_ETH_Init() was executed.	
Description	This function • creates a static MAC record based on the arguments "mac" and "port"; and • obtains a hash key from the MAC address and adds it to the address table.	
Error condition	The address table has no more space to add a new MAC address.	
Supplementary note	None	

Usage example	`uint8_t mac [] = { 0x12, 0x34, 0x56, 0x78, 0x9A, 0xBC };`
	`int32_t hash;`
	`hash = R_ETHSW_AddMaclutStatic (mac,ETHSW_LUT_S_PORT2);`	

(4) R_ETHSW_MacLearning

R_ETHSW_MacLearning		
Synopsis	Learns the MAC address of the Ethernet switch.	
Header	r_eth.h r_eth_sw.h	
Declaration	int32_t R_ETHSW_MacLearning(void)	
Argument	None	
Returned values	0	: The MAC address has been learned.
	-1	: The MAC address has not been learned.
Execution condition	After the initialization function for the Ethernet MAC R_ETH_Init() was executed.	
Description	This function • selects use of the learning interface to learn the MAC addresses of frames that pass through an Ethernet port: the MAC address, port number, and hash key of the frames are acquired; and • creates the dynamic MAC record and adds them to the address table.	
Error condition	None	
Supplementary note	None	
Usage example	`R_ETHSW_MacLearning();`	

(5) ethsw_update_maclut

ethsw_update_maclut	
Synopsis	Updates the address table of the Ethernet switch.
Header	r_eth.h r_eth_sw.h
Declaration	static int32_t ethsw_update_maclut(RIN_ETHSW_LUT_ADR_Typedef *macinfo,uint8_t hash)
Argument	RIN_ETHSW_LUT_ADR_Typedef *macinfo : A pointer to the MAC record uint8_t hash : The position for including the information in the address table (hash key)
Returned values	>=0 : The position for including the information in the address table (hash key) -1 : Registration of the MAC record failed.
Execution condition	After the initialization function for the Ethernet MAC R_ETH_Init() was executed.
Description	This function • adds the MAC record to the entry position within eight entries from the position specified by "hash" in the address table; • overwrites the MAC record at the entry position within eight entries from the position specified by "hash" in the address table if there is an entry for the same MAC address in the table; and • writes the MAC record to an empty entry if there is one and, if there is no entry for the same MAC address, overwrites the entry position with the oldest timestamp.
Error condition	No dynamic MAC records can be added because all eight entries are filled with the static MAC records.
Supplementary note	None
Usage example	None

(6) ethsw_getCRC8

ethsw_getCRC8	
Synopsis	Obtains a hash key by using the CRC-8 polynomial of the Ethernet switch.
Header	r_eth.h r_eth_sw.h
Declaration	static crc ethsw_getCRC8(uint8_t *buff , uint16_t size)
Argument	uint8_t *buff : A pointer to the data for use in the calculation. uint16_t size : Size of the data used for the calculation.
Returned values	uint8_t : Hash key
Execution condition	None
Description	This function obtains the hash key by using a CRC-8 polynomial with the data pointed by "buff".
Error condition	None
Supplementary note	None
Usage example	None

(7) ethsw_cap_timer

ethsw_cap_timer	
Synopsis	Captures the timer in the Ethernet switch.
Header	r_eth.h r_eth_sw.h
Declaration	static void ethsw_cap_timer(void)
Argument	None
Returned value	None
Execution conditions	After the initialization function for the Ethernet MAC R_ETH_Init() was executed.
Description	This function • instructs capturing of the value of the timer in the Ethernet switch; and • updates the timer register value in response to the capture instruction.
Error condition	None
Supplement	None
Usage example	None

5.9 Setup Procedure for the Ethernet MAC

Setup procedure for the Ethernet MAC, from its initialization until reception of data being enabled, is described in the flowcharts.

5.9.1 Flow of Initialization

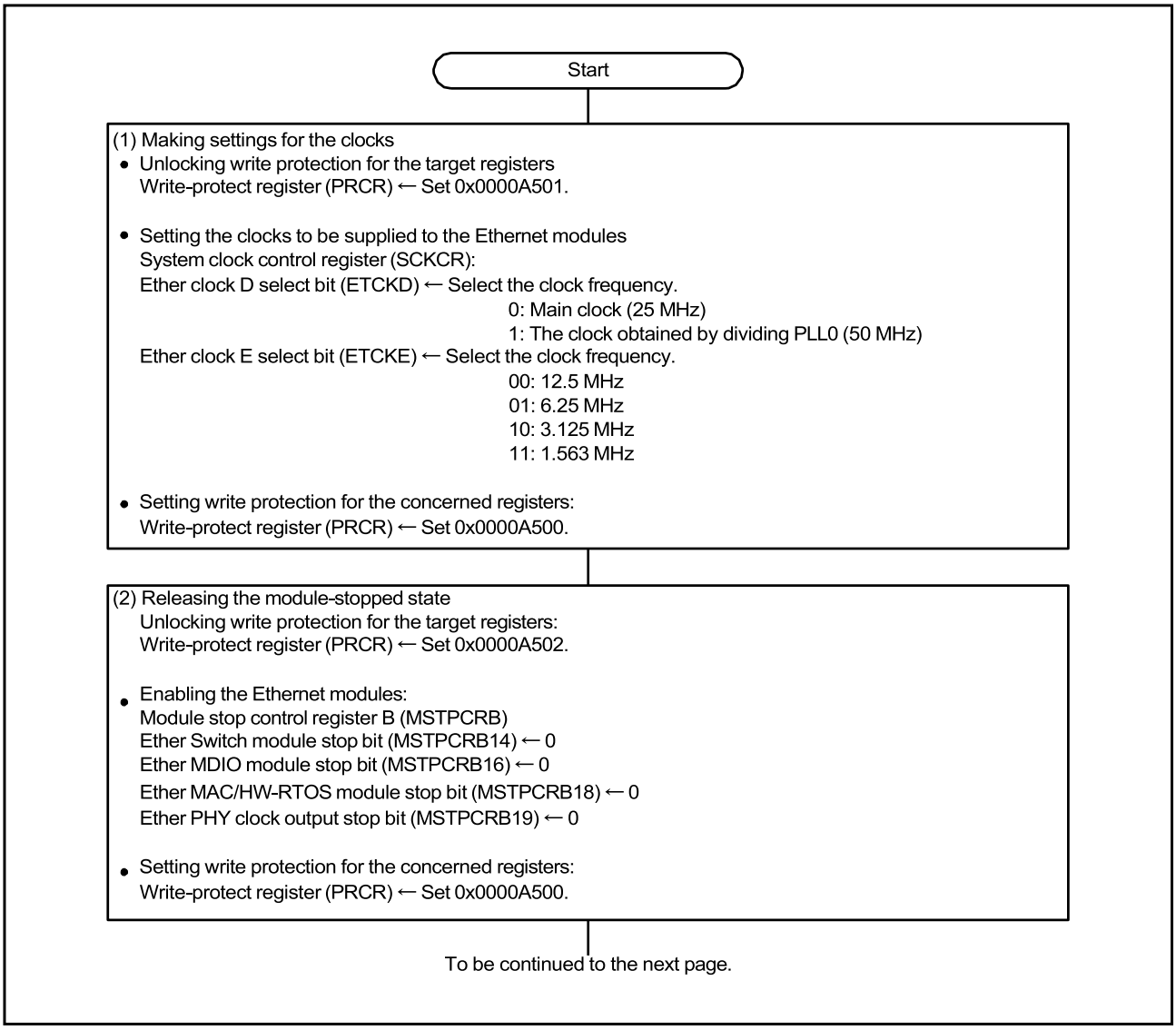


Figure 5.1 Flow of Initialization of the Ethernet MAC (1)

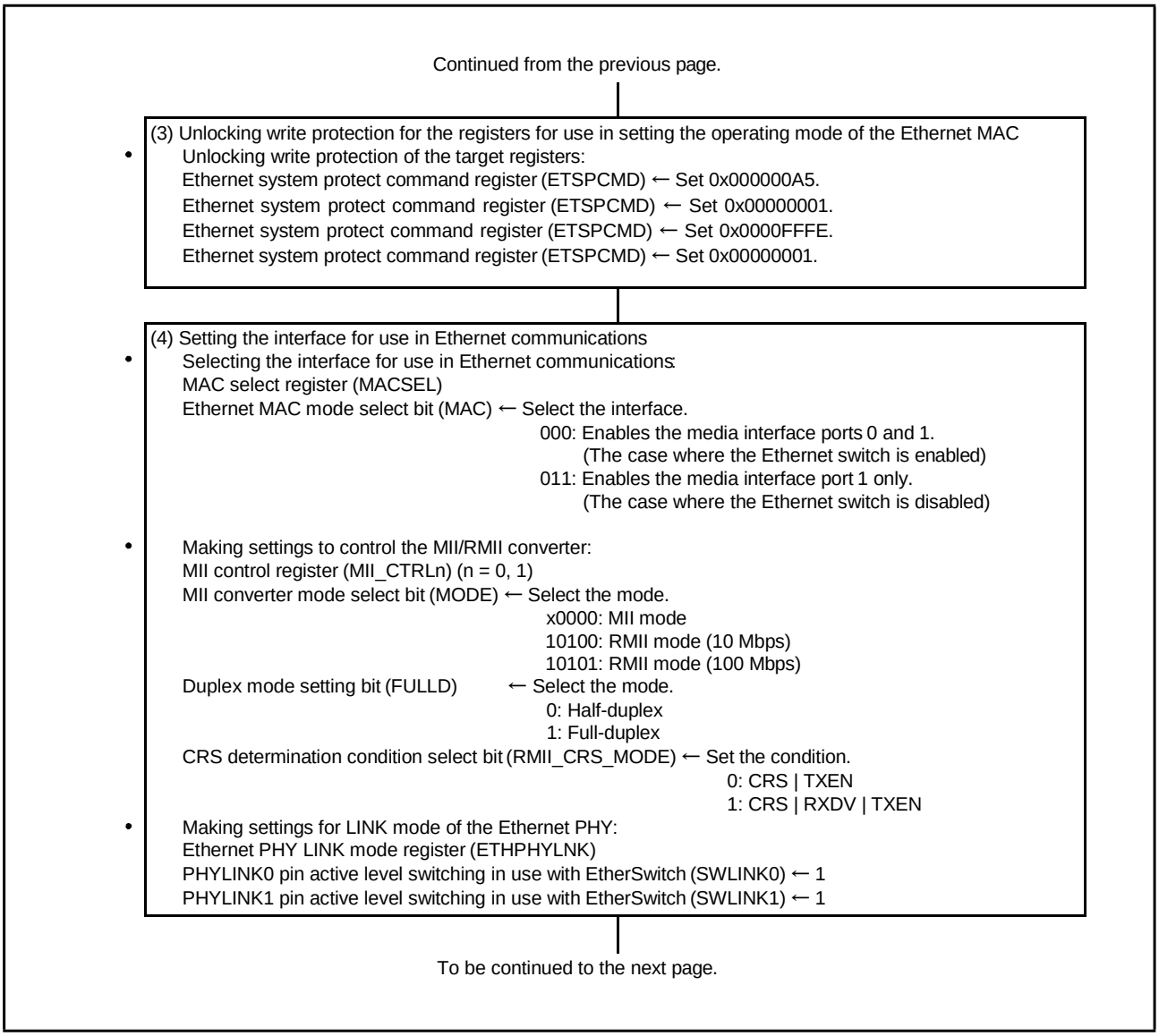


Figure 5.2 Flow of Initialization of the Ethernet MAC (2)

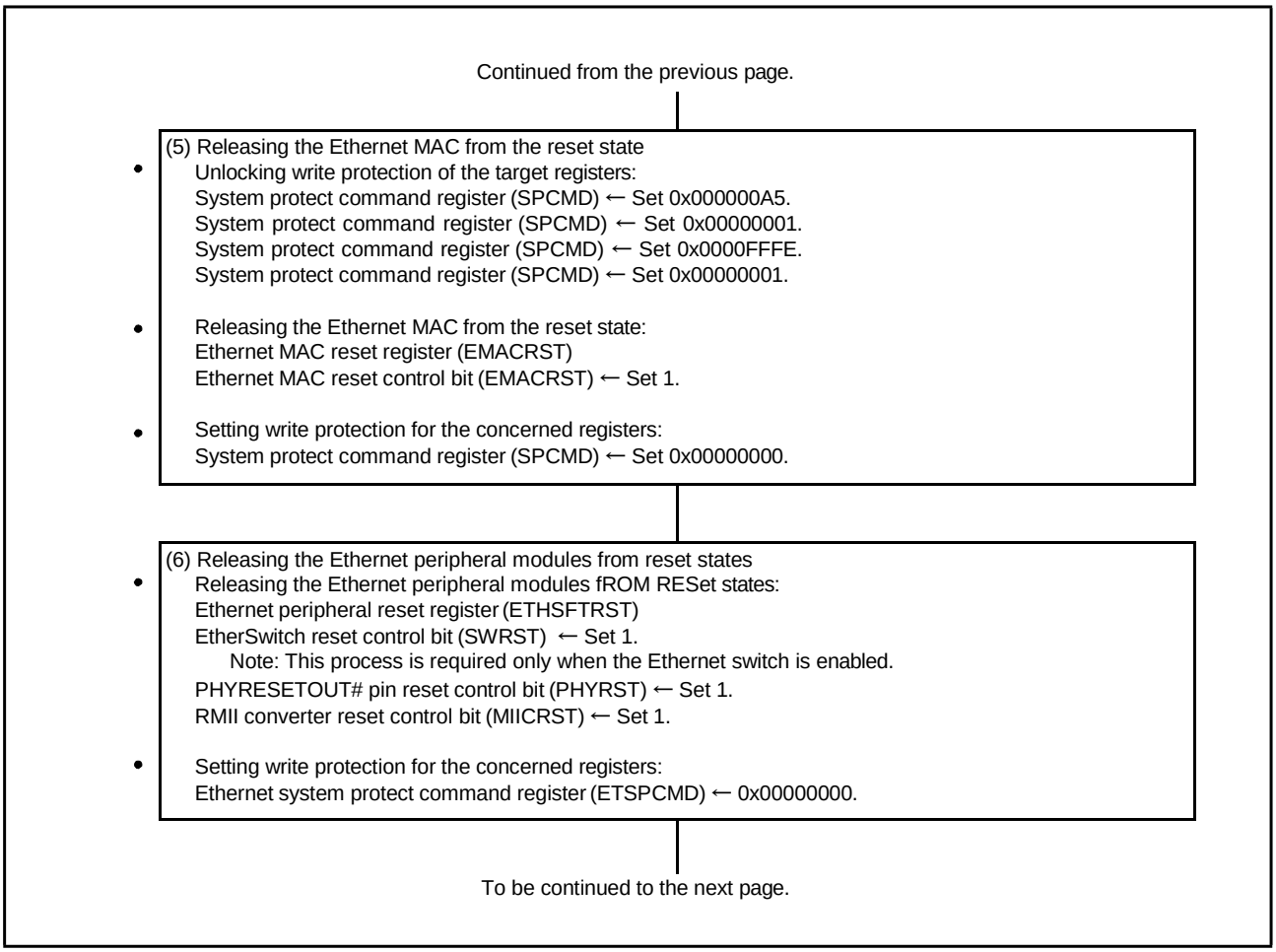


Figure 5.3 Flow of Initialization of the Ethernet MAC (3)

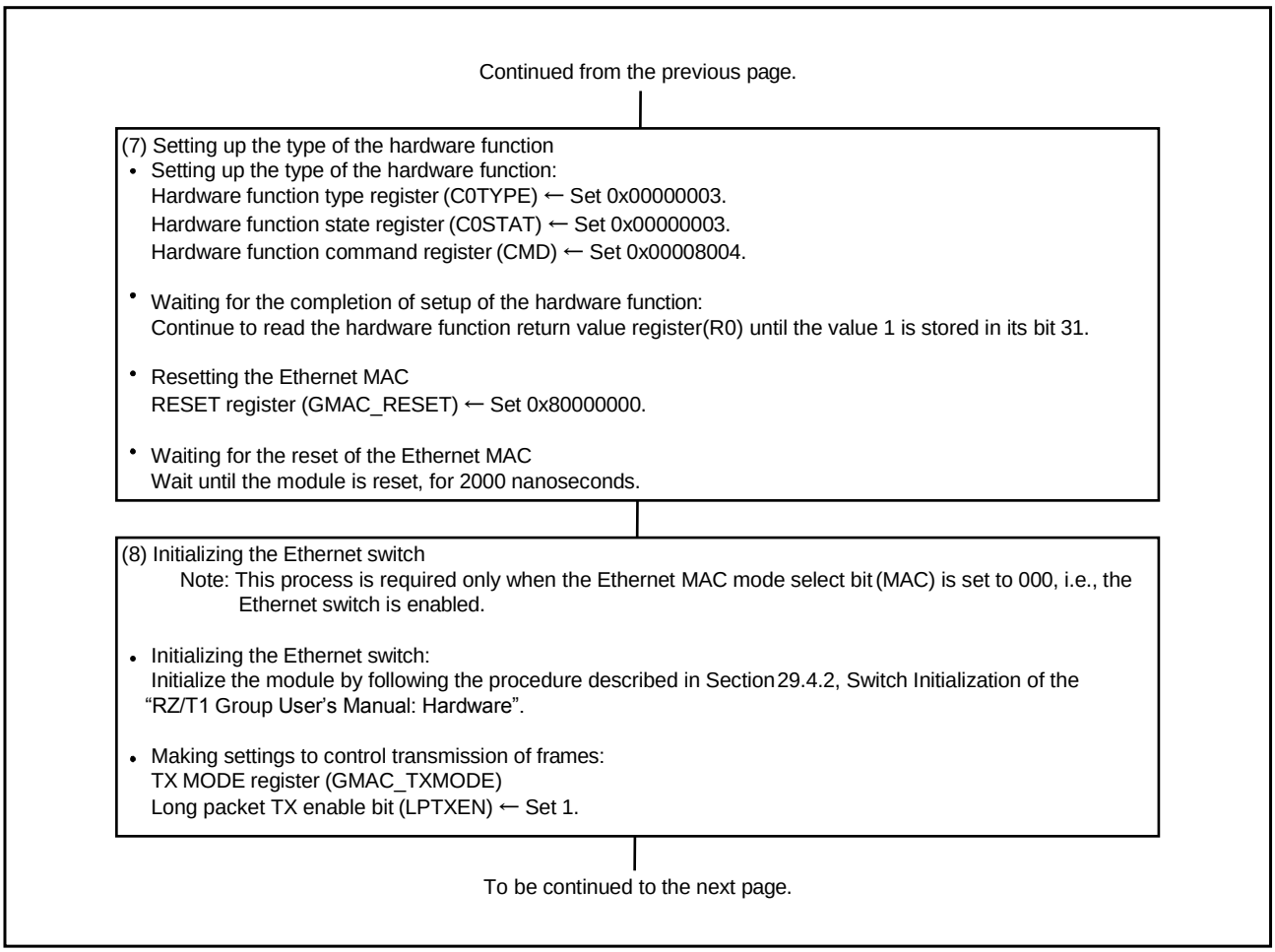


Figure 5.4 Flow of Initialization of the Ethernet MAC (4)

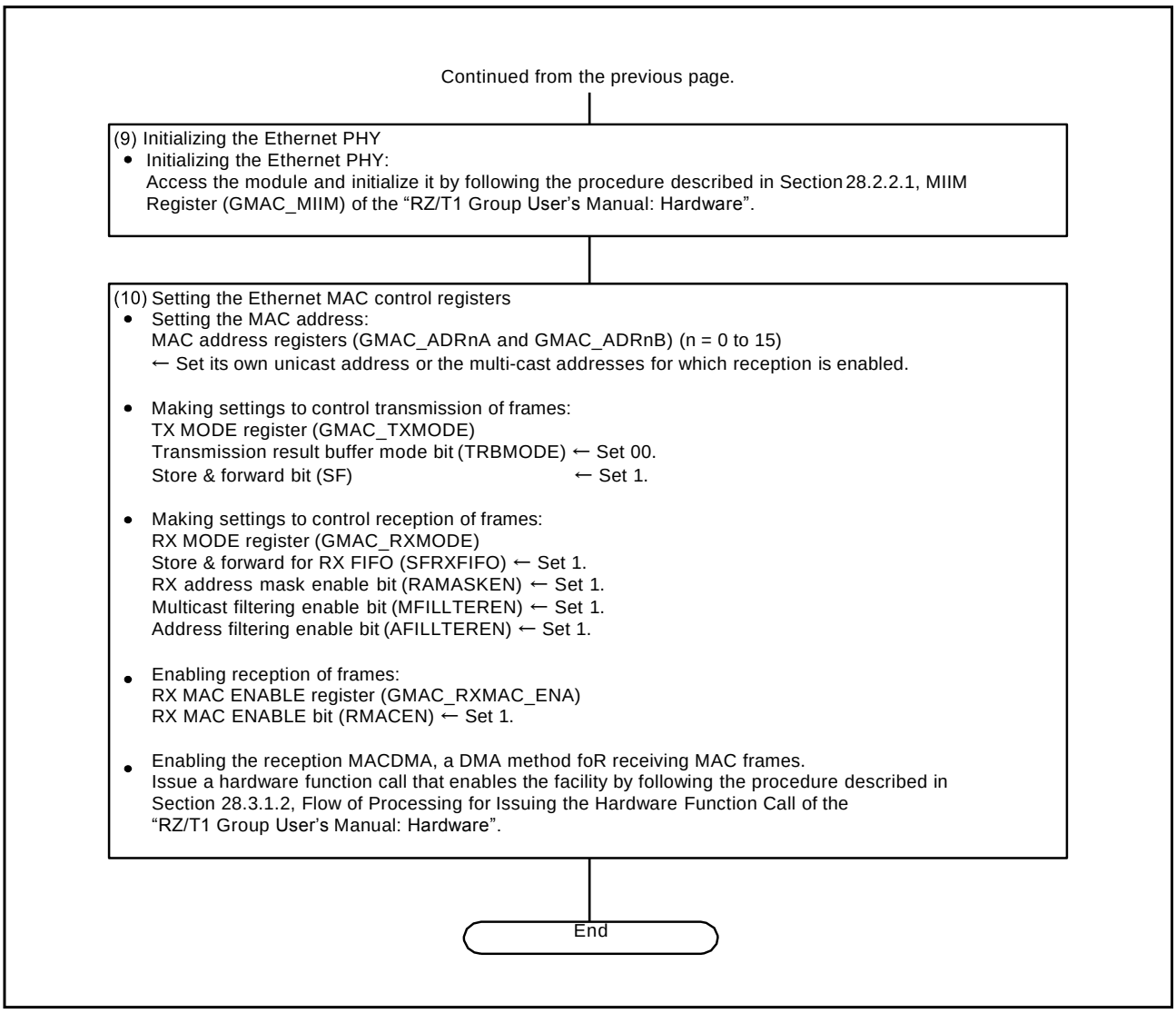


Figure 5.5 Flow of Initialization of the Ethernet MAC (5)

6. Software (Application)

6.1 Outline of the Application

Along with the drivers to control Ethernet communications, this sample program demonstrates functionality for response to ping packets through the transmission and reception of Ethernet frames and simple analysis of the frames.

6.2 Software Block Diagram

A software block diagram is illustrated below.

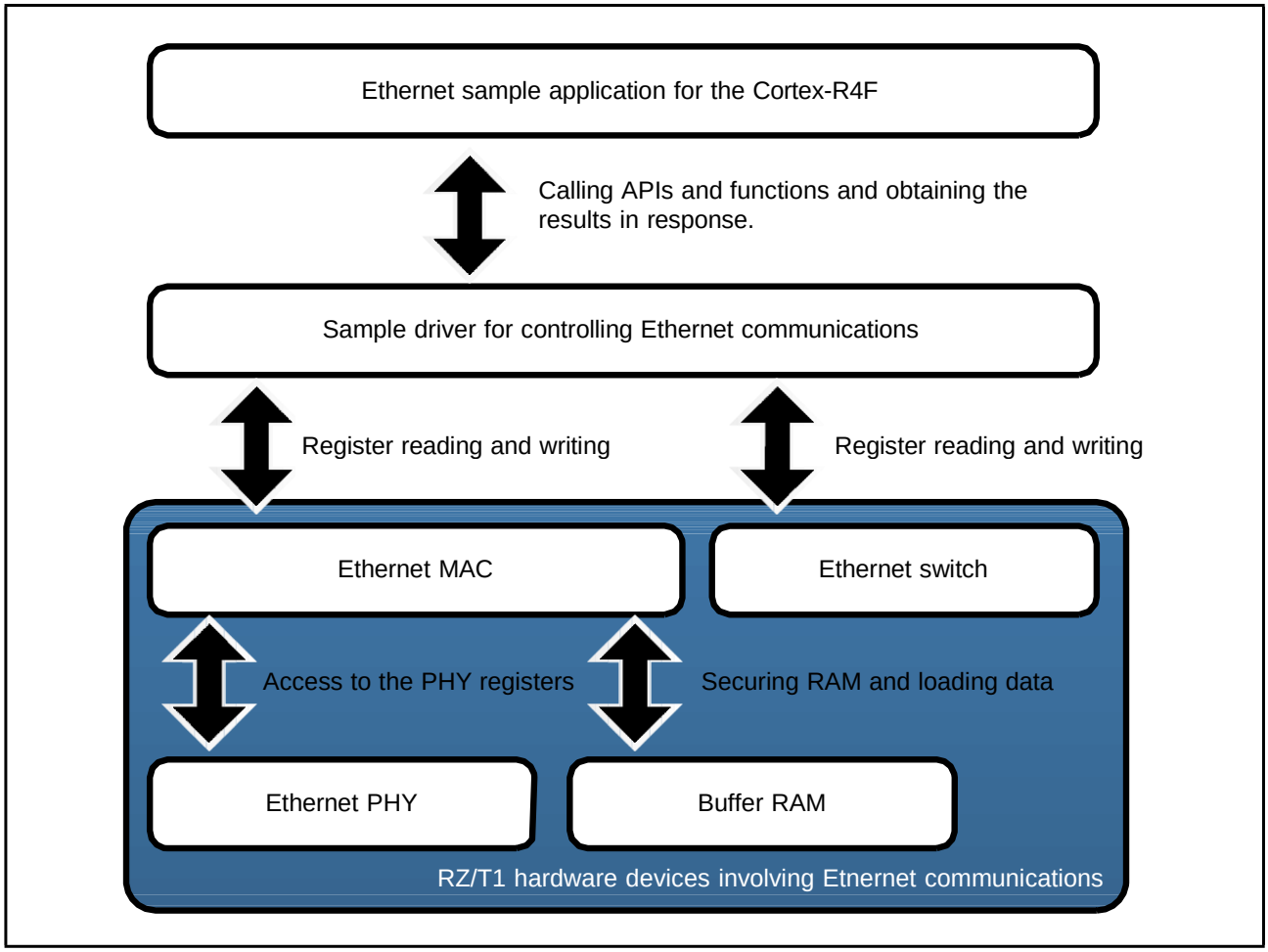


Figure 6.1 Block Diagram of Ethernet Application Software

6.3 Files

The application files used in this sample program are listed below.

Table 6.1 Files Used in the Sample Program

File name	Outline	Remark
.. *1/src/sample/main.c	The file for the main processing of the Ethernet sample program.	
.. *1/src/sample/app.c	The application file for the Ethernet sample program.	

Note: ".. *1" is the folder name of the sample program.

6.4 Constants

Constants used in this sample program are listed below.

Table 6.2 Constants

Constant Name	Setting	Description	Definition File
ARP_HRD_ETH	0x0001	ARP Hardware Type : Ethernet	app.c
ARP_PRO_IP	0x0800	ARP Protocol Type : IP	app.c
ARP_OP_REQ	0x0001	ARP Operation : Request	app.c
ARP_OP_REP	0x0002	ARP Operation : Reply	app.c
ARP_PKT_SIZE	28	ARP Packet Size	app.c
IP_PRO_ICMP	1	IP Protocol Type : ICMP	app.c
IP_PRO_TCP	6	IP Protocol Type : TCP	app.c
IP_PRO_UDP	17	IP Protocol Type : UDP	app.c
IP_HEAD_SIZE	20	IP Packet Header Size	app.c
ICMP_TYPE_ECHO_REP	0	ICMP Type : Echo Reply	app.c
ICMP_TYPE_ECHO_REQ	8	ICMP Type : Echo Request	app.c

6.5 Structures and Unions

A list of structures and unions is given below. Details are described in the subsequent tables.

Table 6.3 List of Structures and Unions

Type Definition	Description	Definition File
arp_pkt_t Structure	Structure for the ARP packet	app.c
icmp_echo_pkt_t Structure	Structure for the ICMP packet	app.c

6.5.1 Details on the Structures and Unions

Table 6.4 arp_pkt_t Structure

Member Name	Description
uint16_t hwtype	Hardware type
uint16_t ptype	Protocol type
uint8_t hwlen	Hardware address length
uint8_t plen	Protocol address length
uint16_t op	Operation code
uint8_t sha[6]	Source hardware address (MAC address)
uint8_t spa[4]	Source protocol address (IP address)
uint8_t dha[6]	Destination hardware address (MAC address)
uint8_t dpa[4]	Destination protocol address (IP address)

Table 6.5 icmp_echo_pkt_t Structure

Member Name	Description
uint8_t type	ICMP type
uint8_t code	Type of service
uint16_t cs	Total length
uint16_t ident	Identifier
uint16_T seq	Sequence number
uint8_t data[2]	The beginning of the payload

6.6 Global Variables

The global variables used in this sample program are listed below.

Table 6.6 List of Global Variables

Type	Variable Name	Description	Definition File
eth_macinfo_t	macAddr_t[]	MAC address information table	main.c
uint8_t	rcv_buf[1514]	Buffer for the received Ethernet frames	main.c
uint8_t	snd_buf[1514]	Buffer for the Ethernet frames to be sent	main.c
eth_frm_t *	ethfrm_rx	A pointer from which the received Ethernet frame is referred	app.c
eth_frm_t *	ethfrm_tx	A pointer from which the Ethernet frame to be sent is referred	app.c
uint32_t	lcl_ipa	Local IP address	app.c

6.7 Error Code

This driver returns negative integers to indicate errors and zero or a positive integer to indicate normal execution.

6.8 Functions

The functions used in this sample program are listed below.

Table 6.7 List of Functions

Function Name	Description	Scope	Definition File
main	Main processing	Global	main.c
app_main	Main processing for the Ethernet frame reception application	Local	app.c
app_arp_recv	ARP packet reception application	Local	app.c
app_ip4_recv	IPv4 packet reception application	Local	app.c
app_icmp_recv	ICMP packet reception application	Local	app.c
app_ip4_echo_res	IPv4 packet echo response application	Local	app.c
app_echo_res	Echo response application	Local	app.c
checksum	For calculaling checksum	Local	app.c

6.8.1 Details on Functions

(1) main

main	
Synopsis	Main processing
Header	r_eth.h r_eth_sw.h
Declaration	int main (void)
Argument	None
Returned value	None
Execution condition	After the boot
processing. Description	This function • initializes the common part of the sample program; • initializes the Ethernet-related modules; • configures the information of transmit and receive buffers; • checks if the link with the ETHPHY module is up; • learns the MAC address of the Ethernet switch; • obtains Ethernet-related events; • checks if Ethernet frames are received or not; and • analyzes the received Ethernet frames.
Error condition	None
Supplement	None
Usage example	None

(2) app_main

app_main		
Synopsis	Main processing for the Ethernet frame reception application.	
Header	r_eth.h	
Declaration	int32_t app_main(eth_frminfo_t *rx , eth_frminfo_t *tx)	
Arguments	eth_frminfo_t *rx	: A pointer to the information of the reception frame.
	eth_frminfo_t *tx	: A pointer to the information of the transmission frame.
Returned values	0	: Normal termination
	-1	: An API error
Execution condition	This function runs the main processing following the reception of Ethernet	
frames. Description	This function • analyzes the received Ethernet frames; • invokes the IPv4 packet reception application if the Ethernet type is of that type; • invokes the ARP packet reception application if the Ethernet type is of that type; and • invokes the echo response application if the Ethernet type is neither of the above two.	
Error condition	None	
Supplement	None	
Usage example	None	

(3) app_arp_recv

app_arp_recv		
Synopsis	ARP packet reception application	
Header	r_eth.h	
Declaration	static int32_t app_arp_recv(eth_frminfo_t *rx , eth_frminfo_t *tx)	
Arguments	eth_frminfo_t *rx	: A pointer to the information of the reception frame.
	eth_frminfo_t *tx	: A pointer to the information of the transmission frame.
Returned values	0	: Normal termination
	-1	: An API error
Execution condition	This function runs the main processing for the Ethernet frames reception application following the reception of ARP packets.	
Description	This function • analyzes the header of the ARP (address resolution protocol) packet; • creates a response packet at a match of the hardware types, protocol types, and the IP addresses and sends it; and • does nothing if there is no match.	
Error condition	An API error occurred.	
Supplement	None	
Usage example	None	

(4) app_ip4_recv

app_ip4_recv		
Synopsis	IPv4 packet reception application	
Header	r_eth.h	
Declaration	static int32_t app_ip4_recv(eth_frminfo_t *rx , eth_frminfo_t *tx)	
Arguments	eth_frminfo_t *rx	: A pointer to the information of the reception frame
	eth_frminfo_t *tx	: A pointer to the information of the transmission frame.
Returned values	0	: Normal termination
	-1	: An API error
Execution condition	This function runs the main processing for the Ethernet frame reception application following the reception of IPv4 packets.	
Description	This function • analyses the header of the IPv4 packets; • invokes the ICMP packet reception application if the protocol type is ICMP; • invokes the IPv4 echo response application if the protocol type is TCP or UDP; and • invokes the echo response application if the protocol type is not ICMP, TCP or UDP.	
Error condition	None	
Supplementary note	None	
Usage example	None	

(5) app_icmp_recv

app_icmp_recv		
Synopsis	ICMP packet reception application	
Header	r_eth.h	
Declaration	static int32_t app_icmp_recv(eth_frminfo_t *rx , eth_frminfo_t *tx) Arguments	
	eth_frminfo_t *rx	: A pointer to the information of the reception frame.
	eth_frminfo_t *tx	: A pointer to the information of the transmission frame.
Returned values	0	: Normal termination
	-1	: An API error
Execution condition	This function runs the IPv4 packet reception application following the reception of ICMP packets.	
Description	This function • checks the checksum of the ICMP (Internet Control Message Protocol) packets; • creates a response packet if no checksum errors are found and the ICMP type is "echo request" and returns it; and • does nothing in the cases other than the above.	
Error condition	A checksum error or an API error occurred.	
Supplementary note	None	
Usage example	None	

(6) app_ip4_echo_res

app_ip4_echo_res		
Synopsis	IPv4 packet echo response application	
Header	r_eth.h	
Declaration	static int32_t app_ip4_echo_res(eth_frminfo_t *rx , eth_frminfo_t *tx)	
Arguments	eth_frminfo_t *rx	: A pointer to the information of the reception frame.
	eth_frminfo_t *tx	: A pointer to the information of the transmission frame.
Returned values	0	: Normal termination
	-1	: An API error
Execution condition	This function is executed from the IPv4 packet reception application when a TCP or UDP packet is received.	
Description	This function • sets the destination IP address, sender IP address, and transmit packet size; and • returns the payload part of the received IPv4 packet to where the packet was transmitted from.	
Error condition	An API error occurred.	
Supplementary note	None	
Usage example	None	

(7) app_echo_res

app_echo_res		
Synopsis	Echo response application	
Header	r_eth.h	
Declaration	static int32_t app_echo_res(eth_frminfo_t *rx , eth_frminfo_t *tx)	
Arguments	eth_frminfo_t *rx	: A pointer to the information of the reception frame.
	eth_frminfo_t *tx	: A pointer to the information of the transmission frame.
Returned values	0	: Normal termination
	-1	: An API error
Execution condition	Reception of unsupported Ethernet frames.	
Description	This function returns the payload part of the reception frame to where the frame was transmitted from.	
Error condition	An API error occurred.	
Supplementary note	None	
Usage example	None	

(8) checksum

checksum		
Synopsis	Calculates checksum.	
Header	r_eth.h	
Declaration	static uint16_t checksum(uint8_t *buf , uint16_t count)	
Arguments	uint8_t *buf	: A pointer to the data used for calculating checksum.
	eth_frminfo_t *tx	: The size of the data used for calculating checksum.
Returned values	0	: Normal termination
	-1	: An API error
Execution condition	None	
Description	This function calculates the checksum of the specified data range.	
Error condition	None	
Supplementary note	Checksum calculation is based on the internet checksum algorithm RFC1071.	
Usage example	None	

6.9 Flowcharts

Flowcharts of the applications are shown below.

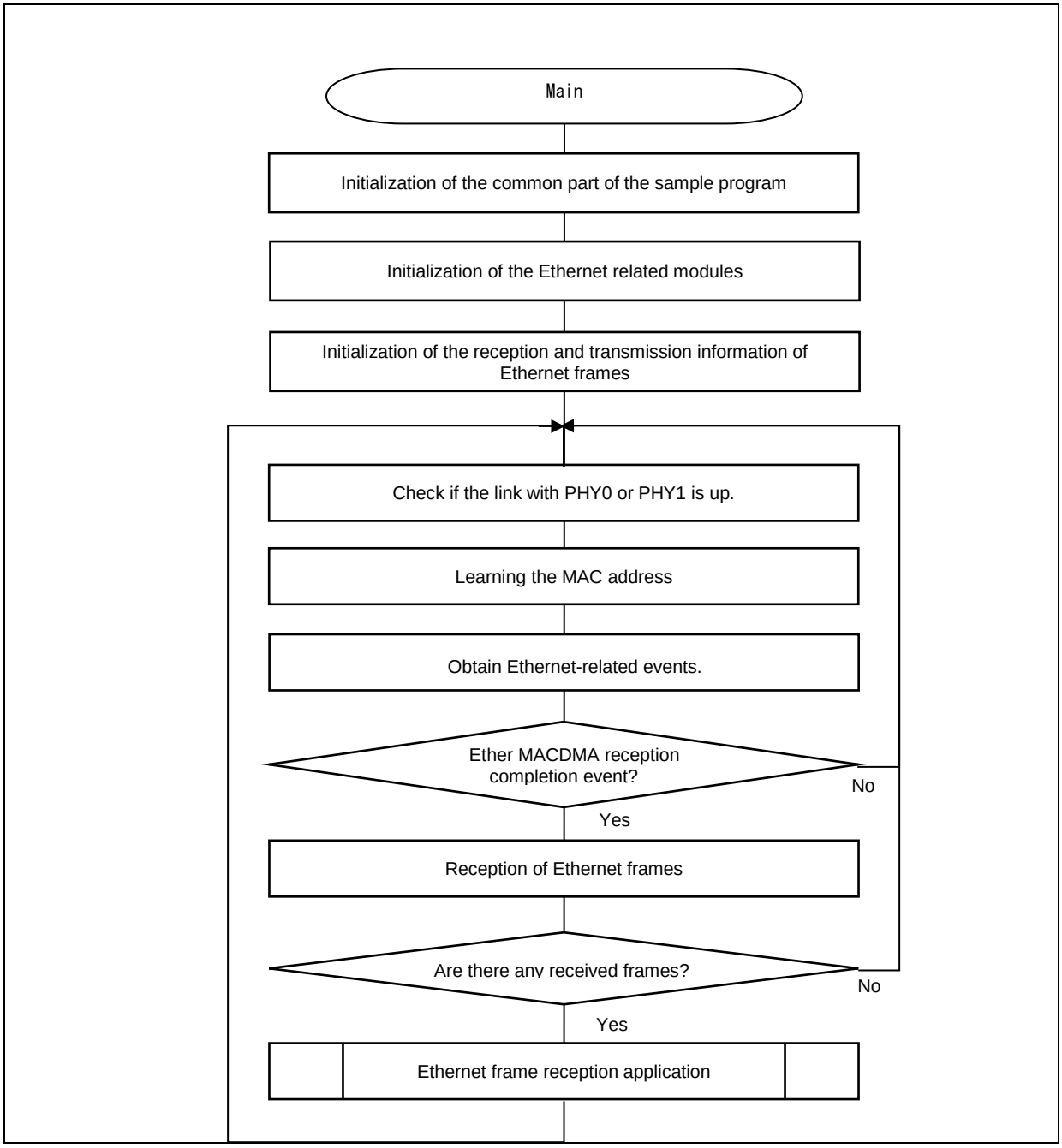


Figure 6.2 Main Processing (main)

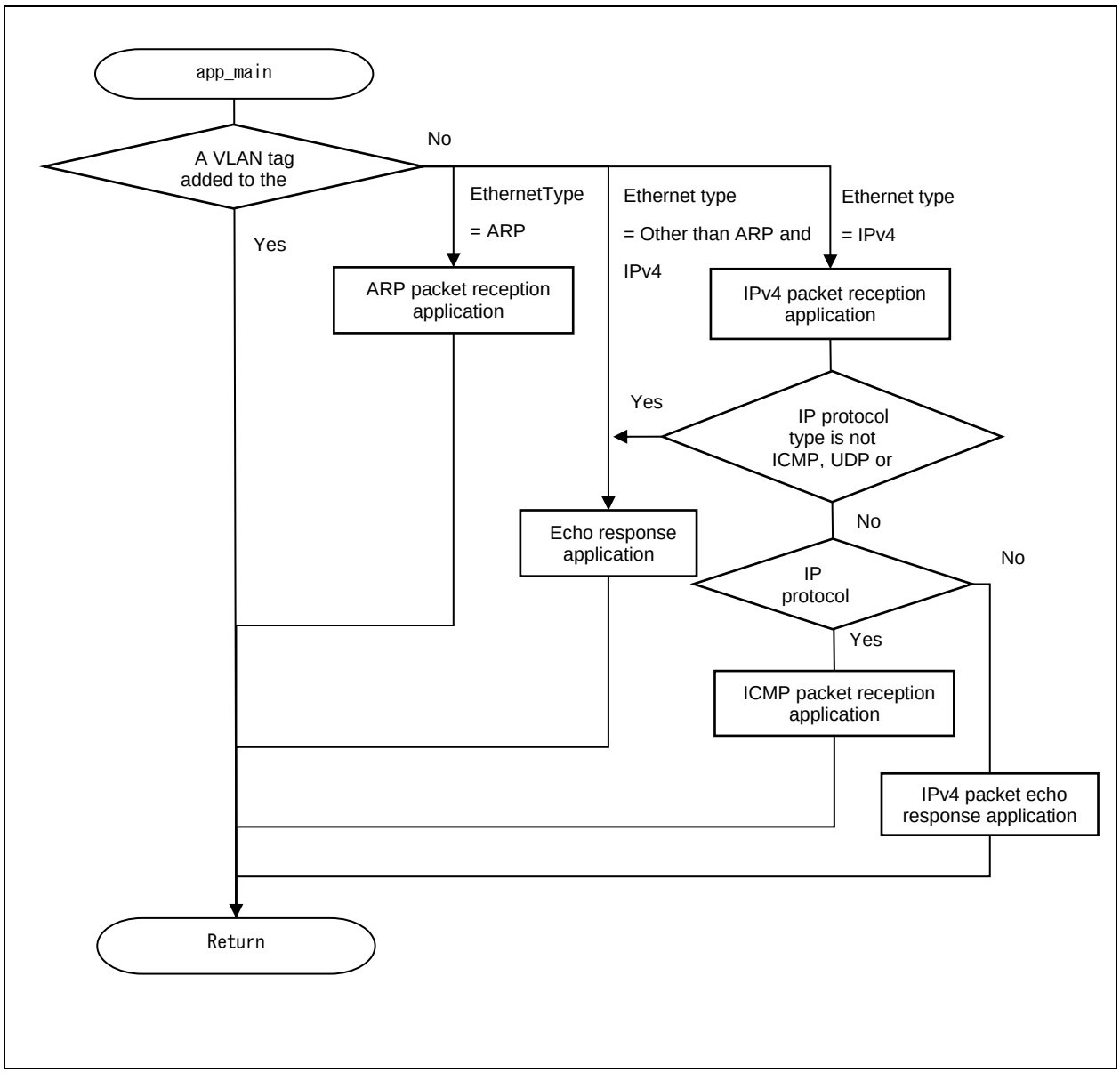


Figure 6.3 Main Processing for the Ethernet Frame Reception Application (app_main)

7. Sample Program

The sample program is available on the Renesas Electronics website.

8. Website and Support

Renesas Electronics website

http://www.renesas.com/

Inquiries

http://www.renesas.com/contact

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Nov. 28, 2016	—	First Edition issued
1.10	Sep 1, 2017	9	Table 5-3 Constant List Event Flag, Ethernet Type Addition
		12	5.5 Configuration Ethernet Port Selection Option Description Add
		14	5.6 Structure / Union List IPv4 Packet Structure Addition
		16	5.8 Function list Add event flag related function
		44	6. Software description (application) IPv4 Packet Echo Response Application related addition

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. 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 Microprocessing unit or Microcontroller unit products 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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other disputes involving 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, including but not limited to, the product data, drawing, chart, program, algorithm, application examples.
3. 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 shall not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics products.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended 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 (traffic lights); large-scale communication equipment; key financial terminal systems; safety control 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 (space and undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability 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. When using the Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident 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 Renesas Electronics 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 ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of failure or malfunction of Renesas Electronics products, 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 by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, 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 investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall 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 shall not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
11. This document shall not be reprinted, 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.

(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.

(Rev.3.0-1 November 2016)

RENESAS

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
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

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 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

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 1207, 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 India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 6.0