V850E2/ML4

Ethernet Send/Receive Setup Example

R01AN1018EJ0100
Rev.1.00
Jun 15, 2012

Introduction

This application note describes an example of the settings required to set up Ethernet/IEEE 802.3 frame transmission and reception using the V850E2/ML4 Ethernet controller.

Target Devices

- V850E2/ML4 group microcontrollers (model uPD70F4022)
- LAN8700i manufactured by Standard Microsystems Corporation

Target Board

V850E2/ML4 CPU board (product number: R0K0F4022C000BR)

Notes

- Since the internal ROM and RAM capacities differ between the different V850E2/ML4 models, the section allocations in table 5.1 must be corrected to match the ROM/RAM capacities of the microcontroller actually used.
- The sample program uses an autonegotiation function to select the communications method. If there is a large difference between the times until the completion of autonegotiation between the V850E2/ML4 and the remote system, communication may fail even though autonegotiation succeeded. If this problem occurs, perform the adjustment described in section 4.7.1, Operating Environment Note 1.

Contents

1. Introduction

1.1 Specifications

- Transmission/reception error handling is not included in this sample program. If required, the user must implement this error handling.
- After a reset is cleared, the sample program sets up the H bus, sets up the I/O registers, and sets up the timers.
- The Standard Microsystems Corporation LAN8700i is used as the Ethernet PHY IC.
- An autonegotiation function is used for the Ethernet PHY IC link.
- The sample program allows one of the following two types of processing to be selected.
  — 10-frame transmission of Ethernet frames
  — 10-frame reception of Ethernet frames

1.2 Functions Used

- H bus
- I/O ports
- Timers
- Ethernet controller
- Interrupts

1.3 Conditions for Application

- Microcontroller: V850E2/ML4
- Evaluation board: V850E2/ML4 CPU board (product number: R0K0F4022C000BR)
- Operating frequencies
  — Input clock: 10 MHz
  — Internal system clock (fCLK): 200 MHz
  — MII transmit clock (fMIITX): 25 MHz
  — MII receive clock (fMIIRX): 25 MHz
  — H bus clock: (fHCLK): 33.3 MHz
- Operating mode
  — Normal operating mode
- Integrated development environment*[1]
  — Renesas Electronics Corporation CubeSuite+ Version 1.01.00 [October 20, 2011]
    C compiler: Renesas Electronics Corporation CX Version 1.21
  — Green Hills Software MULTI Version 5.1.7D
- Compiler options
  Default compiler options
  -cpu=v850 -output=obj="$(CONFIGDIR)\$(FILELEAF).obj" -nologo

Note: 1. Both a project for CubeSuite+ and a project for MULTI are included together in this application note's sample code workspace directory. The Ethernet driver source code and the following descriptions apply to both.

2. Initialization

An initialization program that initializes the H bus and other hardware is required to use the Ethernet driver included in the sample code. This section describes the settings used in the sample program.

2.1 Initialization Program

After a reset is cleared, the first thing the main program does is to set up the H bus, the ports used for the Ethernet, and the timer (TAUA0) and to enable interrupts. Figure 2.1 shows the flowchart for this initialization processing.

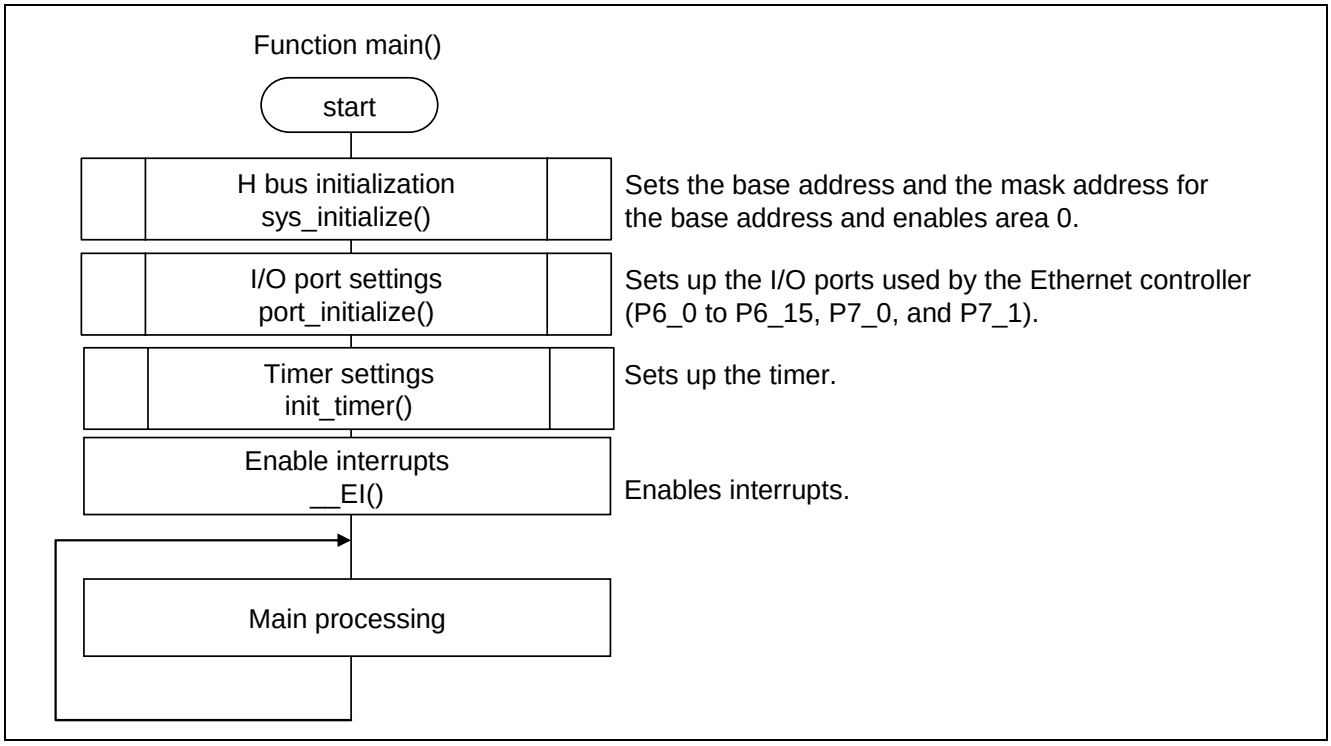


Figure 2.1 Post-Reset Initialization Flowchart

2.2 Settings During Initialization

Table 2.1 lists the settings made by the sample program.

Table 2.1 Sample Program Settings

Module	Settings
Operating mode	Normal operating mode
H bus settings	Enables area 0
I/O ports	Settings for the pins used by the Ethernet (P6_0 to P6_15, P7_0, and P7_1)
Timer settings	TAUA0 is set to issue an interrupt every 1 ms.

3. PHY IC Autonegotiation

The sample program uses the autonegotiation function provided by the Ethernet PHY IC. The result of autonegotiation is read from the port pin read register (PPR7).

3.1 Operational Overview of Functions Used

The physical layer link processing is performed by the Ethernet PHY IC. Thus the V850E2/ML4 internal Ethernet controller can acquire the link results simply by reading those results from the Ethernet PHY IC. The sample program enables the autonegotiation function provided by the PHY IC. See the datasheet for the PHY IC itself for details on the Ethernet PHY IC's functions.

The interface between the Ethernet controller and the Ethernet PHY IC conforms to the IEEE 802.3 MII (Media Independent Interface) and RMII (Reduced Media Independent Interface) standards.

Figure 3.1 shows the connections between the V850E2/ML4 and the LAN8700i.

The autonegotiation result is stored in an Ethernet PHY IC internal register and is read out using a serial interface (Serial Management Interface) using the MDC and MDIO pins. The V850E2/ML4 can read and write these pins using port control. See section 3.2, MII/RMII Register Access Procedure, for details on the procedure for accessing the PHY IC internal registers.

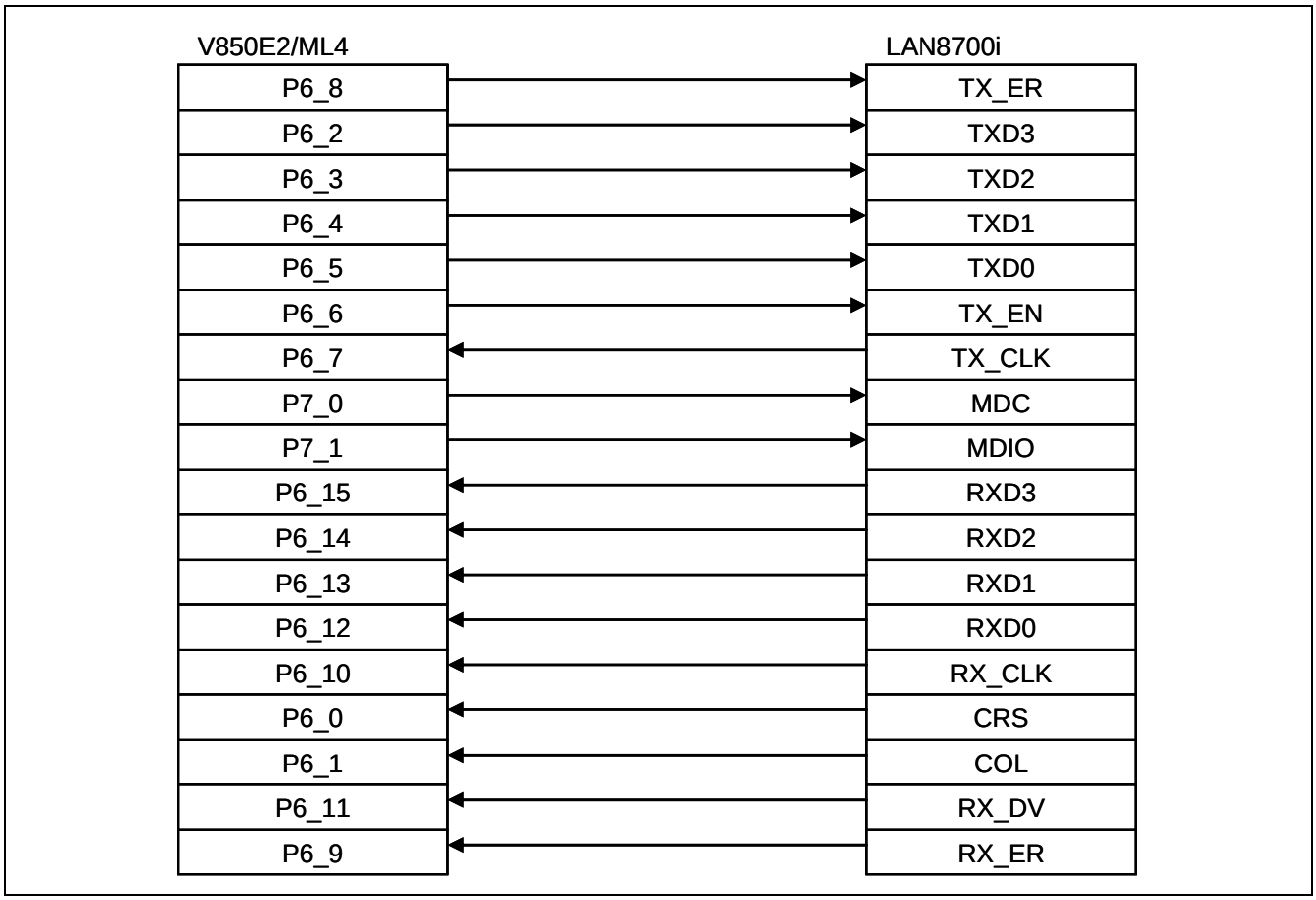


Figure 3.1 Connections with the LAN8700i (MII)

3.2 MII/RMII Register Access Procedure

This section describes the procedure for accessing the MII/RMII registers, which are Ethernet PHY IC internal registers.

The serial interface (Serial Management Interface) used to access the MII/RMII registers consists of two pins, the MDC and MDIO pins (which are both Ethernet controller pins); MDC is the synchronization clock pin and MDIO is the data I/O pin. The states of these pins can be referenced and modified using port control. Data that conforms to the stipulated format (the MII/RMII management frame) must be output by MII. Figure 3.2 shows the MII management frame. The sample program performs Z0 output for 1 bit in the IDLE state. Although the IEEE 802.3 standard does not mention clock input, there are cases where correct connection by the PHY IC is not possible, and this is performed for safety.

Input and output of MII/RMII management frames is performed in 1-bit units starting from PRE. Figures 3.3 to 3.6 show the I/O flow in 1-bit units. The MDC and MDIO timings must conform to the IEEE 802.3 standard. Table 3.1 and figure 3.7 show the I/O timings for the IEEE 802.3 standard.

Access ID	MII/RMII Management Frame							
Item	PRE	ST	OP	PHYAD	REGAD	TA	DATA	IDLE
Number of bits	32	2	2	5	5	2	16	—
Read	1..1	01	10	00001	RRRRR	Z0	D..D	—
Write	1..1	01	01	00001	RRRRR	10	D..D	X

Legend:

PRE: 32 consecutive bits that are all one

ST: A write of 01 that indicates the start of a frame

OP: Write of a code that indicates the access ID

PHYAD: Write 00001 (write in order starting with the MSB) if the PHY IC address is first. These bits change depending on the PHY IC address.

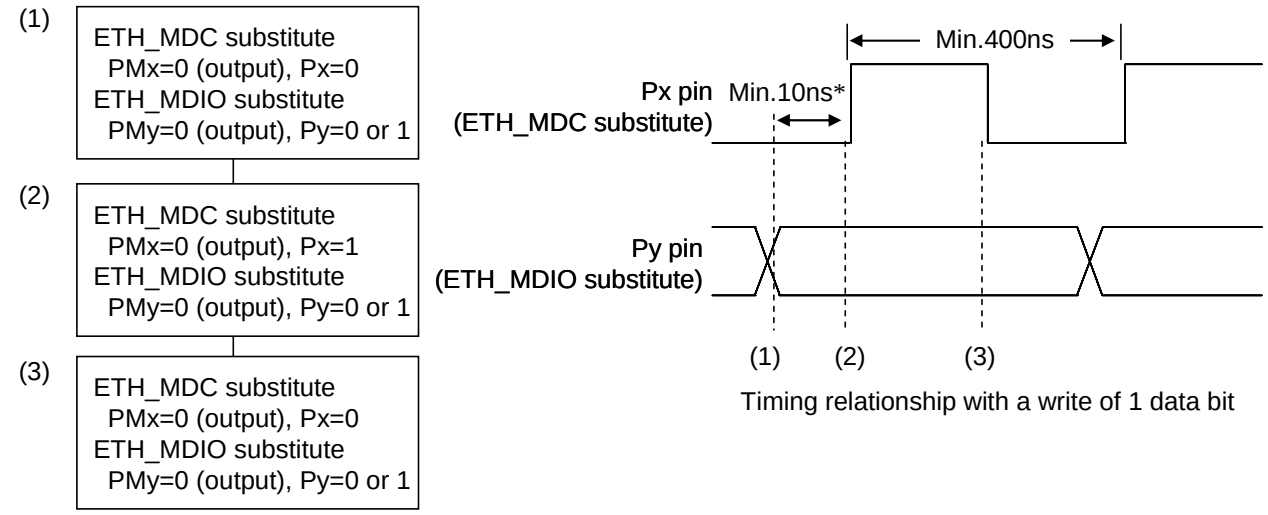
REGAD: Write 00001 (write in order starting with the MSB) if the register address is first. These bits change depending on the PHY IC register address.

TA: The time to switch data transfer sources in the MII/RMII interface
(a) On write: Write 10.
(b) On read: Perform a bus release (notated as Z0)

DATA: 16 bits of data. These bits are read or written in order starting with the MSB.
(a) On write: Write 16 bits of data.
(b) On read: Read 16 bits of data.

IDLE: The wait time until the next MII management frame input
(a) On write: Perform an independent bus release (notated as X)
(b) On read: The bus has already been released at TA and control is not required.

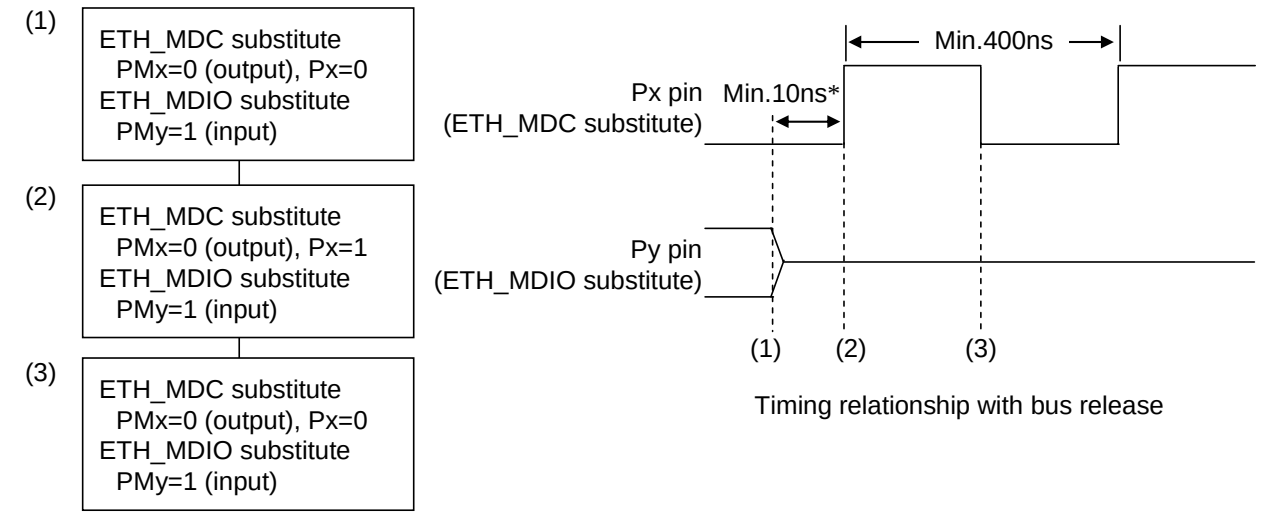
Figure 3.2 MII/RMII Management Frame Format



Timing relationship with a write of 1 data bit

Note: * The 10 ns delay between (1) and (2) can be assured by executing different instructions, even if those instructions are consecutive.

Figure 3.3 One Bit Data Write Flow



Timing relationship with bus release

Note: * The 10 ns delay between (1) and (2) can be assured by executing different instructions, even if those instructions are consecutive.

Figure 3.4 Bus Release Flow (for TA during read)

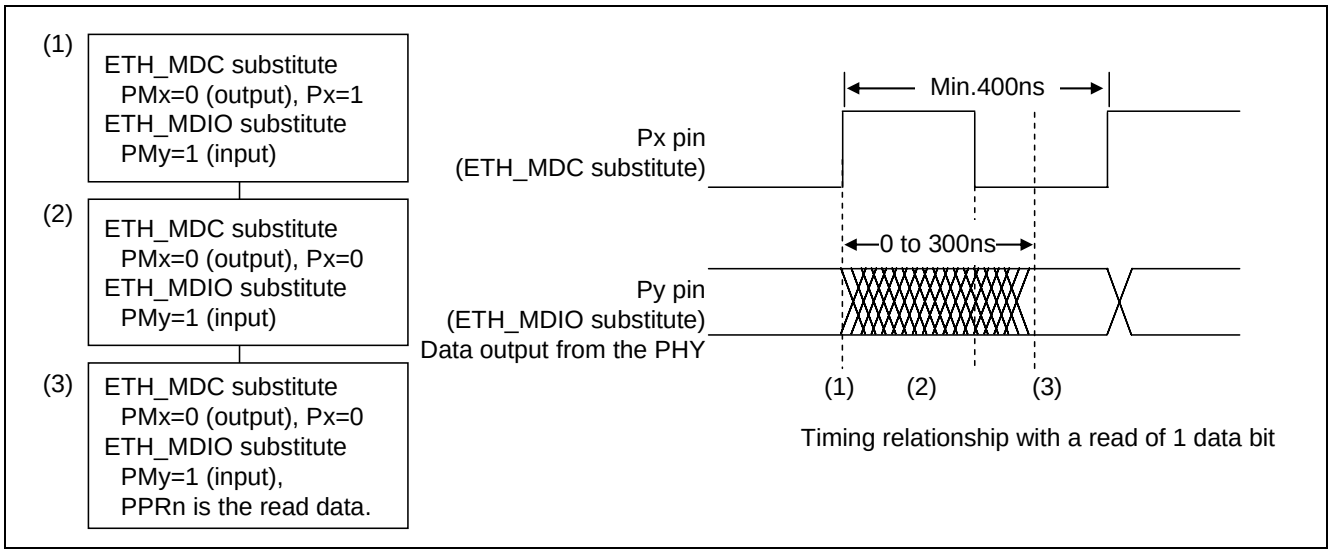


Figure 3.5 One Bit Data Read Flow

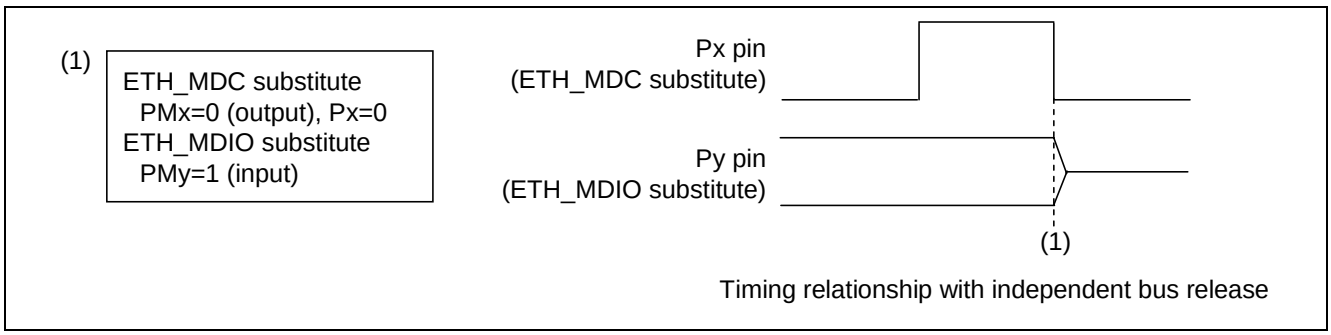


Figure 3.6 Independent Bus Release Flow (IDLE during write)

Table 3.1 MDC/MDIO I/O Timing

Item	Symbol	Min.	Max.	Unit
MDC high-level pulse width	t_1	160		ns
MDC low-level pulse width	t_2	160		ns
MDC cycle time	t_3	400		ns
MDIO setup time	t_4	10		ns
MDIO hold time	t_5	10		ns
MDIO output delay time	t_6	0	300	ns

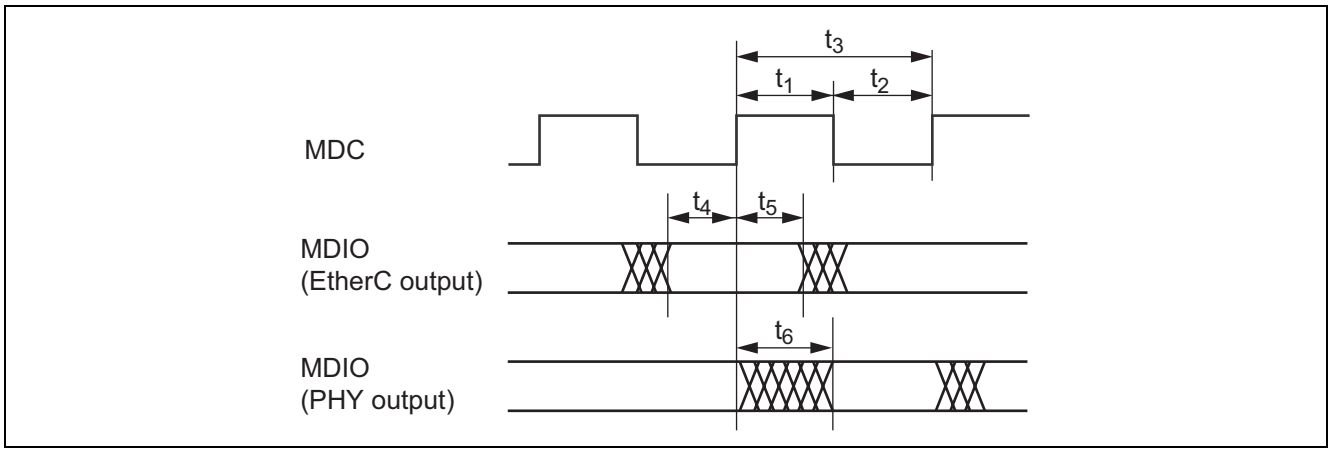


Figure 3.7 MDC/MDIO I/O Timing

3.3 PHY IC Autonegotiation Setup

ether_phy.c

The PHY IC reset function (phy_reset()) and the PHY IC initialization function (phy_initialize()) are coded in this file.

Figure 3.8 shows the flowchart for phy_reset() and figure 3.9 shows the flowchart for phy_initialize().

Also, figures 3.10 to 3.16 show the flowcharts for the processing called from these functions.

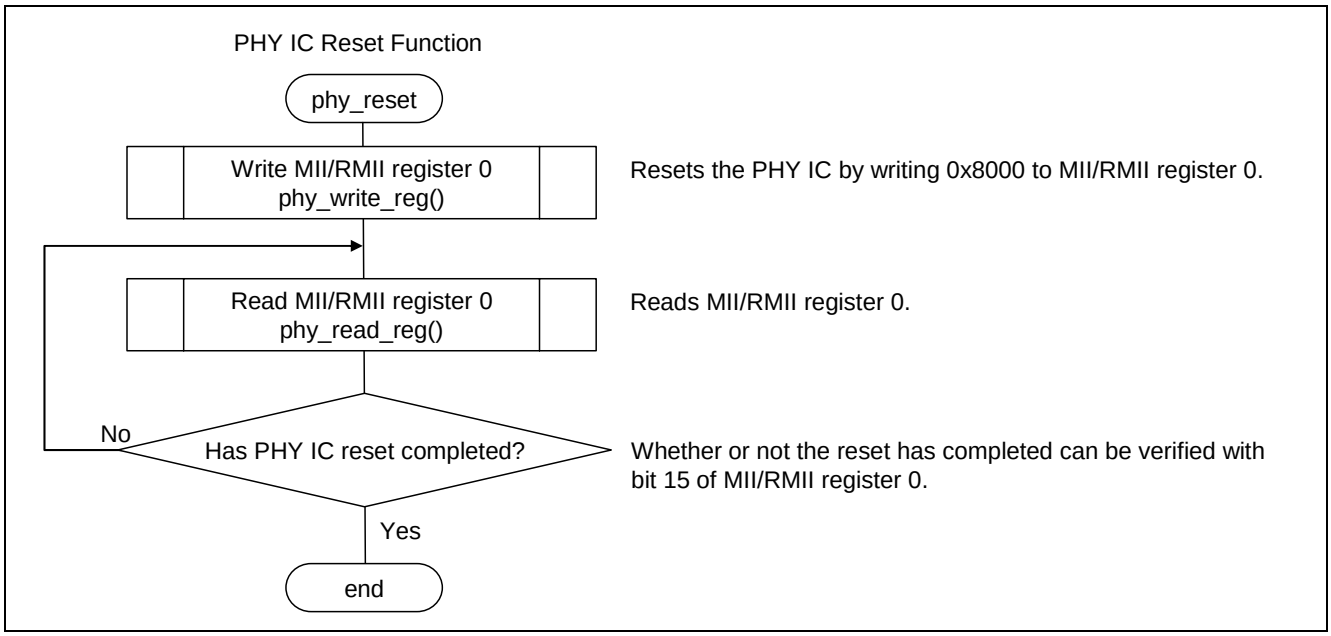


Figure 3.8 PHY IC Reset Function Flowchart

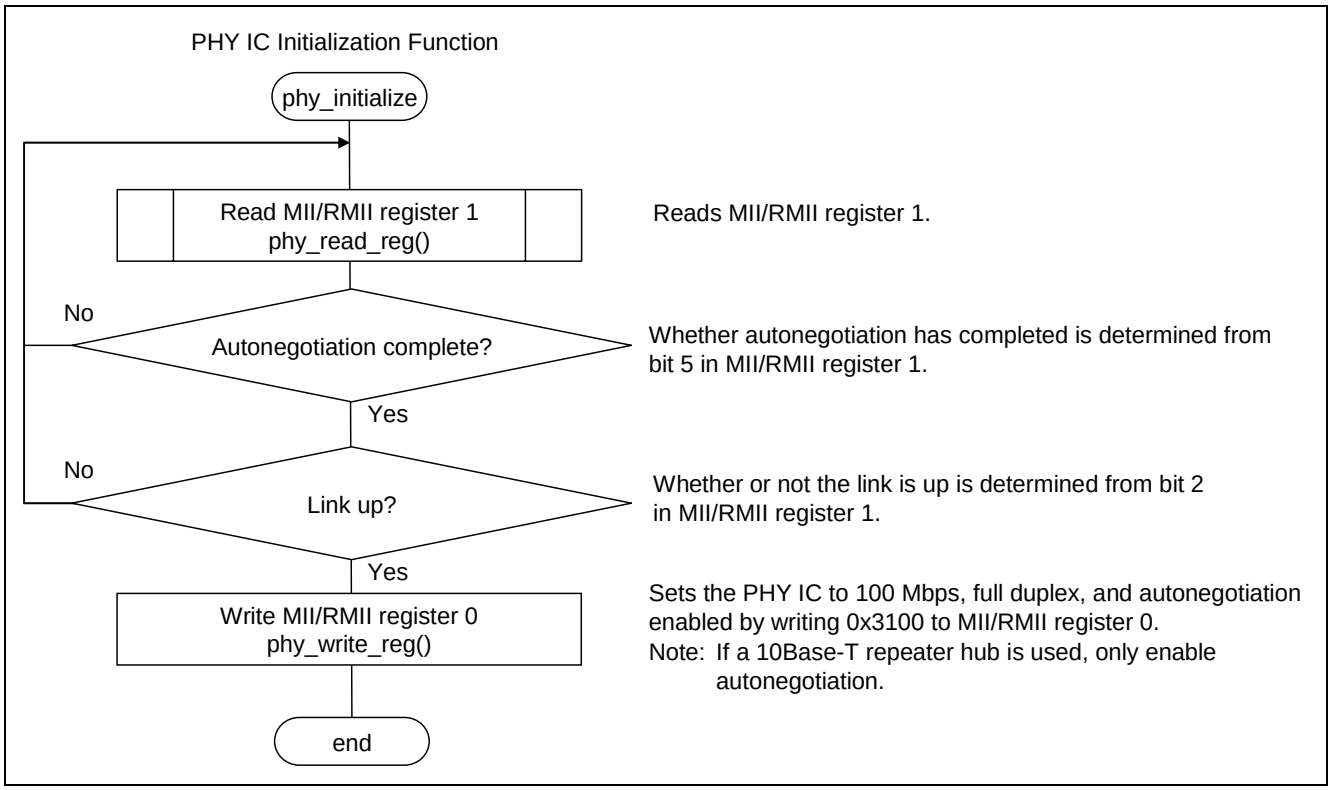


Figure 3.9 PHY IC Initialization Function Flowchart

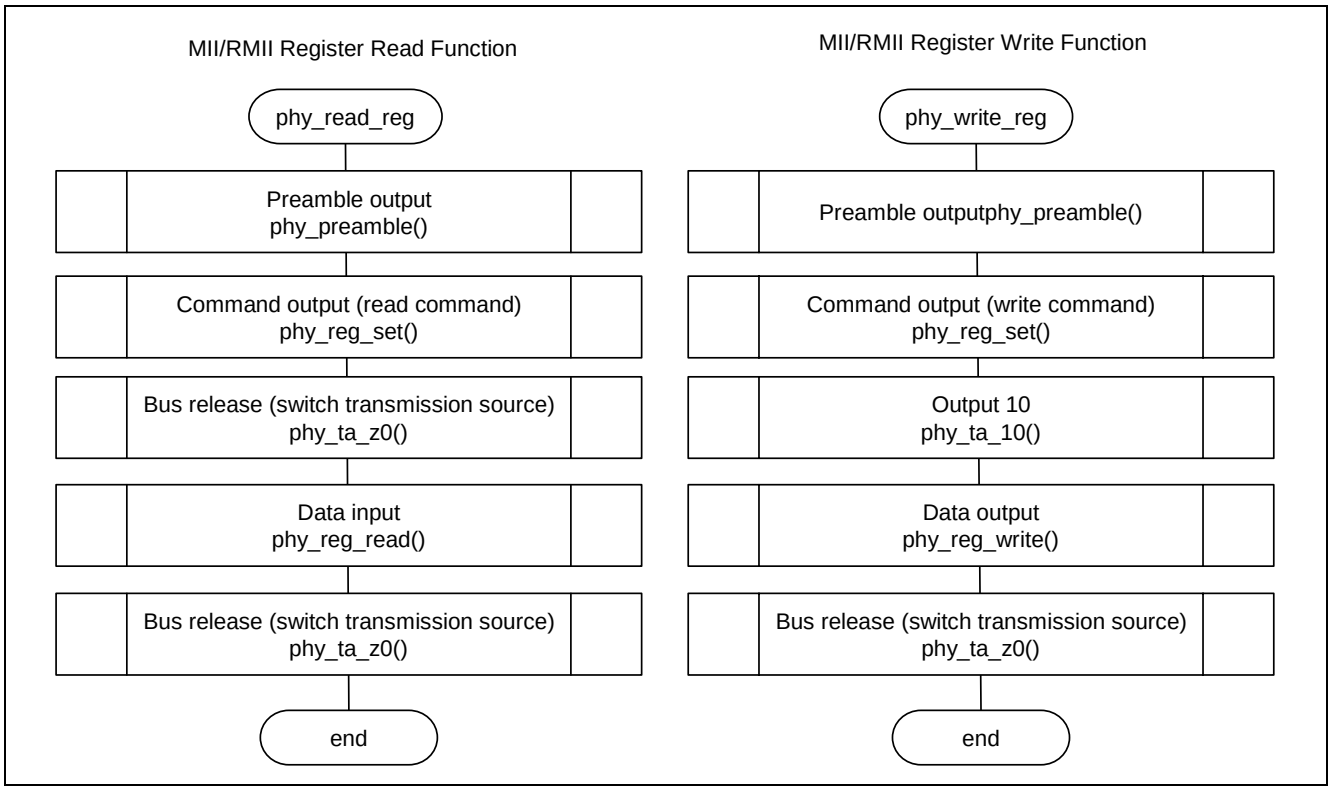


Figure 3.10 MII/RMII Register Access Processing Flowchart (1)

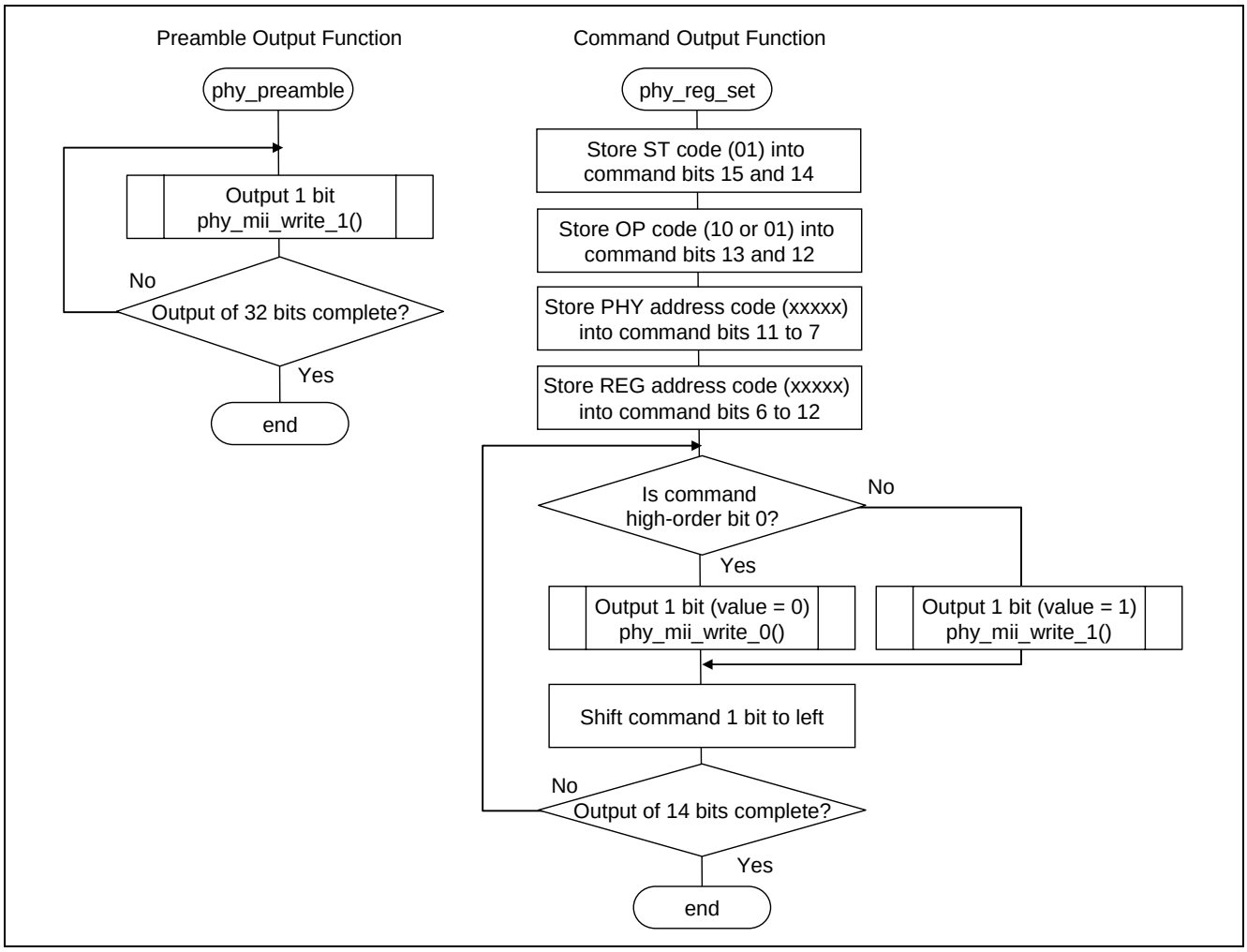


Figure 3.11 MII/RMII Register Access Processing Flowchart (2)

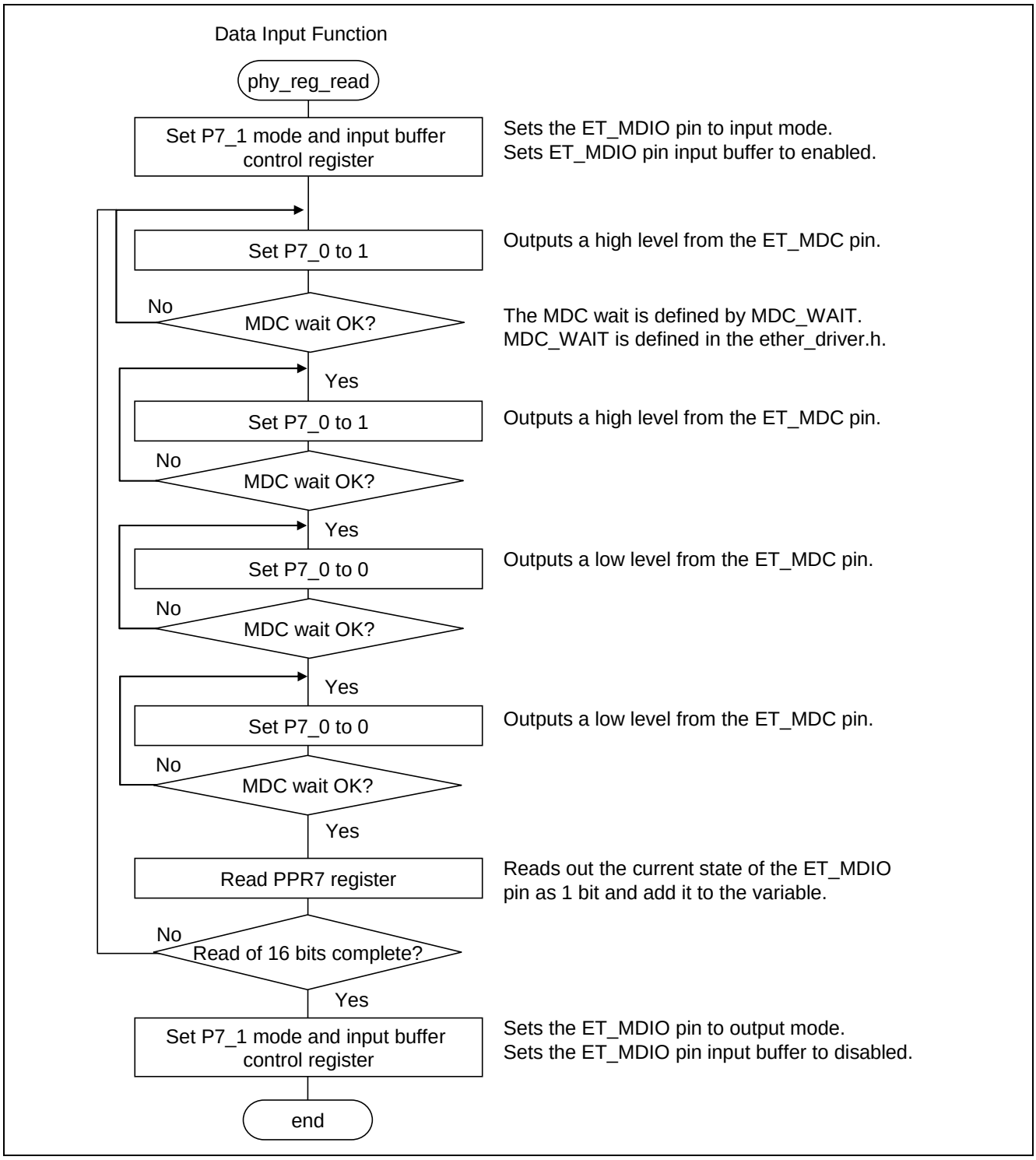


Figure 3.12 MII/RMII Register Access Processing Flowchart (3)

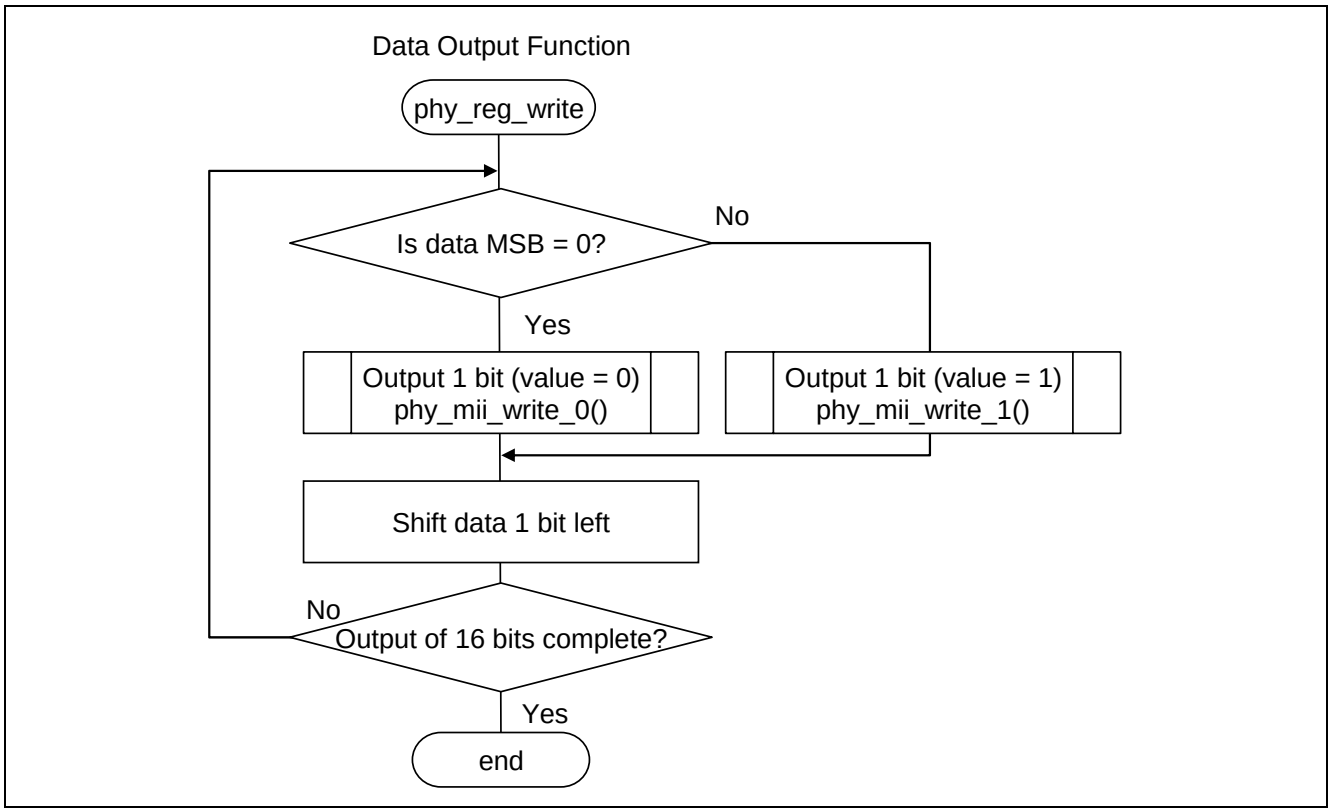


Figure 3.13 MII/RMII Register Access Processing Flowchart (4)

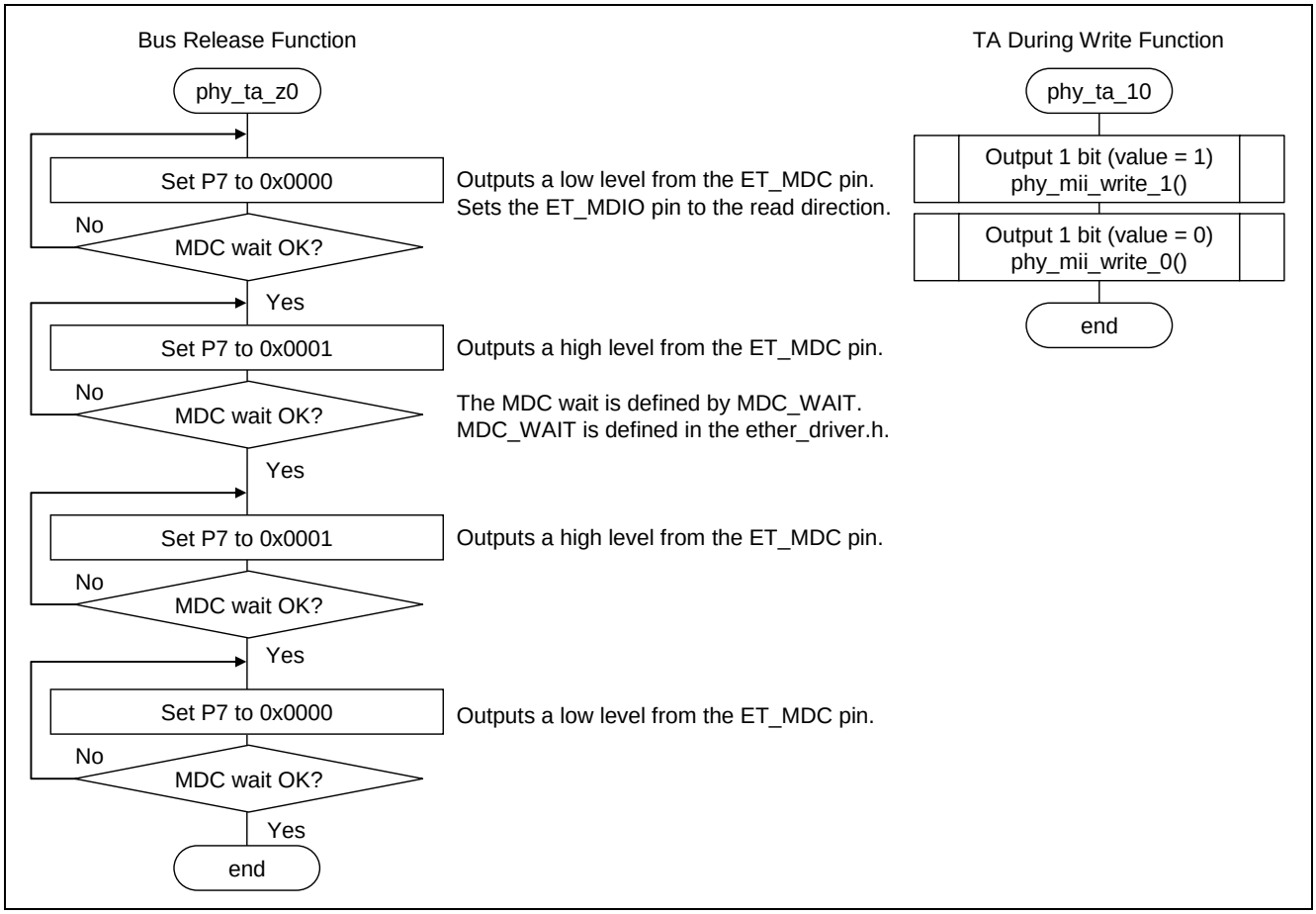


Figure 3.14 MII/RMII Register Access Processing Flowchart (5)

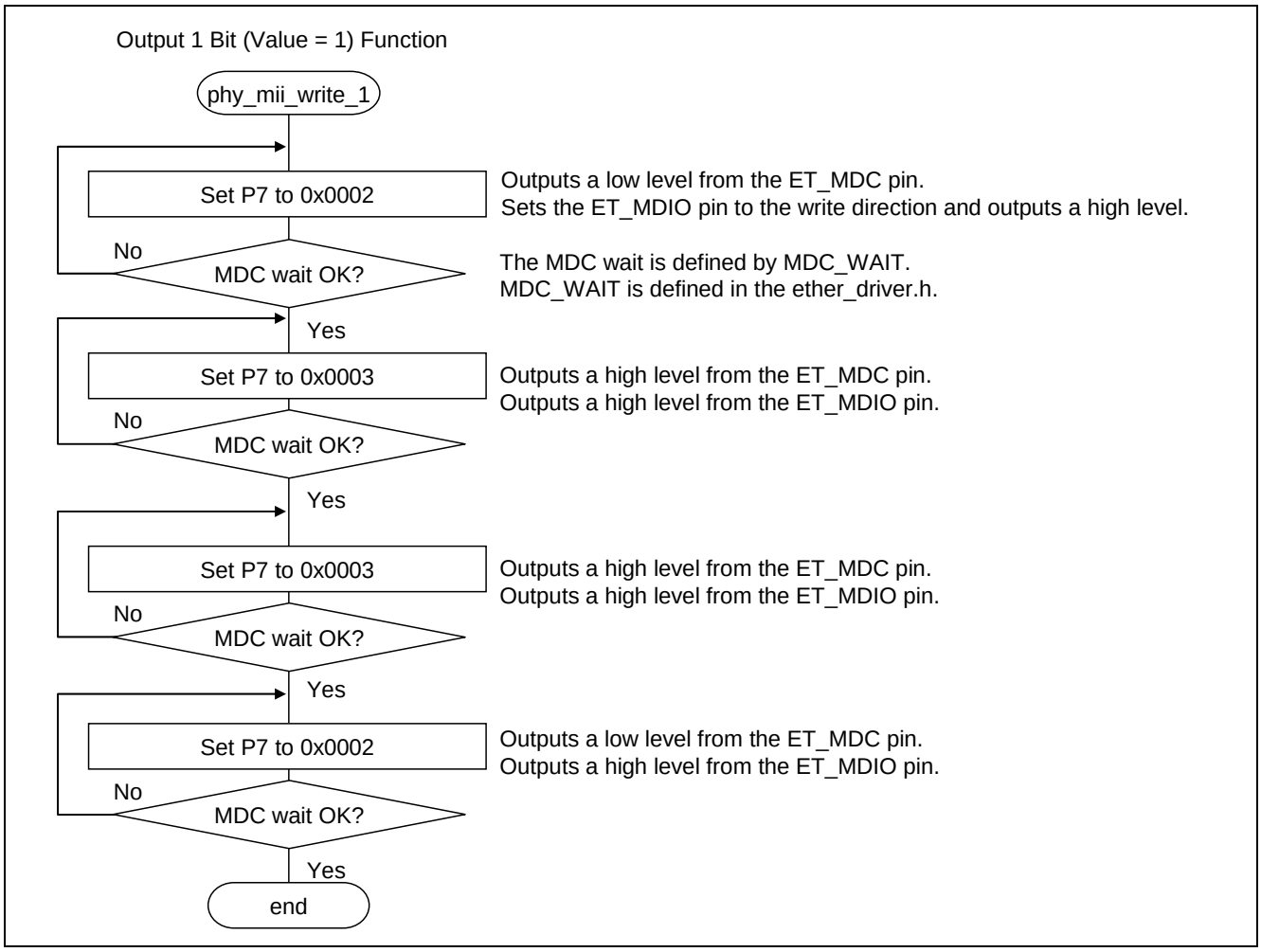


Figure 3.15 MII/RMII Register Access Processing Flowchart (6)

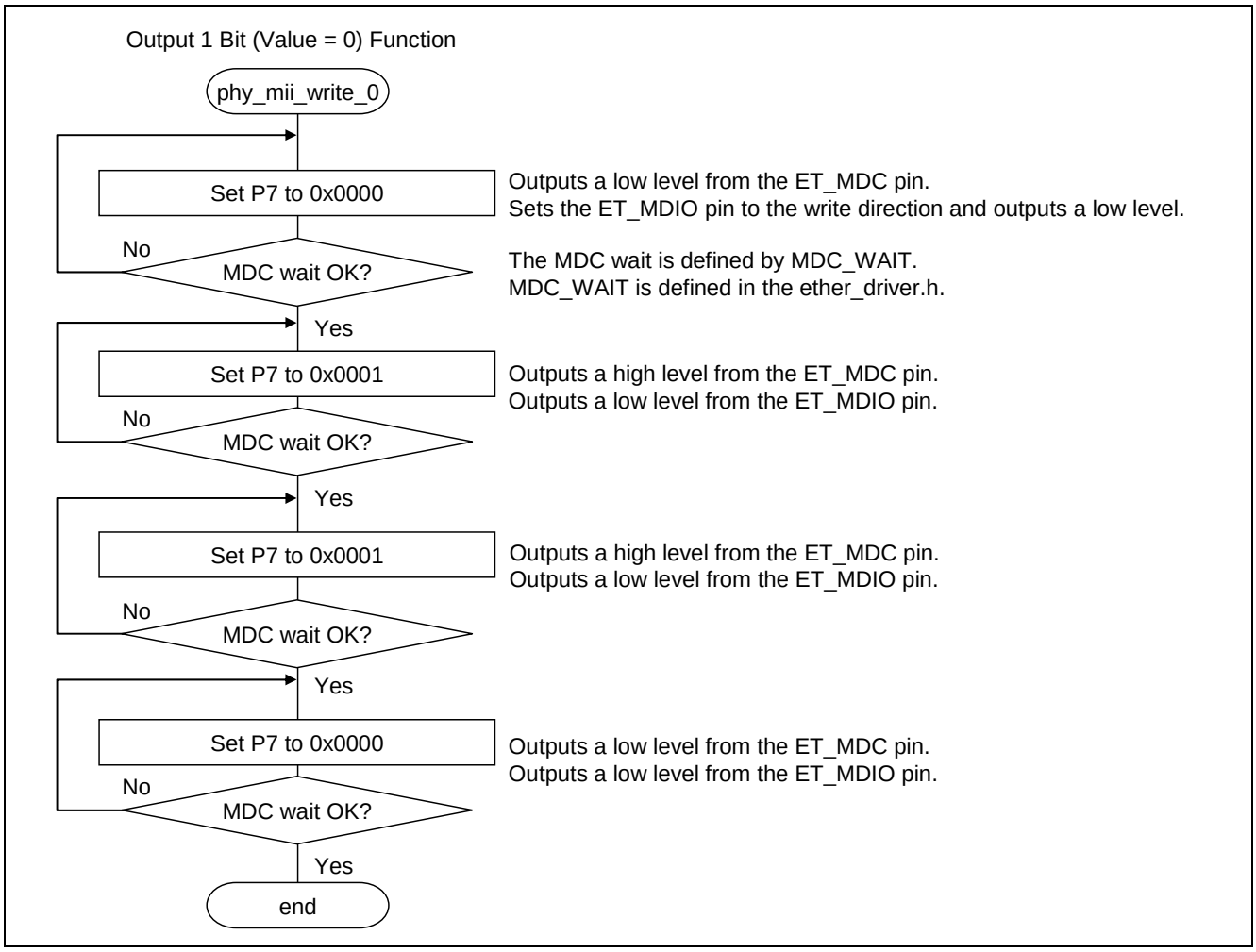


Figure 3.16 MII/RMII Register Access Processing Flowchart (7)

3.4 PSY IC Autonegotiation Settings

Table 3.2 lists these sample program settings.

Table 3.2 Sample Program Settings

Item	Description
PHY IC model number	LAN8700i manufactured by Standard Microsystems Corporation
Link mode	100 Mbps (full duplex, half duplex) and 10 Mbps (full duplex, half duplex)
Link determination method	Autonegotiation
PHY address	0x1F*[1]
MII registers set	Register 0 — Basic Control (Address: 0x00) Register 1 — Basic Status (Address: 0x01) Register 4 — Auto Negotiation Advertisement (Address: 0x04) Register 5 — Auto Negotiation Link Partner Ability (Address: 0x05)

Note: 1. The V850E2/ML4 CPU board (product number: R0K0F422C000BR) setting is 0x1F. This must be changed to match the PHY address actually used.

3.5 Notes on PHY IC Autonegotiation

- The sample program uses autonegotiation as the method for PHY link determination.
- If the remote system uses autonegotiation, the link mode will be determined according to the priorities listed in table 3.3.

Table 3.3 Link Mode Priority

Priority		Link mode
High	1	100 Mbps full duplex
	2	100 Mbps half duplex
	3	10 Mbps full duplex
Low	4	10 Mbps half duplex

- The MII/RMII register access timing can be changed with a macro definition in the ether_driver.h file. Set this macro to a value of 1 or larger.

```
#define  MDC_WAIT  2
```

- The V850E2/ML4 CPU board (product number: R0K0F422C000BR) PHY address is set to 0x1F. The PHY address can be changed with the macro definition in the ether_driver.h file shown below.

```
#define PHY_ADDRESS        ( 31 )
```

4. Transmission/Reception Settings

The sample program uses the Ethernet controller and the dedicated Ethernet controller DMAC.

4.1 Operational Overview of Functions Used

The V850 family microcontrollers use the Ethernet controller and the dedicated Ethernet controller DMAC to implement Ethernet communication. The Ethernet controller performs the transmission/reception control. The dedicated Ethernet controller DMAC performs DMA transfers between the transmit/receive FIFOs and the data storage targets (data buffers).

4.1.1 Ethernet Controller Overview

The V850 family microcontrollers include an Ethernet controller that conforms to the Ethernet or IEEE 802.3 MAC (media access control) layer standards. By connecting to a physical layer IC (PHY IC) that matches that standard, the Ethernet controller can send and receive Ethernet/IEEE 802.3 frames. The Ethernet controller includes a single MAC layer interface. Also, the Ethernet controller is connected internally to the dedicated Ethernet controller DMAC and supports high-speed memory access.

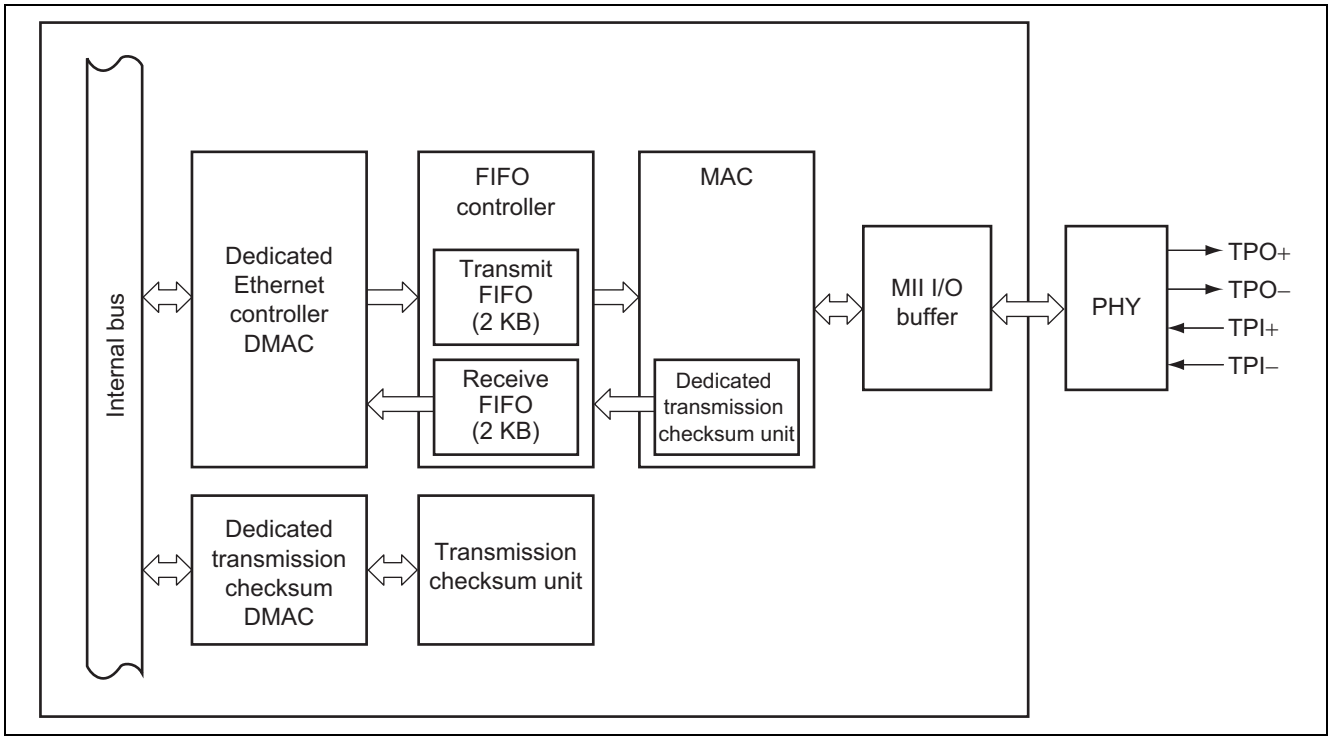


Figure 4.1 Ethernet Controller Structure

4.1.2 Frame Formats Supported By the Ethernet Controller

The Ethernet controller supports the following three frame formats.

- Basic frame
- VLAN frame
- Pause control frame

(1) Basic Frame

This is the basic frame format used by Ethernet and consists of 7 elements.

- PA: Preamble
- SFD: Frame start delimiter
- DA: Destination address
- SA: Source address
- Type/LEN: Type/length field
- DATA: Data field
- FCS: Frame check sequence

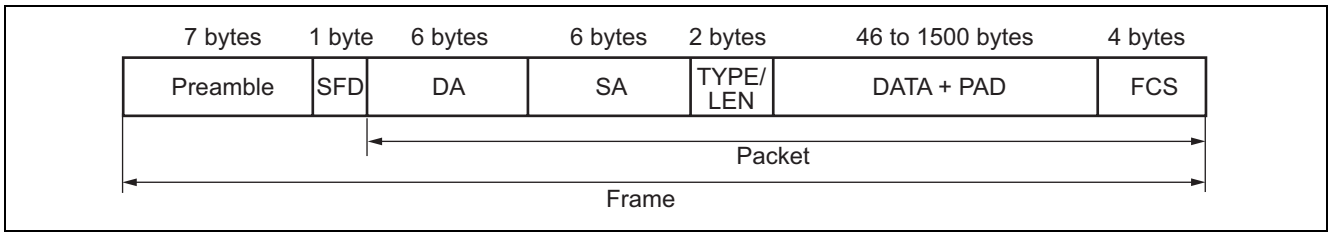
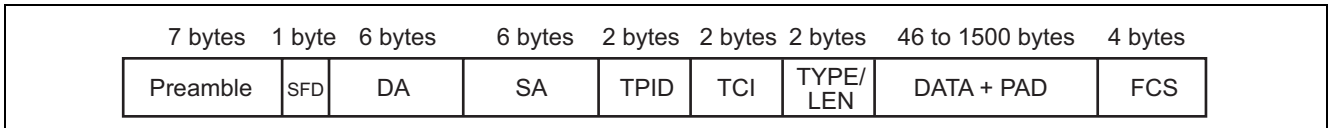


Figure 4.2 Basic Frame Structure

(2) VLAN Frame

The Ethernet controller has a VLAN frame detection function and if a transmit packet or a receive packet is detected as a VLAN frame, packet processing is performed based on that receive packet length.

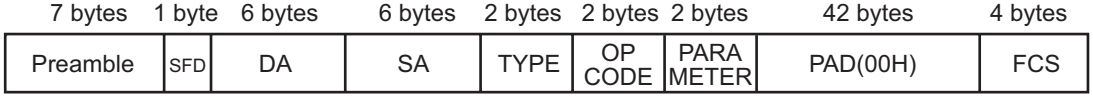


Notes: TPID: Tag protocol ID
TCI: Tag control information
TPCID + TCI: VLAN header

Figure 4.3 VLAN Frame Structure

(3) Pause Control Frame

The Ethernet controller has a function that automatically transmits pause control frames according to the amount of data remaining in the receive FIFO.



Notes: DA: Fixed address (01-80-C2-00-00-01H)
TYPE/LEN: Type of MAC control frame (88-08H)
OPCODE: Pause Op-code (00-01H)
PARAMATER: Pause command parameter (set with the ETHA0PAUSETM register)
PAD: All bits are filled zero (00H)

Figure 4.4 Pause Control Frame Structure

4.1.3 Ethernet Controller Transmit Block Overview

The Ethernet controller transmit function generates transmit frames that conform to the IEEE 802.3 standard from the transmit packet data stored in the transmit FIFO by DMA transfers using the dedicated Ethernet controller DMAC and outputs those frames to the PHY. It uses a random backoff algorithm to retransmit when a collision is detected.

1. Transmit clock
   The Ethernet controller operates in synchronization with a transmit clock (TXCLK) supplied externally (from the PHY). Transmit packet data stored in the transmit FIFO by DMA transfers is output to the PHY synchronized with TXCLK internally to the FIFO.

2. Carrier sense signal (CRS)
   In half-duplex communication, the Ethernet controller finishes storing the transmit data in a FIFO and then when transmission is possible, if a carrier is detected (CRS = 1), delays transmission until the carrier terminates (CRS = 0). After the carrier terminates, the Ethernet controller waits until the inter-packet gap (IPG) count set by the ETHA0IPGT register has completed and then starts transmission.

3. Collision detection (COL) and retransmission
   In half-duplex communication, when the Ethernet controller detects a collision, it stops transmission after transmitting the jam signal (error CRC data). If a collision within the collision window is detected within the maximum collision detection count (initial value: 15 times), the Ethernet controller performs a transmission standby according to the random backoff algorithm and then retransmits the data in the transmit FIFO.

4. Inter-packet gap (IPG)
   The Ethernet controller starts an IPG count after it or some remote station completes a transmission. After a transmission of its own, if a next transmission request is indicated by the FIFO before the IPG count reaches the value of the ETHA0IPGT register, it recognizes back-to-back (continuous) transmission and immediately starts transmission after the count completes.

5. Preamble, CRC, and pad addition
   The Ethernet controller adds a 7-byte preamble and 1-byte start of frame delimiter (SFD) in front of the transmit packet supplied from the FIFO.
   If the ETHA0MACC1.CRCEN bit is set to 1, an internally generated frame check sequence (FCS) is added at the end of the transmit packet.

6. Transmit abort
   If one of the following conditions occurs, the Ethernet controller will abort the transmission.
   Note that within the range of normal usage, the Ethernet controller will not generate an abort due to transmit FIFO underrun.
   — Collisions that exceed the maximum number of collisions (MAX collisions)
   — Collisions outside the collision window (rate collisions)
   — Excessive transmission delay
   — An attempt to transmit a packet whose length exceeds the frame length set in the ETHA0LMAX register. (Note, however, that if the ETHA0MACC1.HUGEN bit is set to 1, there is no limit on the transmit frame length.)

7. Full-duplex operation
   Full-duplex operation is possible if the ETHA0MACC1.FULLD bit is set to 1. The IPG will always be the value set with the ETHA0IPGT register.

8. Flow control function and back pressure function
   The Ethernet controller provides back pressure and flow control functions linked to the receive FIFO. These functions operate automatically when the amount of space remaining in the receive FIFO becomes too small and work powerfully to prevent receive FIFO overflow.

4.1.4 Ethernet Controller Receive Block Overview

The Ethernet controller generates a receive packet for the FIFO from the receive frame and performs a variety of functions including SFD detection, length filed check, FCS check, and VLAN frame detection.

1. Receive block
   The Ethernet controller receives data in synchronization with the receive clock (RXCLK) supplied externally (from the PHY).

2. MII data reception
   The Ethernet controller, during periods when the RXDV signal is asserted, recognizes data on its signals as receive frames and sees the time when the RXDV signal is deasserted as the end of the frame.

3. Preamble and SFD detection
   The Ethernet controller detects the preamble and SFD at the start of the receive frame and takes the data that follows as the receive packet.

4. Length field check
   The Ethernet controller counts the length of the receive packet and sees the 2 bytes following the source address as the length field. Based on that field, it checks the length of the data field.

5. CRC check
   The Ethernet controller calculates a 4-byte frame check sequence (FCS) from the receive packet and compares it to the FCS data added to the end of the receive packet.

6. Transfer of data to the FIFO
   The Ethernet controller accepts packets of 6 or more bytes as valid and discards packets with less than 6 bytes.

7. Huge packet detection
   When the ETHA0MACC1.HUGEN bit is set to 0, the Ethernet will only receive packets with up to the maximum frame length (initial value: 1536 bytes) set in the ETHA0LMAX register. Packets that exceed that length will be discarded during reception.

8. VLAN frame detection
   The Ethernet controller checks every packet to determined whether or not it is a VLAN frame.
   If the value of the TPID field (the 2 bytes following the source address) in a receive packet matches the value set in the ETHA0VLTP register, the ETHAS0RXSTMONI.VLAN flag is set as a VLAN packet.

4.1.5 Dedicated Ethernet Controller DMAC Overview

The V850 family microcontrollers have a built-in dedicated DMAC directly connected to the Ethernet controller. This dedicated Ethernet controller DMAC uses descriptors to control most aspects of buffer management. This reduces the load on the CPU and allows highly efficient data transmission and reception.

The dedicated Ethernet controller DMAC is connected to the Ethernet controller and performs highly efficient transfers, without CPU intervention, with memory (data buffers). The dedicated Ethernet controller DMAC reads in itself control information that hold buffer pointers, called descriptors, that correspond to the various buffers. It reads transmit data from transmit data buffers and writes receive data to receive data buffers according to this control information. Transmission and reception operations can be performed sequentially by allocating multiple descriptors in series (in a descriptor list).

Table 4.1 lists the specifications of the dedicated Ethernet controller DMAC.

Table 4.1 Dedicated Ethernet Controller DMAC Specifications

Item	Description
DMA transfer modes	• Single transfer mode • 4-bit incremental burst transfer • 8-bit incremental burst transfer • 16-bit incremental burst transfer
Objects that can be accessed by DMA transfers	H bus shared memory

4.1.6 Descriptor Overview

Descriptors, to which the data buffer and transfer data storage addresses and other information have been written, are required for the Ethernet controller to perform DMA transfers.

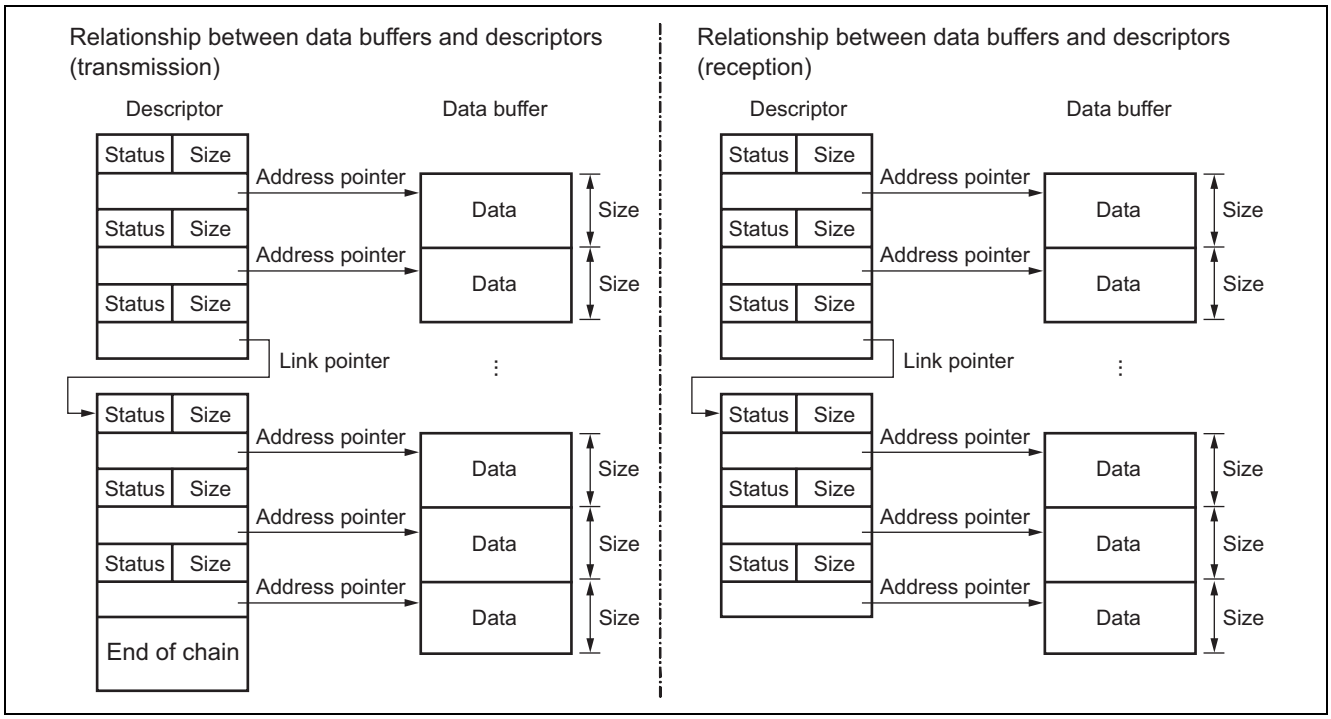


Figure 4.5 Relationship Between Data Buffers and Descriptors

4.1.7 Descriptor Types

The descriptor mechanism was adopted to support cases where the memory spaces that hold transmit data/receive data are not contiguous when using the Ethernet controller.

The Ethernet controller uses the following three types of descriptor.

- Buffer descriptor
- Link pointer
- End of chain

Each of these descriptors consists of two words (64 bits) of word-aligned data.

(1) Buffer Descriptor

The buffer descriptor format consists of two words (64 bits). The low-order word holds control bits and the high-order word holds the start address of the data buffer that the descriptor points to.

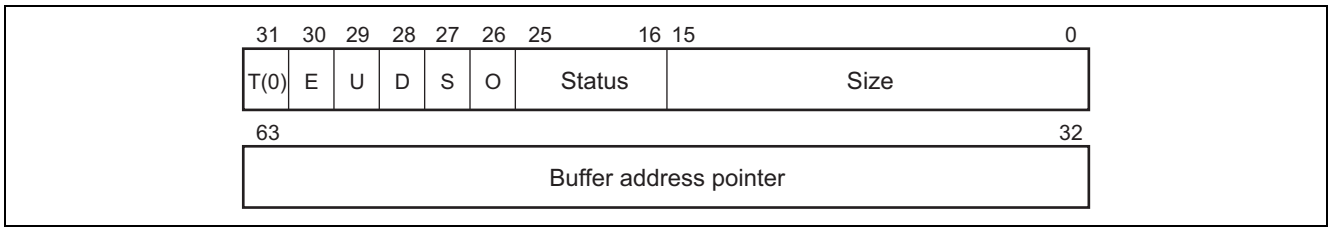


Figure 4.6 Buffer Descriptor Structure

Table 4.2 Buffer Descriptor Data Format

Bit Position	Symbol	Description
63 to 32	BAP	Address pointer that indicates the data buffer start address.
31	T	Descriptor type Indicates the descriptor type. For buffer descriptors, this bit is set to 0.
30	E	Last buffer flag Control bit that indicates the end of the packet data. 0: Indicates that this is normal buffer data (not the last data). 1: Indicates that this is the last data buffer for the current packet.
29	U	Used bit Indicates whether the DMA transfer is complete or not transferred (including transfer in progress). 0: Not transferred (including transfer in progress) 1: Transfer complete
28	D	Bit that indicates data buffer access errors 0: No error 1: A data buffer access error occurred.
27	S	Indicates whether the receive status information has been written to the Status field. 0: Status information is not included. 1: The receive packet status information is included. Only control bit S in the descriptor for the start of the packet during reception is valid. This bit is not used in transmission.
26	O	Reports overflows that occur during reception. 0: No overflow occurred. 1: An overflow occurred. This bit is not used in transmission.
25 to 16	Status	Indicates the reception status information. The Status field value is valid when control bit S is 1. This field is not used in transmission.
15 to 0	Size	Indicates the size (in bytes) of the buffer data pointed to by the descriptor.

(2) Link Pointer

The link pointer format consists of two words, The low-order word holds control bits and the high-order word holds the address of the next descriptor.

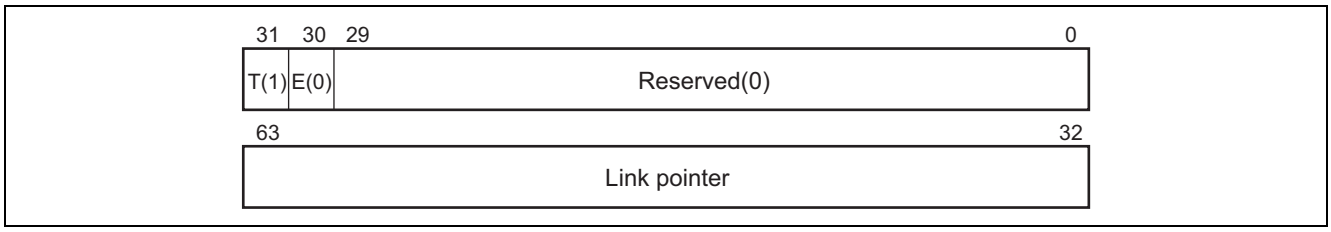


Figure 4.7 Link Pointer Structure

Table 4.3 Link Pointer Data Format

Bit position	Symbol	Description
63 to 32	Link Pointer	Indicates the address of the next descriptor.
31	T	In a link pointer, this bit must be 1.
30	E	In a link pointer, this bit must be 0.
29 to 0	Reserved	Reserved area

(3) End of Chain

The end of chain format consists of two words, the low-order word holds control bits and the high-order word is set to 0. When the Ethernet controller detects an end of chain, it terminates the DMA transfer and issues an RECI/TECI interrupt.

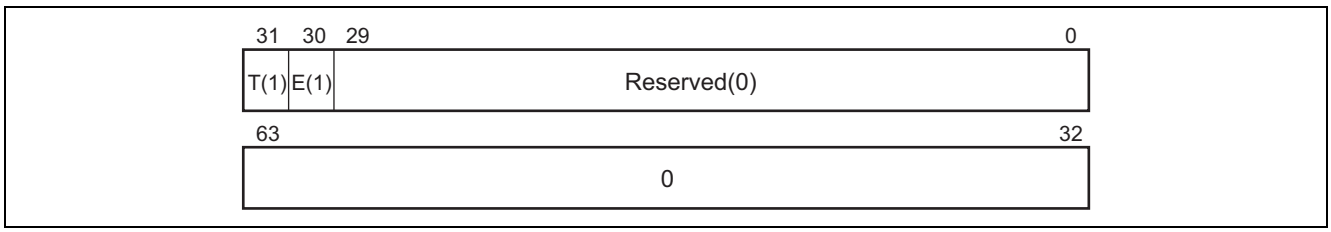


Figure 4.8 End of Chain Structure

Table 4.4 End of Chain Data Format

Bit position	Symbol	Description
63 to 32	BAP	In an end of chain, this field is set to NULL (all zeros).
31	T	In an end of chain, this bit must be 1.
30	E	In an end of chain, this bit must be 1.
29 to 0	Reserved	Reserved area

4.1.8 Descriptor Operation Overview

The Ethernet controller can process multiple descriptors sequentially in a single DMA transfer activation. Reception DMA transfer and transmission DMA transfer are started by setting the ETHA0RXDP register to the start address of a receive descriptor chain, setting the ETHA0TXDP register to the start address of a transmit descriptor chain, and setting the RXS and TXS bits in the ETHA0MODE register.

There must be an end of chain descriptor at the end of each descriptor chain.

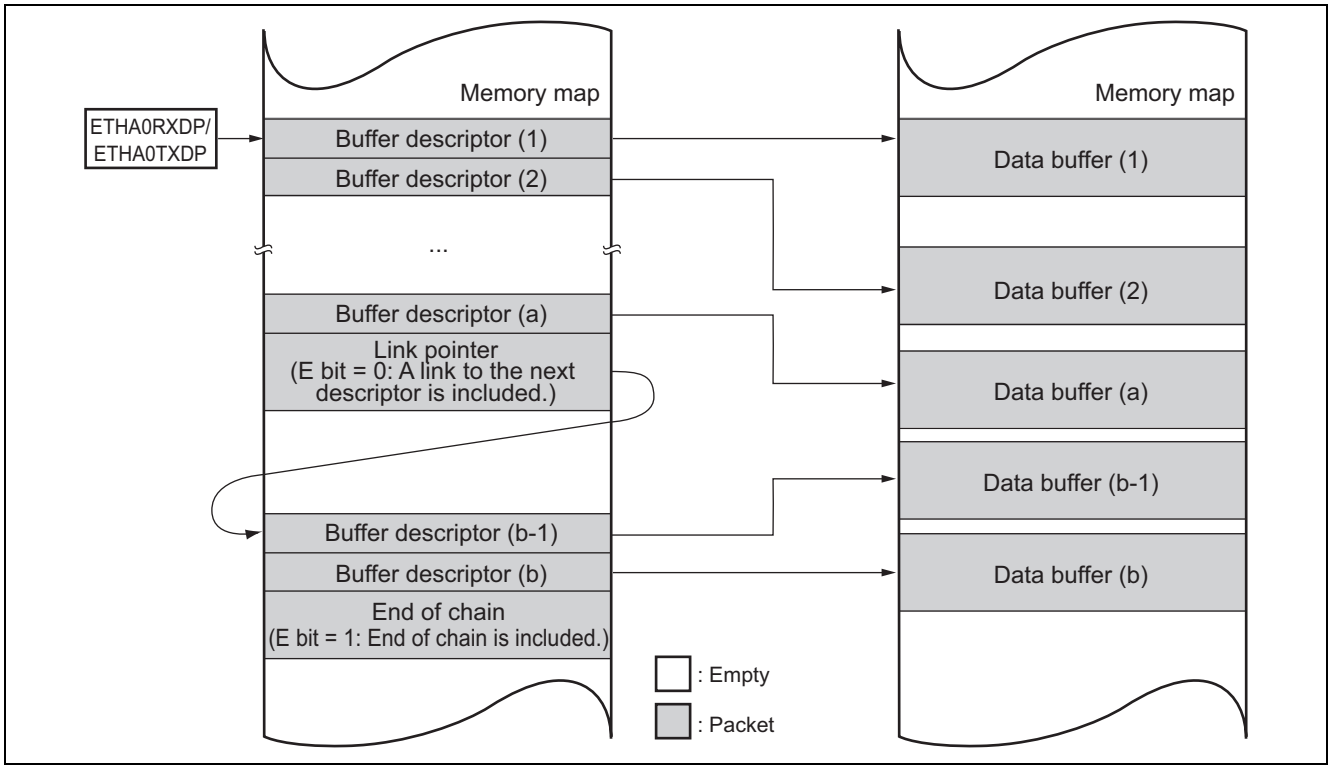


Figure 4.9 Relationship Between Descriptors and Data Buffers (for descriptor chains)

4.1.9 When a Descriptor Chain Is Set Up As a Ring Buffer

If a descriptor chain is set up as a ring buffer, the descriptors can be updated by reading the ETHA0LSTRXDP and ETHA0LSTTXDP registers using the INTSCTX internal TXI flag (or the INTSCRX interrupt RXI flag) as the trigger.

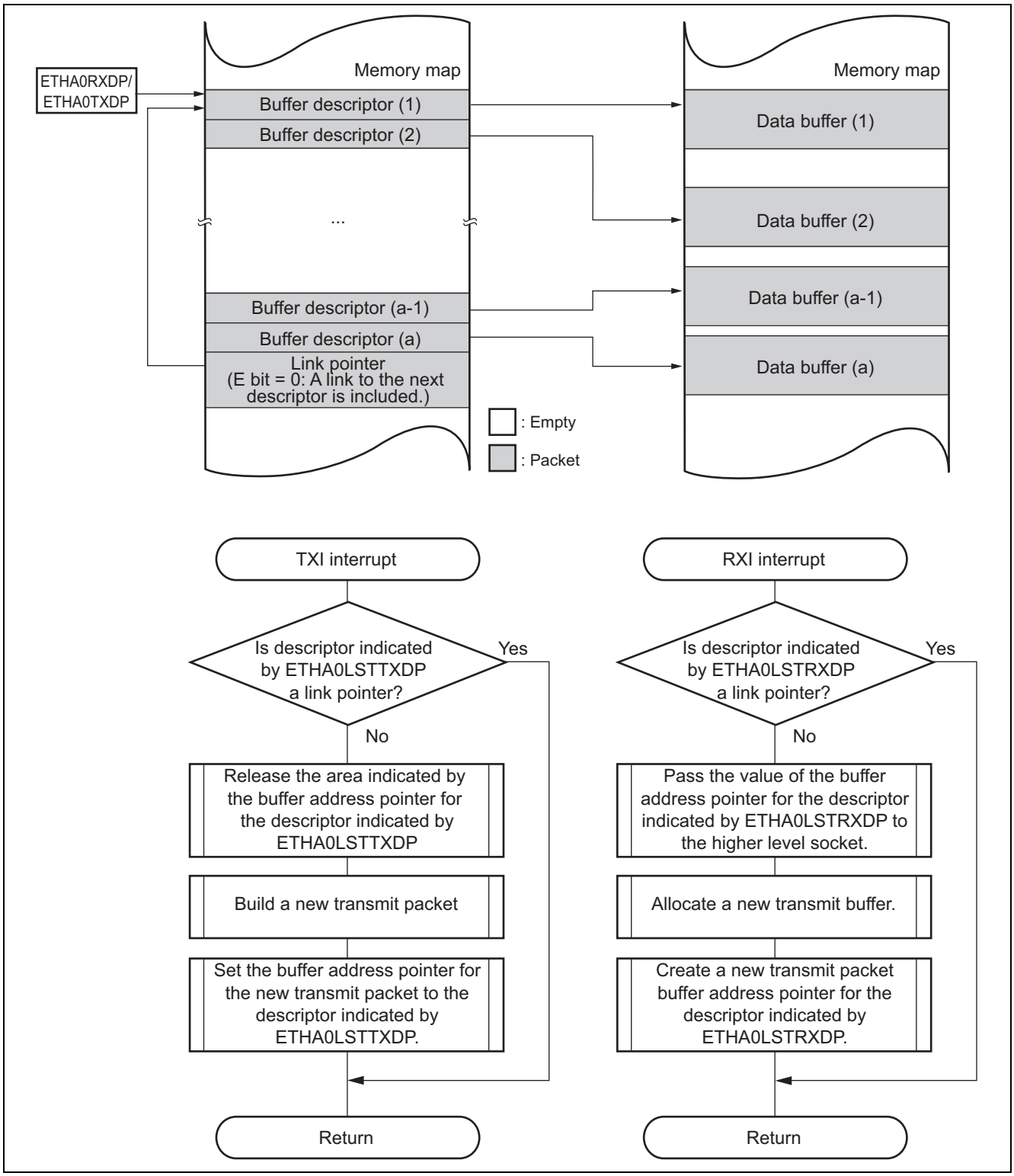


Figure 4.10 Relationship Between Descriptors and Buffers (ring buffer)

4.1.10 Transmit Descriptor Setup Example

Figure 4.11 shows a sample descriptor chain and its operation for the case where two transmit buffers are used.

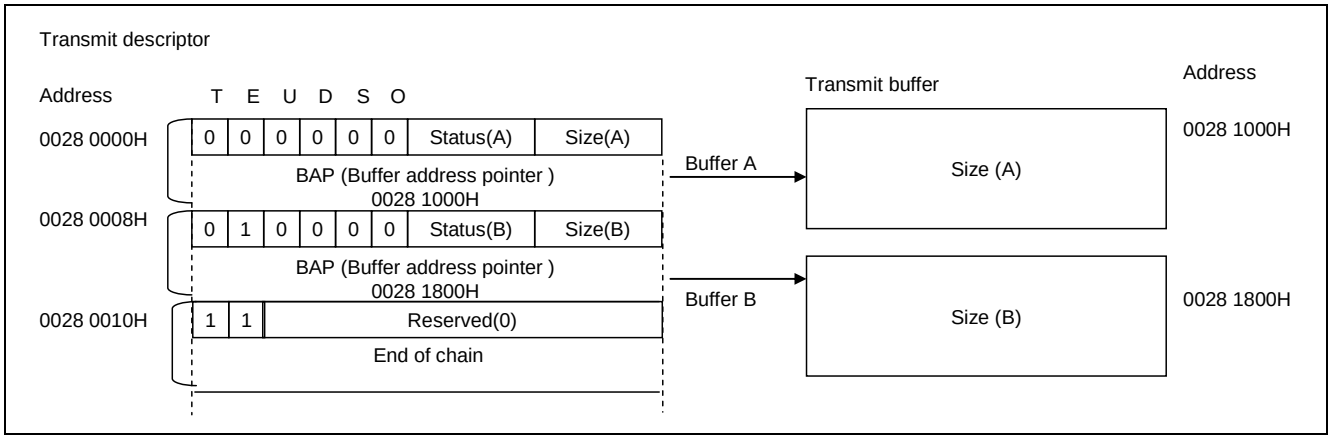


Figure 4.11 Transmit Descriptor Setup Example

1. Set the core function settings register (ETHA0MODEETHA0MODE) TXS bit to 1 in software.
2. The Ethernet controller will then read the starting descriptor from the address (0028 0000H) indicated by the transmit descriptor pointer (ETHA0TXDP).
3. The DMA transfer start address is set to the buffer address pointer (0028 1000H) and the data in the buffer is transferred to the FIFO.
4. Since the transmit descriptor E bit is not 0, it indicates that this is not the last data. Therefore, the next buffer descriptor (0028 0008H) is read, the buffer address pointer is set (0028 1800H), and the data in the buffer is transferred to the FIFO.
5. When the transfer of the data in a buffer pointed to by a buffer descriptor whose E bit is 1 completes, the U bit is set and transfer processing completes. Also, an interrupt request (TXI) is generated.

4.1.11 Receive Descriptor Setup Example

Figure 4.12 shows a sample descriptor chain and its operation for the case where two receive buffers are used.

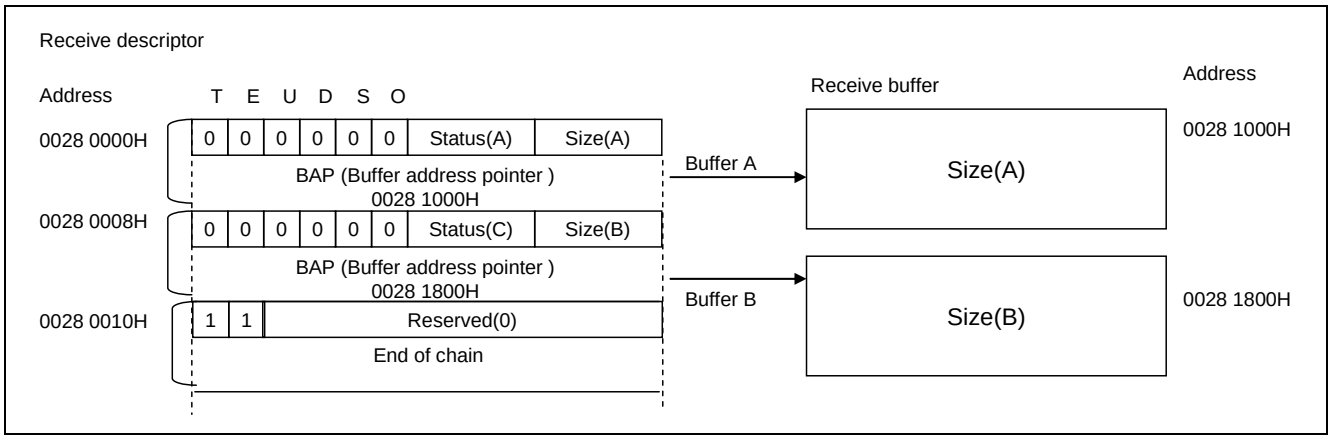


Figure 4.12 Receive Descriptor Setup Example

1. Set the core function settings register (ETHA0MODEETHA0MODE) RXS bit to 1 in software.
2. The Ethernet controller will then read the starting descriptor from the address (0028 0000H) indicated by the receive descriptor pointer (ETHA0RXDP).
3. The Ethernet controller sets the DMA transfer start address to the first buffer address pointer (0028 1000H) and transfers the receive data in the FIFO to buffer A.
4. Then, when receive buffer A becomes full, the next descriptor (0028 0008H) is read, the DMA transfer start address is set to the buffer analysis pointer (0028 1800H), and the receive data in the FIFO is transferred to buffer B.
5. For the last descriptor, the E and U bits are set to 1 and the number of bytes in the transferred data is written back to the Size field. Also, after all packet data has been transferred, the start descriptor U and S bits are set to 1 and the receive data status information is written back to the Status (A) field.

4.1.12 Frame Transmission Procedure

When the CPU prepares transmit descriptors and transmit data as the transfer object for the dedicated Ethernet controller DMAC, sets the transmit descriptor register (ETHA0TXDP), and sets the TXS bit in the ETHA0MODEETHA0MODE register, the dedicated DMAC fetches a transmit buffer descriptor from the address set in the descriptor register, reads the transmit data from the data buffer and transfers it to the transmit FIFO.

The data transmitted to the FIFO is output to the PHY in synchronization with TXCLK in the order preamble, SFD, and then frame data.

If the CRCEN bit in the ETHA0MACC1 register is set, an FCS field is added after the data.

If the PADEN bit in the ETHA0MACC1 register is set, a PAD field is automatically added during short frame transmission.

If the currently indicated descriptor does not include the end of the frame, the next descriptor is read and the data from the data buffer specified by that descriptor is read out.

After transmission completes, the transmit status is written to the last descriptor. After that, the next transmit buffer descriptor is fetched and if the next data transmission operation is possible, transmission using the same method starts.

Also, an interrupt (TXI) that indicates the completion of DMA transmission is issued after DMA processing of each buffer descriptor completes.

If the next transmit buffer descriptor is an end of chain descriptor, the interrupt (TECI) that indicates end of chain is generated and the DMA transmission operation terminates.

To start another DMA transmission, set up the ETHA0TXDP register and the buffer descriptors again.

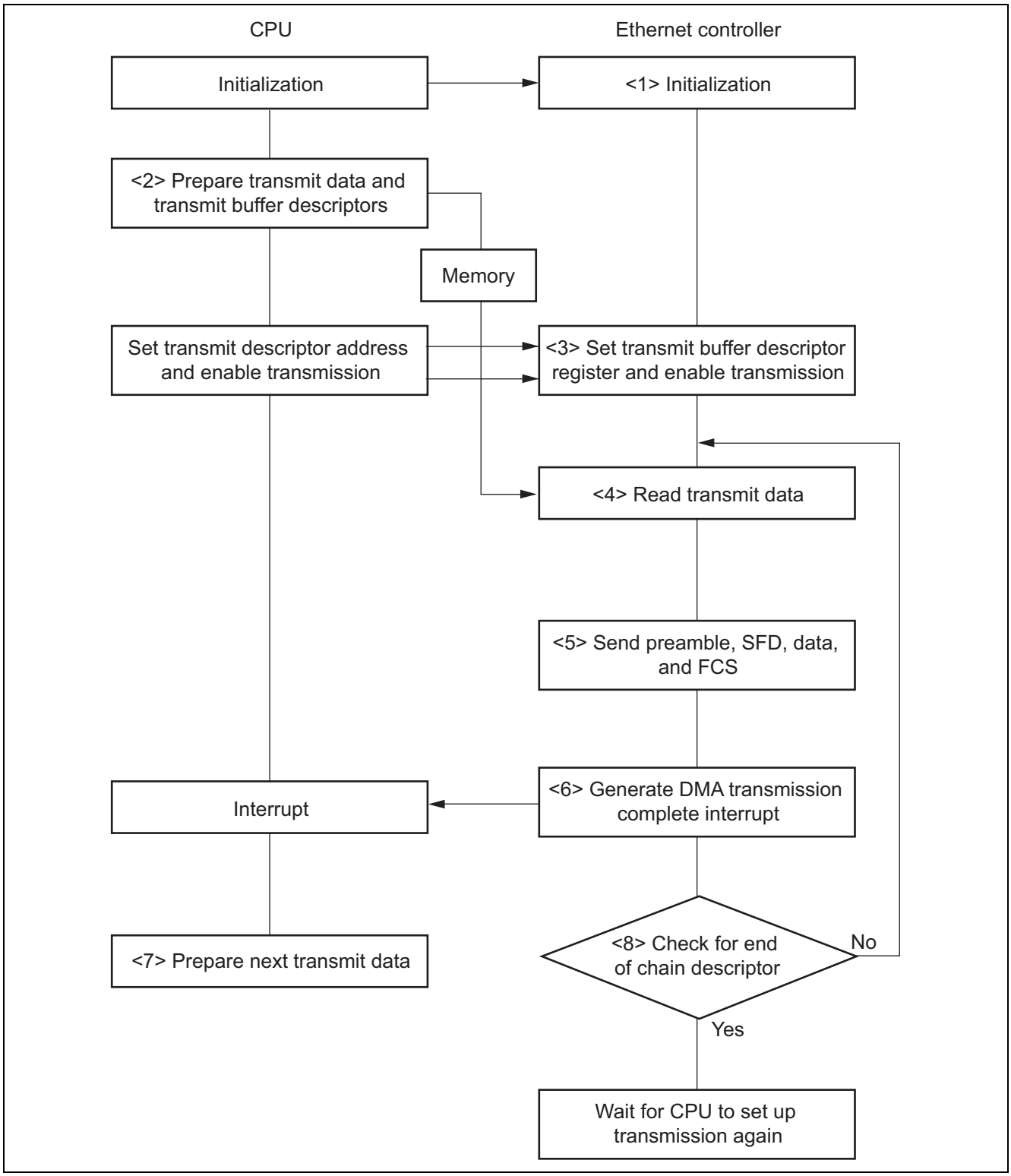


Figure 4.13 Frame Transmission Procedure

4.1.13 Frame Reception Procedure

After the SRXEN (receive enable) bit in the MAC configuration register (ETHA0MACC1) is set, the RXS bit (DMA reception enable) in the ETHA0MODEETHA0MODE register is set, and the receive descriptor pointer register ETHA0RXDP is set, receive frame processing will start as soon as the MAC receives data.

When the MAC receives data, whether the preamble and frame start delimiter (FSD) are valid is checked.

If the preamble and SFD are valid, the received frame is processed.

If a valid preamble and SFD are not found, the frame is ignored.

If a collision occurred or if the frame is discarded by address filtering, no data will be written to the receive buffer.

A receive frame that is received correctly and not discarded by address filtering is transferred to the data buffer specified by the receive buffer descriptor.

The Ethernet controller checks whether or not the frame length is appropriate during reception.

At the end of the frame, the FCS is checked and written to the buffer descriptor. Note that frames less than 64 bytes (short packets) will be transferred by DMA.

When frame reception completes, the E and U bits in the last descriptor are set to 1 and the number of bytes of data transferred is written back to the Size field. Also, after all the packet data has been transmitted, the start descriptor U and S bits are set to 1 and the receive status information is written back to the Status field. An interrupt request (RXI) is also generated.

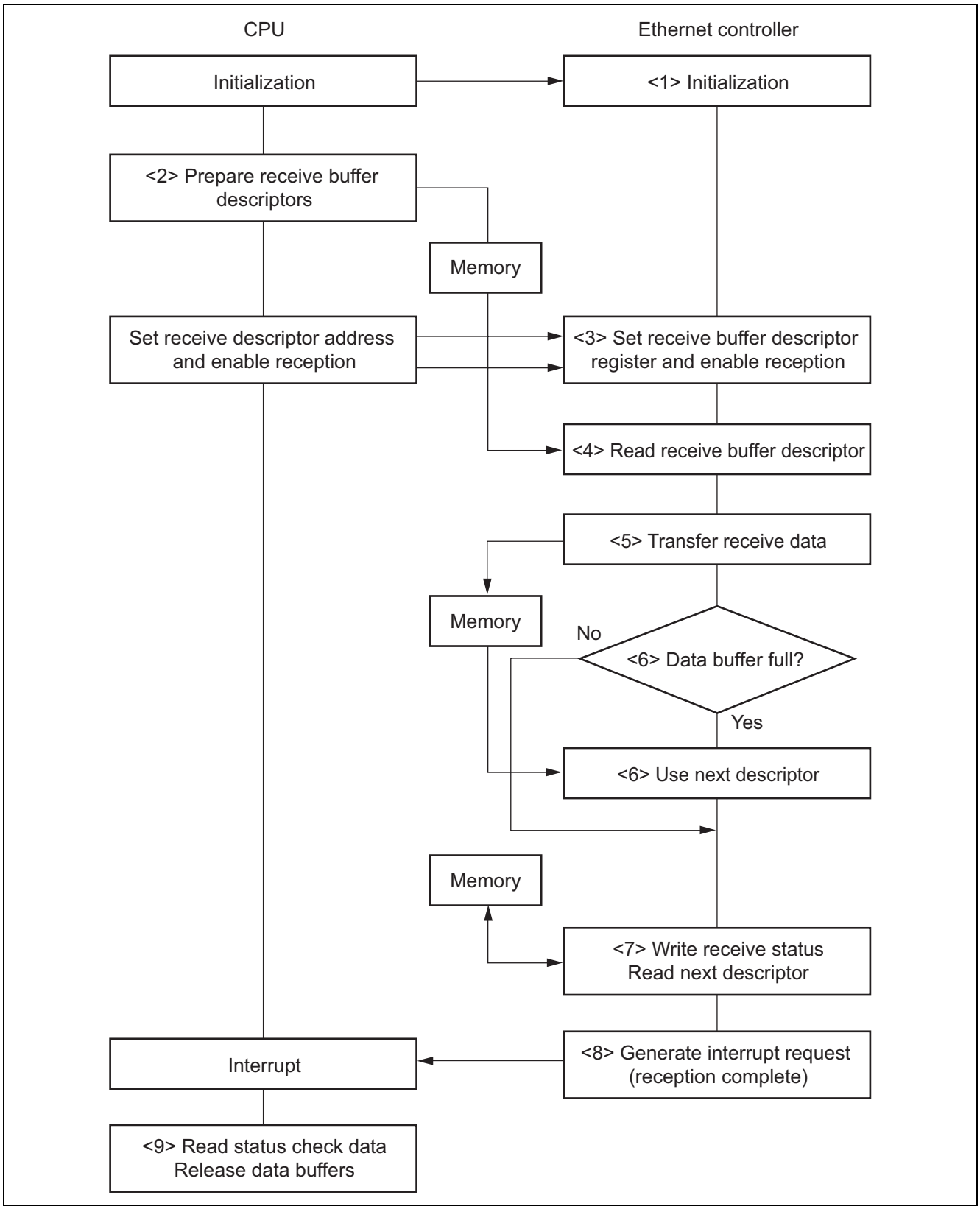


Figure 4.14 Frame Transmission Procedure

4.2 Sample Program Operation

The sample program performs the following two types of processing depending on the test type selected in the main routine.

- Transmission of 10 Ethernet frames
- Reception of 10 Ethernet frames

4.2.1 Sample Program Operation (Transmission)

When the transmission test is selected, the sample program uses the Ethernet controller and the dedicated Ethernet controller DMAC to transmit 10 frames to the remote host.

The sample program provides five transmit descriptors and four 1520-byte transmit buffers (data buffers). The transmit descriptors are used in a ring configuration.

Of the five transmit descriptors, the first to fourth are used as buffer descriptors and the fifth is used as a link pointer.

After writing 10 frames of transmit data to the transmit buffers, the sample program recognizes 10 frames of transmission as having completed from the transmit status but (TREN_STA) in the transfer control register (STHA0TRANSCTL) and terminates the test.

4.2.2 Sample Program Operation (Reception)

When the reception test is selected, the sample program uses the Ethernet controller and the dedicated Ethernet controller DMAC to receive 10 frames from the remote host.

The sample program provides four receive descriptors and three 1520-byte receive buffers (data buffers).

Of the four receive descriptors, the first to third are used as buffer descriptors and the fourth is used as a link pointer.

The sample program checks the U, S, and O bits (bits 29, 27, and 26 in the buffer descriptor) in the receive descriptor and if there were no errors and furthermore the Size field (bits [15:0] in the buffer descriptor) is a value other than 0, it reads out the receive data from the data buffer indicated by the receive descriptor.

The sections of the Ethernet frame other than the preamble, SFD, and CRC are transferred to the receive buffer.

4.2.3 Sample Program Operating Environment

Figure 4.15 shows the sample program's operating environment.

See section 4.7.1, Operating Environment Note 1, and section 4.7.2, Operating Environment Note 2, for notes on the operating environment.

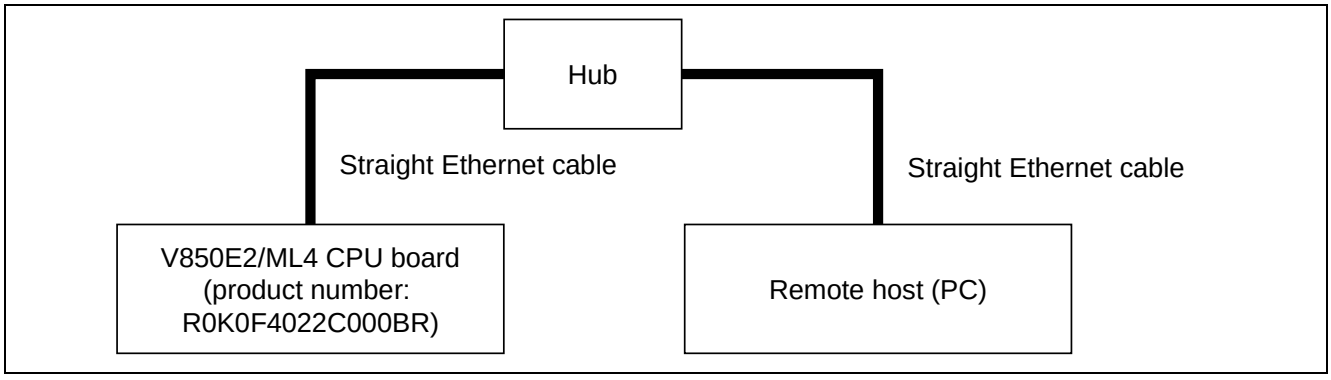


Figure 4.15 Sample Program Operating Environment

4.2.4 Ethernet Frame Format

For transmit data, the application must provide the components of the Ethernet frame other than the preamble, start of frame delimiter (SFD), and CRC sections. The destination MAC address and source MAC address in the header block must be changed to the MAC addresses of the products used. Note that the Ethernet controller does not check the source MAC address.

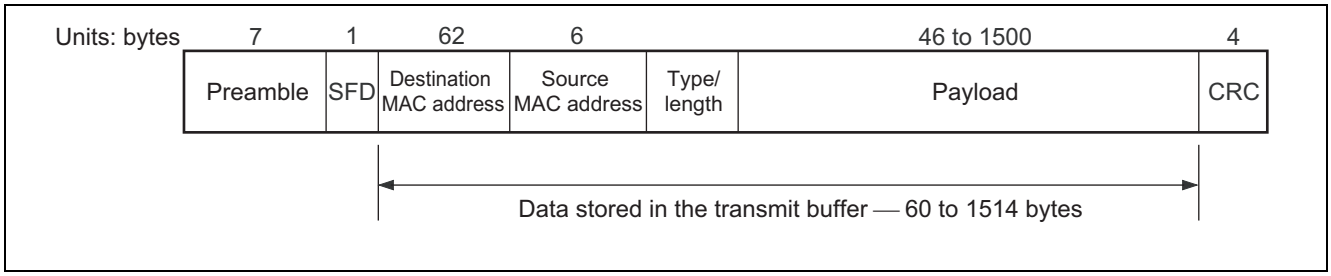


Figure 4.16 Ethernet Frame Format (transmission)

The components of the Ethernet frame other than the preamble, start of frame delimiter (SFD), and CRC sections.

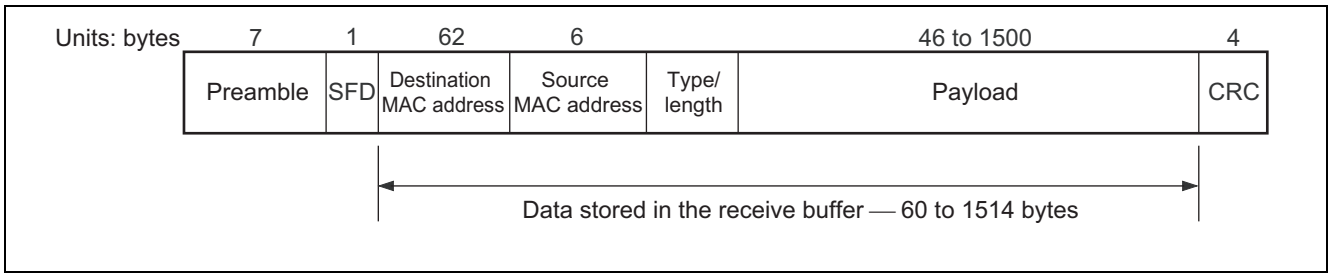


Figure 4.17 Ethernet Frame Format (reception)

4.3 Sample Program Descriptor Definitions

In the V850E2/ML4, the descriptors and data buffers are allocated in the H bus shared memory area.

The sample program sets the next descriptor start addresses to be in this area and implements a ring structure in software.

Figure 4.18 shows the transmit and receive descriptors and data buffers used in the sample program.

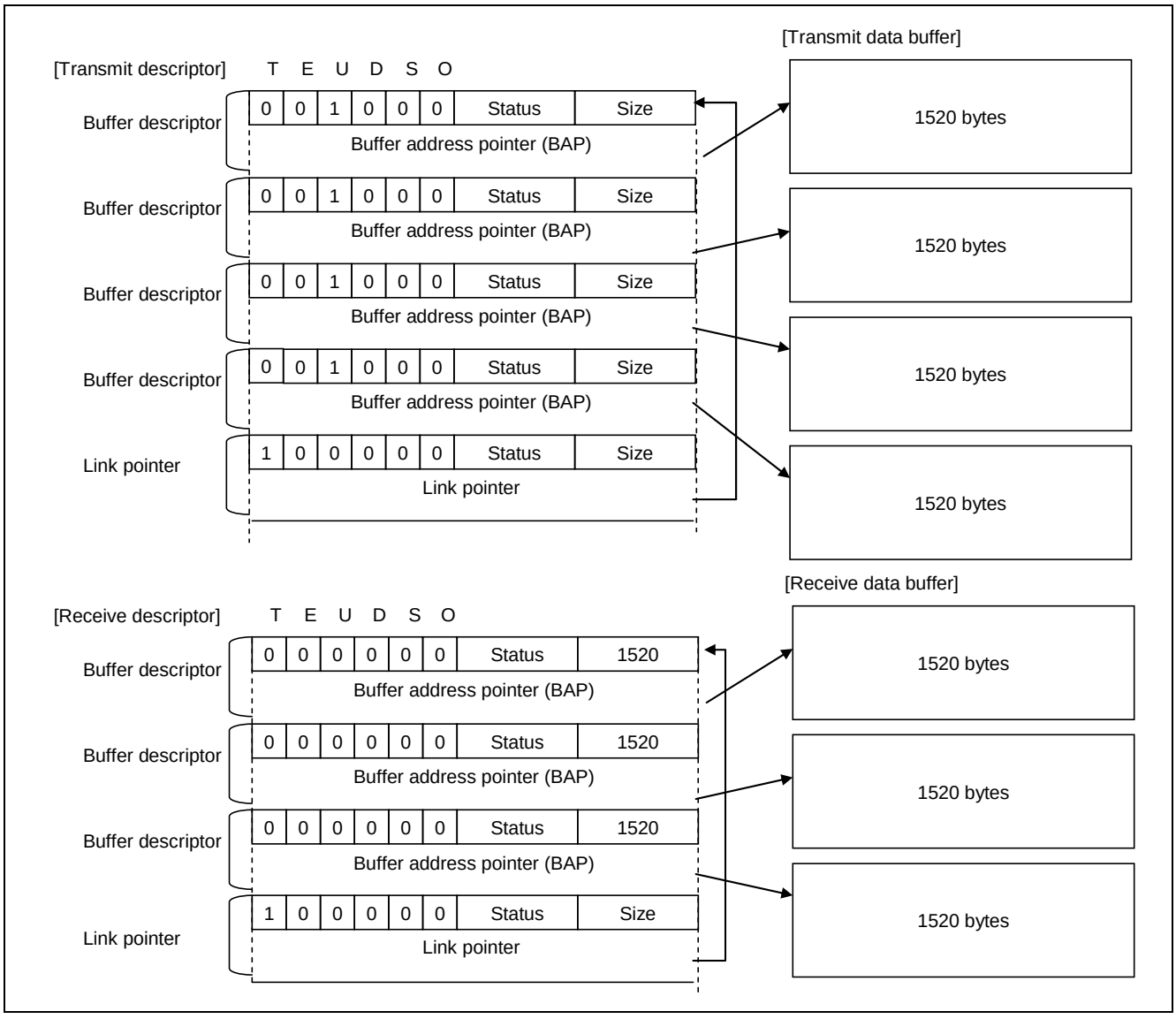


Figure 4.18 Descriptor Settings in the Sample Program

The descriptors in the sample program are defined as structures as shown below using macros defined by the compiler.

```
typedef struct tst_ctrl_desc {
 uint32_t value;            /* descriptor 31- 0 bit          */
 uint8_t* addr;             /* descriptor 63-32 bit          */
} CTRL_DESC;

typedef struct tst_ctrl_buf {
 CTRL_DESC*  tx_next;       /* top address of active TX DMA descriptor   */
 CTRL_DESC*  tx_write;      /* write address of active TX DMA descriptor */
 uint8_t     tx_next_buf;   /* write address of active TX DMA buffer     */
 CTRL_DESC*  rx_read;       /* read address of active RX DMA descriptor  */
} CTRL_BUF;
```

The number of descriptors and the data buffer size can be changed by modifying the following macros defined in ether_driver.h. FRAME_SIZE is the data buffer size, TX_DESCRIPTOR_MAX is the number of transmit descriptors, and RX_DESCRIPTOR_MAX is the number of receive descriptors

```
#define  FRAME_SIZE        ( 1518 )
#define  TX_DESCRIPTOR_MAX ( 5 )
#define  RX_DESCRIPTOR_MAX ( 4 )
```

The data buffer size specifies the size excluding the checksum (2 bytes).

4.4 Interrupts Used by the Sample Program

The sample program uses the transmit (DMA) end of chain interrupt (TECI), which is an Ethernet packet transmit interrupt (INTETMTX). When the TECI interrupt occurs, the transmit descriptor is returned to the start and if there is transmit data, transmission starts.

4.5 Ethernet Driver API

The following functions are provided as a TCP/IP stack driver interface. These form the standard Renesas API (RAPI) for the Renesas Ethernet device.

- R_Ether_Open
- R_Ether_Close
- R_Ether_Read
- R_Ether_Write

4.5.1 R_Ether_Open()

The R_Ether_Open() function initializes the Ethernet controller, the dedicated Ethernet controller DMAC, and the transmit/receive data buffers.

- Prototype
  int32_t R_Ether_Open(uint32_t ch, uint8_t mac_addr[]);

- Arguments
  — ch
  Specifies the Ethernet controller channel
  — mac_addr
  Specifies the Ethernet controller's MAC address.

- Return value
  R_ETHER_OK (0): Normal completion
  R_ETHER_ERROR (-1): An error occurred.

- Properties
  Declared in the file ether_driver.h.
  Defined in the file ether_driver.c.

- Description
  The R_Ether_Open() function initializes the Ethernet controller and the dedicated Ethernet controller DMAC. The Ethernet controller descriptors and data buffers are set up in their initial states. The MAC address is used to initialize the Ethernet controller MAC address.
  The PHY IC is set to autonegotiation mode by this initialization.
  The V850E2/ML4 has only one Ethernet channel and the Ethernet driver does not differentiate its processing based on the channel number. Although the Ethernet driver will operate correctly regardless of the channel number, we recommend always specifying a channel number of 0 when using this Ethernet driver.
  The V850E2/ML4 Ethernet driver does not implement any processing for the case where the MAC address is 0. The user must either use a value other than 0 for the MAC address or else add the processing required for the case where the MAC address is 0.

4.5.2 R_Ether_Close()

The R_Ether_Close() function disables the functions of the Ethernet controller and the dedicated Ethernet controller DMAC.

- Prototype
  int32_t R_Ether_Close(uint32_t ch);
- Arguments
  — ch
  Specifies the Ethernet controller channel
- Return value
  R_ETHER_OK (0): Normal completion
  R_ETHER_ERROR (-1): An error occurred.
- Properties
  Declared in the file ether_driver.h.
  Defined in the file ether_driver.c.
- Description
  The R_Ether_Close() function disables the functions of the Ethernet controller and the dedicated Ethernet controller DMAC.
  The V850E2/ML4 has only one Ethernet channel and the Ethernet driver does not differentiate its processing based on the channel number. Although the Ethernet driver will operate correctly regardless of the channel number, we recommend always specifying a channel number of 0 when using this function.

4.5.3 R_Ether_Read()

The R_Ether_Read() function receives data and stores it in an application receive buffer.

- Prototype
  int32_t R_Ether_Read(uint32_t ch, void *buf);
- Arguments
  — ch
  Specifies the Ethernet controller channel
  — *buf
  Pointer to a receive data buffer
- Return value
  A value greater than 0 specifies the number of bytes read. A value of 0 indicates that there was no receive data.
  R_ETHER_ERROR (-1): An error occurred. (This includes both hardware and software errors.)
  R_ETHER_HARD_ERROR (-3): A hardware error occurred. (A software reset is required for recovery.)
  R_ETHER_RECOVERABLE (-4): A recoverable error occurred. (A software reset is not required for recovery.)
  R_ETHER_NODATA (-5): There was no receive data.

  Note: This sample progam does not use the R_ETHER_HARD_ERROR (-3), R_ETHER_RECOVERABLE (-4), and R_ETHER_NODATA (-5) return values.
- Properties
  Declared in the file ether_driver.h.
  Defined in the file ether_driver.c.

- Description
  The R_Ether_Read() function reads data from the buffer pointed to by the receive descriptor. The data read is copied to the receive data buffer.
  The V850E2/ML4 has only one Ethernet channel and the Ethernet driver does not differentiate its processing based on the channel number. Although the Ethernet driver will operate correctly regardless of the channel number, we recommend always specifying a channel number of 0 when using this function.
  This function discards the data for a descriptor for which a receive frame error occurred, clears the status, and continues reading.

4.5.4 R_Ether_Write()

The R_Ether_Read() function transmits data from the application's transmit buffer.

- Prototype
  int32_t R_Ether_Write(uint32_t ch, void *buf, uint32_t len);

- Arguments
  — ch
  Specifies the Ethernet controller channel
  — *buf
  Pointer to Ethernet to transmit
  — len
  Length of the Ethernet frame

- Return value
  R_ETHER_OK (0): Normal completion
  R_ETHER_ERROR (-1): An error occurred.

- Properties
  Declared in the file ether_driver.h.
  Defined in the file ether_driver.c.

- Description
  The R_Ether_Write() function writes transmit data to the buffer specified by the transmit descriptor. The written data is transmitted by the Ethernet controller.
  The R_Ether_Write() function does not perform a transmit complete check.
  The V850E2/ML4 has only one Ethernet channel and the Ethernet driver does not differentiate its processing based on the channel number. Although the Ethernet driver will operate correctly regardless of the channel number, we recommend always specifying a channel number of 0 when using this function.
  This function does not check for transmit frame errors.

4.6 Sample Program Flowcharts

Figure 4.19 and 4.20 shows processing flow of the sample program that uses the Ethernet driver API and figure 4.21 to 4.27 shows the processing flow of the Ethernet driver API functions and the associated lower-level functions.

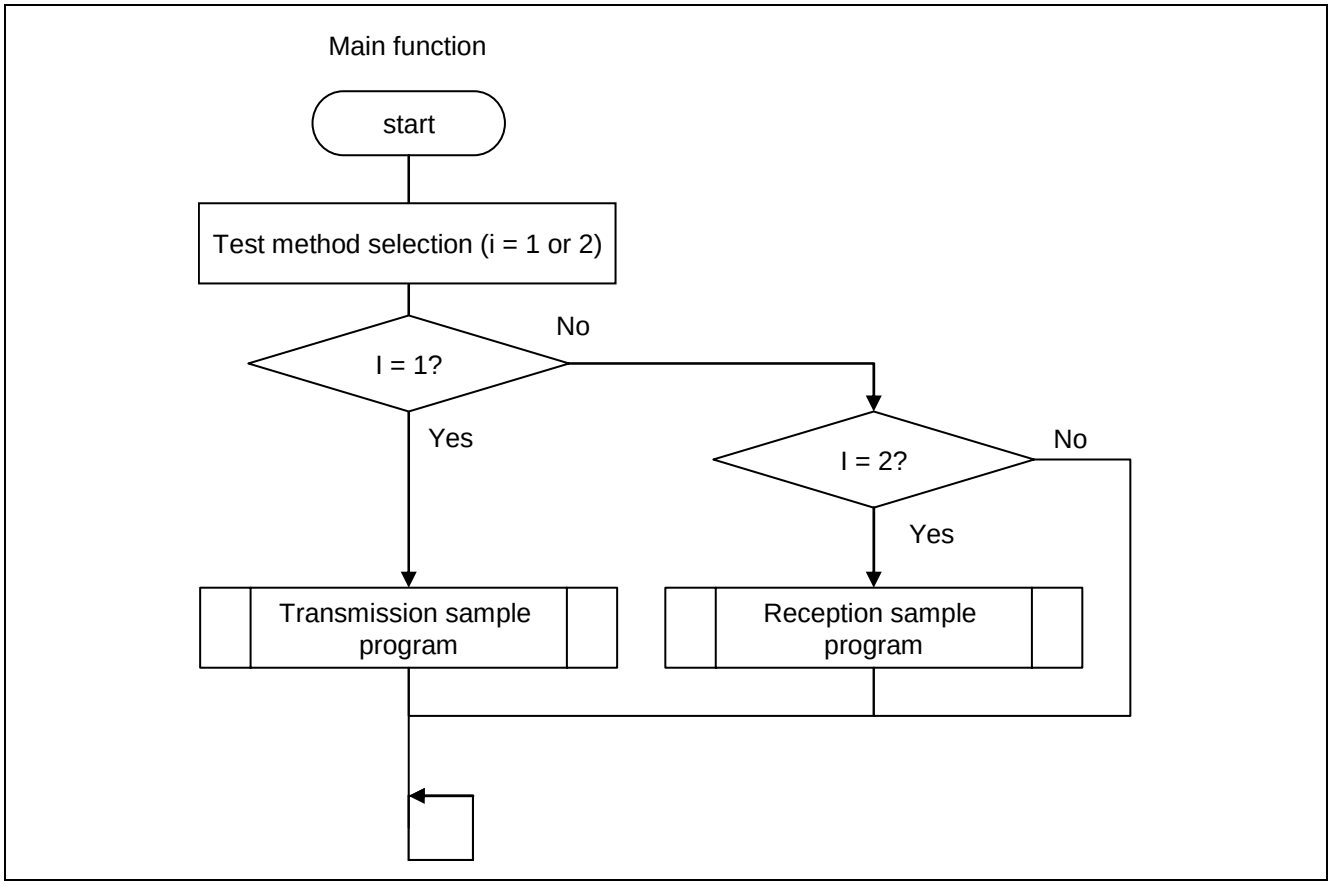


Figure 4.19 Sample Program Main Function Flowchart (1)

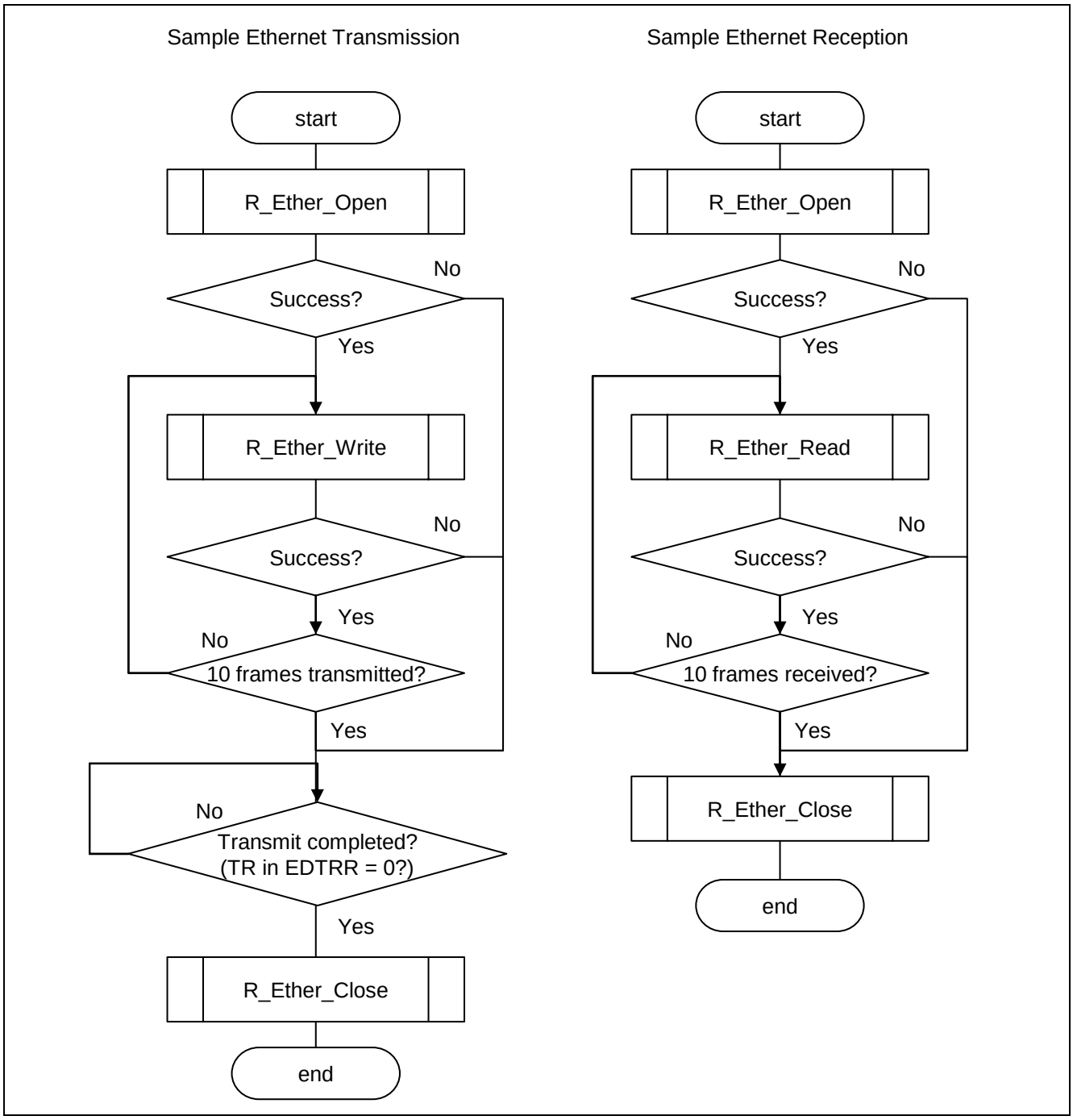


Figure 4.20 Sample Program Main Function Flowchart (2)

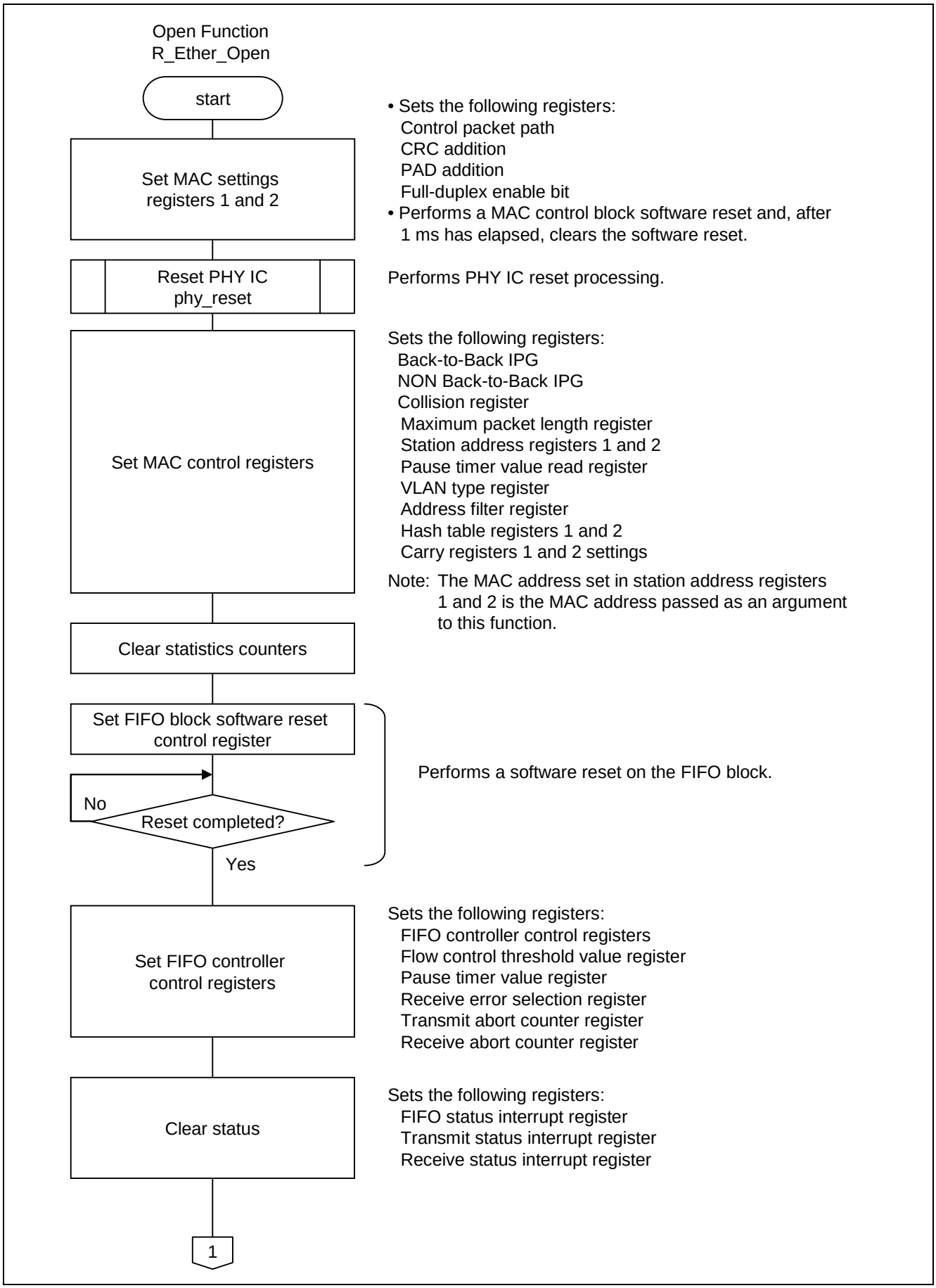


Figure 4.21 Sample Program Ethernet API Flowcharts (1)

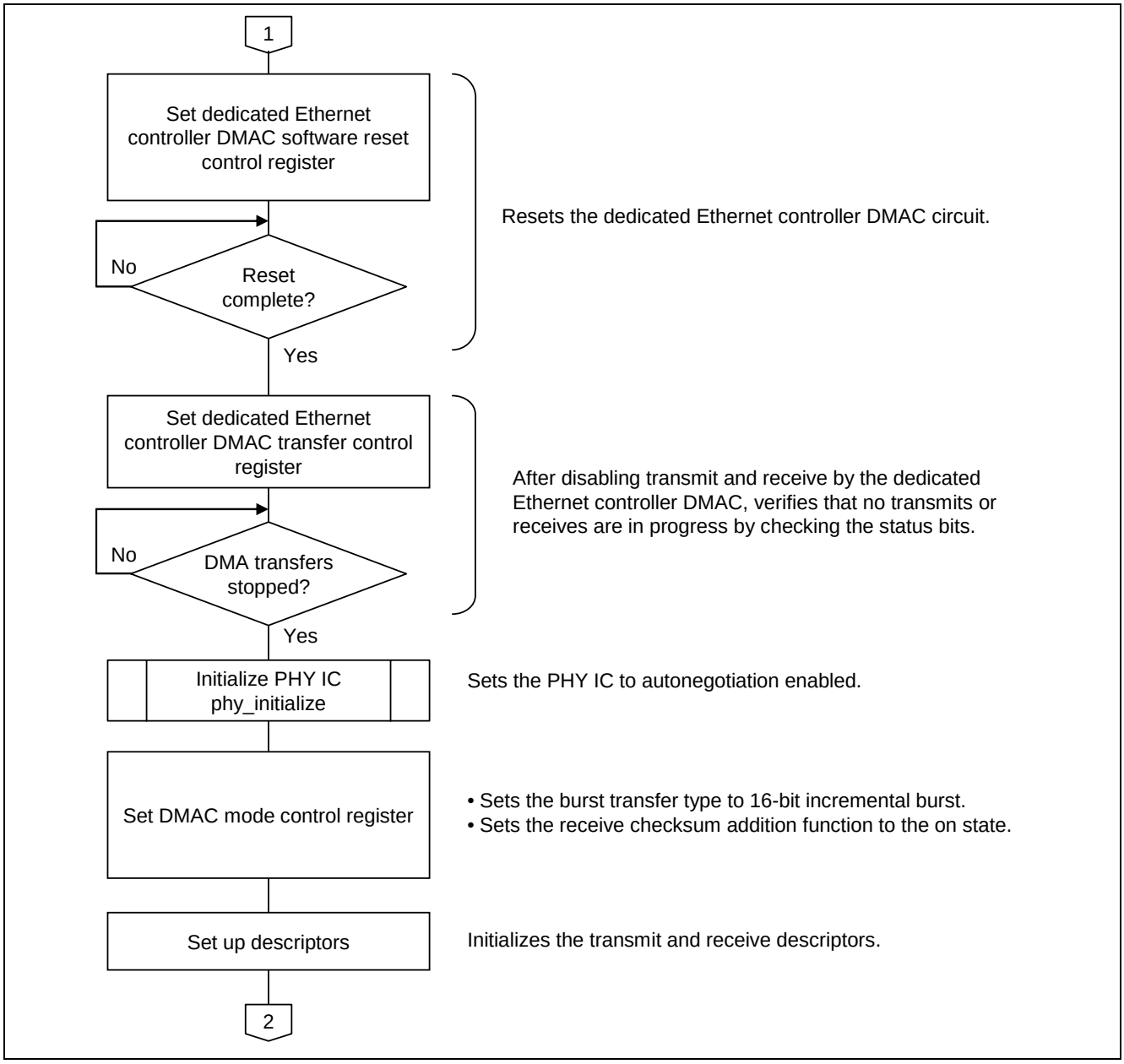


Figure 4.22 Sample Program Ethernet API Flowcharts (2)

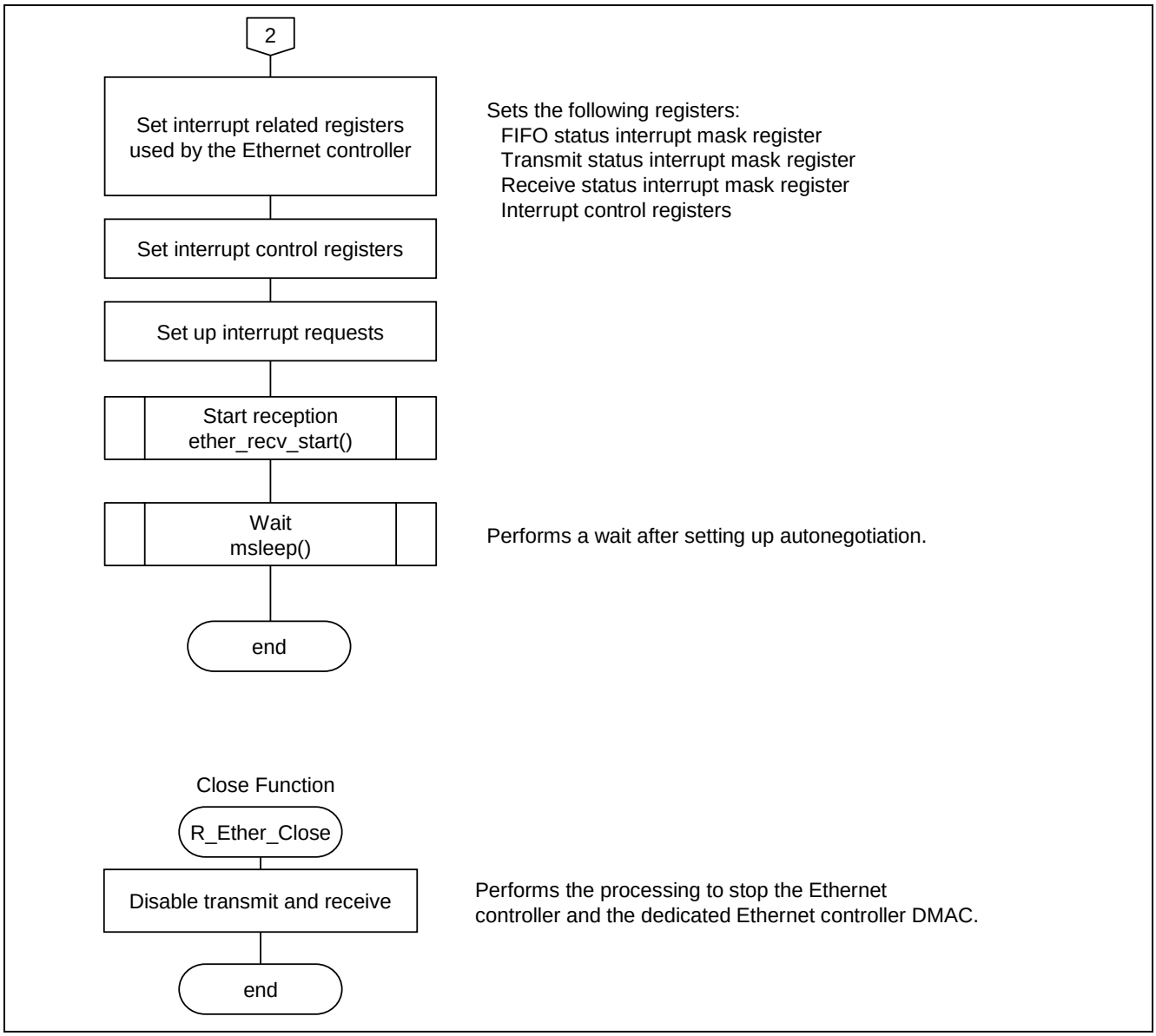
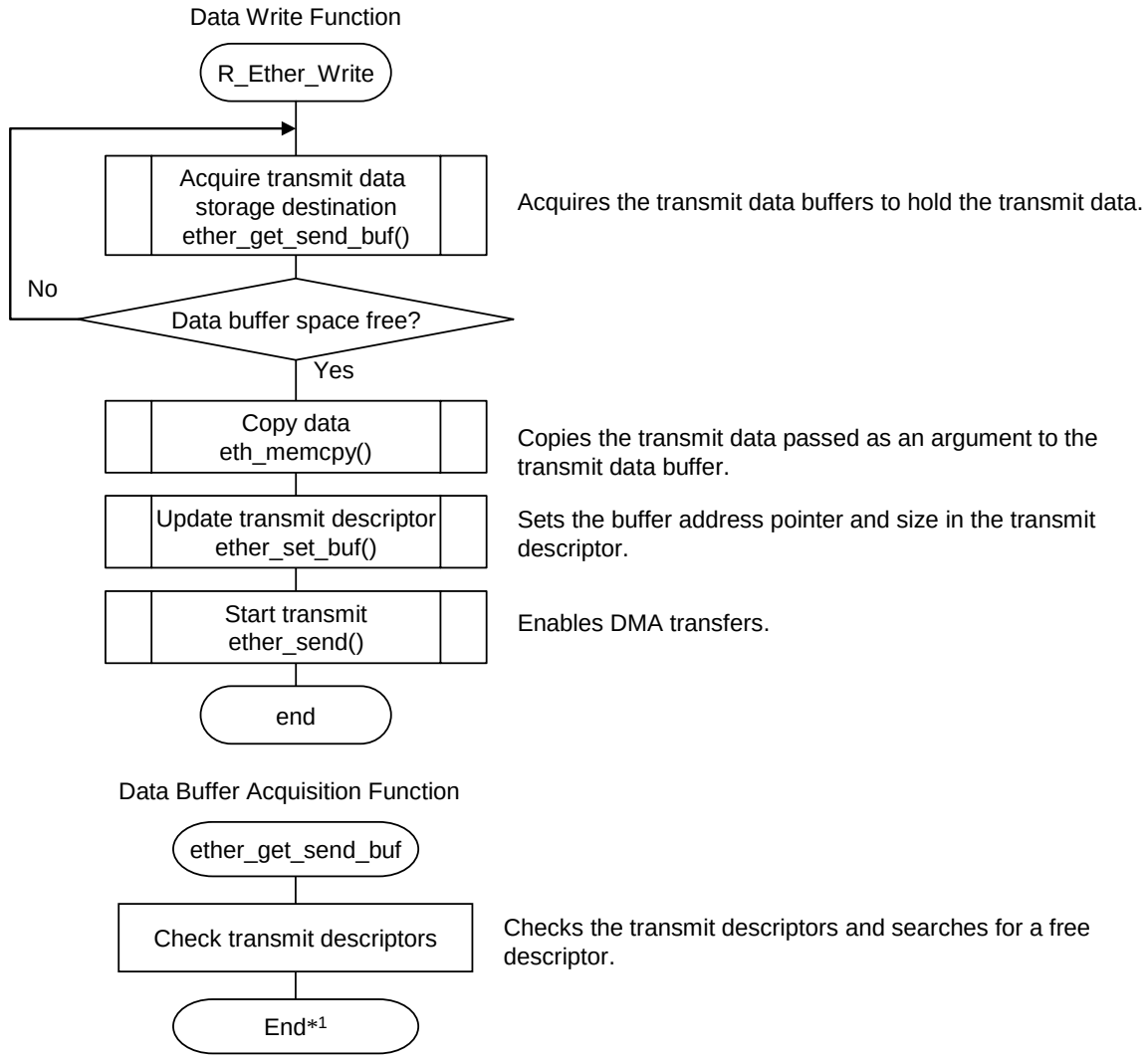


Figure 4.23 Sample Program Ethernet API Flowcharts (3)



Note: 1. Returns the DMA transfer target (transmit data buffer) as the return value.

Figure 4.24 Sample Program Ethernet API Flowcharts (4)

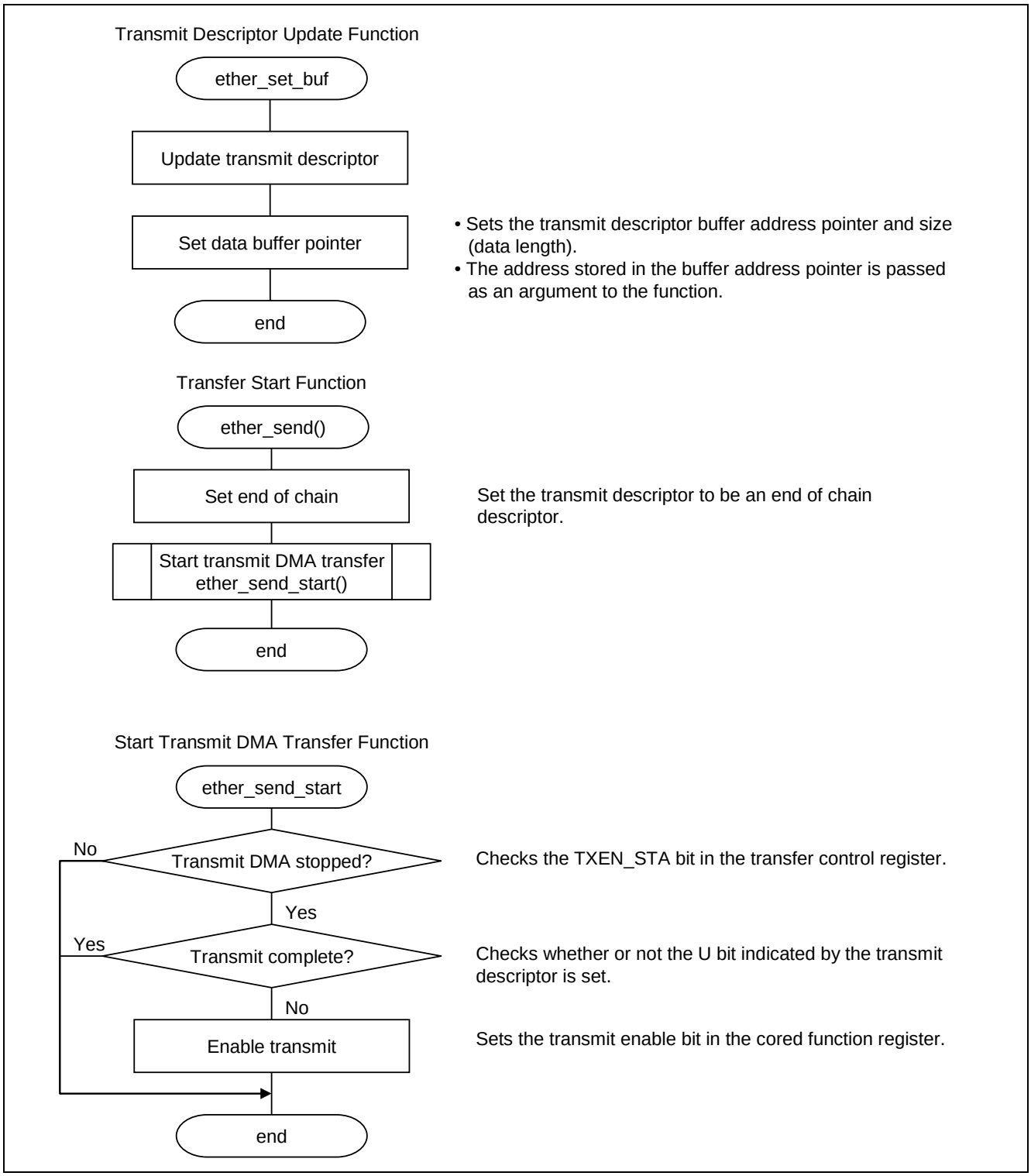
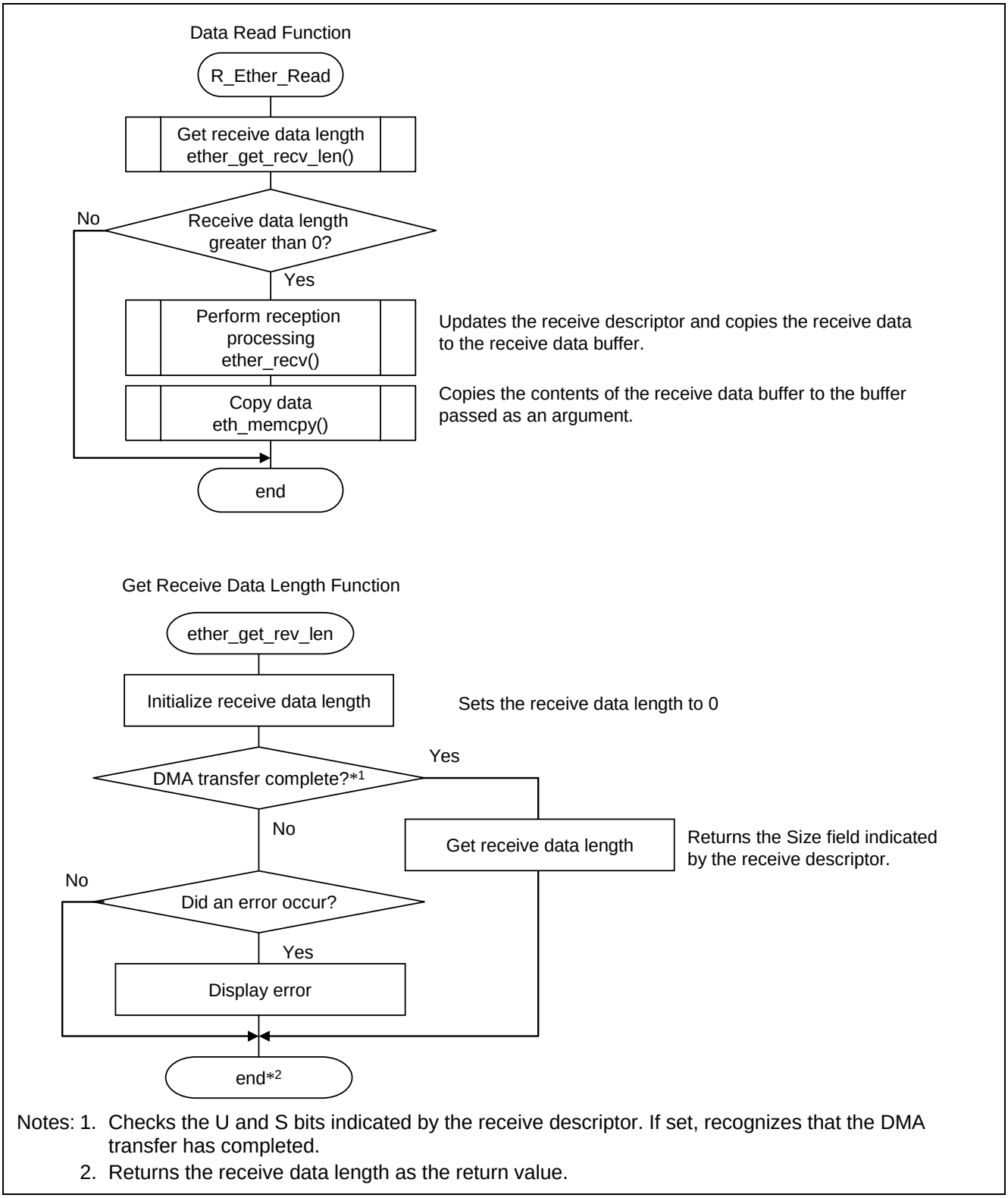
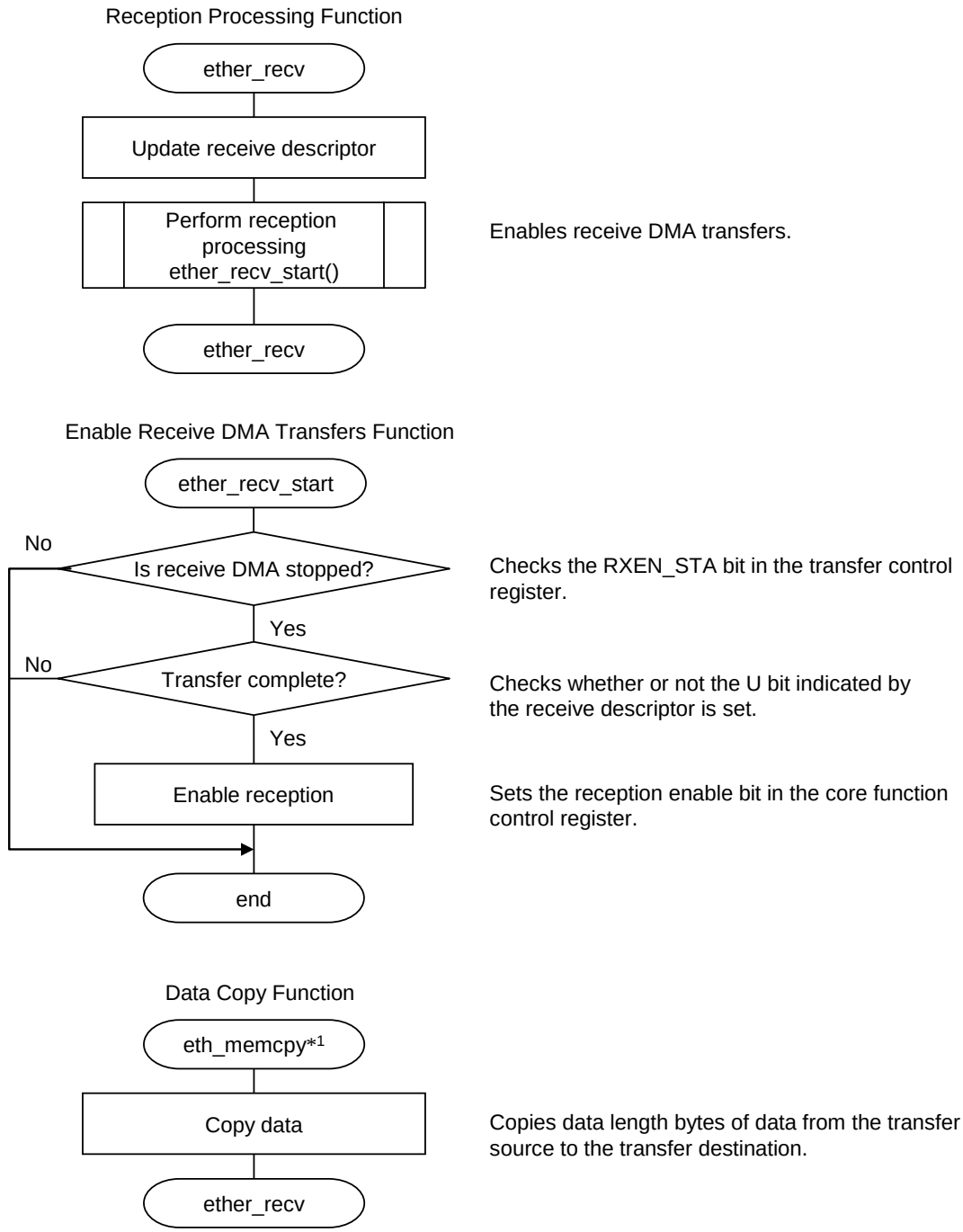


Figure 4.25 Sample Program Ethernet API Flowcharts (5)



Notes: 1. Checks the U and S bits indicated by the receive descriptor. If set, recognizes that the DMA transfer has completed.
2. Returns the receive data length as the return value.

Figure 4.26 Sample Program Ethernet API Flowcharts (6)



Note: 1. The arguments to this function are set to the buffer address pointer (transfer source), the data length, and the data storage destination address (transfer destination).

Figure 4.27 Sample Program Ethernet API Flowcharts (7)

4.7 Notes on Transmission/Reception Settings

4.7.1 Operating Environment Note 1

The sample program uses the autonegotiation function to select the communication method. When the time for the completion of autonegotiation with the remote device (which is the hub in figure 4.15) is comparatively long, communications may fail even if autonegotiation succeeded.

If the V850E2/ML4's remote device cannot receive even though autonegotiation succeeded, insert additional wait time in the V850E2/ML4's processing until the remote device goes to a state where reception is possible. Since the amount of wait time required until the remote device becomes capable of reception depends on the system, the user must determine the amount of wait time required based on thorough evaluation.

To change the wait time, modify the argument in the following code, which is in the R-Ether_Open() function in the ether_driver.c file (line 979).

The initial value is 1000 (1 second). The time given by this argument is 1 ms per count.

```
msleep(1000);
```

4.7.2 Operating Environment Note 2

Since 10Base-T repeater hubs do not support autonegotiation, change the following code in the phy_initialize() function in the ether_phy.c file (line 150) as shown.

<Before modification>

setval = PHY_CONTROL_SPEEDSEL | PHY_CONTROL_DUPLEX | PHY_CONTROL_ANEGENB;

<After modification>

setval = PHY_CONTROL_ANEGENB;

4.7.3 Notes on Error Handling

The sample program does not include error handling. If error processing is required, the user must implement that error processing.

4.7.4 Notes on Transmit/Receive Buffer Definition RAM

The only memory area that the V850E2/ML4 dedicated Ethernet controller DMAC can access is the H bus shared memory area. Thus the transmit/receive descriptors and data buffers used by the Ethernet controller must be allocated in H bus shared memory.

5. Sample Program Section Allocation

Table 5.1 lists the section allocations used in the sample program.

See the CubeSuite+ V1.00.00 Integrated Development Environment User's Manual - V850 Build for details on compiler support.

Table 5.1 Sample Program Section Allocation

Section	Address	Size	Description
RESET	0x00000000	0x00000004	Reset
INTTAUA0I0	0x000003b0	0x00000004	TAUA0 channel 0
INTETHA0SRX	0x00000ba0	0x00000004	Ethernet receive packet read requests
INTETHA0SCRX	0x00000bb0	0x00000004	Ethernet packet reception
INTETHA0SCTX	0x00000bc0	0x00000004	Ethernet packet transmission
INTETHA0RS	0x00000bd0	0x00000004	Ethernet receive status detection
INTETHA0TS	0x00000be0	0x00000004	Ethernet transmit status detection
INTETHA0FS	0x00000bf0	0x00000004	Ethernet FIFO status detection
INTETHA0MAC	0x00000c00	0x00000004	Ethernet statistics counter overflow
.pro_epi_runtime	0x00000c08	0x000001e0	Prolog/epilog runtime calls
.text	0x00000c08	0x000001e0	.text section
eth_memory.bss	0xf9800000	0x000029d8	Ethernet descriptors and data buffers
.sdata	0xfedf0000	0x00000030	.sdata section (initialized variables)
.sbss	0xfedf0030	0x00003cb8	.sbss section (uninitialized variables)
.bss	0xfedf3ce8	0x00000200	.bss section

6. Reference Documents

- V850E2/ML4 Hardware (r01uh0262ej0001_v850e2ml4.pdf)
  CubeSuite+ V1.00.00 Integrated Development Environment User's Manual: Coding for CX Compiler (r20ut0554ej0100_qscdcx.pdf)
  CubeSuite+ V1.00.00 Integrated Development Environment User's Manual: V850 Build (r20ut0557ej0100_qsbd850.pdf)
- LAN8700/LAN8700i Datasheet
  (Download the latest versions of these datasheets from the Standard Microsystems Corporation web site.)

Website and Support

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

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Jun.15.12	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord 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 one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

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.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

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

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

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

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

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

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

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 2.0