SH7670 Group

Example of Setting for Reception of Ethernet Frames

R01AN0303EJ0101
Rev. 1.01
Oct. 15, 2010

Summary

This application note describes an example of settings for connecting the Ethernet controller of the SH7670, SH7671, SH7672 and SH7673.

Target Device

SH7670 MCU

Contents

1. Introduction

1.1 Specifications

- In this sample program, ten Ethernet frames are received. Every time an Ethernet frame is received, the frame-received interrupt is used to initiate copying of the frame to a user buffer.

1.2 Module Used

- Ethernet controller (EtherC)
- Ethernet controller direct memory access controller (E-DMAC)
- Interrupt controller (INTC)
- I^2C bus interface 3 (IIC3)
- Pin function controller (PFC)

1.3 Applicable Conditions

MCU	SH7670
Operating Frequency	Internal clock: 200 MHz Bus clock: 66.6 MHz Peripheral clock: 33.3 MHz
Integrated Development Environment	Renesas Electronics High-performance Embedded Workshop Ver.4.03.00
C Compiler	Renesas Electronics SuperH RISC engine Family C/C++ compiler package Ver.9.01 Release 01
Compiler Options	Default setting in the High-performance Embedded Workshop (-cpu=sh2afpu -fpu=single -debug -gbr=auto -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1)

1.4 Related Application Notes

For more information, refer to the following application notes:

- SH7670 Group Example of Initialization
- SH7670 Group Example of Setting for Automatic Negotiation by Ethernet PHY-LSI
- SH7670 Group Example of Setting for Transmission of Ethernet Frames

2. Description of the Sample Application

- This sample application employs an Ethernet controller (EtherC) and a direct memory access controller for Ethernet controller (E-DMAC).

2.1 Operational Overview of Module Used

Be sure to use the EtherC and E-DMAC modules to handle Ethernet communications for this LSI. The EtherC module controls the transmission and reception of Ethernet frames. E-DMAC specifically handles DMA transfer between its transmission/reception FIFO and data-storage areas (buffers) specified by the user.

2.1.1 Overview of the EtherC

This LSI has an on-chip Ethernet controller (EtherC) that conforms to the Ethernet or the IEEE802.3 MAC (Media Access Control) layer standard. Connecting a physical-layer LSI (PHY-LSI) complying with this standard enables the EtherC to perform transmission and reception of Ethernet/IEEE802.3 frames. The EtherC with this on-chip LSI has one MAC layer interface. The Ethernet controller is connected to the direct memory access controller for Ethernet controller (E-DMAC) inside this LSI, and carries out high-speed data transfer to and from the memory.

Figure 1 shows a configuration of the EtherC.

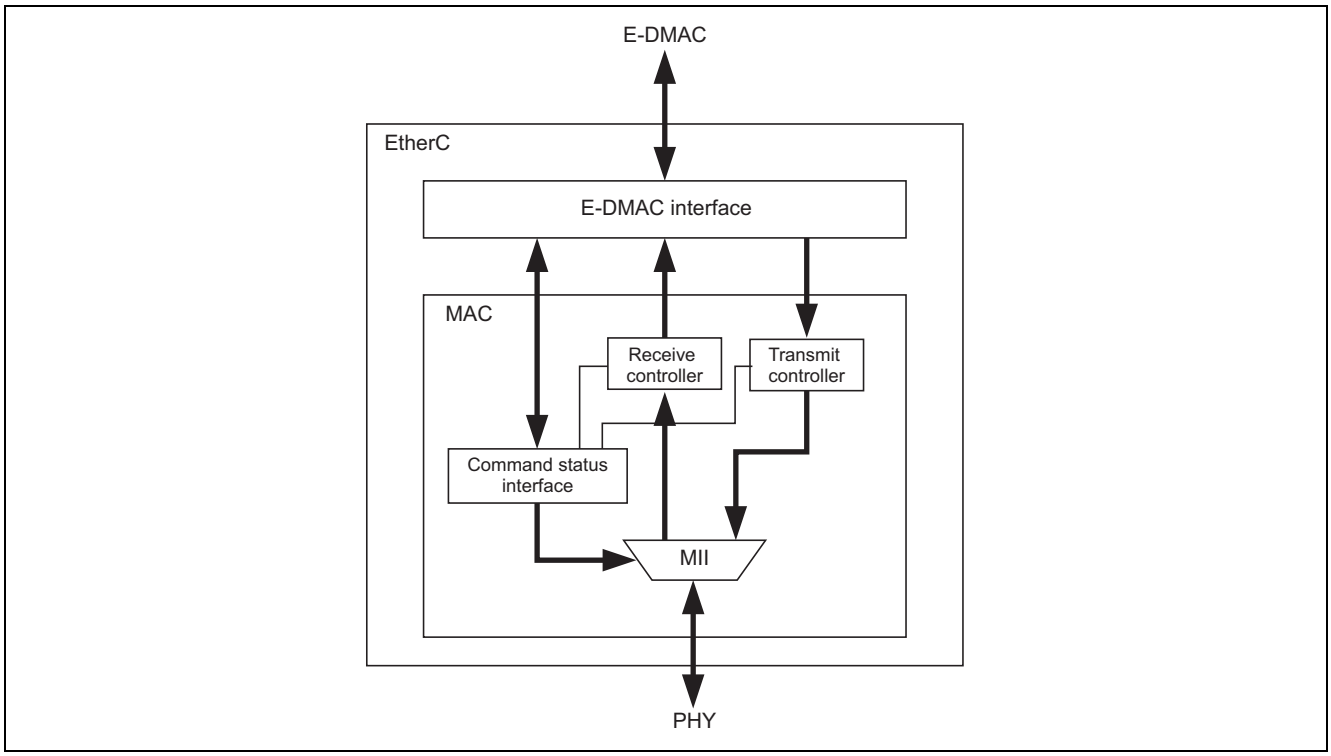

Figure 1 Configuration of the EtherC

2.1.2 Overview of the EtherC Receiver

The EtherC receiver separates the frame of data which have been input from the MII (Media Independent Interface) into preamble, SFD (Start Frame Delimiter), data and CRC (Cyclic Redundancy Check) code. Then it outputs the portion other than preamble, SFD and CRC code to the receiver E-DMAC. Figure 2 shows the state transitions of the EtherC receiver. The flow of operations in reception is described below.

1. When the receive enable (RE) bit of the EtherC mode register (ECMR) is set, the EtherC receiver enters the idle state.
2. When the start frame delimiter (SFD) is detected after the preamble of a frame to be received, the EtherC receiver starts processing for reception. A frame with an invalid pattern is discarded.
3. In normal mode, the EtherC receiver starts reception of data (i) if the destination MAC address matches the receiver's own address, (ii) in the case of a broadcast frame, and (iii) in the case of multicast frame. If promiscuous mode has been specified, the EtherC receiver starts reception of data irrespective of the frame type.
4. After a frame has been received from the MII, the EtherC receiver carries out a CRC of the frame data. The result is indicated as a status bit in the descriptor after the frame of data has been written to memory. If an error is found, the error state is reported to the EtherC/E-DMAC status register (EESR).
5. After one frame has been received, the EtherC receiver enters the idle state in readiness for receiving the next frame.

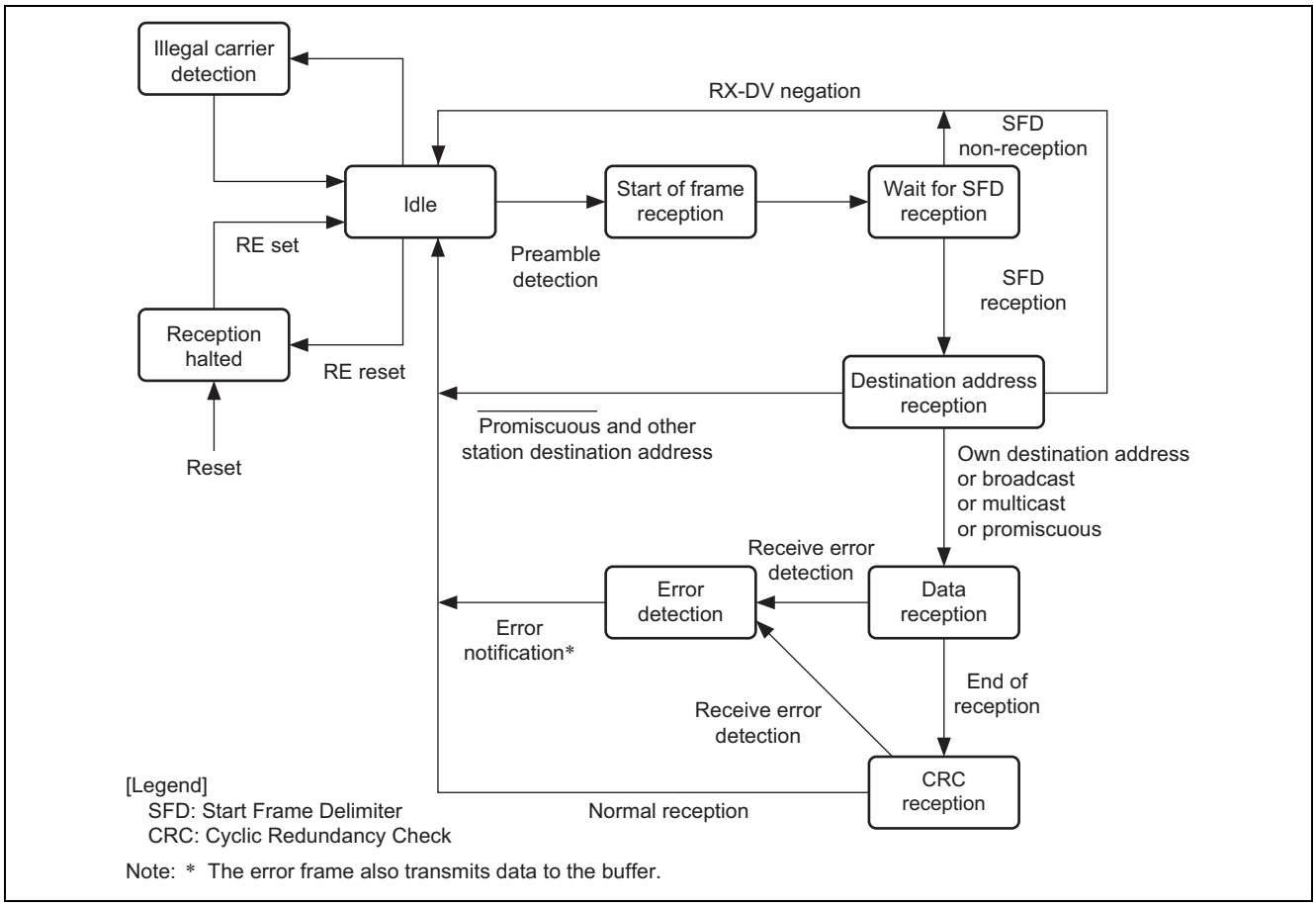

Figure 2 State Transmissions of the EtherC Receiver

2.1.3 Overview of the E-DMAC

This LSI includes a direct memory access controller (E-DMAC) directly connected to the Ethernet controller (EtherC). The E-DMAC transfers data for transmission and reception between transmit/receive FIFO in the E-DMAC and data storage location (transmit/receive buffer) specified by user using DMA transfer. Directly writing data to or reading data from the transmit/receive FIFO by the CPU is not possible. During DMA transfer, the E-DMAC refers to information called transmit and receive descriptors (details to be described in the next section); these are placed in memory by the user. The E-DMAC reads the descriptor information before transmitting or receiving an Ethernet frame, and follows the descriptor in reading data for transmission from the transmission buffer or writing received data to the receiving buffer. By setting up a number of consecutive descriptors (a descriptor list), it is possible to execute the consecutive transfer of multiple Ethernet frames. This E-DMAC function lightens the load on the CPU and enables efficiency in data transfer control.

Figure 3 shows the configuration of the E-DMAC, and of the related descriptors and buffers.

The E-DMAC has the following features;

- Equipped with two independent on-chip DMACs for transmission and reception
- The load on the CPU is reduced by means of a descriptor management system
- Transmit/receive frame status information is indicated in descriptors
- Block transfer by using DMA (16-byte units) achieves efficient utilization of the system bus.
- Supports one-frame/one-descriptor, one-frame/multi-frame (multi-buffer) operation (see section 2.1.5)

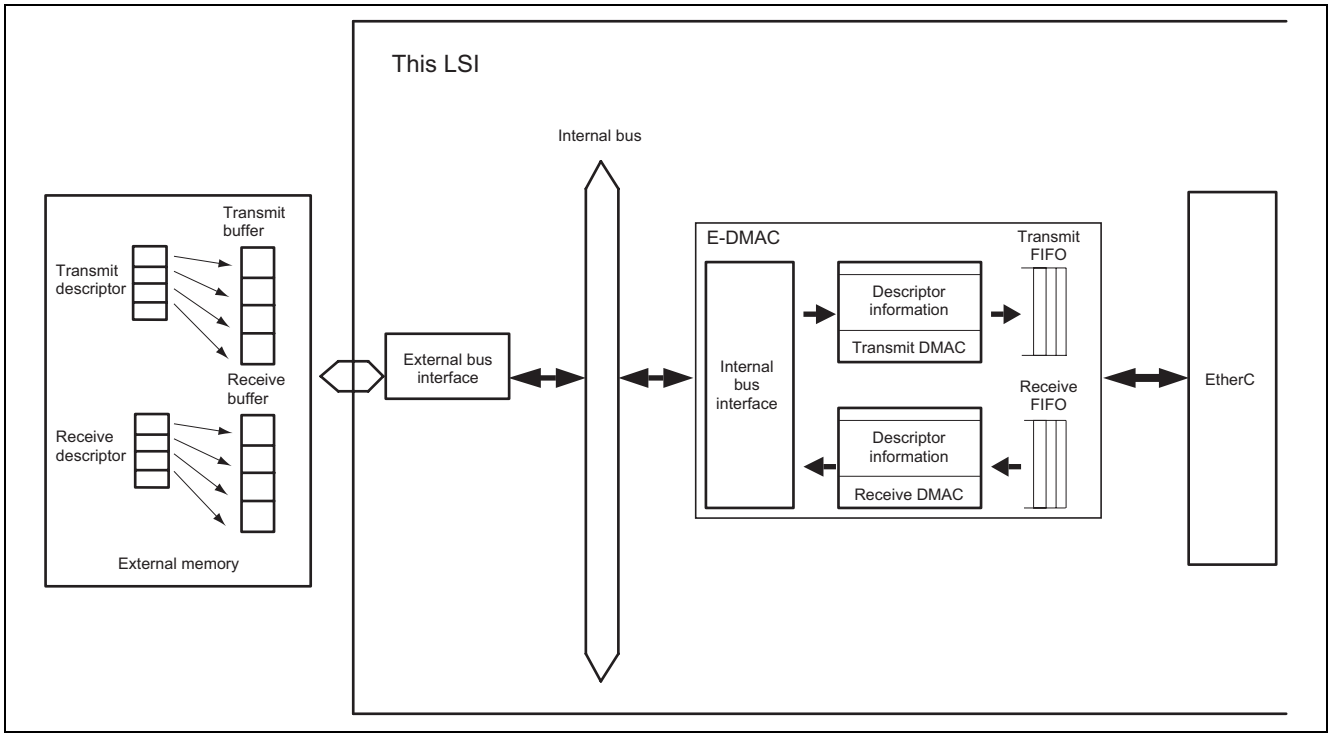

Figure 3 Configuration of E-DMAC, and Descriptors and Buffers

2.1.4 Overview of E-DMAC Descriptors

When the E-DMAC performs DMA transfer, it employs descriptor information that includes the storage address for the data for transfer, etc. There are two types of descriptors: transmit descriptors and receive descriptors. When the TR bit in the E-DMAC transmit request register (EDTRR) is set to 1, the E-DMAC automatically starts reading a transmit descriptor. When the RR bit in the E-DMAC receive request register (EDRRR) is set to 1, the E-DMAC automatically starts reading a receive descriptor. The user must enter information related to the DMA transfer of Ethernet data in the transmit/receive descriptors before the transfer can proceed. After transmission or reception of an Ethernet frame has been completed, the E-DMAC switches the descriptor active/inactive bit (TACT bit for transmission, RACT bit for reception) to the inactive setting and indicates the result of transmission or reception in the status bits (TFS26 to TFS0 for transmission, RFS26 to RFS0 for reception).
Descriptors are placed in readable and writable memory, and the address where the first descriptors start (the addresses of the first descriptors of each type to be read by the E-DMAC) are set in the transmit descriptor list address register (TDLAR) and receive descriptor list address register (RDLAR). When multiple descriptors are set up in a descriptor list, the descriptors are placed in contiguous address ranges in accord with the descriptor length as indicated by bits DL1 and DL0 in the E-DMAC mode register (EDMR).

2.1.5 Overview of Receive Descriptors

Figure 4 shows the relationship between a receive descriptor and a receive buffer.

In order from its first address, a receive descriptor consists of RD0, RD1, RD2 (each is a 32-bit unit), and padding. RD0 indicates whether the descriptor is active or inactive, describes the configuration of the descriptor, and contains state information. RD1 indicates the size of the receiving buffer (RBL) to which the descriptor refers, and the length of the received frame (RDL). RD2 indicates the address where the receiving buffer starts. The length of padding is determined by the descriptor length as specified by bits DL0 and DL1 in the EDMR register.

According to the settings of receive descriptors, either a single descriptor or multiple descriptors can specify a single frame of received data (one frame/one descriptor and one frame/multi-descriptor, respectively). In one frame/multi-descriptor cases, multiple descriptors are prepared in advance to form a descriptor list. If a frame is longer than the setting of the descriptor's RBL field, the E-DMAC uses the next descriptor in the sequence to continue transferring the frame to the receiving buffer. For example, if the E-DMAC receives an Ethernet frame with 1,514 bytes while the RBL of each descriptor is 500 bytes, the received Ethernet frame is transferred to the receiving buffer in 500-byte portions until the final 14 bytes that remain are transferred to the fourth buffer.

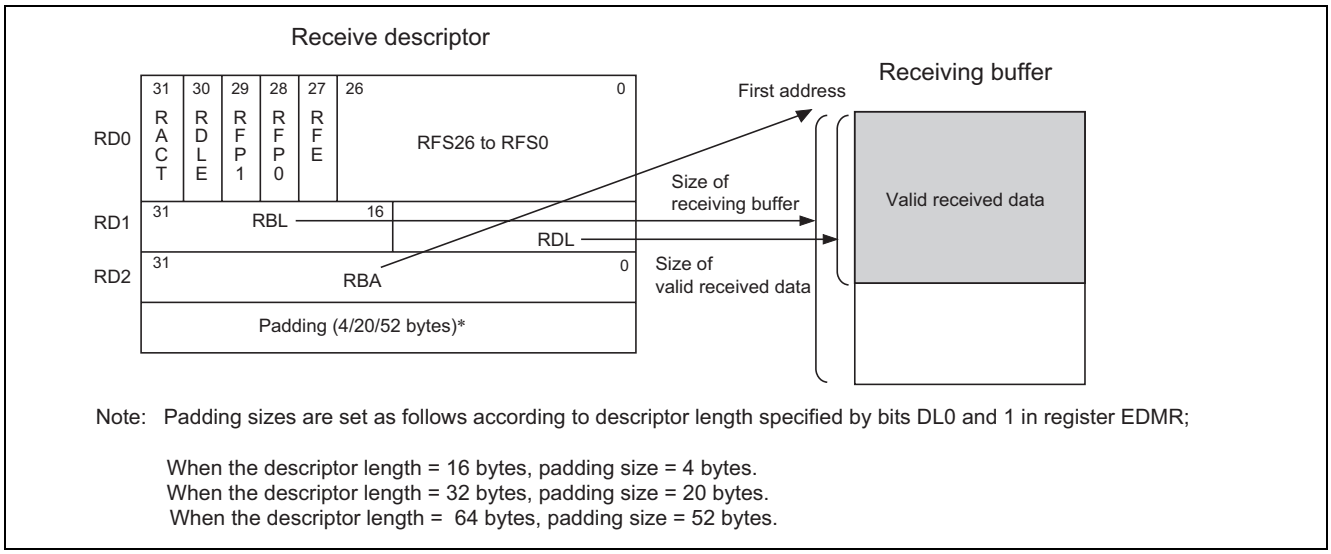

Figure 4 Relationship between Receive Descriptor and Receiving Buffer

2.1.6 Example of Setting Receive Descriptors

Figure 5 shows an example where three receive descriptors and three areas of the receiving buffer are in use. Each area of the receiving buffer has a size of 1,520 bytes, and operation is of the one-frame/one-descriptor type. The receive descriptors are simplified in the figure, with only RD0 being shown. Numbers (1), (2), etc. in the figure indicate the sequence of execution.

1. Bits RFP1, RFP0, RFE, and RFS26 to RFS0 of all descriptors are set to 0.
2. In the first and second descriptors, the RDLE bit is set to 0. The RDLE bit of the third descriptor is set to 1, so the E-DMAC reads the first descriptor on completion of processing of the third descriptor. Settings like this can be used to arrange descriptors in a ring structure.
3. Although the following settings for each of the descriptors have been left out of figure 5, prior to the start of reception, the RBL of RD1 is set for a size of each area of the receiving buffer, 1,520 bytes, and the RBA of RD2 is set to the address where the corresponding area of the receiving buffer starts.
4. To enable continuous reception, the RACT bit of each descriptor is set to 1.

The next section describes the details on the procedure in receiving operation.

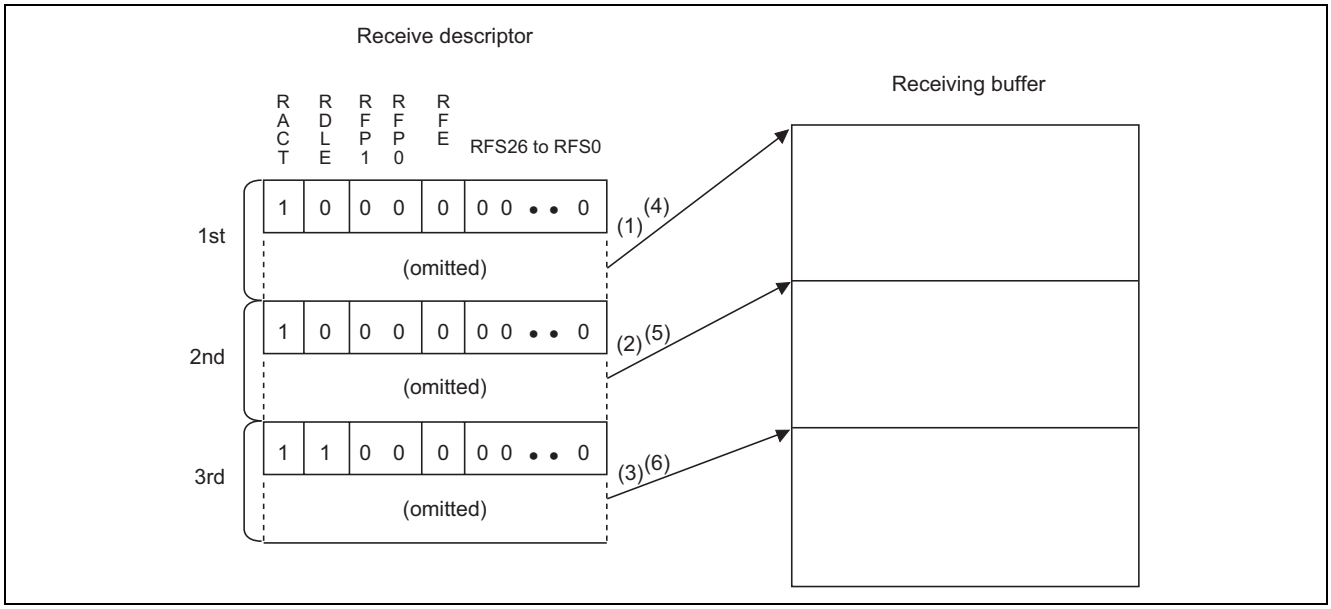

Figure 5 Relationship between Receive Descriptor and Receiving Buffer

2.1.7 Operation of the Sample Program

When the setting of the reception enable (RE) bit of the EtherC mode register (ECMR) is 1, and 1 is written to the receive request (RR) bit in the E-DMAC receive request register (EDRRR), the reception section of the E-DMAC is activated. After a software reset of the EtherC and E-DMAC modules, the E-DMAC reads the descriptor indicated by the receive descriptor list address register (RDLAR), and enters the reception-standby state if the setting of the RACT bit is 1 (active). If the EtherC module then receives a frame addressed to itself (the address of the frame allows for reception by the EtherC module), it stores the received data in the receive FIFO. If the setting of the RACT bit of the receive descriptor is 1, the received data are transferred to the receiving buffer specified by RD2 (if the setting of the RACT bit is 0 (inactive), the RR bit is cleared to 0 and E-DMAC operation for reception is halted). If the received frame contains more data than the buffer length given by RD1, the E-DMAC writes back to the descriptor when the buffer is full (to set RFP = B'10 or B'00), and then reads the next descriptor.
When reception of the frame is completed or is suspended because of any kind of error, the E-DMAC writes back to the current descriptor (to set RFP = B'11 or B'01). If continuous reception has been selected (i.e. cases where the setting of the receive enable control (RNC) bit in the receiving method control register (RMCR) is 1), the E-DMAC then reads the next descriptor and enters the reception-standby state if the setting of the RACT bit is 1. If continuous reception has not been selected (i.e. cases where the setting of the RNC bit in the RMCR is 0), the RR bit in EDRRR is cleared to 0 and E-DMAC operation for reception is halted. If the RR bit is again set to 1, the E-DMAC reads the descriptor which follows the last descriptor to have been used in reception, and then enters the reception-standby state.
Figure 15 shows an example of the flow of reception (in the one-frame/one-descriptor and continuous-reception cases).

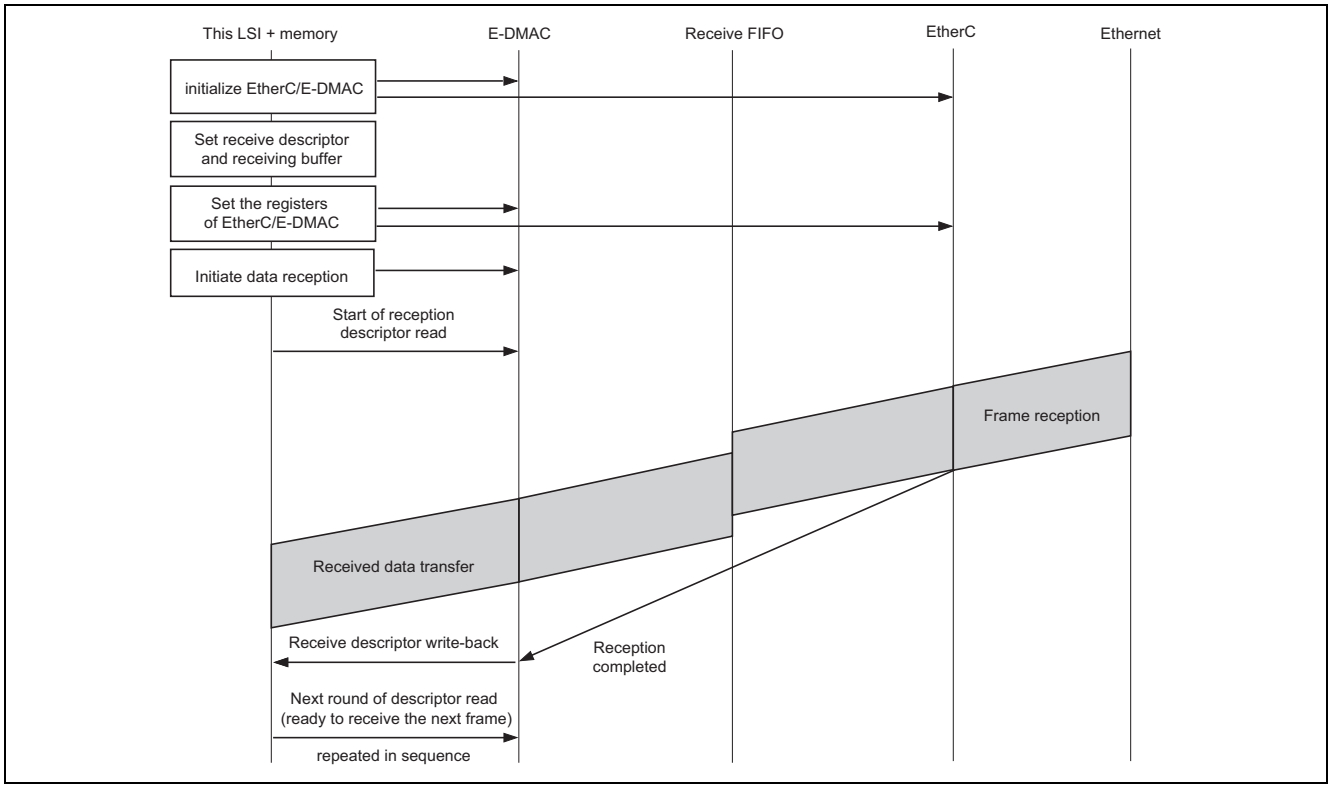

Figure 6 Sample Reception Flowchart

2.1.8 Procedure for Setting Module Used

This section describes an example of fundamental settings for reception of the Ethernet frames.
Figures 7 and 8 show an example of flowchart for setting the reception of Ethernet frames.

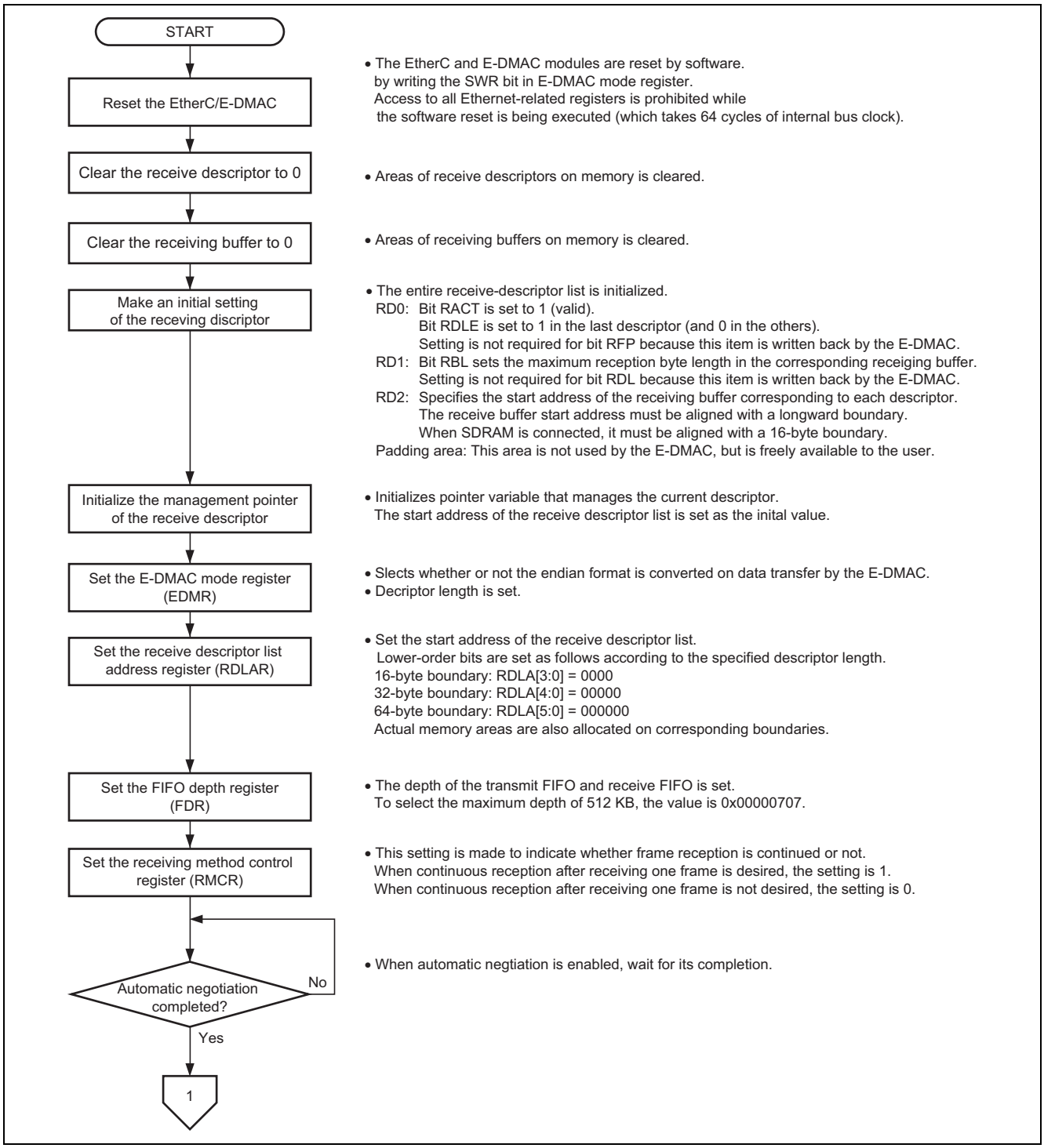

Figure 7 Example of Flowchart for Ethernet Setting (1)

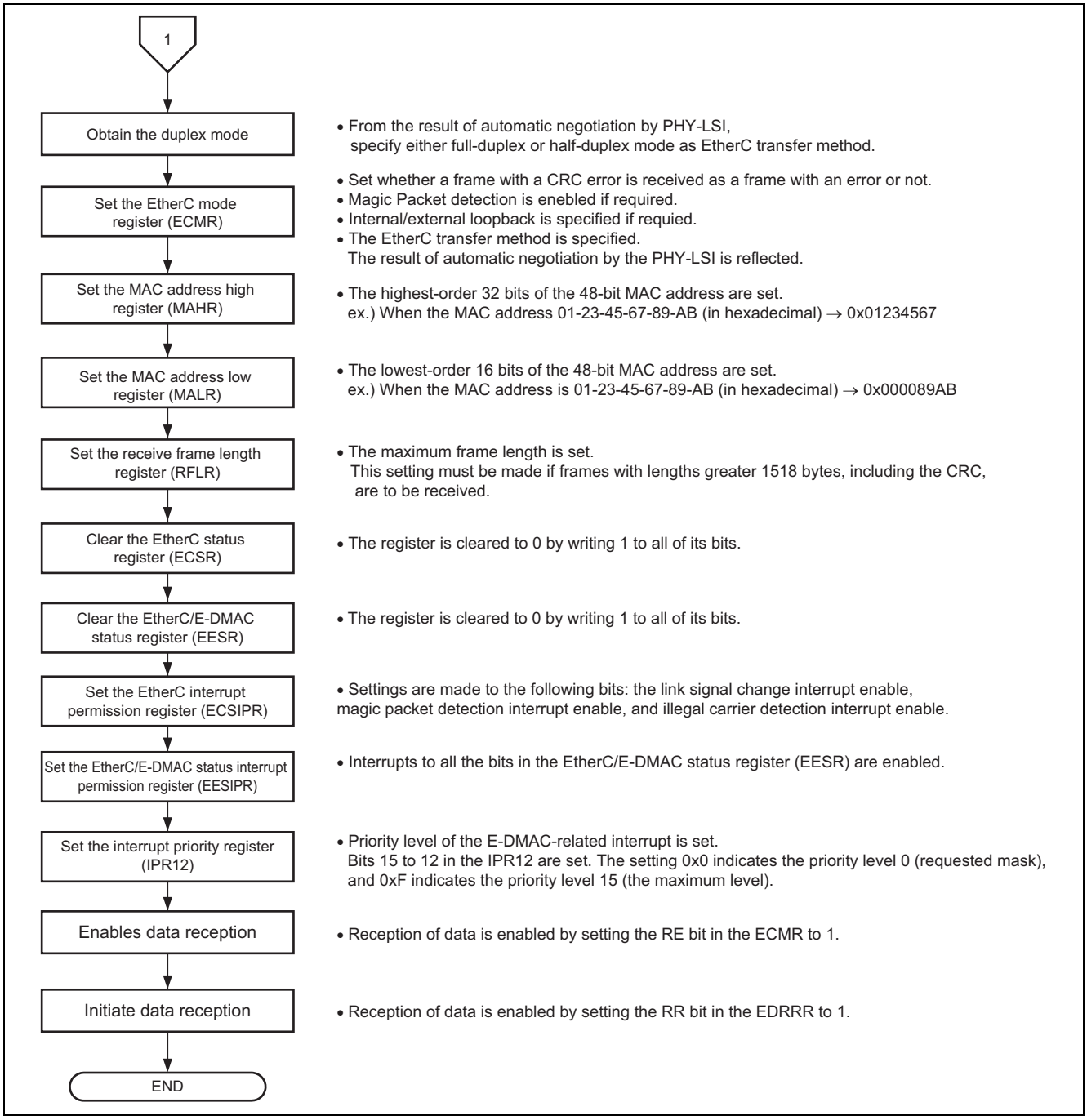

Figure 8 Example of Flowchart for Ethernet Setting (2)

2.2 Operation of the Sample Program

This sample program employs the EtherC and the E-DMAC modules to receive 10 Ethernet frames from the host personal computer at the other end. In this sample program, there are four receive descriptors, and four areas of the receiving buffer each with 1,520 bytes. The receive enable control (RNC) bit in the receiving method control register (RMCR) is set to 1 to enable continuous reception operations. Every time an interrupt related to reception such as frame reception (FR), etc. is generated, the RFE bit (bit 27 in the RD0) of the receive descriptor is checked, and if no errors are found (i.e. RFE = 0) the single frame of data in the receiving buffer is copied to the user buffer. The corresponding descriptor is then initialized in readiness for its next round of reception. If an error is found (i.e. RFE = 1), data in the receiving buffer are not copied to the user buffer but the corresponding descriptor is initialized.
Additionally, data other than the preamble, SFD, and CRC in the Ethernet frame are transferred to the receiving buffer.
Figure 9 shows operating environment of the sample program, and figure 10 shows a format of the Ethernet frame.

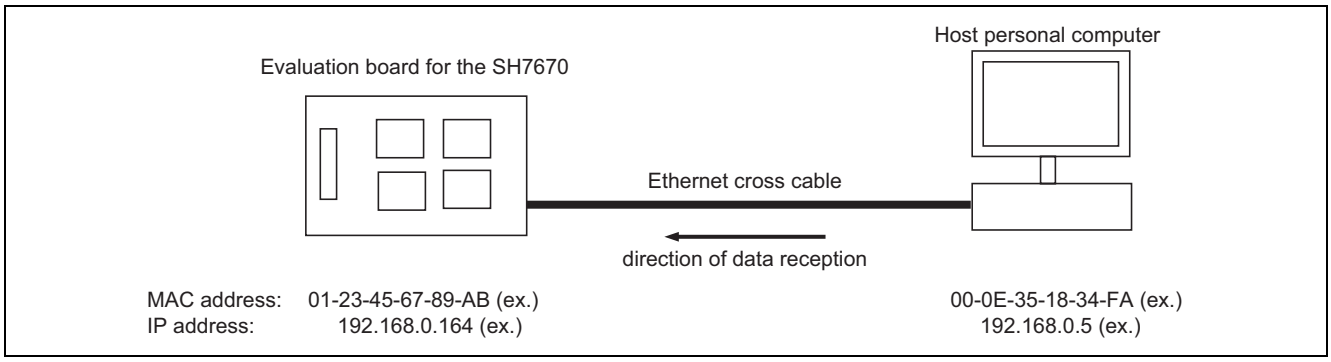

Figure 9 Operating Environment of the Sample Program

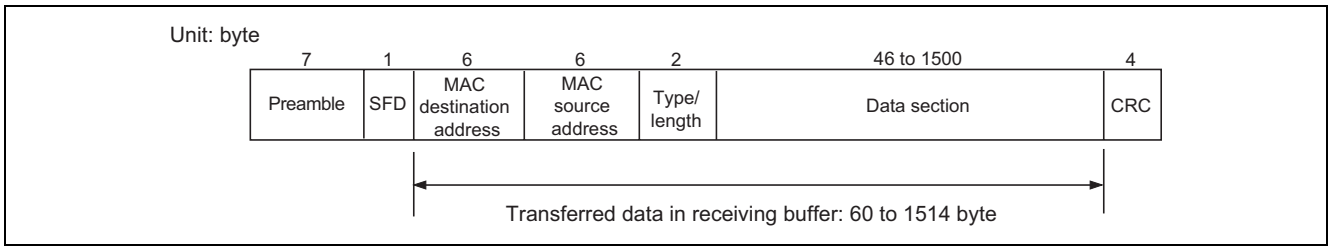

Figure 10 Ethernet Frame Format

2.3 Definition of Descriptors Used in the Sample Program

The E-DMAC does not use the padding area of a descriptor, this area is freely available to the user. In this sample program, this area is used to specify the address where the next descriptor starts, and this in conjunction with software is used to arrange the descriptors in a ring structure.
Figure 11 shows the definition of the receive-descriptor structure in the sample program and an example of how the array of receive descriptors is used.

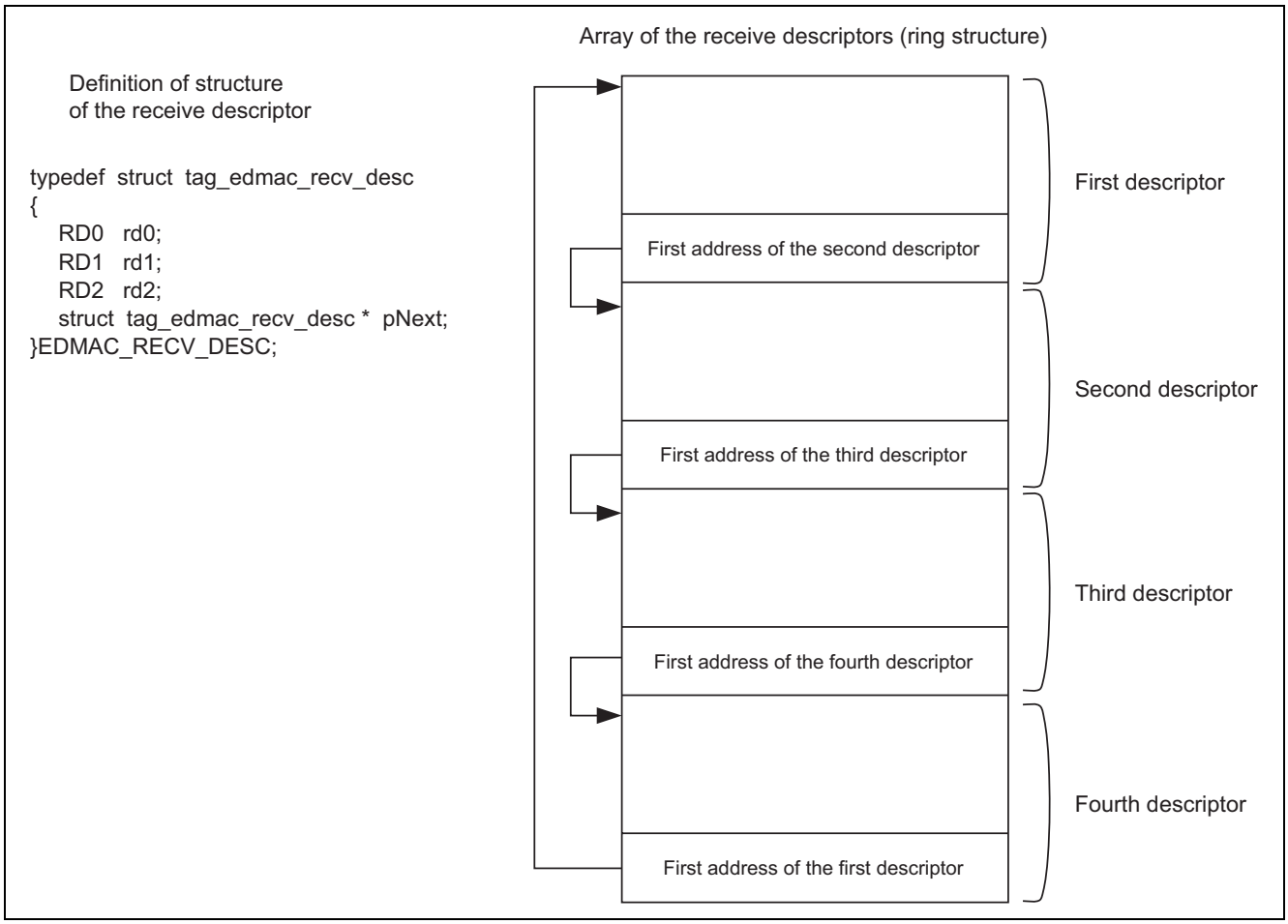

Figure 11 Definition of Receive Descriptor and Usage Example of Receive Descriptor Array

2.4 Sequence of Processing by the Sample Program

Figures 12 to 16 show flows of handling the sample program. For details on the automatic negotiation function phy_autonego, see the application note “SH7670 Example of Setting for Automatic Negotiation by Ethernet PHY-LSI (REJ06B0800)”.

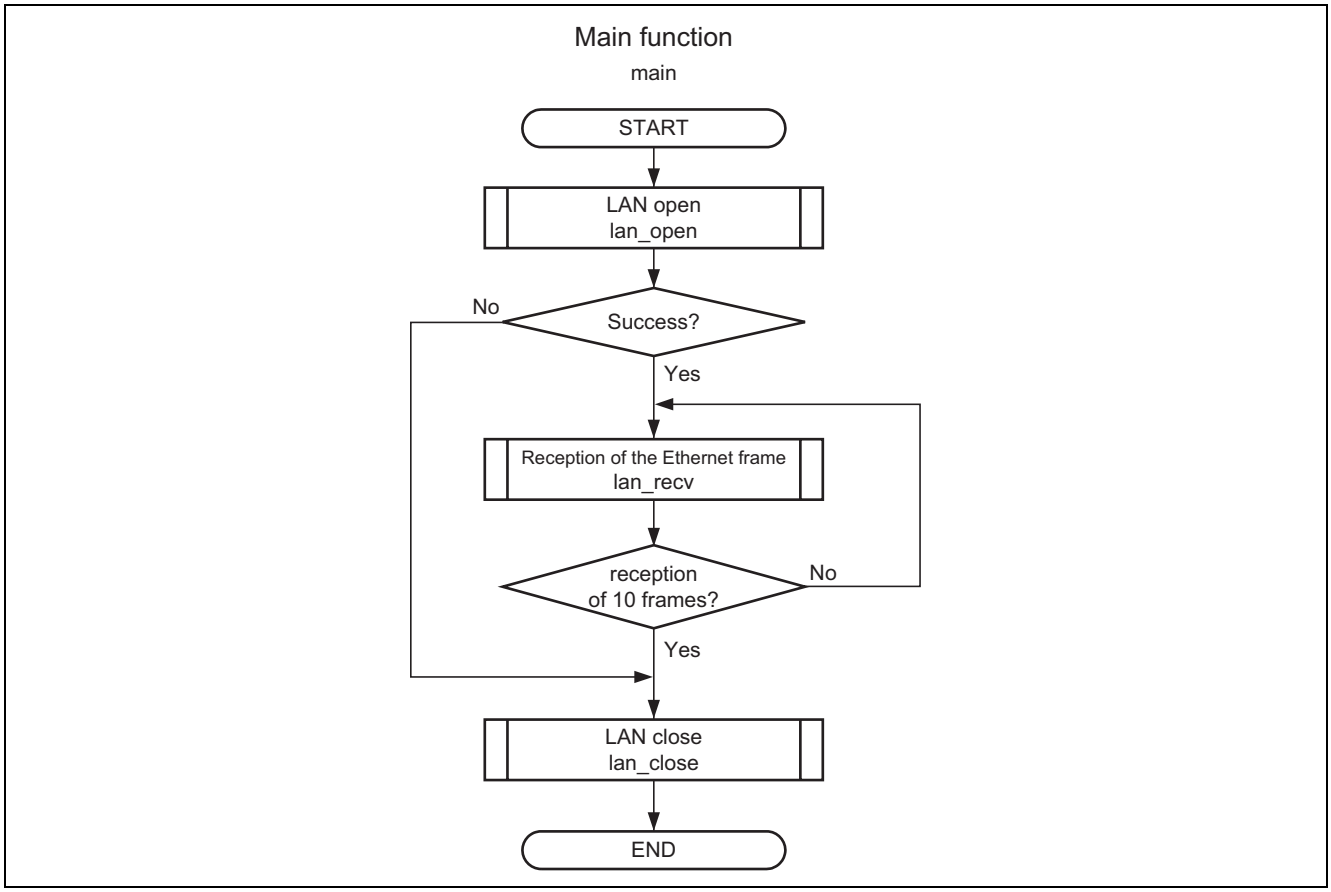

Figure 12 Flow of Handling in the Sample Program (1)

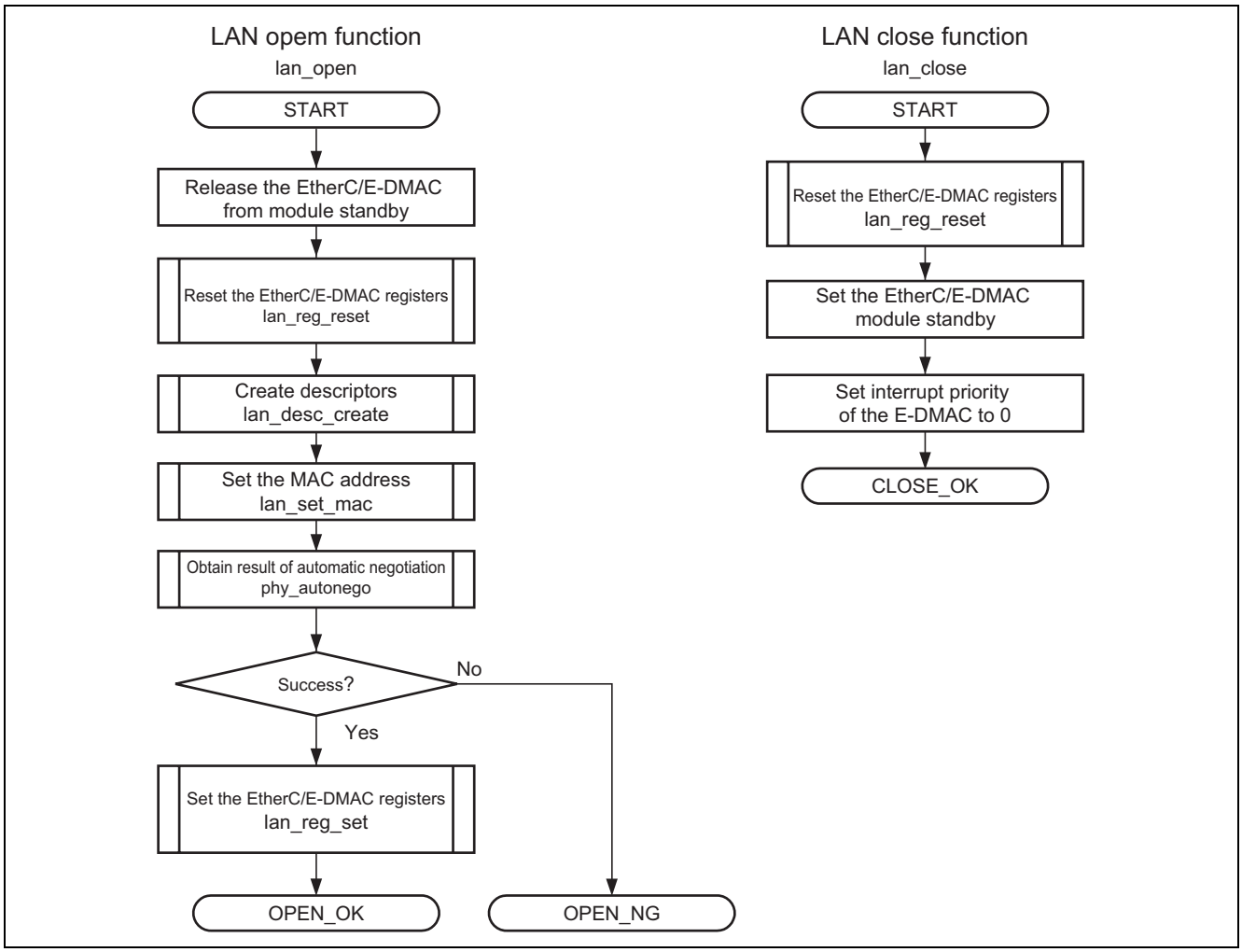

Figure 13 Flow of Handling in the Sample Program (2)

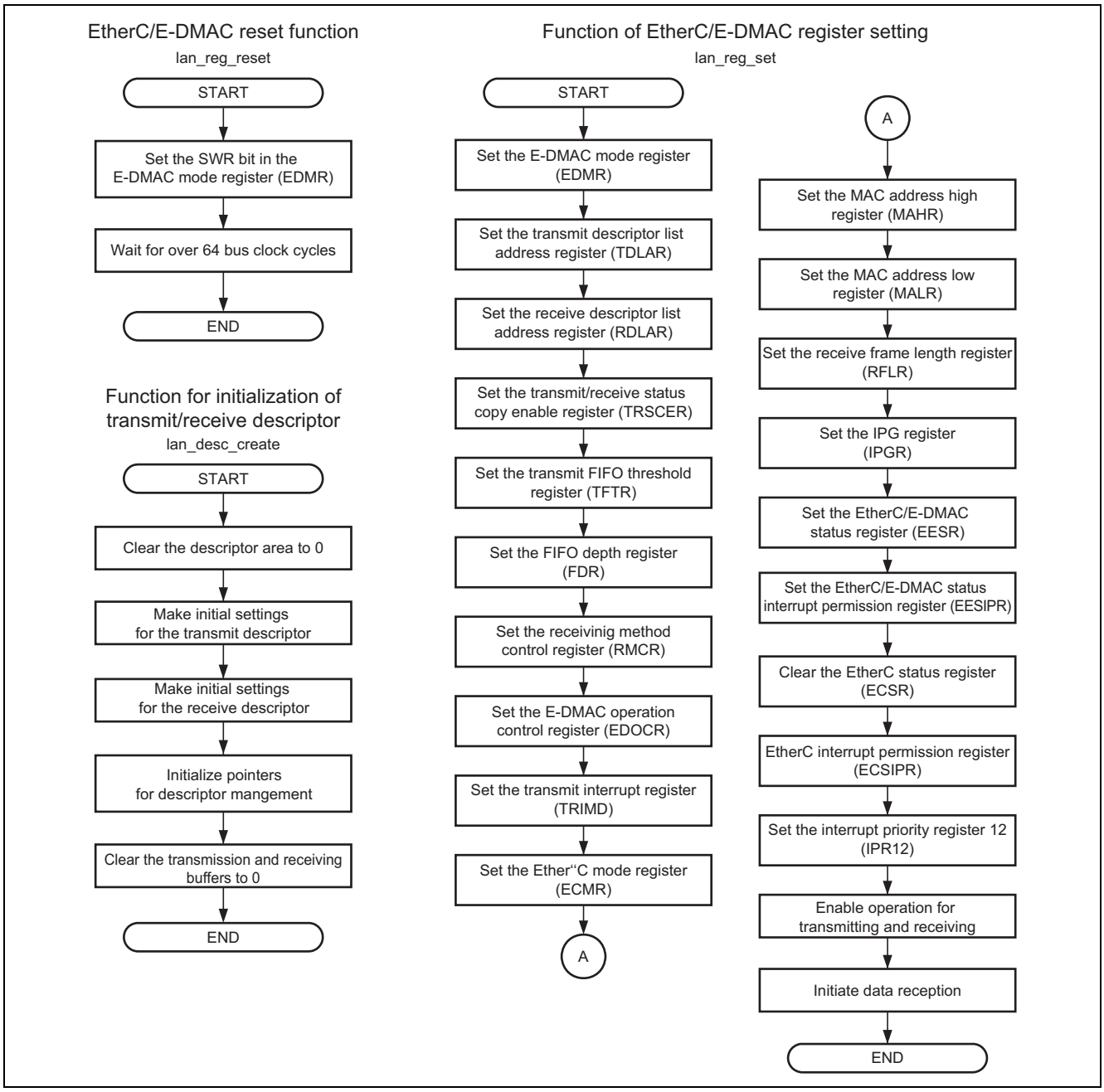

Figure 14 Flow of Handling in the Sample Program (3)

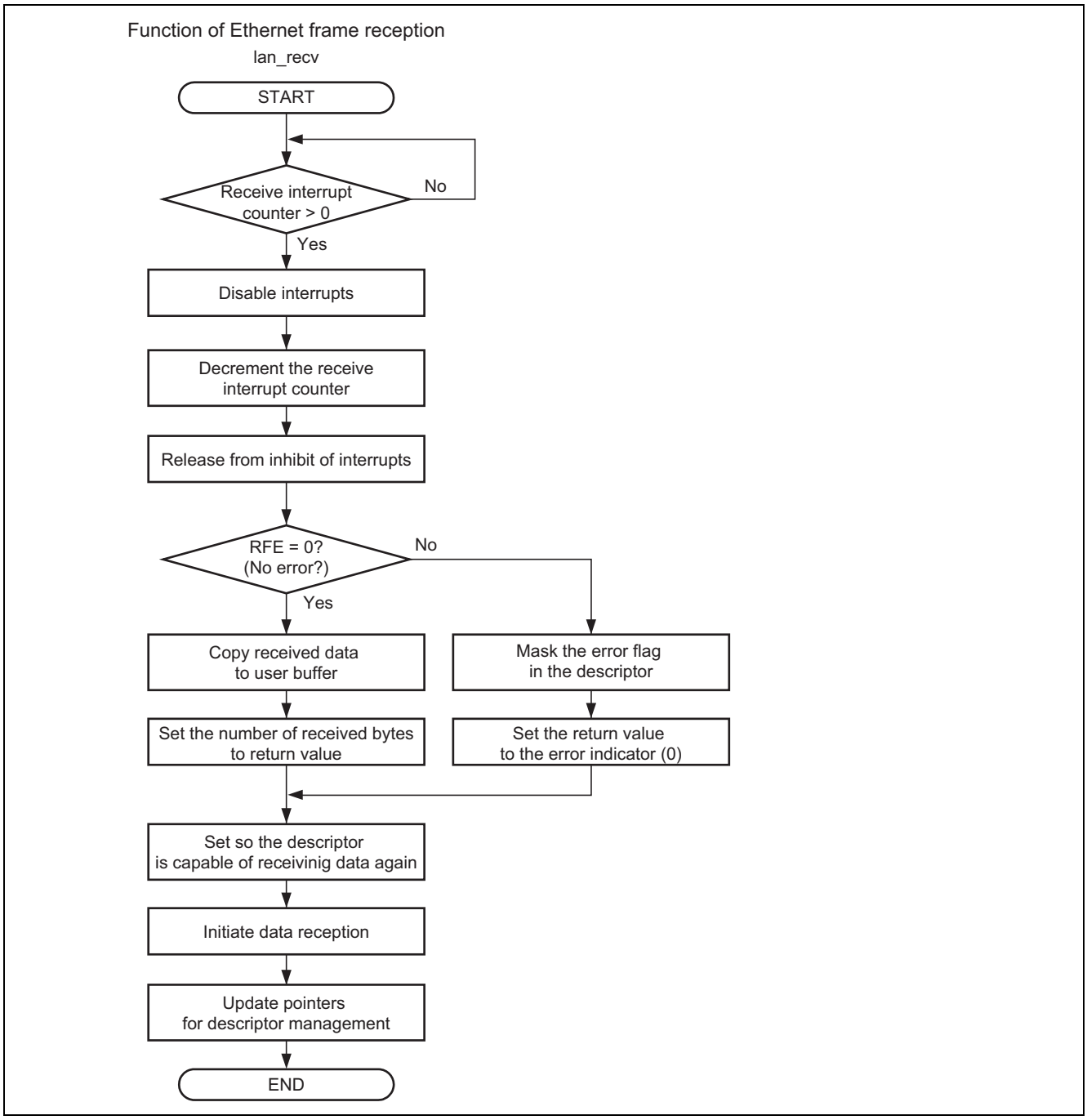

Figure 15 Flow of Handling in the Sample Program (4)

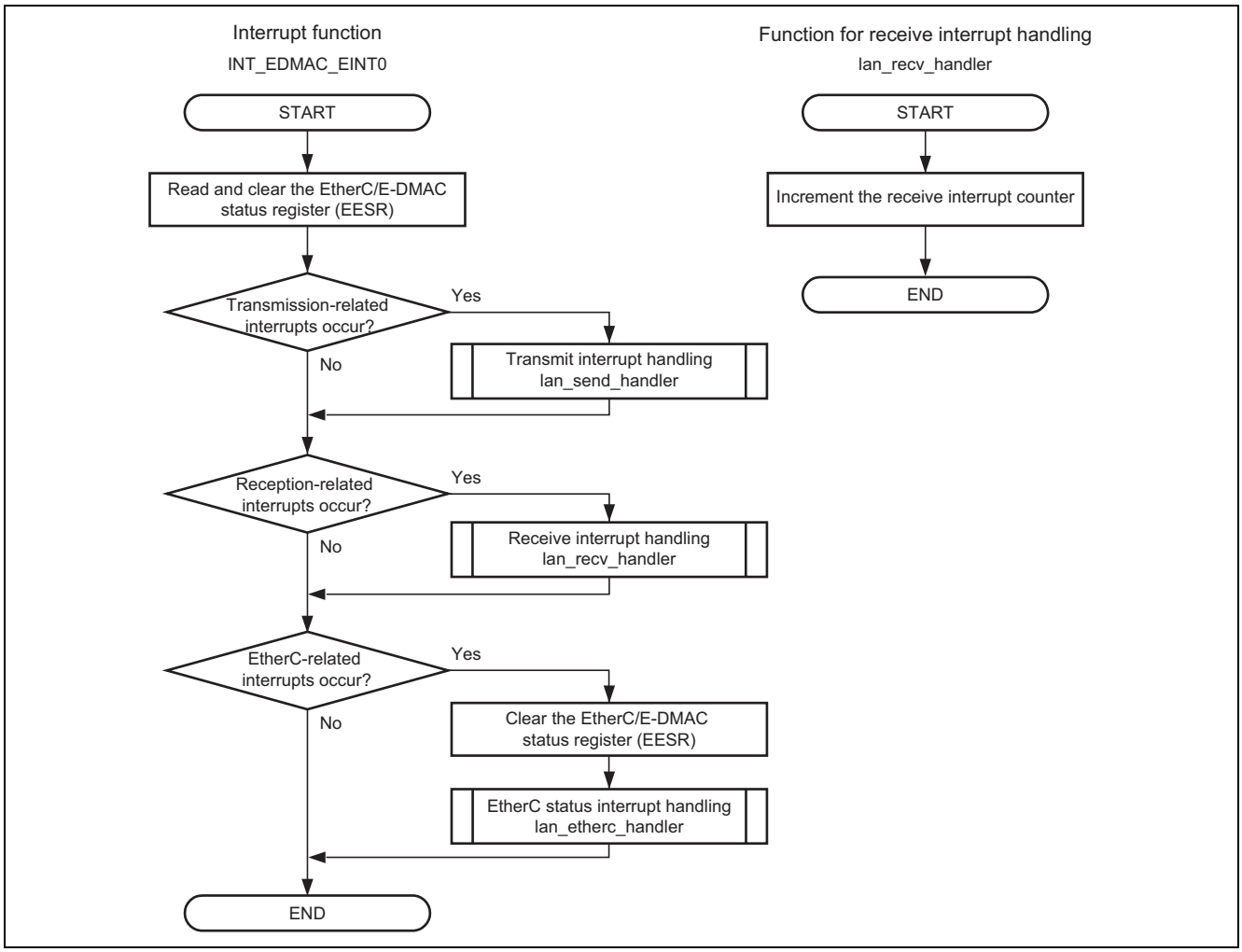

Figure 16 Flow of Handling in the Sample Program (5)

3. Sample Program Listing

3.1 Sample program list "main.c" (1)

```
1   /***********************************************************************
2   *   DISCLAIMER
3   *
4   *   This software is supplied by Renesas Electronics Corporation and is only
5   *   intended for use with Renesas products. No other uses are authorized.
6   *
7   *   This software is owned by Renesas Electronics Corporation and is protected under
8   *   all applicable laws, including copyright laws.
9   *
10  *   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11  *   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12  *   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13  *   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14  *   DISCLAIMED.
15  *
16  *   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17  *   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18  *   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19  *   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20  *   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21  *
22  *   Renesas reserves the right, without notice, to make changes to this
23  *   software and to discontinue the availability of this software.
24  *   By using this software, you agree to the additional terms and
25  *   conditions found by accessing the following link:
26  *   http://www.renesas.com/disclaimer
27  ***********************************************************************
28  * Copyright (C) 2007(2010) Renesas Electronics Corporation. All rights reserved.
29  *""FILE COMMENT""*********** Technical reference data *************************
30  *   System Name : SH7671 Sample Program
31  *   File Name   : main.c
32  *   Abstract    : Sample Ethernet Reception Setting
33  *   Version     : 1.00.01
34  *   Device      : SH7671
35  *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.03.00).
36  *               : C/C++ compiler package for the SuperH RISC engine family
37  *               :                               (Ver.9.01 Release01).
38  *   OS          : None
39  *   H/W Platform: M3A-HS71(CPU board)
40  *   Description :
41  ***********************************************************************
42  *   History     : Jul.04,2007 ver.1.00.00
43  *               : May 10,2010 ver.1.00.01 Changed the company name and device name
44  *""FILE COMMENT END""*****************************************************/
45  #include "iodefine.h"
46  #include "defs.h"
47  #include "ether.h"
48
49  /* **** Prototype Declaration **** */
50  void main(void);
51
```

3.2 Sample program list "main.c" (2)

```
52    /* **** Variable Declaration **** */
53    #pragma section ETH_BUFF                /* Allocated in SDRAM since capacity is large */
54    typedef struct{
55      unsigned char frame[SIZE_OF_BUFFER];
56      int len;
57      unsigned char wk[12];
58    }USER_BUFFER;
59    static USER_BUFFER recv[10];
60    #pragma section
61
62    /*""FUNC COMMENT""*************************************************************
63     * ID          :
64     * Outline     : Ethernet reception sample program main function
65     *------------------------------------------------------------------------------
66     * Include     : #include "iodefine.h"
67     *------------------------------------------------------------------------------
68     * Declaration : void main(void)
69     *------------------------------------------------------------------------------
70     * Function    : Ethernet frames are received using on-chip Ethernet controller (EtherC)
71     *             : and dynamic memory access controller (E-DMAC) for Ethernet controller.
72     *             : The RTL8201CP from REALTEK is used for PHY module.
73     *             : Multiple planes of receive discriptor is used for continuous reception.
74     *------------------------------------------------------------------------------
75     * Argument    : void
76     *------------------------------------------------------------------------------
77     * ReturnValue : void
78     *------------------------------------------------------------------------------
79     * Notice      :
80     *""FUNC COMMENT END""*********************************************************/
81    void main(void)
82    {
83      int i,j;
84      int ret;
85
86      /* ==== Ethernet initial setting ==== */
87      ret = lan_open();
88      if( ret == OPEN_OK ){
89        /* ==== Start reception of 10 frames ==== */
90        for(i=0; i<10; i++){
91            /* ---- Reception ---- */
92            recv[i].len = lan_recv( recv[i].frame );
93            if( recv[i].len == 0 ){
94                i--;
95            }
96        }
97      }
98      /* ==== Ethernet transmission/reception halted ==== */
99      lan_close();
100   }
101   /* End of file */
```

3.3 Sample program list "ether.c" (1)

```
/***********************************************************************
*   DISCLAIMER
*
*   This software is supplied by Renesas Electronics Corporation and is only
*   intended for use with Renesas products. No other uses are authorized.
*
*   This software is owned by Renesas Electronics Corporation and is protected under
*   all applicable laws, including copyright laws.
*
*   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
*   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
*   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
*   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
*   DISCLAIMED.
*
*   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
*   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
*   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
*   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
*   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
*
*   Renesas reserves the right, without notice, to make changes to this
*   software and to discontinue the availability of this software.
*   By using this software, you agree to the additional terms and
*   conditions found by accessing the following link:
*   http://www.renesas.com/disclaimer
***********************************************************************
* Copyright (C) 2008(2010) Renesas Electronics Corporation. All rights reserved.
*""FILE COMMENT""************ Technical reference data *************************
*   System Name : SH7671 Sample Program
*   File Name   : ether.c
*   Abstract    : Example of setting for reception of Ethernet frames
*   Version     : 1.00.01
*   Device      : SH7671
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.03.00).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                               (Ver.9.01 Release01).
*   OS          : None
*   H/W Platform: M3A-HS71(CPU board)
*   Description :
***********************************************************************
*   History     : Mar.05,2008 ver.1.00.00
*               : May 10,2010 ver.1.00.01 Changed the company name and device name
*""FILE COMMENT END""*************************************************************/
#include "machine.h"
#include "string.h"
#include "iodefine.h"
#include "defs.h"
#include "phy.h"
#include "ether.h"
#include "siic.h"

```

3.4 Sample program list "ether.c" (2)

```
53    /* **** Macro definition **** */
54    #define DEVADDR_EEPROM 0          /* Dependence on the pin allocation of the EEPROM */
55    #define ROMADDR_MAC 0         /* Location for storage of the MAC address in the EEPROM */
56    #define DEFAULT_MAC_H 0x00010203  /* For debugging */
57    #define DEFAULT_MAC_L 0x00000405
58    #define MACSET_OK 0
59    #define MACSET_NG -1
60
61    /* **** Prototype declaration **** */
62    void main(void);
63    void lan_send_handler( unsigned long status );
64    static void lan_desc_create( void );
65    static void lan_reg_reset( void );
66    static void lan_reg_set( int link );
67    static int lan_set_mac( void );
68    /* **** Declaration of variables **** */
69    /* ---- Descriptor ---- */
70    #pragma section ETH_DESC                  /* Allocated to a 16-byte boundary */
71    static volatile TXRX_DESCRIPTOR_SET desc;    /* Descriptor area */
72    #pragma section
73    /* ---- Buffer ---- */
74    #pragma section ETH_BUFF                  /* Allocated to a 16-byte boundary */
75    static volatile TXRX_BUFFER_SET buf;      /* Area for transmission/reception buffer */
76    #pragma section
77    /* ---- MAC address ---- */
78    static unsigned long my_macaddr_h;
79    static unsigned long my_macaddr_l;
80    /* ---- Other ---- */
81    static volatile int c_recv = 0;           /* received frame counter */
82
```

3.5 Sample program list "ether.c" (3)

```
83    /*""FUNC COMMENT""*****************************************************************
84    * ID          :
85    * Outline     : Ethernet open function
86    *-------------------------------------------------------------------------------
87    * Include     : #include "iodefine.h"
88    *             : #include "phy.h"
89    *             : #include "ether.h"
90    *-------------------------------------------------------------------------------
91    * Declaration : int lan_open(void)
92    *-------------------------------------------------------------------------------
93    * Function    : Initializes E-DMAC, EtherC, PHY, and buffer memory.
94    *             : Initialization required for Ethernet is performed within this function,
95    *             : and it enables operations for transmission/reception.
96    *             : If the setting to enable transmission/reception operations is not
97    *             : possible, an error code is returned.
98    *-------------------------------------------------------------------------------
99    * Argument    : void
100   *-------------------------------------------------------------------------------
101   * ReturnValue : OPEN_OK(0) : Success in opening
102   *             : OPEN_NG(-1): Failure in opening
103   *-------------------------------------------------------------------------------
104   * Notice      :
105   *""FUNC COMMENT END""*************************************************************/
106   int lan_open(void)
107   {
108     int link;
109
110     /* ==== PFC setting ==== */
111     // PORT.PBCRL1.BIT.PB6MD = 1; /* Setting for usage on the DK30686 board */
112     PORT.PCCRH1.WORD = 0x0155; /* EtherC function */
113     PORT.PCCRL1.WORD = 0x5555;
114     PORT.PCCRL2.WORD = 0x5555;
115     /* ==== Release of EtherC/EDMAC from module standby ==== */
116     CPG.STBCR4.BIT.MSTP40 = 0;
117     /* ==== Stop EtherC and E-DMAC === */
118     lan_reg_reset();
119     /* ==== Initialize buffer memory ==== */
120     lan_desc_create();
121     /* ==== Acquire the MAC address ==== */
122     lan_set_mac();
123     /* ==== Setting of EtherC and E-DMAC ==== */
124     link = phy_autonego();  /* Confirm duplex mode */
125     if( link == NEGO_FAIL ){
126       return OPEN_NG;    /* fail in opening */
127     }
128     else{
129       lan_reg_set(link);
130     }
131     return OPEN_OK;
132   }
```

3.6 Sample program list "ether.c" (4)

```
133  /*""FUNC COMMENT""*****************************************************************
134  * ID          :
135  * Outline     : Function to close the Ethernet link
136  *-------------------------------------------------------------------------------
137  * Include     : #include "iodefine.h"
138  *             : #include "ether.h"
139  *-------------------------------------------------------------------------------
140  * Declaration : int lan_close(void)
141  *-------------------------------------------------------------------------------
142  * Function    : Stops EDMAC or EtherC.
143  *             : Also stops supply of the clock signals to EDMAC and EtherC.
144  *-------------------------------------------------------------------------------
145  * Argument    : void
146  *-------------------------------------------------------------------------------
147  * ReturnValue : int CLOSE_OK( 0): Success in closing
148  *             : CLOSE_NG(-1): Failure in closing
149  *-------------------------------------------------------------------------------
150  * Notice      :
151  *""FUNC COMMENT END""*************************************************************/
152  int lan_close( void )
153  {
154    int i;
155
156    /* ==== Reset the EtherC and E-DMAC === */
157    lan_reg_reset();
158    /* ==== Stop the EtherC and E-DMAC === */
159    CPG.STBCR4.BIT.MSTP40 = 1;
160    /* ==== Disable interrupts related to E-DMAC === */
161    INTC.IPR12.BIT._ETC = 0;
162
163    return CLOSE_OK;
164  }
165
166  /*""FUNC COMMENT""*****************************************************************
167  * ID           :
168  * Outline      : Ethernet frame reception function
169  *-------------------------------------------------------------------------------
170  * Include      : #include "ether.h"
171  *              : #include "iodefine.h"
172  *-------------------------------------------------------------------------------
173  * Declaration  : int lan_recv (unsigned char *addr )
174  *-------------------------------------------------------------------------------
175  * Function     : Copies a received frame to the specified buffer.
176  *              : If there is no received frame, a loop is set up to wait for one.
177  *              : Processing should proceed until the number of received frames is
178  *              : the same as the number of descriptors.
179  *-------------------------------------------------------------------------------
180  * Argument     : unsigned char addr: I : First address of the buffer
181  *              :                       : where received frames are to be stored
182  *-------------------------------------------------------------------------------
183  * ReturnValue  : int : Number of bytes in the received frame (or 0 for error in reception)
184  *-------------------------------------------------------------------------------
185  * Notice       :
186  *""FUNC COMMENT END""*************************************************************/
```

3.7 Sample program list "ether.c" (5)

```
187    int lan_recv( unsigned char *addr )
188    {
189      int i;
190      int pri;
191      int ret = 0;
192      EDMAC_RECV_DESC *p;
193
194      /* ==== Wait for reception ==== */
195      while (c_recv <= 0 ){
196        ;/* wait */
197      }
198      /* ==== Decrement the interrupt count ==== */
199      pri = INTC.IPR12.BIT._ETC;              /* Exclusive control (interrupt disabled) */
200      INTC.IPR12.BIT._ETC = 0;
201      --c_recv;
202      INTC.IPR12.BIT._ETC = pri;
203
204      /* ==== Copy the received frame ==== */
205      p = desc.pRecv_end;
206      if( p->rd0.BIT.RFE == 0 ){
207        memcpy(addr, p->rd2.RBA, p->rd1.RDL);
208        ret = p->rd1.RDL;
209      }
210      /* ---- Receive error ---- */
211      else{
212        p->rd0.LONG &= 0x70000000;        /* Processing for the error flags*/
213        ret = 0;                          /* 0 for error in reception */
214      }
215      /* ==== Restore the descriptor to the state where reception is possible" ====*/
216      p->rd0.BIT.RACT = 1;
217      /* ---- Initiate data reception ---- */
218      if(EDMAC.EDRRR.BIT.RR == 0 ){              /* 0 must be read before writing 1 */
219           EDMAC.EDRRR.BIT.RR = 1;
220      }
221      /* ==== Update the current pointer value ==== */
222      desc.pRecv_end = p->pNext;
223
224      return ret;
225    }
226
```

3.8 Sample program list "ether.c" (6)

```
227    /*""FUNC COMMENT""*****************************************************************
228    * ID           :
229    * Outline      : Construction function for descriptors
230    *-----------------------------------------------------------------------------
231    * Include      : #include "ether.h"
232    *-----------------------------------------------------------------------------
233    * Declaration  : static void lan_desc_create (void)
234    *-----------------------------------------------------------------------------
235    * Function     : Initializes the transmission/reception buffer and descriptor
236    *              : required for the Ethernet link. One frame/one buffer is assumed.
237    *-----------------------------------------------------------------------------
238    * Argument     : void
239    *-----------------------------------------------------------------------------
240    * ReturnValue  : void
241    *-----------------------------------------------------------------------------
242    * Notice       :
243    *""FUNC COMMENT END""*************************************************************/
244    static void lan_desc_create( void )
245    {
246      int i;
247      /* ==== Construct the area for the descriptor ==== */
248      /* ---- Memory is cleared ---- */
249      memset(&desc, 0, sizeof(desc) );
250      /* ---- Transmit descriptor ---- */
251      for(i=0; i<NUM_OF_TX_DESCRIPTOR; i++){
252        desc.send[i].td2.TBA = buf.send[i];       /* TD2 */
253        desc.send[i].td1.TDL = 0;                 /* TD1 */
254        desc.send[i].td0.LONG= 0x30000000;        /* TD0:1frame/1buf, transmission disabled */
255        if( i != (NUM_OF_TX_DESCRIPTOR-1) ){      /* pNext */
256            desc.send[i].pNext = &desc.send[i+1];
257        }
258      }
259      desc.send[i-1].td0.BIT.TDLE = 1;
260      desc.send[i-1].pNext = &desc.send[0];
261      /* ---- Receive descriptor ---- */
262      for(i=0; i<NUM_OF_RX_DESCRIPTOR; i++){
263        desc.recv[i].rd2.RBA = buf.recv[i];       /* RD2 */
264        desc.recv[i].rd1.RBL = SIZE_OF_BUFFER;    /* RD1 */
265        desc.recv[i].rd0.LONG= 0xb0000000;        /* RD0:1frame/1buf, reception enabled */
266        if( i != (NUM_OF_RX_DESCRIPTOR-1) ){      /* pNext */
267            desc.recv[i].pNext = &desc.recv[i+1];
268        }
269      }
270      desc.recv[i-1].rd0.BIT.RDLE = 1;                  /* Set the last descriptor */
271      desc.recv[i-1].pNext = &desc.recv[0];
272
273      /* ---- Initialize the descriptor management information ---- */
274      desc.pSend_top = &desc.send[0];
275      desc.pRecv_end = &desc.recv[0];
276
277      /* ==== Construct the buffer area ==== */
278      /* ---- Clear the area ---- */
279      memset(&buf, 0, sizeof(buf) );
280    }
```

3.9 Sample program list "ether.c" (7)

```
281  /*""FUNC COMMENT""*****************************************************************
282  * ID           :
283  * Outline      : Function for initializing the EtherC and E-DMAC registers
284  *-------------------------------------------------------------------------------
285  * Include      : #include "iodefine.h"
286  *-------------------------------------------------------------------------------
287  * Declaration  : static void lan_reg_reset(void)
288  *-------------------------------------------------------------------------------
289  * Function     : Resets the registers of EtherC and E-DMAC. This function secures
290  *              : the required reset period of at least 64 bus clock cycles.
291  *-------------------------------------------------------------------------------
292  * Argument     : void
293  *-------------------------------------------------------------------------------
294  * ReturnValue  : void
295  *-------------------------------------------------------------------------------
296  * Notice       :
297  *""FUNC COMMENT END""*************************************************************/
298  static void lan_reg_reset(void)
299  {
300    volatile int j = 100; /* Wait for over 64 cycles of the bus clock */
301
302    /* ---- Software reset ---- */
303    EDMAC.EDMR.BIT.SWR = 1;
304
305    /* ---- Secure the reset time ---- */
306    while(j--){
307      /* Wait for over 64 cycles of the bus clock  */
308    }
309  }
310  /*""FUNC COMMENT""*****************************************************************
311  * ID           :
312  * Outline      : Setting of the EhterC and E-DMAC registers
313  *-------------------------------------------------------------------------------
314  * Include      : #include "iodefine.h"
315  *              : #include "phy.h"
316  *              : #include "ether.h"
317  *-------------------------------------------------------------------------------
318  * Declaration  : void lan_reg_set(int link)
319  *-------------------------------------------------------------------------------
320  * Function     : Initializes the E-DMAC and EtherC modules.
321  *              : Both transmission and reception are enabled.
322  *-------------------------------------------------------------------------------
323  * Argument     : int link  : I     : Duplex mode makes a setting for EhterC.
324  *              :                   : Uses the value returned by the phy_autonego function.
325  *-------------------------------------------------------------------------------
326  * ReturnValue  : void
327  *-------------------------------------------------------------------------------
328  * Notice       : This function should be executed only when transmission/reception
329  *              : operations are disabled after E-DMAC has been reset by software.
330  *""FUNC COMMENT END""*************************************************************/
```

3.10 Sample program list "ether.c" (8)

```
331   static void lan_reg_set( int link )
332   {
333     /* ==== EDMAC ==== */
334     EDMAC.EDMR.LONG = 0x00000000;   /* No endian conversion (big endian)             */
335                                     /* 16-byte descriptor length                     */
336     EDMAC.TDLAR = &desc.send[0];    /* Start of the transmission descriptor list     */
337     EDMAC.RDLAR = &desc.recv[0];    /* Start of the reception descriptor list        */
338     EDMAC.TRSCER.LONG = 0x00000000; /* Copy all status information to the descriptor */
339     EDMAC.TFTR = 0x00;              /* Transmission FIFO threshold (store & forward) */
340     EDMAC.FDR.BIT.TFD = 1;          /* Transmission FIFO capacity (512 bytes)        */
341     EDMAC.FDR.BIT.RFD = 1;          /* Reception FIFO capacity (512 bytes)           */
342     EDMAC.RMCR.BIT.RNC = 1;         /* Consecutive reception is enabled              */
343     EDMAC.EDOCR.LONG = 0x00000000;  /* Operation continues even when an error occurs in the
      FIFO */
344     EDMAC.FCFTR.LONG = 0x00070000;  /* Set the flow control threshold Disabled by EtherC */
345     EDMAC.TRIMD.BIT.TIS = 0;        /* No notice of write-back completion            */
346     /* ==== EtherC ==== */
347     EtherC.ECMR.LONG = 0x00000000;  /* Flow control is disabled                      */
348                                     /* Recognize the CRC frame as an error           */
349                                     /* Magic packet detection is not permitted       */
350                                     /* Reception is disabled                         */
351                                     /* Transmission is disabled                      */
352                                     /* Internal loopback is not performed            */
353                                     /* External loopback is not performed            */
354                                     /* Duplex mode (half-duplex)                     */
355                                     /* Promiscuous mode operation is not performed   */
356     if( link == FULL_TX || link == FULL_10M ){
357          EtherC.ECMR.BIT.DM = 1;    /* Set the mode as full-duplex                   */
358     }
359     EtherC.MAHR = my_macaddr_h;     /* Set the MAC address                           */
360     EtherC.MALR = my_macaddr_l;
361     EtherC.RFLR = 0x000;            /* Maximum length of received frames (1518 bytes)*/
362     EtherC.IPGR = 0x14;             /* Gap between packets (96-bit time)             */
363     /* ==== interrupt-related ==== */
364     EDMAC.EESR.LONG = 0x47FF0F9F;   /* Clear all status information (cleared by writing 1) */
365     EDMAC.EESIPR.LONG = EDMAC_EESIPR_INI_SEND | EDMAC_EESIPR_INI_RECV |
      EDMAC_EESIPR_INI_EtherC;
366                                 /* Enable Transmission/reception and EtherC interrupts */
367     EtherC.ECSR.LONG = 0x00000017;  /* Clear all status information (cleared by writing 1) */
368     EtherC.ECSIPR.LONG = EtherC_ECSIPR_INI; /* Interrupt enabled                      */
369     INTC.IPR12.BIT._ETC = 5;
370                     /* Assign the fifth priority level to the E-DMAC interrupt (EINT0) */
371     /* ==== Setting to enable transmission and reception ==== */
372     /* ---- EtherC ---- */
373     EtherC.ECMR.BIT.RE = 1;     /* Enable reception                          */
374     EtherC.ECMR.BIT.TE = 1;     /* Enable transmission                   */
375     /* ---- E-DMAC ---- */
376     if(EDMAC.EDRRR.BIT.RR == 0){
377          EDMAC.EDRRR.BIT.RR = 1; /* Initiate data reception                  */
378     }
379   }
```

3.11 Sample program list "ether.c" (9)

```
/*""FUNC COMMENT""*****************************************************************
* ID           :
* Outline      : Transmit interrupt function
*-----------------------------------------------------------------------------
* Include      : #include "iodefine.h"
*              : #include "ether.h"
*-----------------------------------------------------------------------------
* Declaration  : void lan_send_handler (unsigned long status)
*-----------------------------------------------------------------------------
* Function     : The interrupt handler for transmission related to EDMAC (EESR).
*-----------------------------------------------------------------------------
* Argument     : unsigned long status : I    : EESR status (only bits for which
*                                     : interrupts are enabled)
*-----------------------------------------------------------------------------
* ReturnValue  : none
*-----------------------------------------------------------------------------
* Notice       : No operation is performed in this sample program.
*""FUNC COMMENT END""*************************************************************/
void lan_send_handler(unsigned long status )
{
}
/*""FUNC COMMENT""*****************************************************************
* ID           :
* Outline      : receive interrupt function
*-----------------------------------------------------------------------------
* Include      : #include "iodefine.h"
*              : #include "ether.h"
*-----------------------------------------------------------------------------
* Declaration  : void lan_recv_handler (unsigned long status)
*-----------------------------------------------------------------------------
* Function     : The interrupt handler for reception related to EDMAC (EESR)
*-----------------------------------------------------------------------------
* Argument     : unsigned long status : I    : EESR status (only bits for which
*                                     : interrupts are enabled)
*-----------------------------------------------------------------------------
* ReturnValue  : none
*-----------------------------------------------------------------------------
* Notice       :
*""FUNC COMMENT END""*************************************************************/
void lan_recv_handler (unsigned long status )
{
    c_recv++;          /* Increment the counter for the number of reception interrupts*/
}
```

3.12 Sample program list "ether.c" (10)

```
423  /*""FUNC COMMENT""*****************************************************************
424  * ID           :
425  * Outline      : EtherC interrupt function
426  *-----------------------------------------------------------------------------
427  * Include      : #include "iodefine.h"
428  *              : #include "ether.h"
429  *-----------------------------------------------------------------------------
430  * Declaration  : void lan_etherc_handler( unsigned long status )
431  *-----------------------------------------------------------------------------
432  * Function     : The interrupt handler related to EtherC(ECSR)
433  *-----------------------------------------------------------------------------
434  * Argument     : unsigned long status : I    : ECSR status (only bits for which
435  *                                     : interrupts are enabled)
436  *-----------------------------------------------------------------------------
437  * ReturnValue  : none
438  *-----------------------------------------------------------------------------
439  * Notice       : No operation is performed in this sample program.
440  *""FUNC COMMENT END""*************************************************************/
441  void lan_etherc_handler( unsigned long status )
442  {
443  }
444  /*""FUNC COMMENT""*****************************************************************
445  * ID           :
446  * Outline      : MAC address setting
447  *-----------------------------------------------------------------------------
448  * Include      : #include "siic.h"
449  *-----------------------------------------------------------------------------
450  * Declaration  : static int lan_set_mac( void )
451  *-----------------------------------------------------------------------------
452  * Function     : Obtains the MAC address from the EEPROM
453  *-----------------------------------------------------------------------------
454  * Argument     : none
455  *-----------------------------------------------------------------------------
456  * ReturnValue  : MACSET_OK(0) : Success in obtaining
457  *              : MACSET_NG(-1): Failure in obtaining
458  *-----------------------------------------------------------------------------
459  * Notice       :
460  *""FUNC COMMENT END""*************************************************************/
```

3.13 Sample program list "ether.c" (11)

```
461  static int lan_set_mac( void )
462  {
463    volatile int ret, i;
464    unsigned char buf[10];
465
466    /* ==== Initialization of EEPROM driver ==== */
467    siic_Init_Driver();
468
469    /* ==== Reading the EEPROM ==== */
470    ret = siic_EepRomRW(DEVADDR_EEPROM, ROMADDR_MAC, 6, buf, SIIC_MODE_EEP_READ);
471    if (ret < SIIC_OK) {
472      /* ---- Reading failure ---- */
473      my_macaddr_h = DEFAULT_MAC_H;
474      my_macaddr_l = DEFAULT_MAC_L;
475      return MACSET_NG;
476    }
477    do{
478      ret = siic_Chk_Eep();
479      if( ret < SIIC_OK ){
480          /* ---- Reading failure ---- */
481          my_macaddr_h = DEFAULT_MAC_H;
482          my_macaddr_l = DEFAULT_MAC_L;
483          return MACSET_NG;
484      }
485    }while( ret != SIIC_OK);
486    /* ---- Success in reading ---- */
487    for(i=0; i<6; i++){
488      if( buf[i] != 0xff ){
489          break;
490      }
491    }
492    if( i == 6 ){
493      /* ---- Set the default value if the EEPROM setting has not been made ---- */
494      my_macaddr_h = DEFAULT_MAC_H;
495      my_macaddr_l = DEFAULT_MAC_L;
496    }
497    else{
498      /* ---- Set the read address ---- */
499      my_macaddr_h = buf[0];
500      my_macaddr_h <<= 8;
501      my_macaddr_h |= buf[1];
502      my_macaddr_h <<= 8;
503      my_macaddr_h |= buf[2];
504      my_macaddr_h <<= 8;
505      my_macaddr_h |= buf[3];
506      my_macaddr_l = buf[4];
507      my_macaddr_l <<= 8;
508      my_macaddr_l |= buf[5];
509    }
510    return MACSET_OK;
511  }
512
513  /* End of file */
```

3.14 Sample program list "ether.h" (1)

```
/***********************************************************************
*   DISCLAIMER
*
*   This software is supplied by Renesas Electronics Corporation and is only
*   intended for use with Renesas products. No other uses are authorized.
*
*   This software is owned by Renesas Electronics Corporation and is protected under
*   all applicable laws, including copyright laws.
*
*   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
*   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
*   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
*   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
*   DISCLAIMED.
*
*   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
*   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
*   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
*   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
*   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
*
*   Renesas reserves the right, without notice, to make changes to this
*   software and to discontinue the availability of this software.
*   By using this software, you agree to the additional terms and
*   conditions found by accessing the following link:
*   http://www.renesas.com/disclaimer
***********************************************************************
* Copyright (C) 2007(2010) Renesas Electronics Corporation. All rights reserved.
*""FILE COMMENT""************ Technical reference data *************************
*   System Name : SH7671 Sample Program
*   File Name   : ether.h
*   Abstract    : Example of setting for transmission and reception of Ethernet frames
*   Version     : 1.00.01
*   Device      : SH7671
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.03.00).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                               (Ver.9.01 Release01).
*   OS          : None
*   H/W Platform: M3A-HS71(CPU board)
*   Description :
***********************************************************************
*   History     : Jul.04,2007 ver.1.00.00
*               : May 10,2010 ver.1.00.01 Changed the company name and device name
*""FILE COMMENT END""*************************************************************/
#ifndef _ETHER_H
#define _ETHER_H

/* **** Macro definition **** */
#define NUM_OF_TX_DESCRIPTOR      4
#define NUM_OF_RX_DESCRIPTOR      4
#define NUM_OF_TX_BUFFER          4
#define NUM_OF_RX_BUFFER          4
#define SIZE_OF_BUFFER            1520    /* Byte size must be an integer multiple of 16 */

```

3.15 Sample program list "ether.h" (2)

```
55    #define OPEN_OK                   0
56    #define OPEN_NG                   -1
57    #define SEND_OK                   0
58    #define SEND_NG                   -1
59    #define CLOSE_OK                  0
60    #define CLOSE_NG                  -1
61    #define MIN_FRAME_SIZE            60
62    #define MAX_FRAME_SIZE            1514
63
64    #define EDMAC_EESIPR_INI_SEND     0x04280F00
65                                /* 0x04000000 : Transmission abort detection  *
66                                 * 0x00200000 : Transmission of Ethernet frame completed    *
67                                 * 0x00080000 : Transmission FIFO underflow   *
68                                 * 0x00000800 : Carrier not detected  *
69                                 * 0x00000400 : Detection of carrier vanishing   *
70                                 * 0x00000200 : Detection of delay collision  *
71                                 * 0x00000100 : Transmission retry over   */
72    #define EDMAC_EESIPR_INI_RECV     0x0205001F
73                                /* 0x02000000 : Detection of reception aborted   *
74                                 * 0x00040000 : Reception of Ethernet frame completed    *
75                                 * 0x00010000 : Receive FIFO overflow *
76                                 * 0x00000010 : Reception of fraction-bit frames *
77                                 * 0x00000008 : Long frame reception  *
78                                 * 0x00000004 : Short frame reception *
79                                 * 0x00000002 : PHY-LSI reception error   *
80                                 * 0x00000001 : CRC error in received frame   */
81    #define EDMAC_EESIPR_INI_EtherC   0x00400000     /* 0x00400000 : EtherC status register*/
82    #define EtherC_ECSIPR_INI         0x00000004     /* 0x00000004 : Link signal change    */
83
84    /* **** Type definition **** */
85
86    /* ==== Transmit descriptor ==== */
87    typedef union{
88        unsigned long LONG;
89        struct{
90                unsigned int TACT:1;      /* Transmission descriptor enabled               */
91                unsigned int TDLE:1;      /* The last transmit descriptor                  */
92                unsigned int TFP :2;      /* Position of the frame for transmission: 1, 0  */
93                unsigned int TFE :1;      /* Transmission frame error                      */
94                unsigned int reserved :23;/* Reservation: TFS26 to 4                       */
95                unsigned int TFS3:1;      /* No carrier detected (EESR to CND bits)        */
96                unsigned int TFS2:1;   /* Detection of carrier vanishing (EESR to DLC bits) * /
97                unsigned int TFS1:1;
98                           /* Detection of delay collision in transmission (EESR to CD bits)*/
99                unsigned int TFS0:1;      /* Transmission retry over (EESR to TRO bits)     */
100       }BIT;
101   }TD0;
102   typedef struct{
103       unsigned short TDL;               /* Transmission buffer data length    */
104       unsigned short reserved;
105   }TD1;
```

3.16 Sample program list "ether.h" (3)

```
106    typedef struct{
107        unsigned char *TBA;              /* Transmission buffer address       */
108    }TD2;
109    typedef struct tag_edmac_send_desc{
110        TD0 td0;
111        TD1 td1;
112        TD2 td2;
113        struct tag_edmac_send_desc *pNext;
114    }EDMAC_SEND_DESC;
115
116    /* ==== Receive descriptor ==== */
117    typedef union{
118        unsigned long LONG;
119        struct{
120            unsigned int RACT:1;          /* Reception descriptor enabled                  */
121            unsigned int RDLE:1;          /* The last reception descriptor                 */
122            unsigned int RFP :2;          /* Position of the receive frame: 1, 0           */
123            unsigned int RFE :1;          /* Received frame error                          */
124            unsigned int reserved1:17;    /* Reservation: TFS26 to 10                      */
125            unsigned int RFS9:1;          /* Reception FIFO overflow (EESR to RFOF bits)   */
126            unsigned int reserved2:1;     /*  : Reservation                                */
127            unsigned int RFS7:1;      /* Reception of multicast frames (EESR to RMAF bits) */
128            unsigned int reserved3:1; /*  : Reservation                                    */
129            unsigned int reserved4:1; /*  : Reservation                                    */
130            unsigned int RFS4:1;      /* Reception error; frame only contains a fraction of the
131                                         required number of bits (EESR to RRF bits)       */
132            unsigned int RFS3:1;/*Reception error; excessively long frame (EESR to RTLE bits)*/
133            unsigned int RFS2:1;/*Reception error; excessively short frame(EESR to RTSF bits)*/
134            unsigned int RFS1:1;/* PHY-LSI reception error (EESR to PRE bits)             */
135            unsigned int RFS0:1;/* CRC error detected for received frame (EESR to CERF bits)*/
136        }BIT;
137    }RD0;
138    typedef struct{
139        unsigned short RBL;                         /* Reception buffer length "         */
140        unsigned short RDL;                         /* Received data length "            */
141    }RD1;
142    typedef struct{
143        unsigned char *RBA;                         /* Reception buffer address          */
144    }RD2;
145    typedef struct tag_edmac_recv_desc{
146        RD0 rd0;
147        RD1 rd1;
148        RD2 rd2;
149        struct tag_edmac_recv_desc *pNext;
150    }EDMAC_RECV_DESC;
```

3.17 Sample program list "ether.h" (4)

```
151
152    /*====All transmission/reception descriptors (to be allocated on a 16-byte boundary)====*/
153    typedef struct{
154        EDMAC_SEND_DESC send[NUM_OF_TX_DESCRIPTOR];
155        EDMAC_RECV_DESC recv[NUM_OF_RX_DESCRIPTOR];
156        EDMAC_SEND_DESC *pSend_top;/*Position where transmission descriptors are registered */
157        EDMAC_RECV_DESC *pRecv_end; /* Reception complete/registered position of receive
       descriptors */
158    }TXRX_DESCRIPTOR_SET;
159
160    /* ==== Transmission/reception buffer (to be allocated on a 16-byte boundary) ==== */
161    /* ---- Definition of all transmission/reception buffer areas ---- */
162    typedef struct{
163        unsigned char send[NUM_OF_TX_BUFFER][SIZE_OF_BUFFER];
164        unsigned char recv[NUM_OF_RX_BUFFER][SIZE_OF_BUFFER];
165    }TXRX_BUFFER_SET;
166
167    /* **** Prototype declaration **** */
168    int lan_open(void);
169    int lan_close(void);
170    int lan_send(unsigned char *addr, int flen);
171
172
173    #endif
174
175    /* End of File */
```

3.18 Sample program list "intprg_eth.c" (1)

```
/************************************************************************
*   DISCLAIMER
*
*   This software is supplied by Renesas Electronics Corporation and is only
*   intended for use with Renesas products. No other uses are authorized.
*
*   This software is owned by Renesas Electronics Corporation and is protected under
*   all applicable laws, including copyright laws.
*
*   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
*   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
*   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
*   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
*   DISCLAIMED.
*
*   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
*   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
*   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
*   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
*   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
*
*   Renesas reserves the right, without notice, to make changes to this
*   software and to discontinue the availability of this software.
*   By using this software, you agree to the additional terms and
*   conditions found by accessing the following link:
*   http://www.renesas.com/disclaimer
************************************************************************
* Copyright (C) 2007(2010) Renesas Electronics Corporation. All rights reserved.
*""FILE COMMENT""************ Technical reference data *************************
*   System Name : SH7671 Sample Program
*   File Name   : intprg_eth.c
*   Abstract    : interrupt entry function
*   Version     : 1.00.01
*   Device      : SH7671
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.03.00).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                               (Ver.9.01 Release01).
*   OS          : None
*   H/W Platform: M3A-HS71(CPU board)
*   Description :
************************************************************************
*   History     : Sep.18,2007 ver.1.00.00
*               : May 10,2010 ver.1.00.01 Changed the company name and device name
*""FILE COMMENT END""*********************************************************/

(omitted)
```

3.19 Sample program list "intprg_eth.c" (2)

```
670    // 171 ETC EINT0
671    void INT_ETC_EINT0(void)
672    {
673      unsigned long stat_edmac;
674      unsigned long stat_EtherC;
675
676      /* ---- Clear the interrupt request flag ---- */
677      stat_edmac = EDMAC.EESR.LONG & EDMAC.EESIPR.LONG;
678                           /* Targets are restricted to allowed interrupts */
679      EDMAC.EESR.LONG = stat_edmac;
680      /* ==== Transmission-related ==== */
681      if(stat_edmac & EDMAC_EESIPR_INI_SEND ){
682        lan_send_handler(stat_edmac & EDMAC_EESIPR_INI_SEND);
683      }
684      /* ==== Reception-related ==== */
685      if( stat_edmac & EDMAC_EESIPR_INI_RECV ){
686        lan_recv_handler( stat_edmac & EDMAC_EESIPR_INI_RECV );
687      }
688      /* ==== EtherC-related ==== */
689      if( stat_edmac & EDMAC_EESIPR_INI_EtherC ){
690        /* ---- Clear the interrupt request flag ---- */
691        stat_EtherC = EtherC.ECSR.LONG & EtherC.ECSIPR.LONG;
692        /* Targets are restricted to allowed interrupts */
693        EtherC.ECSR.LONG = stat_EtherC;
694        lan_etherc_handler(stat_EtherC);
695      }
696    }
       (omitted)
```

4. References

- Software Manual
  SH-2A/SH2A-FPU Software Manual Rev. 3.00
  The latest version of the software manual can be downloaded from the Renesas Electronics website.

- Hardware Manual
  SH7670 Group Hardware Manual Rev. 2.00
  The latest version of the hardware user's manual can be downloaded from the Renesas Electronics website.

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	Dec.24.08	—	First edition issued
1.01	Oct.15.10	—	Changed the sample program (AC Switching Characteristics are removed)

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. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
   - "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
   - "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
   - "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

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: +49-211-6503-0, 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.
7F, No. 363 Fu Shing North Road Taipei, Taiwan, R.O.C.
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 1.0