SH7670 Group

Example of Setting for Transmission of Ethernet Frames

R01AN0302EJ0101
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. After the transmission of each frame is completed, transmission of the next proceeds.
- The frame transmission complete interrupt is used to judge whether frame transmission has been completed or not.

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 Reception 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) conforming 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 Ethernet controller (EtherC) to perform transmission and reception of Ethernet/IEEE802.3 frames. This 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.

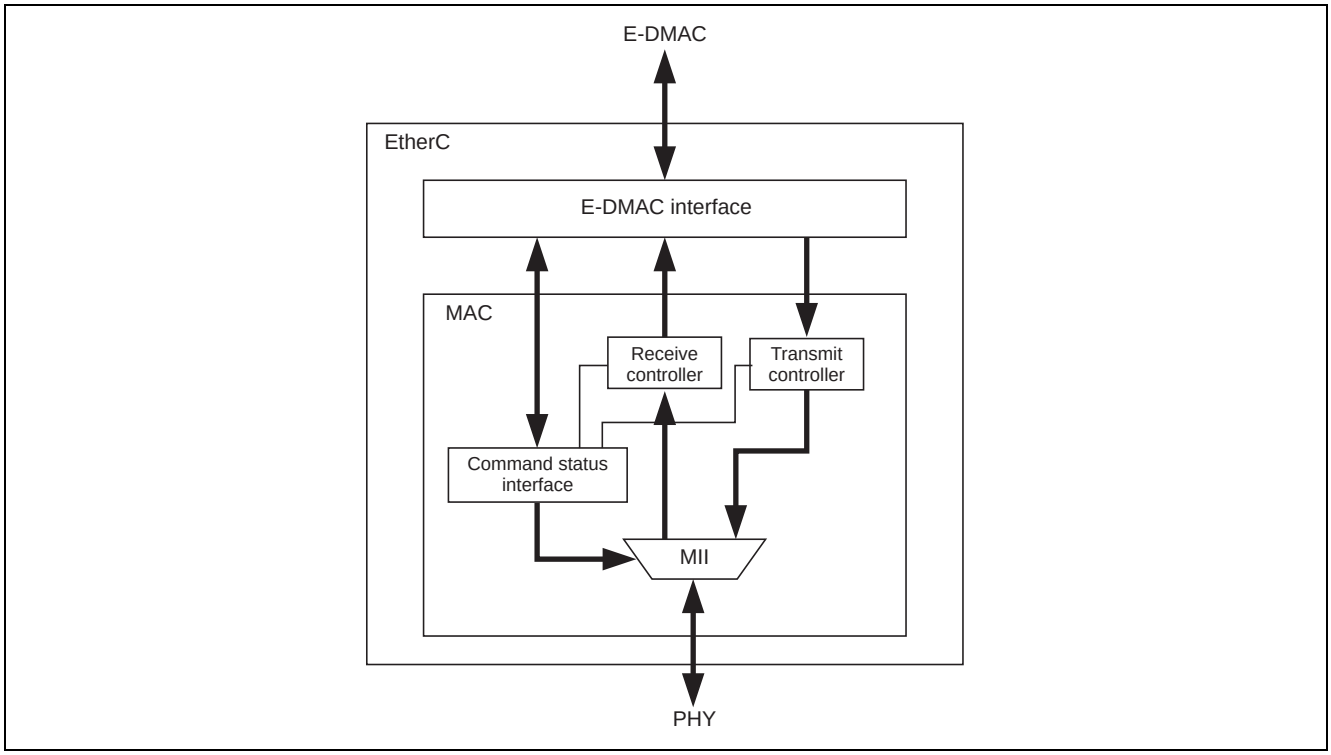


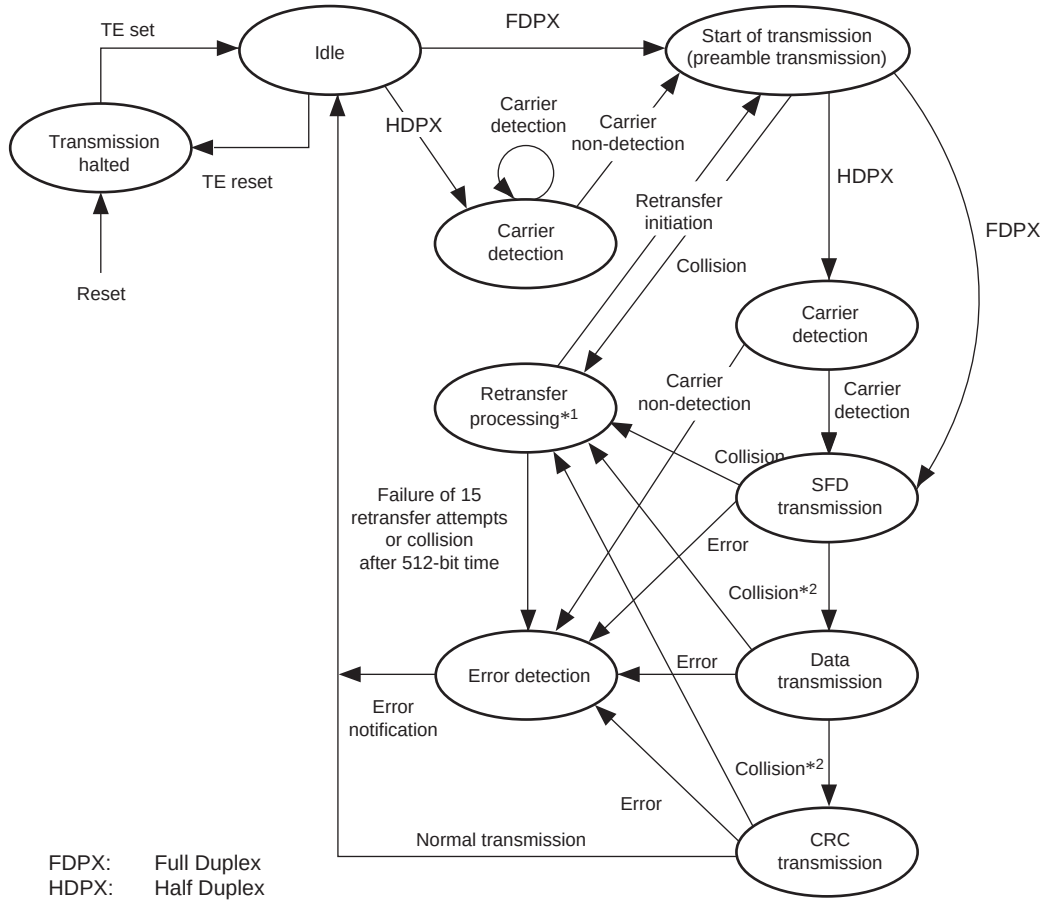
Figure 1 Configuration of EtherC

2.1.2 Overview of the EtherC Transmitter

The EtherC transmitter assembles the transmit data on the frame and outputs to MII when there is a transmit request from the E-DMAC. The data transmitted via the MII is transmitted to the lines by PHY-LSI. Figure 2 shows the state transition of the EtherC transmitter.

The following describes the flow of operations in transmission.

1. When the transmit enable (TE) bit of the EtherC mode register (ECMR) is set, the EtherC transmitter enters the idle state.
2. (A) When a request for transmission is issued by the transmitter E-DMAC while half-duplex transfer has been selected, the EtherC module attempts to detect a carrier. If it does not detect a carrier, the EtherC module sends the preamble to the RMII after a transmission delay equivalent to the time required by the frame interval. If a carrier is detected, the EtherC module waits until the carrier disappears and then sends the preamble to the RMII after a transmission delay equivalent to the time required by the frame interval.
   (B) Full-duplex transfer does not require carrier detection, so if this is selected, the preamble is sent as soon as the request for transmission is issued by the E-DMAC. In continuous transmission, however, the preamble is sent from the frame which has been transmitted at the last minute surely after a transmission delay equivalent to the time required by frame interval.
3. The EtherC transmitter sends the start frame delimiter (SFD), data, and cyclic redundancy check (CRC) code in sequence. At the end of transmission, the transmitter E-DMAC generates a frame transmission complete (TC) interrupt. If a collision occurs or the EtherC transmitter enters the carrier-not-detected state, an interrupt corresponding to the given state will be generated.
4. The EtherC transmitter enters the idle state and then, if there are more data for transmission, continues to transmit.



FDPX: Full Duplex
HDPX: Half Duplex
SFD: Start Frame Delimiter
CRC: Cyclic Redundancy Check

Notes: 1. Transmission retry processing includes both jam transmission that depends on collision detection and the adjustment of transmission intervals based on the back-off algorithm.
2. Transmission is retried only when data of 512 bits or less (including the preamble and SFD) is transmitted. When a collision is detected during the transmission of data greater than 512 bits, only jam is transmitted and transmission based on the back-off algorithm is not retried.

Figure 2 EtherC Transmitter State Transitions

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 the 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)

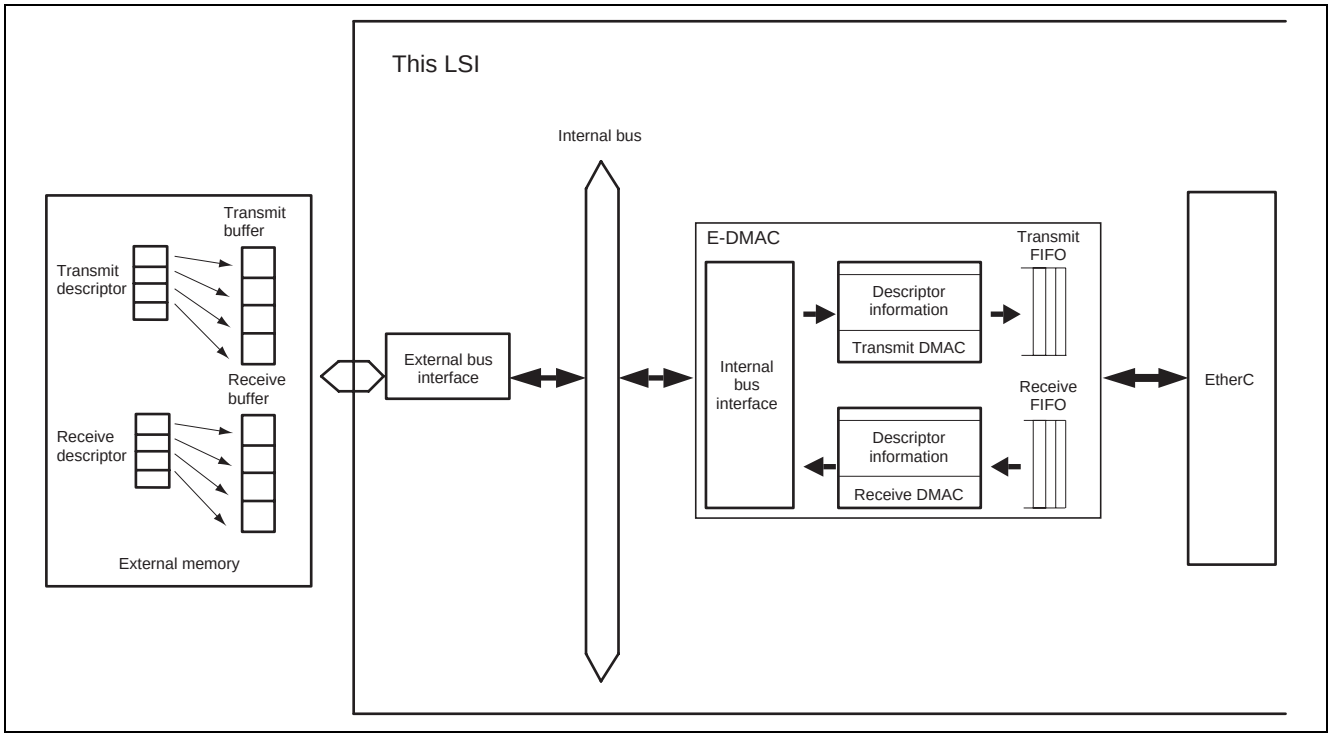


Figure 3 Configuration of E-DMAC, and Descriptors and Buffers

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 Transmit Descriptors

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

In order from its first address, a receive descriptor consists of TD0, TD1, TD2 (each is a 32-bit unit), and padding. TD0 indicates whether the descriptor is active or inactive, describes the configuration of the descriptor, and contains state information. TD1 indicates the size of the transmit buffer to which the descriptor refers, and the length of the transmit frame (TDL). TD2 indicates the address where the transmission 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 transmit descriptors, either a single descriptor or multiple descriptors can specify a single frame of transmit data (one frame/one descriptor and one frame/multi-descriptor, respectively). As an example where the one frame/multi-descriptor type of setting may be useful, multiple descriptors might be set up for data in Ethernet frames which are used in transmission every time. Specifically, data for the destination and source addresses within the Ethernet frame may be shared among multiple descriptors, with the remaining data stored in individual buffers.

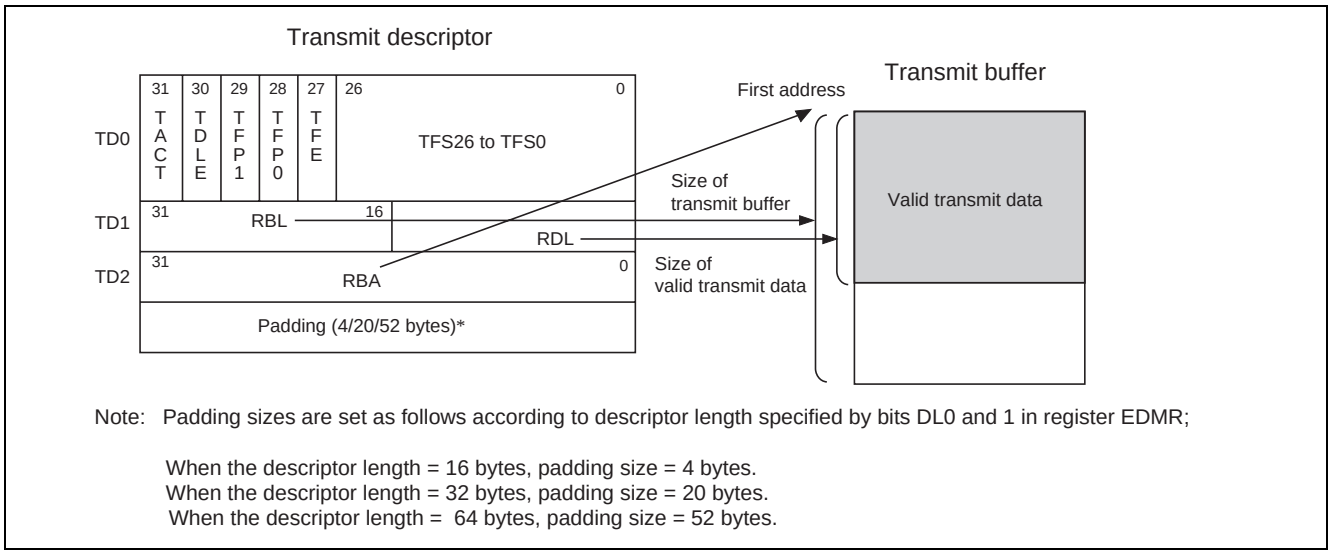


Figure 4 Relationship between Transmit Descriptor and Transmit Buffer

2.1.6 Example of Setting Transmit Descriptors

Figure 5 shows an example (one frame/one descriptor) where three transmit descriptors and three areas of the transmit buffer are in use. In this case, a single frame is transmitted in response to a single request for transmission. The transmit descriptors are simplified in the figure, with only TD0 being shown. Numbers (1), (2), etc. in the figure indicate the sequence of execution.

The Settings are as follows.

1. Due to one-frame/one-descriptor operation, the TFP1 and TFP0 bits of all descriptors are set to B'11.
2. Bits TACT, TFE, and TFS26 to TFS0 of individual descriptors are all set to 0 as the initial value.
3. In the first and second descriptors, the TDLE bit is set to 0. The TDLE 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.
4. Although the following settings have been left out of figure 5, the data length of the transmission buffer referred to by the respective descriptors is set in TDL, and the addresses where individual areas of the transmit buffer start are set in TBA.
5. Since only one frame is transmitted in response to each request in this example, only the TACT bit of the first descriptor is set to 1 for the first transmission. For the next transmission, only the TACT bit of the second descriptor is set to 1.

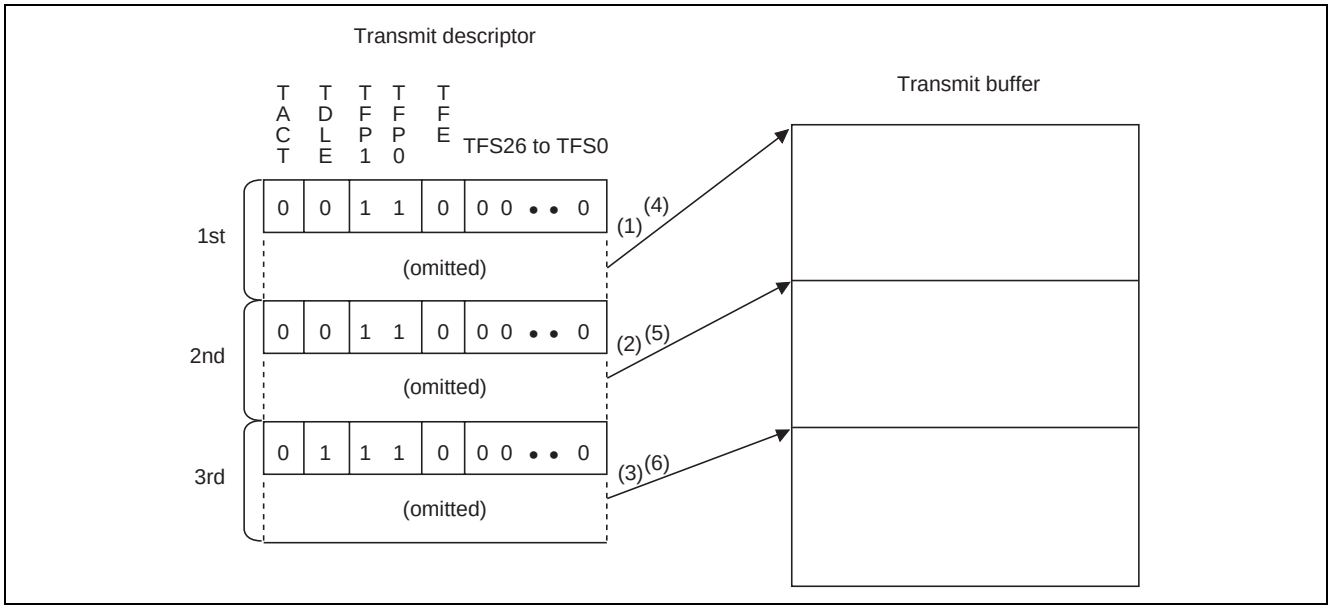


Figure 5 Relationship between Transmit Descriptor and Transmit Buffer

2.1.7 Operation of the Sample Program

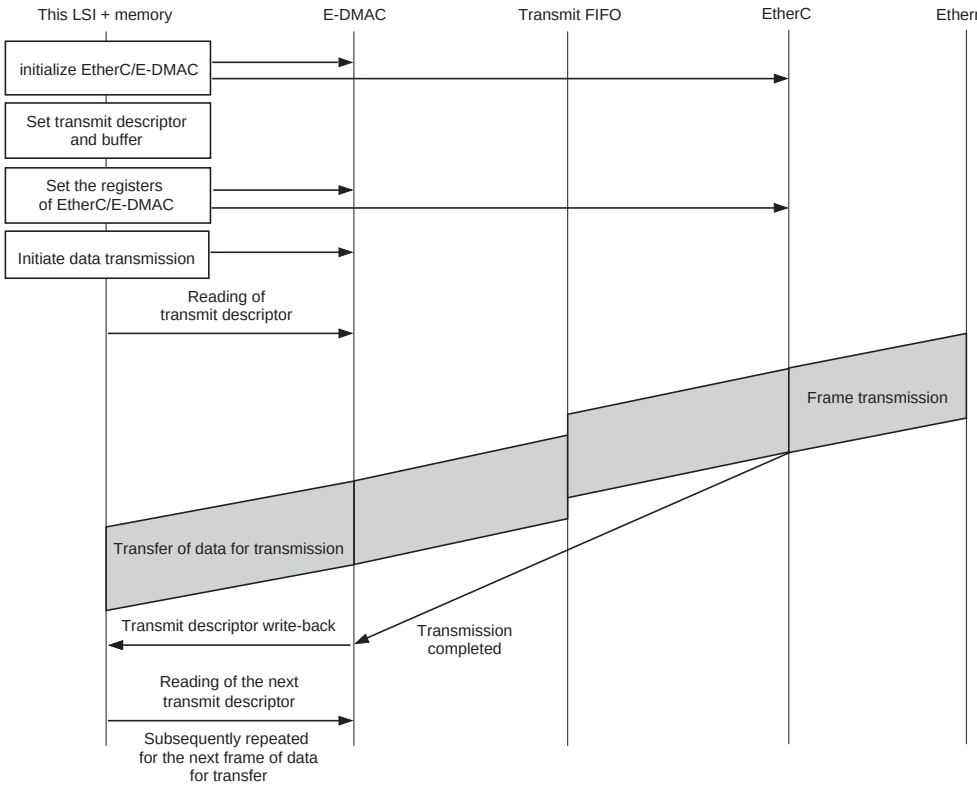
When the setting of the TE bit of the EtherC mode register (ECMR) is 1 and 1 is written to the transmit request (TR) bit in the E-DMAC transmit request register (EDTRR), the transmission section of the E-DMAC is activated. After a software reset of the EtherC and E-DMAC modules, the E-DMAC reads the descriptor indicated by the transmit descriptor list address register (TDLAR). If the setting of the TACT bit of that descriptor is 1 (active), the E-DMAC reads the frame of data for transmission in sequence from the first address for the transmit buffer as specified by TD2 of the transmit descriptor, and transfers it to the EtherC module.

The EtherC module creates a frame for transmission and starts transmitting it to the RMII. After DMA transfer equivalent to the buffer length specified in the descriptor, the value of the TFP bits determines further processing in the way described below.

- TFP = B'00 or B'10 (frame continuation):
  Writing back to the descriptor (to write 0 to the TACT bit) proceeds after the DMA transfer. The TACT bit of the next descriptor is then read.
- TFP = B'01 or B'11 (frame end):
  Writing back to the descriptor (to write 0 to the TACT bit or to write state information) proceeds after transmission of the frame is complete (writing of 0 or status to the TACT bit). The TACT bit of the next descriptor is then read.

If the TACT bit read from the next descriptor is 1, transmission of frames continues and the descriptor itself is read. If the TACT bit read from the next descriptor is 0 (inactive), the E-DMAC sets the TR bit in EDTRR to 0, and transmission ends. When 1 is written to the TR bit after its setting was 0, the transmission section of the E-DMAC is reactivated. In this case, however, the descriptor that is read will be that which follows the last descriptor to have been used in transmission.

Figure 6 shows an example of the flow of transmission (in the one-frame/one-descriptor and multiple-descriptor cases).



Supplementary Notes
EtherC/E-DMAC initialization: Both modules are software reset by setting the SWR bit in the EDMR to 1.
EtherC/E-DMAC settings: 1 is written to the TE bit in the ECMR and the TR bit in the EDTRR.
Initiation of transmission: This is done by the above action (writing 1 to the TE bit in the ECMR and TR bit in the EDTRR).
Reading of transmit descriptor: The E-DMAC automatically reads the transmit descriptor
Transfer of data for transmission: The E-DMAC performs DMA transfer to write the data for transmission to the transmit FIFO.
Transmit descriptor write-back: The E-DMAC writes 0 to the TACT bit and writes the transmission state information to the transmit descriptor.

Figure 6 Sample Flow of Transmission

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.

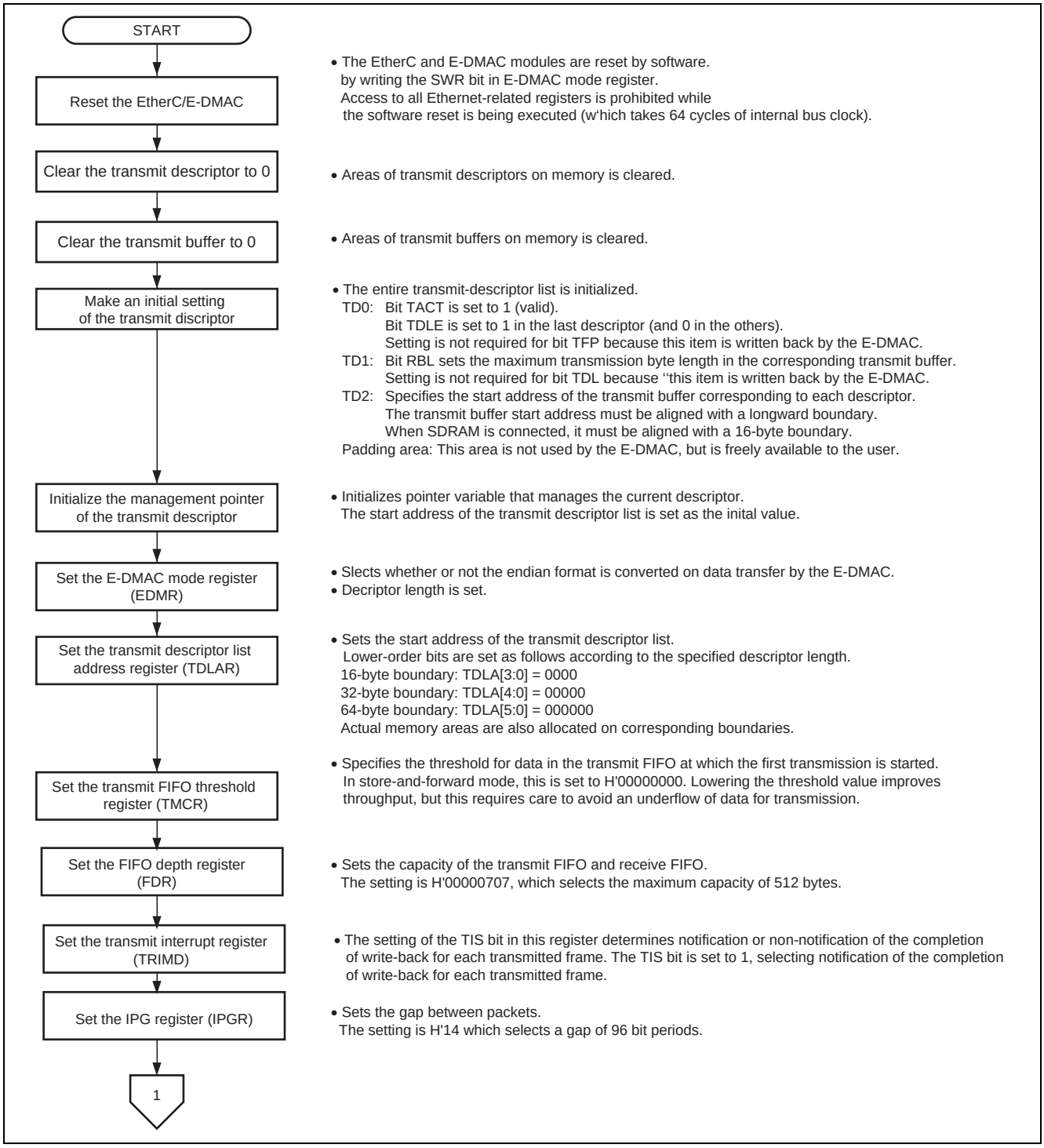
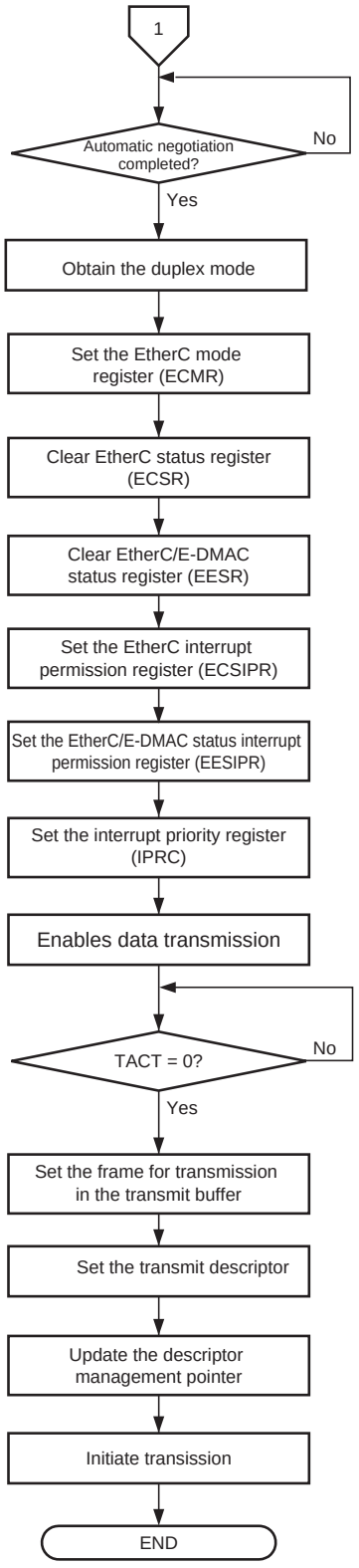


Figure 7 Example of a Flowchart for Ethernet Settings (1)



- When automatic negotiation is enabled, wait until it is completed
- Either full duplex or half duplex transfer method is obtained from the result of automatic negotiation by PHY-LSI
- Set whether CRC error frame is received as an error or not.
- Magic Packet detection is enebled if required.
- Internal/external loopback is specified if requied.
- Specify full duplex or half duplex mode
  The result of automatic negotiation by the PHY-LSI is reflected.
- The register is cleared to 0 by writing 1 to all of its bits
- The register is cleared to 0 by writing 1 to all of its bits
- Settings are made to the following bits: the link signal change interrupt enable, magic packet detection interrupt enable, and illegal carrier detection interrupt enable. Set ECIIP in EESIPR to 1 when this interrupt is used.
- Interrupts to all the bits in the EtherC/E-DMAC status register (EESR) are enabled.
- Priority level of the E-DMAC-related interrupt is set.
  Bits 11 to 8 in the IPRC are set. The setting H'0 indicates the priority level 0 (requested mask), and H'F indicates the priority level 15 (the maximum level).
- Transmission of data is enabled by setting the TE bit in the ECMR to 1.
- Check that operations for the current descriptor are not in progress.
  Use the descriptor management pointer for transmission to check that the TACT bit is cleared to 0 and the transmission has been completed or aborted.
- Data for transmission are set in buffer associated with the current descriptor.
- The setting to enable the transmission of data associated with the current descriptor is made.
  The position within the frame for transmission is set in the TFP
  (the setting is three for one frame/one descriptor).
  The data length of a tranmit frame is set in the TDL ‘
  The length in bytes of the frame for transmission is set in the TDL.
  The TACT bit is set to1 (this bit is the last to be set)
- The pointer is updated from the current descriptor to the next descriptor.
- The TR bit in EDTRR is set to 1 initiating tranmission

Figure 8 Example of a Flowchart for Ethernet Settings (2)

2.2 Operation of the Sample Program

This sample program employs the EtherC and the E-DMAC modules to transmit 10 Ethernet frames from the host personal computer at the other end. In this sample program, there are four transmit descriptors, and four areas of the transmit buffer each with 1,520 bytes.
Transmit descriptors are used in a ring structure. The frame transmit complete interrupt (TC) is used to determine when the transmission of one frame is completed and to start transmission of the next.
Data for transmission in the Ethernet frame (i.e., the frame with the exception of the preamble, start frame delimiter (SFD), and CRC section) needs to be prepared. The destination and source MAC addresses in the header must be changed to the MAC addresses of the devices in use. Note that the EtherC module does not check the source MAC address.

Figure 9 shows operating environment of the sample program, and figure 10 shows a format of the Ethernet frame.

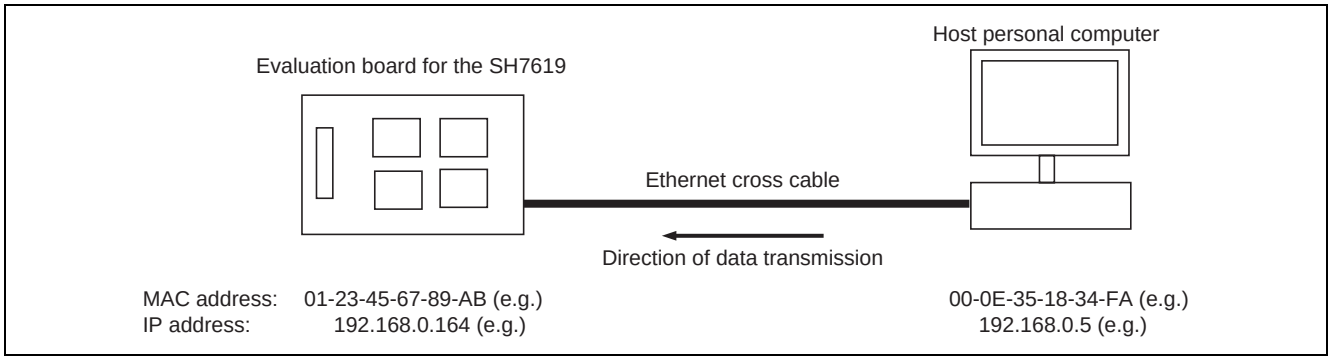


Figure 9 Operating Environment of the Sample Program

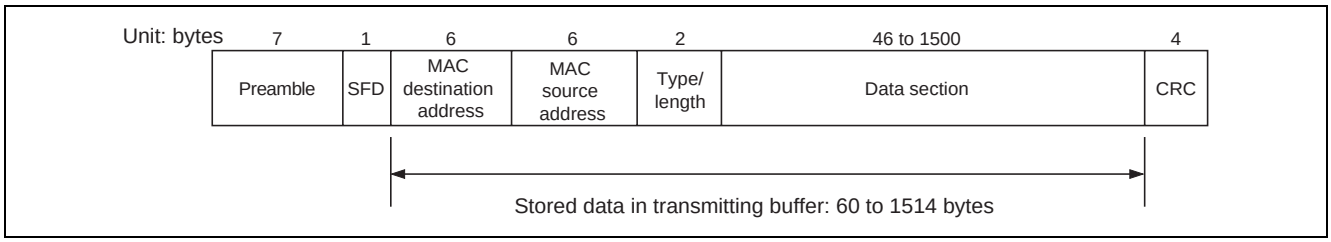


Figure 10 Ethernet Frame Format

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 transmit-descriptor structure in the sample program and an example of how the array of transmit descriptors is used.

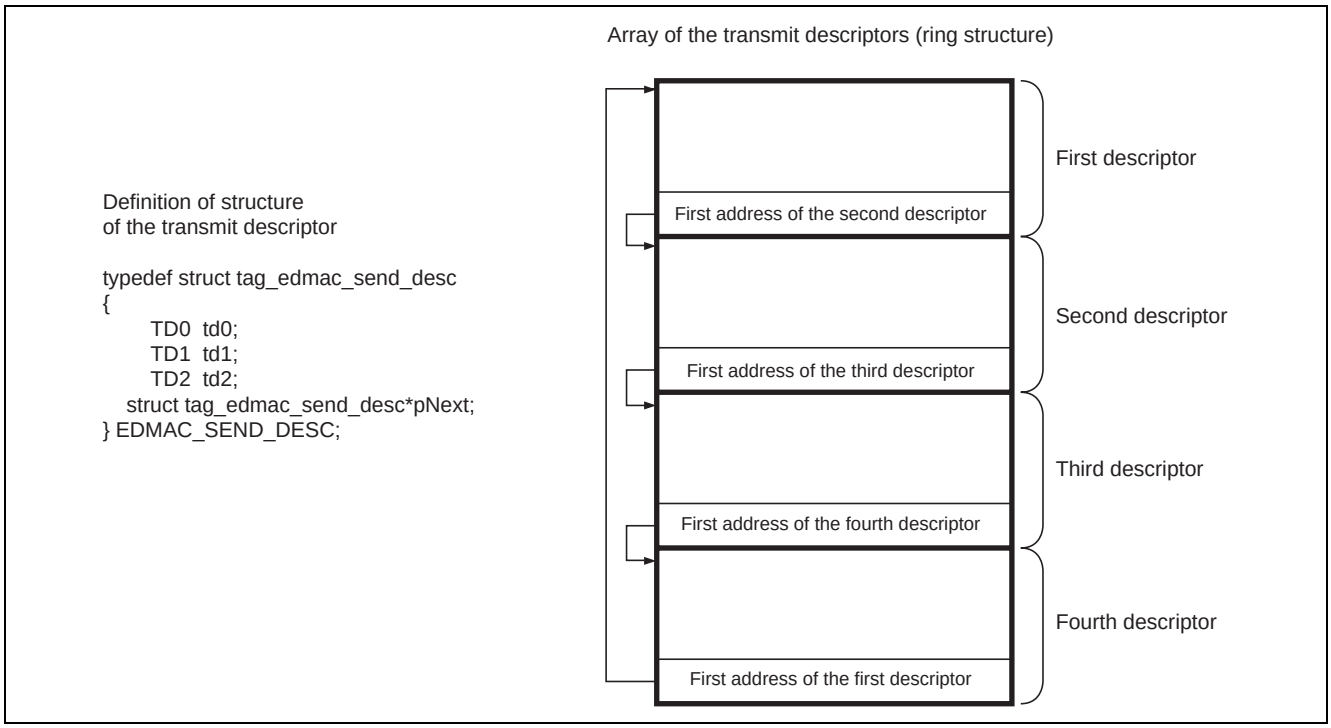


Figure 11 Definition of Transmit Descriptor and Usage Example of Transmit Descriptor Array

2.4 Sequence of Processing by the Sample Program

Figures 12 to15 show the flow of processing in the sample program. Although descriptors and the various registers of the EtherC and E-DMAC modules are initially set up for reception, processing for reception is not performed.

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)”.

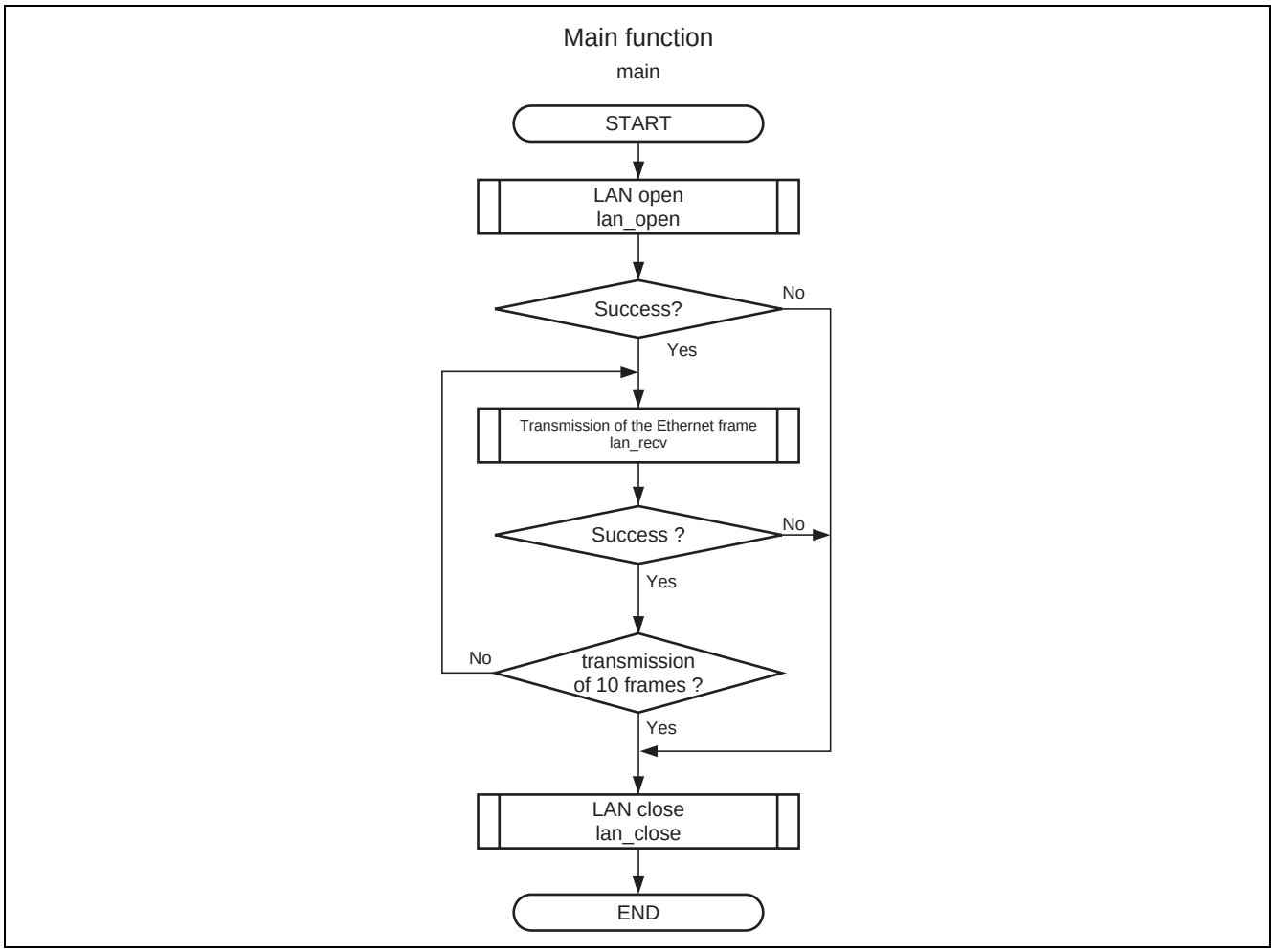


Figure 12 Flow of Handling in the Sample Program (1)

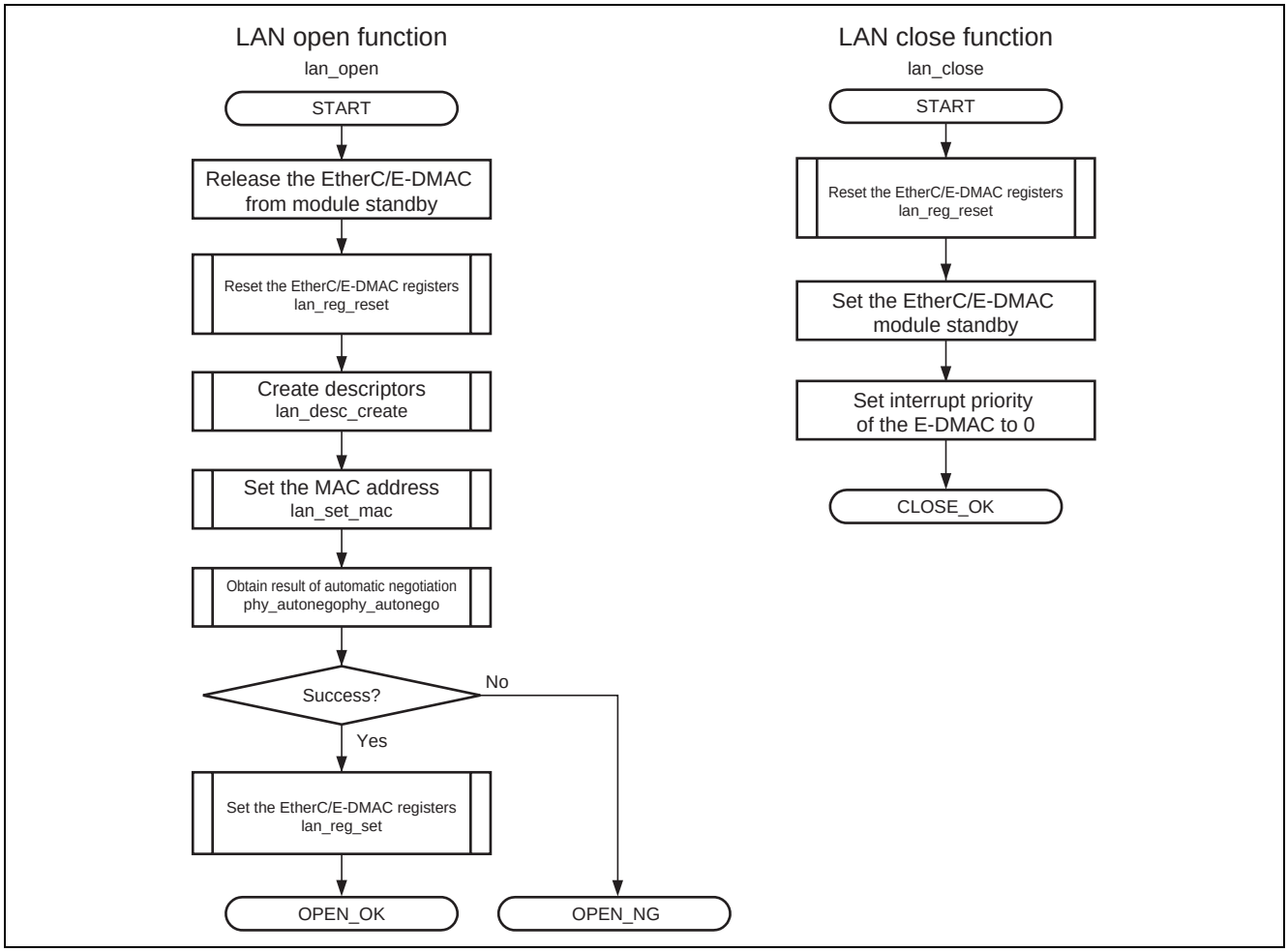


Figure 13 Flow of Handling in the Sample Program (2)

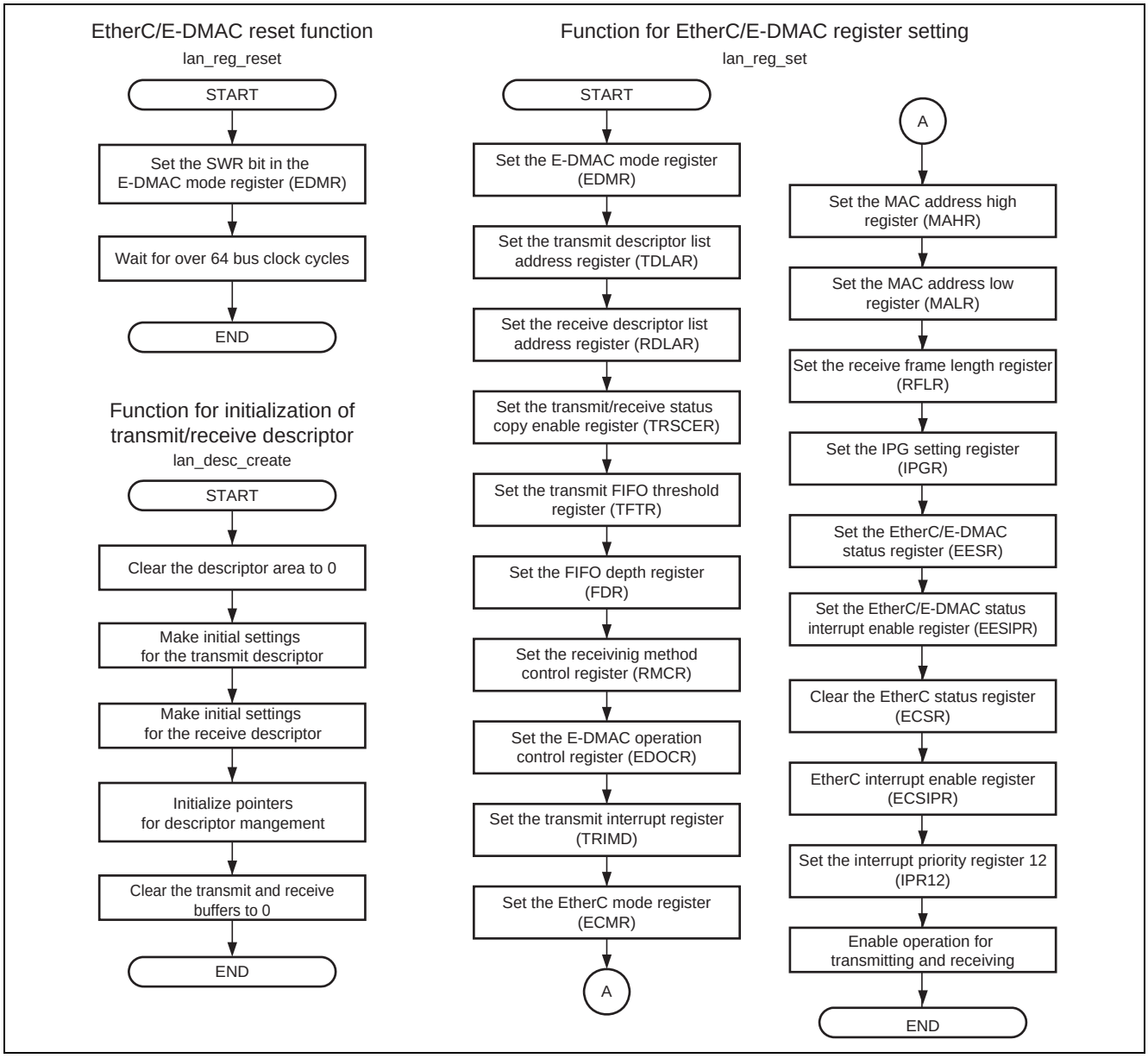


Figure 14 Flow of Handling in the Sample Program (3)

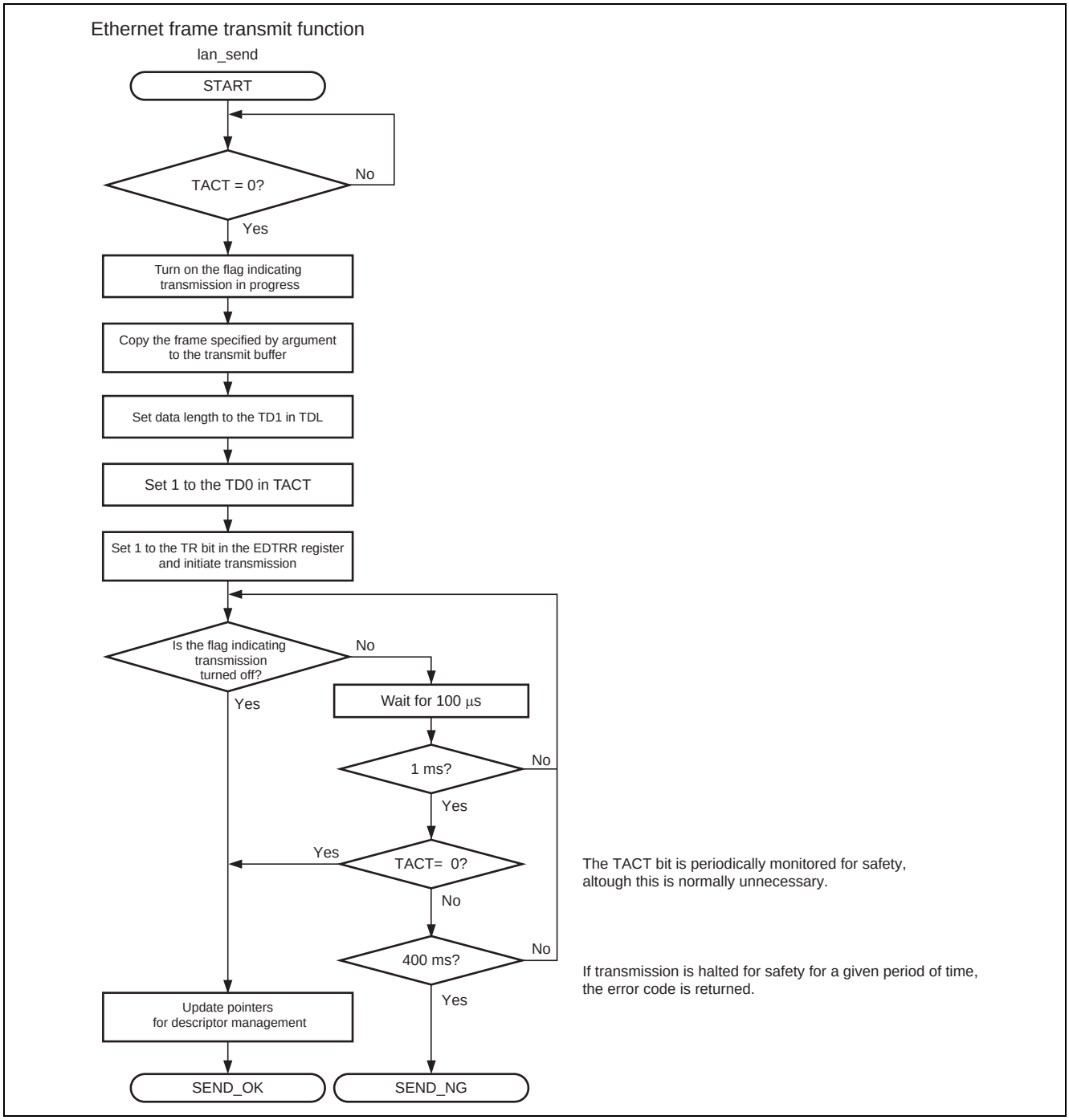


Figure 15 Flow of Handling in the Sample Program (4)

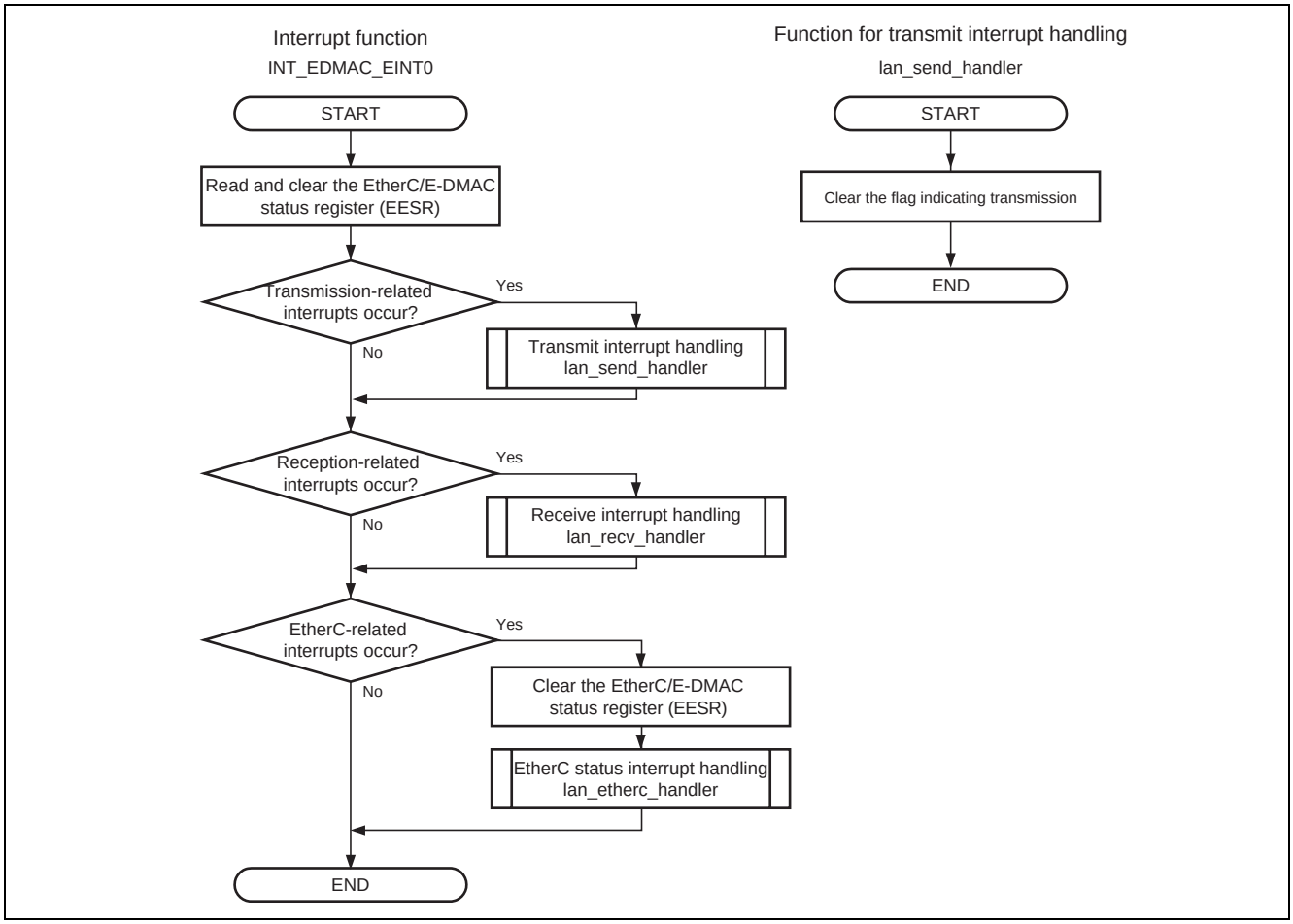


Figure 16 Flow of Handling in the Sample Program (5)

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    : Setting for Transmission of Ethernet Frames
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  *               : Apr.07,2010 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    static unsigned char s_frame[] = {
54      0xff,0xff,0xff,0xff,0xff,0xff,  /* Destination MAC address               */
55      0x00,0x01,0x02,0x03,0x04,0x05,  /* Source MAC address (00:01:02:03:04:05)*/
56      0x08,0x06,                      /* Type (ARP)                           */
57      0x00,0x01,                      /*  +--H/W type= Ethernet               */
58      0x08,0x00,                      /*  +--Protocol type= IP                */
59      0x06,0x04,                      /*  +--HW/protocol address length       */
60      0x00,0x01,                      /*  +--OPCODE= request                  */
61      0x00,0x01,0x02,0x03,0x04,0x05,  /*  +--Source MAC address (00:01:02:03:04:05) */
62      0xc0,0xa8,0x00,0x03,        /*  +--Source IP address (192.168.0.3)    */
63      0x00,0x00,0x00,0x00,0x00,0x00,  /*  +--Inquiry MAC address              */
64      0xc0,0xa8,0x00,0x05,        /*  +--Inquiry IP address (192.168.0.5)   */
65    };
66
67    /*""FUNC COMMENT""*************************************************************
68     * ID          :
69     * Outline     : Ethernet transmission sample program main function
70     *-----------------------------------------------------------------------------
71     * Include     : #include "iodefine.h"
72     *-----------------------------------------------------------------------------
73     * Declaration : void main(void)
74     *-----------------------------------------------------------------------------
75     * Function : On-chip ethernet controller (EtherC) and the dynamic memory access controller
76     *             : (E-DMAC)for the ethernet controller is used to transmit Ethernet frame.
77     *             : RTL8201CP from REALTEK is used for the PHY module.
78     *             : Multiple planes of transmit scripter is used for continuous transmission.
79     *-----------------------------------------------------------------------------
80     * Argument    : void
81     *-----------------------------------------------------------------------------
82     * ReturnValue : void
83     *-----------------------------------------------------------------------------
84     * Notice    : Mac address acquired from EEPROM is not reflected on the transmission frame.
85     *""FUNC COMMENT END""*************************************************************/
86    void main(void)
87    {
88      int i;
89      int ret;
90
91      /* ==== Ethernet initial setting ==== */
92      ret = lan_open();
93      if( ret == OPEN_OK ){
94        /* ==== 10-frame transmission start ==== */
95        for(i=0; i<10; i++){
96            /* ----transmission ---- */
97            ret = lan_send( s_frame, sizeof(s_frame) );
98            if( ret != SEND_OK ){
99                break;
100           }
101       }
102     }
103     /* ==== Ethernet transmission and reception stop ==== */
104     lan_close();
105   }
106   /* End of file */
```

3.3 Sample program list "ether.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) 2008(2010) Renesas Electronics Corporation. All rights reserved.
29  *""FILE COMMENT""************ Technical reference data *************************
30  *   System Name : SH7671 Sample Program
31  *   File Name   : ether.c
32  *   Abstract    : Setting for Transmission of Ethernet Frames
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     : Mar.05,2008 ver.1.00.00
43  *               : Apr.07,2010 ver.1.00.01 Changed the company name and device name
44  *""FILE COMMENT END""*************************************************************/
45  #include "machine.h"
46  #include "string.h"
47  #include "iodefine.h"
48  #include "defs.h"
49  #include "phy.h"
50  #include "ether.h"
51  #include "siic.h"
52
```

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

```
53  /* **** Macro declaration **** */
54  #define DEVADDR_EEPROM    0             /* Depends on the EEPROM PIN setting */
55  #define ROMADDR_MAC       0          /* MAC address storage location in 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
69  /* **** Variable declaration **** */
70  /* ---- Discriptor ---- */
71  #pragma section ETH_DESC                 /* Allocate to 16-byte boundary */
72  static volatile TXRX_DESCRIPTOR_SET desc;    /* Descriptor area */
73  #pragma section
74  /* ---- Buffer ---- */
75  #pragma section ETH_BUFF                 /* Allocate to 16-byte boundary */
76  static volatile TXRX_BUFFER_SET buf;     /* Transmit/receive buffer area */
77  #pragma section
78  /* ---- MAC address ---- */
79  static unsigned long my_macaddr_h;
80  static unsigned long my_macaddr_l;
81  /* ---- Others ---- */
82  static volatile int f_send = 0;          /* Transmitting flag */
83
```

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

```
84    /*""FUNC COMMENT""*****************************************************************
85    * ID           :
86    * Outline      : Ethernet open function
87    *-------------------------------------------------------------------------------
88    * Include      : #include "iodefine.h"
89    *              : #include "phy.h"
90    *              : #include "ether.h"
91    *-------------------------------------------------------------------------------
92    * Declaration  : int lan_open(void)
93    *-------------------------------------------------------------------------------
94    * Function     : Initialize E-DMAC, EtherC, PHY, and buffer memory
95    *              : Initialization required for Ethernet is done within the function, and
96    *              : transmit/receive is enabled. If this fails, an error is returned.
97    *-------------------------------------------------------------------------------
98    * Argument     : void
99    *-------------------------------------------------------------------------------
100   * ReturnValue  : OPEN_OK(0)   : Open successful
101   *              : OPEN_NG(-1)  : Open failed
102   *-------------------------------------------------------------------------------
103   * Notice       :
104   *""FUNC COMMENT END""*************************************************************/
105   int lan_open(void)
106   {
107     int link;
108
109     /* ==== PFC setting ==== */
110   //  PORT.PBCRL1.BIT.PB6MD = 1; /* Setting for using the DK30686 board */
111     PORT.PCCRH1.WORD = 0x0155; /* EtherC function */
112     PORT.PCCRL1.WORD = 0x5555;
113     PORT.PCCRL2.WORD = 0x5555;
114     /* ==== Release EtherC/EDMAC module standby mode ==== */
115     CPG.STBCR4.BIT.MSTP40 = 0;
116     /* ==== EtherC,E-DMAC halted === */
117     lan_reg_reset();
118     /* ==== Buffer initialization ==== */
119     lan_desc_create();
120     /* ==== Acquire MAC address ==== */
121     lan_set_mac();
122     /* ==== EtherC,E-DMAC setting ==== */
123     link = phy_autonego(); /* Check duplex mode */
124     if( link == NEGO_FAIL ){
125       return OPEN_NG; /* OPEN failed */
126     }
127     else{
128       lan_reg_set(link);
129     }
130     return OPEN_OK;
131   }
```

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

```
/*""FUNC COMMENT""*****************************************************************
* ID           :
* Outline      : Ethernet close function
*-----------------------------------------------------------------------------
* Include      : #include "iodefine.h"
*              : #include "ether.h"
*-----------------------------------------------------------------------------
* Declaration  : int lan_close(void)
*-----------------------------------------------------------------------------
* Function     : EDMAC/EtherC EDMAC/EtherC halted
*              : Clock supply to EDMAC/EtherC stops
*-----------------------------------------------------------------------------
* Argument     : void
*-----------------------------------------------------------------------------
* ReturnValue  : int CLOSE_OK( 0) : Close successful
*              : CLOSE_NG(-1)     : Close failed
*-----------------------------------------------------------------------------
* Notice       :
*""FUNC COMMENT END""*************************************************************/
int lan_close( void )
{
  int i;

  /* ==== Reset EtherC,E-DMAC === */
  lan_reg_reset();
  /* ==== EtherC,E-DMAC halted === */
  CPG.STBCR4.BIT.MSTP40 = 1;
  /* ==== E-DMAC-related interrupts are disabled=== */
  INTC.IPR12.BIT._ETC = 0;

  return CLOSE_OK;
}

/*""FUNC COMMENT""*****************************************************************
* ID           :
* Outline      : Frame transmission function
*-----------------------------------------------------------------------------
* Include      : #include "ether.h"
*              : #include "iodefine.h"
*-----------------------------------------------------------------------------
* Declaration  : int lan_send( unsigned char *addr, int flen )
*-----------------------------------------------------------------------------
* Function     : Specified frame is copied and transmitted to the buffer registered
*              : in the transmit descriptor. Wait processing continues until transmission
*              : is completed. For safety, EDMAC is periodically monitored.
*-----------------------------------------------------------------------------
* Argument     : None
*-----------------------------------------------------------------------------
* ReturnValue  : SEND_OK(0)   : Registration successful
*              : SEND_NG(-1)  : Registration failed
*-----------------------------------------------------------------------------
* Notice       :
*""FUNC COMMENT END""*************************************************************/
```

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

```
185  int lan_send( unsigned char *addr, int flen )
186  {
187    int i;
188    int t1ms = 0;
189    int t400ms = 0;
190
191    /* ==== Check that data is not being transmitted ==== */
192    while( desc.pSend_top->td0.BIT.TACT == 1 ){
193      ;/* wait */
194    }
195    /* ==== Set the flag for transmission in process ==== */
196    f_send = 1;                                  /* Clear by transmit-end interrupt */
197
198    /* ==== Updated transmit descriptor ==== */
199    memcpy( desc.pSend_top->td2.TBA, addr, flen ); /* Transmitted data */
200    if( flen < 60 ){                               /* Minimum frame of 60 bytes */
201      memcpy( (desc.pSend_top->td2.TBA)+flen, 0, 60-flen );    /* Padding */
202      flen = 60;
203    }
204    desc.pSend_top->td1.TDL = flen; /* Data length" */
205    desc.pSend_top->td0.BIT.TACT = 1; /* Transmission enabled */
206
207    /* ==== Activate if transmission is stopped ==== */
208    if( EDMAC.EDTRR.BIT.TR == 0 ){ /* Check by reading data */
209         EDMAC.EDTRR.BIT.TR = 1;
210    }
211    /* ==== Check transmission completion ==== */
212    while( f_send ){
213      for( i=LOOP_100us; i>0; i-- ){
214          ;/* 100us wait */
215      }
216      /* ---- Check descriptor when 1 ms elapsed ---- */
217      if( ++t1ms > 10 ){
218          t1ms = 0;
219          if( desc.pSend_top->td0.BIT.TACT == 0 ){
220              break;
221          }
222      }
223      /* ---- If 400 ms has elapsed judge that EDMAC operation stopped ---- */
224      if( ++t400ms > 4000 ){
225          t400ms = 0;
226          return SEND_NG;
227      }
228    }
229    /* ==== Update current pointer ==== */
230    desc.pSend_top = desc.pSend_top->pNext;
231
232    return SEND_OK;
233  }
234
```

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

```
235    /*""FUNC COMMENT""*****************************************************************
236    * ID           :
237    * Outline      : Descriptor configuration function
238    *-----------------------------------------------------------------------------
239    * Include      : #include "ether.h"
240    *-----------------------------------------------------------------------------
241    * Declaration  : static void lan_desc_create( void )
242    *-----------------------------------------------------------------------------
243    * Function     : Initialize transmit/receive buffer required for Ethernet and
244    *              : initialize descriptor. One frame/one buffer is assumed.
245    *-----------------------------------------------------------------------------
246    * Argument     : void
247    *-----------------------------------------------------------------------------
248    * ReturnValue  : void
249    *-----------------------------------------------------------------------------
250    * Notice       :
251    *""FUNC COMMENT END""*************************************************************/
252    static void lan_desc_create( void )
253    {
254      int i;
255      /* ==== Descriptor area configuration ==== */
256      /* ---- Memory area ---- */
257      memset(&desc, 0, sizeof(desc) );
258      /* ---- Transmit descriptor ---- */
259      for(i=0; i<NUM_OF_TX_DESCRIPTOR; i++){
260        desc.send[i].td2.TBA = buf.send[i];   /* TD2 */
261        desc.send[i].td1.TDL = 0;             /* TD1 */
262        desc.send[i].td0.LONG= 0x30000000;    /* TD0:1frame/1buf, transmission disabled*/
263        if( i != (NUM_OF_TX_DESCRIPTOR-1) ){      /* pNext */
264            desc.send[i].pNext = &desc.send[i+1];
265        }
266      }
267      desc.send[i-1].td0.BIT.TDLE = 1;
268      desc.send[i-1].pNext = &desc.send[0];
269      /* ---- Receive descriptor ---- */
270      for(i=0; i<NUM_OF_RX_DESCRIPTOR; i++){
271        desc.recv[i].rd2.RBA = buf.recv[i];       /* RD2 */
272        desc.recv[i].rd1.RBL = SIZE_OF_BUFFER;    /* RD1 */
273        desc.recv[i].rd0.LONG= 0xb0000000;        /* RD0:1frame/1buf reception enabled */
274        if( i != (NUM_OF_RX_DESCRIPTOR-1) ){      /* pNext */
275            desc.recv[i].pNext = &desc.recv[i+1];
276        }
277      }
278      desc.recv[i-1].rd0.BIT.RDLE = 1;                  /* Set the last descriptor */
279      desc.recv[i-1].pNext = &desc.recv[0];
280
281      /* ---- Initialize descriptor management information ---- */
282      desc.pSend_top = &desc.send[0];
283      desc.pRecv_end = &desc.recv[0];
284
285      /* ==== Buffer area configuration ==== */
286      /* ---- Clear the area ---- */
287      memset(&buf, 0, sizeof(buf) );
288    }
```

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

```
289  /*""FUNC COMMENT""*****************************************************************
290  * ID           :
291  * Outline      : EtherC,E-DMAC registers initialization function
292  *-------------------------------------------------------------------------------
293  * Include      : #include "iodefine.h"
294  *-------------------------------------------------------------------------------
295  * Declaration  : static void lan_reg_reset( void )
296  *-------------------------------------------------------------------------------
297  * Function     : Reset EtherC and E-DMAC registers
298  *              : Secure a reset period of Bf・4 cycles or more within the function
299  *-------------------------------------------------------------------------------
300  * Argument     : void
301  *-------------------------------------------------------------------------------
302  * ReturnValue  : void
303  *-------------------------------------------------------------------------------
304  * Notice       :
305  *""FUNC COMMENT END""*************************************************************/
306  static void lan_reg_reset( void )
307  {
308    volatile int j = 100;                  /* Wait for Bf・4 cycles */
309
310    /* ---- Software reset ---- */
311    EDMAC.EDMR.BIT.SWR = 1;
312    /* ---- Secure reset period ---- */
313    while(j--){
314      /* Wait for Bf・4 cycles */
315    }
316  }
317  /*""FUNC COMMENT""*****************************************************************
318  * ID           :
319  * Outline      : Setting EhterC,E-DMAC registers
320  *-------------------------------------------------------------------------------
321  * Include      : #include "iodefine.h"
322  *              : #include "phy.h"
323  *              : #include "ether.h"
324  *-------------------------------------------------------------------------------
325  * Declaration  : void lan_reg_set(int link)
326  *-------------------------------------------------------------------------------
327  * Function     : E-DMAC, EtherC initialization
328  *              : Both transmission and reception are enabled by the setting
329  *-------------------------------------------------------------------------------
330  * Argument     : int link  : I     :Duplex-mode is set to EhterC
331  *              :                   :Return value of phy_autonego function is used
332  *-------------------------------------------------------------------------------
333  * ReturnValue  : void
334  *-------------------------------------------------------------------------------
335  * Notice: Execute this function in transmit/receive stopped state after EDMAC software
     reset.
336  *""FUNC COMMENT END""*************************************************************/
```

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

```
337   static void lan_reg_set( int link )
338   {
339     /* ==== EDMAC ==== */
340     EDMAC.EDMR.LONG = 0x00000000;          /* Endian not changed (big endian) */
341                                            /* descriptor length is 16 bytes */
342     EDMAC.TDLAR = &desc.send[0];           /* Transmit descriptor start */
343     EDMAC.RDLAR = &desc.recv[0];           /* Receive descriptor start */
344     EDMAC.TRSCER.LONG = 0x00000000;            /* Copy all status to descriptor */
345     EDMAC.TFTR = 0x00;                     /* Transmit FIFO threshold: store&forward */
346     EDMAC.FDR.BIT.TFD = 1;                 /* Receive FIFO threshold: store&forward */
347     EDMAC.FDR.BIT.RFD = 1;                 /* Transmit FIFO capacity of 512 bytes */
348     EDMAC.RMCR.BIT.RNC = 1;                /* Receive FIFO capacity of 512 bytes */
349     EDMAC.EDOCR.LONG = 0x00000000;         /* Continuous reception enabled */
350     EDMAC.FCFTR.LONG = 0x00070000;         /* Operation continues on FIFO error */
351     EDMAC.TRIMD.BIT.TIS = 1;        /* Flow control threshold setting, disabled by EtherC */
352     /* ==== EtherC ==== */
353     EtherC.ECMR.LONG = 0x00000000;         /* Flow control disabled */
354                                            /* CRC frame is recognized as an error */
355                                            /* Magic Packet detection is disabled */
356                                            /* Reception disabled */
357                                            /* Transmission disabled */
358                                            /* No internal loopback */
359                                            /* No external loopback */
360                                            /* Duplex mode (half-duplex mode) */
361                                            /* No promiscuous-mode operation */
362     if( link == FULL_TX || link == FULL_10M ){
363       EtherC.ECMR.BIT.DM = 1;          /* Set to full-duplex mode */
364     }
365     EtherC.MAHR = my_macaddr_h;            /* MAC address setting */
366     EtherC.MALR = my_macaddr_l;
367     EtherC.RFLR = 0x000;                   /* Maximum receive frame length of 1518 bytes */
368     EtherC.IPGR = 0x14;                    /* Gap between packets (96-bit period) */
369     /* ==== Interrupt-related ==== */
370     EDMAC.EESR.LONG = 0x47FF0F9F;              /* Clear all status ( clear by writing 1) */
371     EDMAC.EESIPR.LONG = EDMAC_EESIPR_INI_SEND | EDMAC_EESIPR_INI_RECV |
      EDMAC_EESIPR_INI_EtherC;
372     /* Transmit/receive enable setting */
373     EtherC.ECSR.LONG = 0x00000017;     /* Transmit/receive and EtherC interrupts enabled */
374     EtherC.ECSIPR.LONG = EtherC_ECSIPR_INI;    /* Enable interrupts */
375     INTC.IPR12.BIT._ETC = 5; /* E-DMAC(EINT0) interrupt priority level */
376     /* ==== Set to enable transmission/reception ==== */
377     /* ---- EtherC ---- */
378     EtherC.ECMR.BIT.RE = 1;                /* Reception enabled */
379     EtherC.ECMR.BIT.TE = 1;                /* Transmission enabled */
380     /* ---- E-DMAC ---- */
381     if(EDMAC.EDRRR.BIT.RR == 0){
382       EDMAC.EDRRR.BIT.RR = 0;          /* Reception disabled */
383     }
384   }
```

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

```
385  /*""FUNC COMMENT""*************************************************************
386  * ID           :
387  * Outline      : Transmit interrupt function
388  *-----------------------------------------------------------------------------
389  * Include      : #include "iodefine.h"
390  *              : #include "ether.h"
391  *-----------------------------------------------------------------------------
392  * Declaration  : void lan_send_handler( unsigned long status )
393  *-----------------------------------------------------------------------------
394  * Function     : Interrupt handler related to transmission regarding EDMAC(EESR)
395  *              :
396  *-----------------------------------------------------------------------------
397  * Argument     : unsigned long status : I : EESR state (interrupt-enabled bits only)
398  *-----------------------------------------------------------------------------
399  * ReturnValue  : None
400  *-----------------------------------------------------------------------------
401  * Notice       :
402  *""FUNC COMMENT END""*********************************************************/
403  void lan_send_handler( unsigned long status )
404  {
405    /* ==== Clear the flag for transmission in progress ==== */
406    f_send = 0;
407  }
408  /*""FUNC COMMENT""*************************************************************
409  * ID           :
410  * Outline      : Interrupt handler related to reception regarding EDMAC (EESSR)
411  *-----------------------------------------------------------------------------
412  * Include      : #include "iodefine.h"
413  *              : #include "ether.h"
414  *-----------------------------------------------------------------------------
415  * Declaration  : void lan_recv_handler( unsigned long status )
416  *-----------------------------------------------------------------------------
417  * Function     : EDMAC(EESR) EESR state (interrupt-enabled bits only)
418  *              :
419  *-----------------------------------------------------------------------------
420  * Argument     : unsigned long status : I : EESR state (interrupt-enabled bits only)
421  *-----------------------------------------------------------------------------
422  * ReturnValue  : none
423  *-----------------------------------------------------------------------------
424  * Notice       : Nothing is done by this sample task
425  *""FUNC COMMENT END""*********************************************************/
426  void lan_recv_handler( unsigned long status )
427  {
428  }
```

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

```
429    /*""FUNC COMMENT""*************************************************************
430    * ID           :
431    * Outline      : EtherC interrupt function
432    *-----------------------------------------------------------------------------
433    * Include      : #include "iodefine.h"
434    *              : #include "ether.h"
435    *-----------------------------------------------------------------------------
436    * Declaration  : void lan_etherc_handler( unsigned long status )
437    *-----------------------------------------------------------------------------
438    * Function     : Interrupt handler regarding EtherC(ECSR)
439    *              :
440    *-----------------------------------------------------------------------------
441    * Argument     : unsigned long status : I : ECSR state (interrupt-enabled bits only)
442    *-----------------------------------------------------------------------------
443    * ReturnValue  : none
444    *-----------------------------------------------------------------------------
445    * Notice       : Nothing is done by this sample task
446    *""FUNC COMMENT END""*********************************************************/
447    void lan_etherc_handler( unsigned long status )
448    {
449    }
450
451    /*""FUNC COMMENT""*************************************************************
452    * ID           :
453    * Outline      : MAC address setting
454    *-----------------------------------------------------------------------------
455    * Include      : #include "siic.h"
456    *-----------------------------------------------------------------------------
457    * Declaration  : static int lan_set_mac( void )
458    *-----------------------------------------------------------------------------
459    * Function     : Function to obtain MAC address from EEPROM
460    *-----------------------------------------------------------------------------
461    * Argument     : None
462    *-----------------------------------------------------------------------------
463    * ReturnValue  : MACSET_OK(0) : Acquisition successful
464    *              : MACSET_NG(-1): Acquisition failed
465    *-----------------------------------------------------------------------------
466    * Notice       :
467    *""FUNC COMMENT END""*********************************************************/
```

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

```
468   static int lan_set_mac( void )
469   {
470     volatile int ret, i;
471     unsigned char buf[10];
472
473     /* ==== EEPROM driver initial setting ==== */
474     siic_Init_Driver();
475     /* ==== Read data from EEPROM ==== */
476     ret = siic_EepRomRW(DEVADDR_EEPROM, ROMADDR_MAC, 6, buf, SIIC_MODE_EEP_READ);
477     if (ret < SIIC_OK) {
478       /* ---- Read failed ---- */
479       my_macaddr_h = DEFAULT_MAC_H;
480       my_macaddr_l = DEFAULT_MAC_L;
481       return MACSET_NG;
482     }
483     do{
484       ret = siic_Chk_Eep();
485       if( ret < SIIC_OK ){
486           /* ---- Read failed ---- */
487           my_macaddr_h = DEFAULT_MAC_H;
488           my_macaddr_l = DEFAULT_MAC_L;
489           return MACSET_NG;
490       }
491     }while( ret != SIIC_OK);
492     /* ---- Read successful ---- */
493     for(i=0; i<6; i++){
494       if( buf[i] != 0xff ){
495           break;
496       }
497     }
498     if( i == 6 ){
499       /* ---- Set the default value when EEPROM is not set ---- */
500       my_macaddr_h = DEFAULT_MAC_H;
501       my_macaddr_l = DEFAULT_MAC_L;
502     }
503     else{
504       /* ---- Set the read address ---- */
505       my_macaddr_h = buf[0];
506       my_macaddr_h <<= 8;
507       my_macaddr_h |= buf[1];
508       my_macaddr_h <<= 8;
509       my_macaddr_h |= buf[2];
510       my_macaddr_h <<= 8;
511       my_macaddr_h |= buf[3];
512       my_macaddr_l = buf[4];
513       my_macaddr_l <<= 8;
514       my_macaddr_l |= buf[5];
515     }
516     return MACSET_OK;
517   }
518
519   /* End of file */
```

3.14 Sample program list "ether.h" (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   : ether.h
32  *   Abstract    : Setting for Transmission of Ethernet Frames
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     : Nov.07,2007 ver.1.00.00
43  *               : Apr.07,2010 ver.1.00.01 Changed the company name and device name
44  *""FILE COMMENT END""*************************************************************/
45  #ifndef _ETHER_H
46  #define _ETHER_H
47
48  /* **** Macro definition **** */
49  #define NUM_OF_TX_DESCRIPTOR      4
50  #define NUM_OF_RX_DESCRIPTOR      4
51  #define NUM_OF_TX_BUFFER          4
52  #define NUM_OF_RX_BUFFER          4
53  #define SIZE_OF_BUFFER            1520          /* Must be an integral multiple of 16 */
54
```

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 0x44080F00 /* 0x40000000 : Write-back completed     *
65                                            * 0x04000000 : Detect transmit suspended    *
66                                           * Not used 0x00200000 : Frame transmission completed*
67                                            * 0x00200000 : transmit FIFO underflow   *
68                                            * 0x00080000 : Carrier not detected     *
69                                            * 0x00000800 : Carrier lost detected        *
70                                            * 0x00000400 : Delayed collision detected   *
71                                            * 0x00000200 : Transmit retry-over condition*
72                                            * 0x00000100 : Detect reception suspended  */
73    #define EDMAC_EESIPR_INI_RECV 0x0205001F /* 0x02000000 : Detect frame reception       *
74                                            * 0x00040000 : Frame reception              *
75                                            * 0x00010000 : Receive FIFO overflow     *
76                                            * 0x00000010 : Residual bit frame reception*
77                                            * 0x00000008 : Long frame reception      *
78                                            * 0x00000004 : Short frame reception     *
79                                            * 0x00000002 : PHY-LSI reception error   *
80                                            * 0x00000001 : Receive frame CRC error      */
81    #define EDMAC_EESIPR_INI_EtherC 0x00400000  /* 0x00400000 : EtherC status register */
82    #define EtherC_ECSIPR_INI     0x00000004 /* 0x00000004 : Ling signal change           */
83
84    /* **** Type definition **** */
85
86    /* ==== Transmit descriptor ==== */
87    typedef union{
88      unsigned long LONG;
89      struct{
90        unsigned int TACT:1;   /* Transmit descriptor enabled        */
91        unsigned int TDLE:1;   /* Transmit descriptor end               */
92        unsigned int TFP :2;   /* Location 1, 0 within transmit frame    */
93        unsigned int TFE :1;   /* Transmit frame error              */
94        unsigned int reserved :23;    /* TFS26 to 4 (reserved)          */
95        unsigned int TFS3:1;   /* No carrier is detected (CND bit in EESR)  */
96        unsigned int TFS2:1;   /* Carrier lost is detected (DLC bit in EESR)    */
97        unsigned int TFS1:1;   /* Delayed collision detected during transmission (CD bit in
      EESR)*/
98        unsigned int TFS0:1;   /* Transmit retry over condition (TRO bit in EESR)*/
99      }BIT;
100   }TD0;
101   typedef struct{
102     unsigned short TDL;                      /* Transmit buffer data length */
103     unsigned short reserved;
104   }TD1;
```

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

```
105    typedef struct{
106      unsigned char *TBA;                        /* Address of transmit buffer */
107    }TD2;
108    typedef struct tag_edmac_send_desc{
109      TD0 td0;
110      TD1 td1;
111      TD2 td2;
112      struct tag_edmac_send_desc *pNext;
113    }EDMAC_SEND_DESC;
114
115    /* ==== xxxxxxxxxxxxxx ==== */
116    typedef union{
117      unsigned long LONG;
118      struct{
119        unsigned int RACT:1;              /* Receive descriptor enabled */
120        unsigned int RDLE:1;              /* End of receive descriptor*/
121        unsigned int RFP :2;              /* Location 1,0 within receive frame */
122        unsigned int RFE :1;              /* Receive frame error */
123        unsigned int reserved1:17;        /* TFS26 to 10: reserved */
124        unsigned int RFS9:1;              /* Receive FIFO overflow (RFOF bit in EESR)*/
125        unsigned int reserved2:1;         /* Reserved */
126        unsigned int RFS7:1;          /* Receive multicast frames (RMAF bit in EESR)*/
127        unsigned int reserved3:1;     /* Reserved */
128        unsigned int reserved4:1;     /* Reserved */
129        unsigned int RFS4:1;   /* Residual bits frame receive error (RRF bit in ESESR)*/
130        unsigned int RFS3:1;   /* Long frame receive error (RTLE bit in EESR)*/
131        unsigned int RFS2:1;   /* Short frame receive error (RTSP bit in EESR)*/
132        unsigned int RFS1:1;   /* PHY-LSI receive error (PRE bit in EESR)*/
133        unsigned int RFS0:1;   /* Receive frame CRC error detected (CERF bit in EESCR)*/
134      }BIT;
135    }RD0;
136    typedef struct{
137      unsigned short RBL;                       /* Receive buffer length */
138      unsigned short RDL;                       /* Receive data length */
139    }RD1;
140    typedef struct{
141      unsigned char *RBA;                       /* Receive buffer address */
142    }RD2;
143    typedef struct tag_edmac_recv_desc{
144      RD0 rd0;
145      RD1 rd1;
146      RD2 rd2;
147      struct tag_edmac_recv_desc *pNext;
148    }EDMAC_RECV_DESC;
149
150    /*== The whole transmit/receive descriptors (must be allocated in 16-byte boundaries) ==*/
151    typedef struct{
152      EDMAC_SEND_DESC send[NUM_OF_TX_DESCRIPTOR];
153      EDMAC_RECV_DESC recv[NUM_OF_RX_DESCRIPTOR];
154      EDMAC_SEND_DESC *pSend_top; /* Registration location of transmit descriptors */
155      EDMAC_RECV_DESC *pRecv_end; /* Registration location and reception end of transmit
       descriptors */
156    }TXRX_DESCRIPTOR_SET;
```

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

```
157
158    /* ==== Transmit/receive buffers (must be allocated in 16-byte boundaries) ==== */
159    /* ---- Definition of all transmit/receive buffer areas ---- */
160    typedef struct{
161      unsigned char send[NUM_OF_TX_BUFFER][SIZE_OF_BUFFER];
162      unsigned char recv[NUM_OF_RX_BUFFER][SIZE_OF_BUFFER];
163    }TXRX_BUFFER_SET;
164
165    /* **** Prototype Declaration **** */
166    int lan_open( void );
167    int lan_close( void );
168    int lan_send( unsigned char *addr, int flen );
169
170
171    #endif
172
173    /* 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