SH7216 Group

Configuration to Transmit Ethernet Frames

R01AN0051EJ0200
Rev. 2.00
Sep. 17, 2010

Summary

This application note describes the configuration example of the SH7216 microcomputers (MCUs) to transmit Ethernet frames.

Target Device

SH7216 MCU

Contents

1. Introduction

1.1 Specifications

- This application transmits 10 Ethernet frames. After transmitting one Ethernet frame is completed, it starts transmitting the next frame.

1.2 Modules Used

- Pin Function Controller (PFC)
- Ethernet Controller (EtherC)
- Ethernet Controller Direct Memory Access Controller (E-DMAC)

1.3 Applicable Conditions

MCU	SH7216
Operating Frequencies	Internal clock: 200 MHz Bus clock: 50 MHz Peripheral clock: 50 MHz
Integrated Development Environment	Renesas Electronics Corporation High-performance Embedded Workshop Ver.4.05.01
C Compiler	Renesas Electronics SuperH RISC engine Family C/C++ Compiler Package Ver.9.03 Release 00
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:

- SH7216 Group Example of Initialization
- SH7216 Group Configuring the Ethernet PHY-LSI Auto-Negotiation
- SH7216 Group Configuration to Receive Ethernet Frames

2. Applications

This application uses the Ethernet Controller, and the Ethernet Controller Direct Memory Access Controller (E-DMAC).

2.1 Overview

The SH7216 always uses the EtherC and the E-DMAC in the Ethernet communication. The EtherC controls both transmission and reception, and the E-DMAC handles the DMA transfer between the transmit or receive FIFOs and the area to store data specified by user (buffer).

2.1.1 EtherC Overview

The SH7216 includes an Ethernet controller (EtherC) which is compliant with the IEEE 802.3 MAC (Media Access Control) protocol. Connect the EtherC with the IEEE 802.3-compliant physical layer LSI (PHY-LSI) to transmit and receive Ethernet/IEEE 802.3 frames. The EtherC includes one MAC layer interface. As it is connected with the E-DMAC internally, the EtherC can access memory in high-speed.

Figure 1 shows the EtherC configuration.

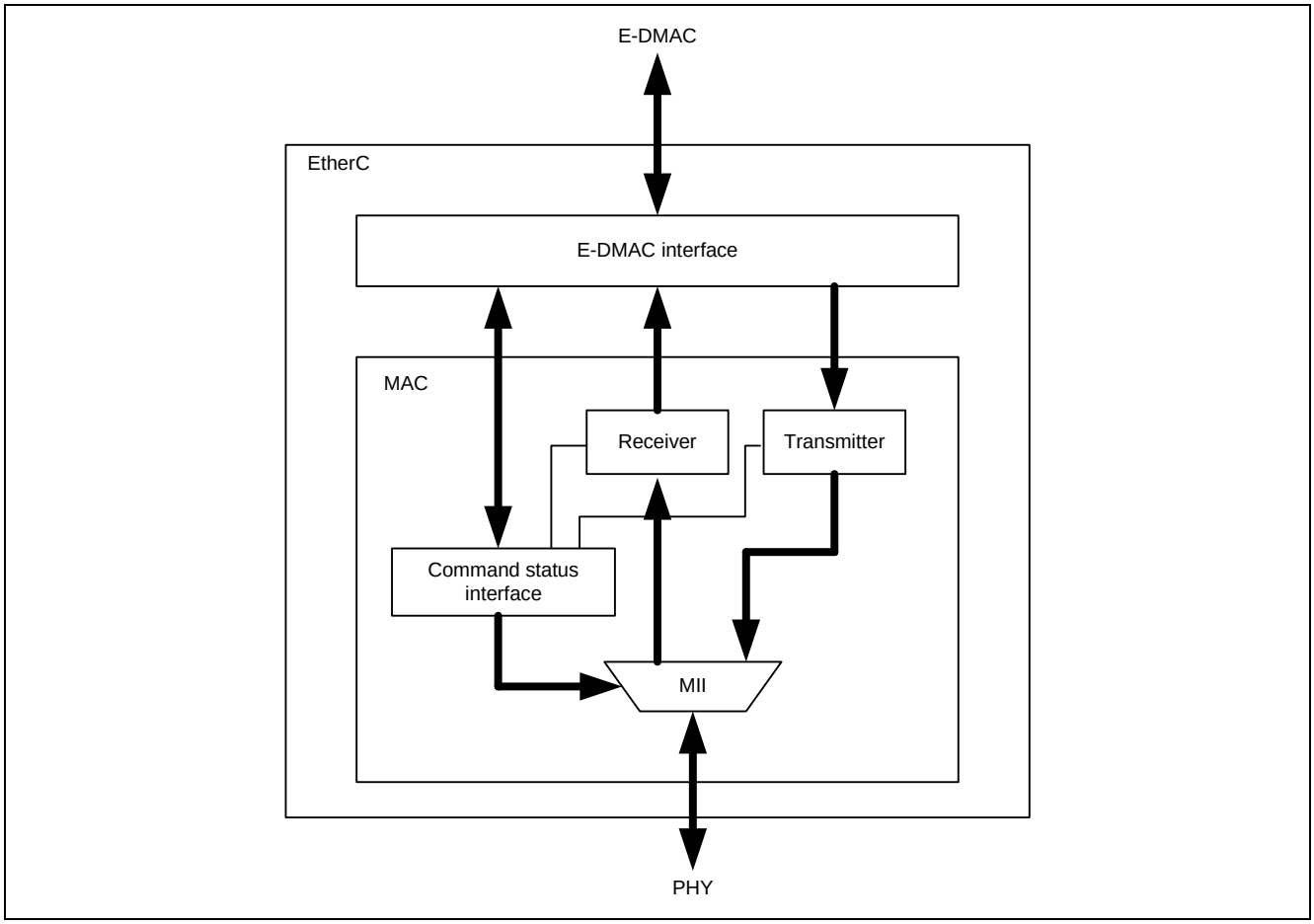
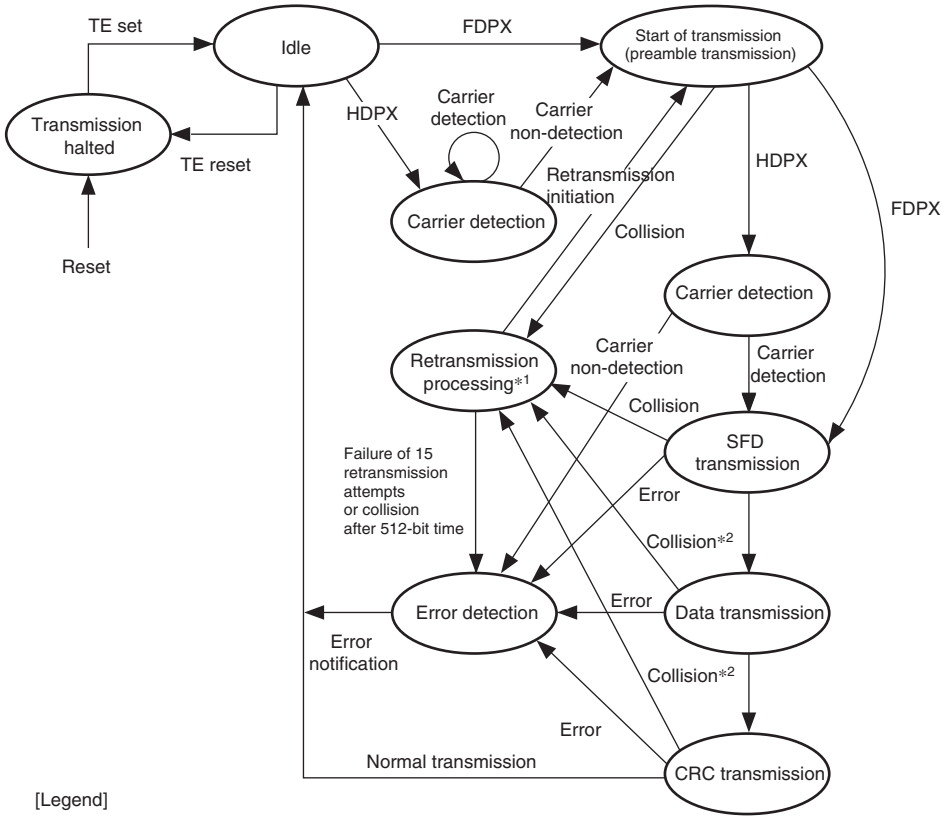


Figure 1 EtherC Configuration

2.1.2 EtherC Transmitter Overview

When the EtherC receives the transmit request from the E-DMAC transmitter, the EtherC transmitter assembles the transmit data into Ethernet frames, and outputs the frames to the MII (Media Independent Interface). Then the Ethernet PHY-LSI transmits data on the communication line. Figure 2 shows the EtherC transmitter state transition diagram of the EtherC transmitter. The transmission sequence is as follows;

1. When the TE (Transmit Enable) bit in the EtherC mode register (ECMR) is set, the EtherC transitions to the transmit idle state.
2. (A) In half-duplex mode (HDPX):
   When the EtherC receives the transmit request from the E-DMAC transmitter, it detects carrier. When no carrier is present, it transmits the preamble to the MII after delaying any transmission for an interframe gap.
   (B) In full-duplex mode (FDPX):
   The EtherC does not require detecting carrier, and transmits the preamble immediately after receiving the transmission request from the E-DMAC transmitter. When transmitting frames continuously, the EtherC transmits the preamble after delaying the transmission for an interframe gap following the last transmitted frame.
3. The EtherC transmits the SFD (Start Frame Delimiter), data, CRC (Cyclic Redundancy Check) in sequence. After the transmission is completed, the frame transmission complete interrupt (TC) occurs. When the data collision occurs or no carrier is present during transmission, interrupts occur by these sources.
4. The EtherC transitions to the idling state. Then, it continues to transmit data when there are data to transmit.



[Legend]
FDPX: Full Duplex
HDPX: Half Duplex
SFD: Start Frame Delimiter

Notes:1. Retransmission processing includes both jam transmission resultant from collision detection and the adjustment of transmission intervals by the back-off algorithm.
2. Retransmission is performed only during transmission of 512-bit (or less) data including the preamble and SFD. If a collision is detected while data exceeding 512 bits is being transmitted, only jam is transmitted and the retransmission processing by the back-off algorithm is not performed.

Figure 2 EtherC Transmitter State Transition Diagram

2.1.3 E-DMAC Overview

The SH7216 includes the Direct Memory Access Controller (E-DMAC) which is directly connected with the EtherC. The E-DMAC uses its internal DMAC to handle the DMA transfer between the transmit or receive FIFOs in the E-DMAC and the area to store data specified by user (transmit or receive buffer). CPU cannot read or write the FIFO data directly. The information that the E-DMAC refers during the DMA transfer is the transmit or receive descriptor, and user must allocate these descriptors on memory. The E-DMAC retrieves the descriptor information before transmitting or receiving Ethernet frames. Then, it reads the transmit data from the transmit buffer or writes the receive data to the receive buffer, according to the descriptor information. Allocating multiple descriptors to make up the descriptor strings (list) allows for transmitting or receiving multiple Ethernet frames sequentially.

This E-DMAC feature reduces the load on the CPU to transmit or receive data efficiently. Figure 3 shows the configuration of the E-DMAC, descriptors, and buffer.

The features of the E-DMAC are as follows;

- Includes two channels (transmit and receive) of the DMAC
- Manages descriptors to reduce the load on the CPU
- Reflects the transmit/receive frame status to the descriptor
- Uses the system bus efficiently by the DMA block transfer (in units of 16-byte)

Supports one frame per one descriptor, and one frame per multiple frames (multi buffer) - refer to section 2.1.5.

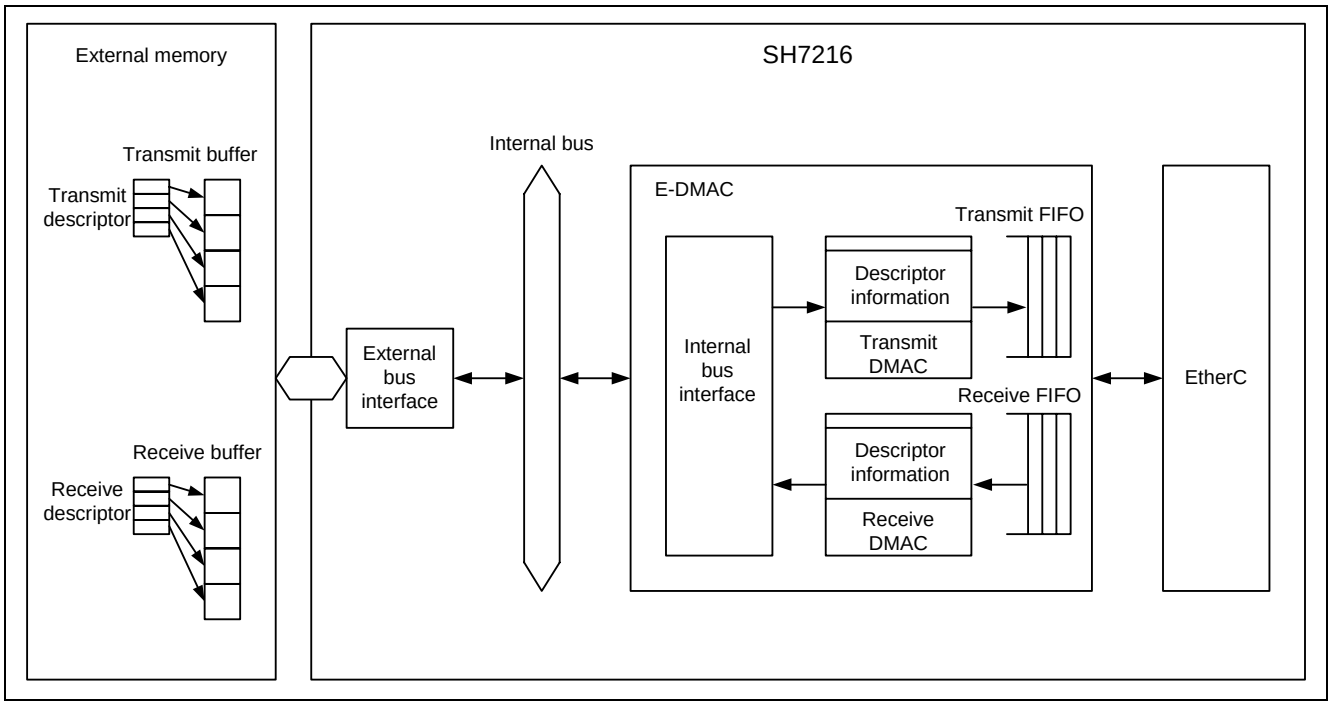


Figure 3 E-DMAC, Descriptors, and Buffer Configuration

2.1.4 Descriptor Overview

The E-DMAC requires the descriptor information (data), which includes the address storing transmit or receive data to use the DMA transfer. There are two types of descriptors; the transmit descriptor and receive descriptor. When the TR bit in the E-DMAC transmit request register (EDTRR) is set to 1, the E-DMAC automatically starts reading the transmit descriptor. When the RR bit in the E-DMAC receive request register (EDRRR) is set to 1, the E-DMAC automatically starts reading the receive descriptor. Before starting transmission or reception, user must include information regarding the DMA transfer of the transmit or receive data in the transmit or receive descriptor. After transmitting or receiving Ethernet frames are completed, the E-DMAC sets the enable/disable bit in the descriptor (TACT bit when transmitting, RACT bit when receiving), and reflects the transmission or reception result in the status bit (TFS25 to TFS0 when transmitting, RFS26 to RFS0 when receiving).

Align the descriptor on the read- and write-enabled memory, and set the starting descriptor (The first descriptor read by the E-DMAC) address in the Transmit descriptor list start address register (TDLAR) or the Receive descriptor list start address register (RDLAR). When using multiple descriptors as the descriptor string (descriptor list), align the descriptors on the contiguous addresses, according to the length of descriptor set in bits DL0 and DL1 in the E-DMAC mode register (EDMR).

2.1.5 Transmit Descriptor Overview

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

The transmit descriptor consists of TD0, TD1, TD2, and padding in units of 32-bit from the top of the data. TD0 indicates if the transmit descriptor is valid or invalid, the descriptor configuration information and status information. TD1 indicates the data length of the buffer to transmit in the descriptor. TD2 indicates the starting address in the transmit buffer. The length of padding is determined by the descriptor length specified in bits DL0 and DL1 in the EDMR register.

According to the transmit descriptor setting, both storing all transmit data in one frame in the transmit buffer by one descriptor (one frame per one descriptor), and storing all transmit data in one frame in the transmit buffer by multiple descriptors (one frame per multiple descriptors) are allowed. As example of using one frame per multiple descriptors, specify blocks of data which are always used in transmission as multiple descriptors. Specifically, allocate the destination and source addresses in the Ethernet frames to multiple descriptors, and store the rest of data in each buffer.

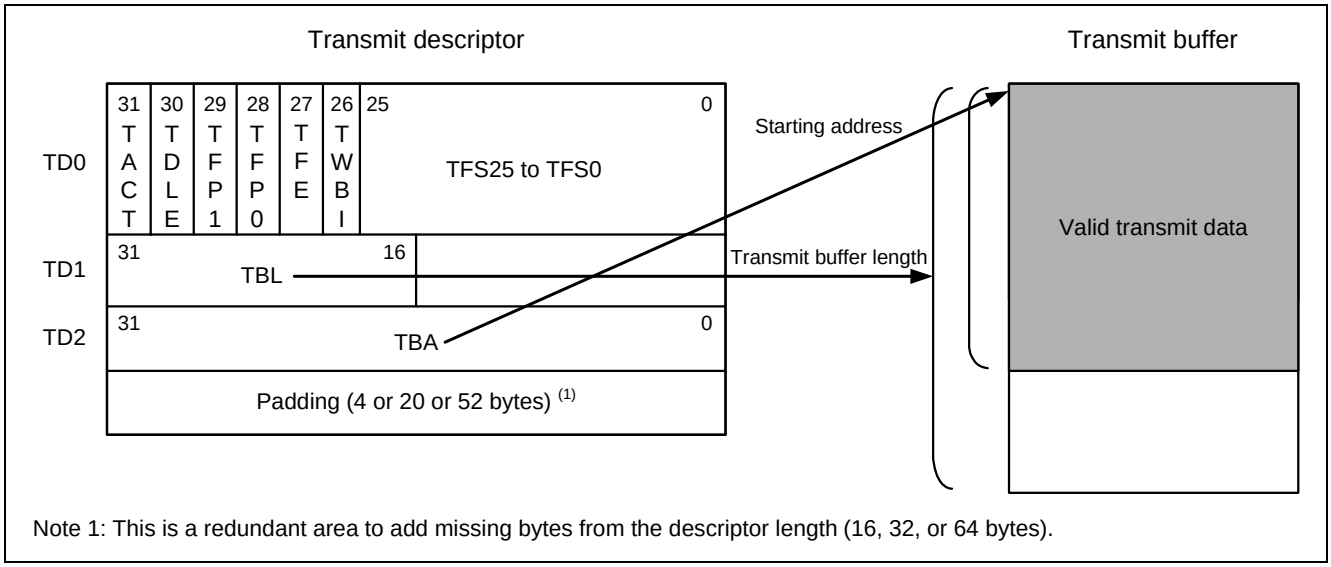


Figure 4 Relationship between the Transmit Descriptor and the Transmit Buffer

2.1.6 Setting the Transmit Descriptor

Figure 5 shows an example of using three transmit descriptors and three transmit buffers, allocating one frame per one descriptor. This example transmits one frame by a transmission request. The figure below simplifies the transmit descriptors as only TD0. Numbers shown in the figure indicates sequence to execute.

Set the transmit descriptor as following steps;

1. To allow one frame per one descriptor, set B'11 in bits TFP1, and TFP0 in all descriptors.
2. Set 0 in bits TACT, TFE, TWBI, TFS25 to TFS0 in all descriptors as the initial value.
3 . Set 0 in the TDLE bit in descriptors 1 and 2. Then, set 1 in the TDLE bit in descriptor 3 and complete the descriptor processing to read descriptor 1. These settings create the descriptor ring structure.
4. Set the data length of the transmit buffer corresponding to the said descriptor in the TBL bit, and specify the TBA bit as the starting address in the transmit buffer. (This step is not described in Figure 5.)
5. As this example transmits one frame by a transmission request, set 1 in the TACT bit in descriptor 1. Then, set 1 in the TACT bit in descriptor 2. Details on the setting procedure are described in the next chapter.

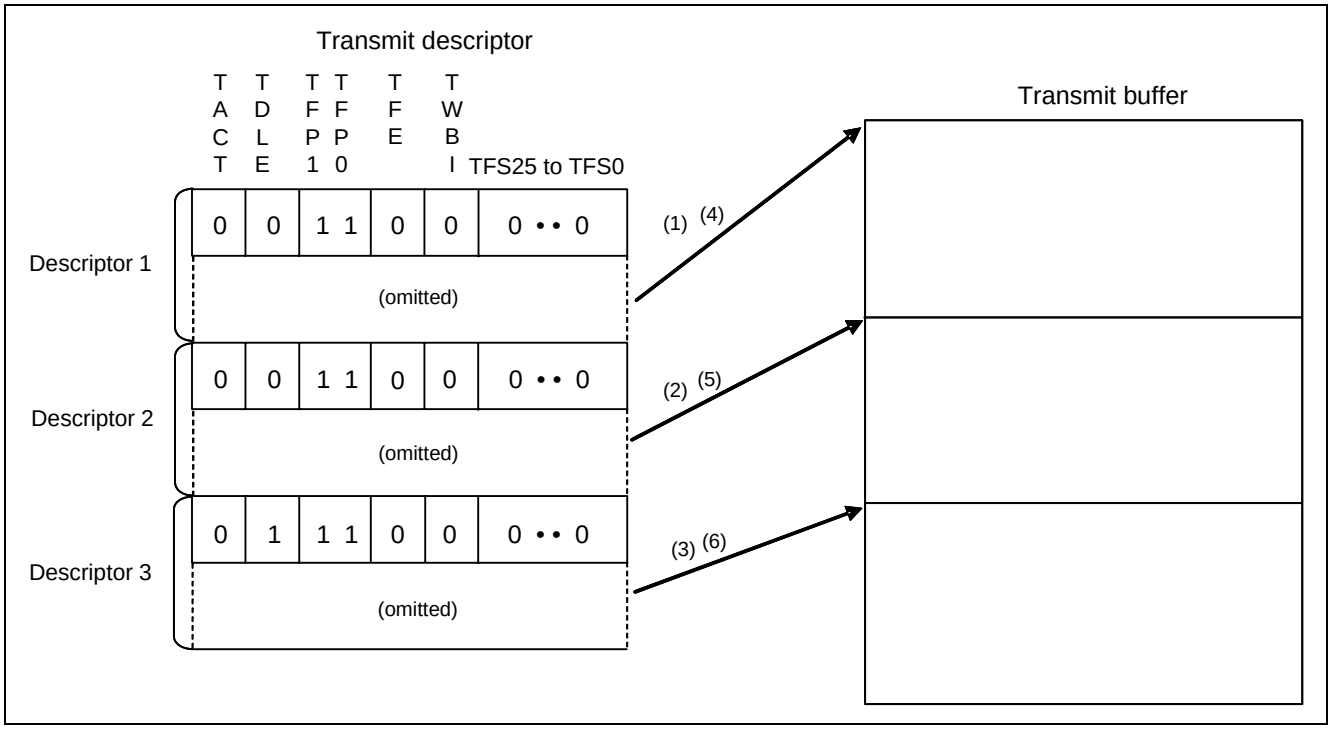


Figure 5 Relationship between 3 Transmit Descriptors and 3 Transmit Buffers

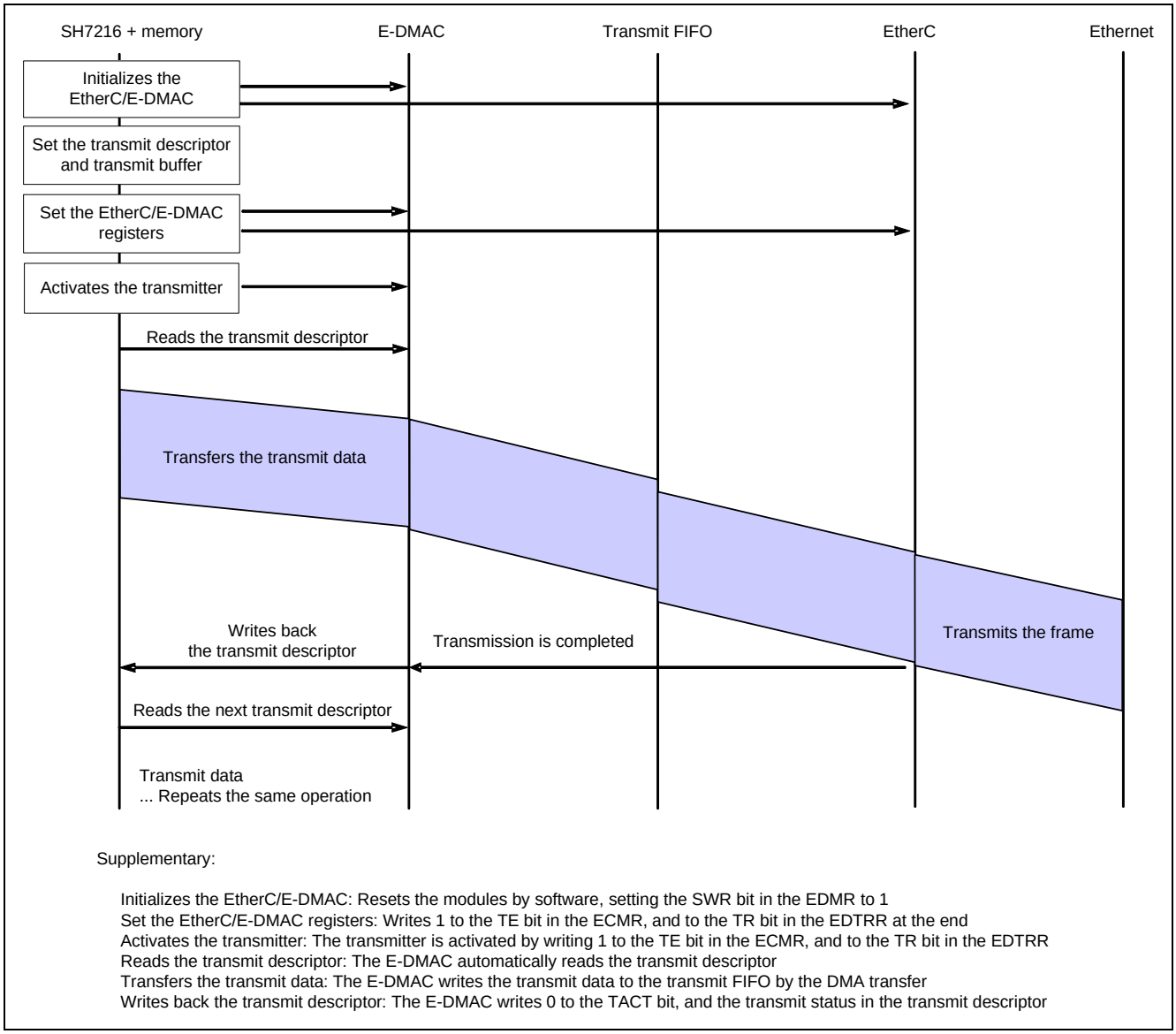
2.1.7 Operation Procedure (When Transmitting)

The E-DMAC transmitter is activated when writing 1 to the transmit request (TR) bit in the E-DMAC transmit request register (EDTRR) upon the TE bit in the EtherC mode register (ECMR) is 1. After the EtherC and E-DMAC are reset by software, the E-DMAC reads the descriptor specified by the Transmit descriptor list start address register (TDLAR). When the TACT bit in the read descriptor is 1 (valid), the E-DMAC sequentially reads the frame data from the transmit buffer starting address specified in TD2 (transmit descriptor) and transfers the data to the EtherC. Then, the EtherC assembles the transmit frames, and transmits the frames to the MII. After the DMA transfer for the buffer length specified in the descriptor is completed, the E-DMAC handles the following processing according to the value in TFP bit in the transmit descriptor.

- TFP = B'00 or B'10 (Frame is continuing)
  After the DMA transfer is completed, the E-DMAC writes back the descriptor (write 0 to the TACT bit). Then, it reads the TACT bit in the next descriptor.
- TFP = B'01 or B'11 (End of frame)
  After transmitting the frame is completed, the E-DMAC writes back the descriptor (write 0 to the TACT bit and the status in the descriptor). Then, it reads the TACT bit in the next descriptor.

When the TACT bit in the read descriptor is 1, the E-DMAC continues to transmit frames and reads the next descriptor. When it reads the descriptor with the TACT bit is 0 (invalid), the E-DMAC sets the TR bit in the EDTRR to 0, and completes transmission. If writing 1 to the TR bit after the bit is set to 0, the E-DMAC transmitter is activated again. In such case, the E-DMAC reads the descriptor following the last transmitted descriptor.

Figure 6 shows the flow chart of transmitting frames (one frame per one descriptor, using multiple descriptors).



Supplementary:

Initializes the EtherC/E-DMAC: Resets the modules by software, setting the SWR bit in the EDMR to 1
Set the EtherC/E-DMAC registers: Writes 1 to the TE bit in the ECMR, and to the TR bit in the EDTRR at the end
Activates the transmitter: The transmitter is activated by writing 1 to the TE bit in the ECMR, and to the TR bit in the EDTRR
Reads the transmit descriptor: The E-DMAC automatically reads the transmit descriptor
Transfers the transmit data: The E-DMAC writes the transmit data to the transmit FIFO by the DMA transfer
Writes back the transmit descriptor: The E-DMAC writes 0 to the TACT bit, and the transmit status in the transmit descriptor

Figure 6 Flow Chart of Transmitting Frames (One Frame per One Descriptor)

2.1.8 Setting Procedure (When Transmitting)

This section describes the basic settings for transmitting Ethernet frames. Figure 7 and Figure 8 show flow charts of transmitting Ethernet frames.

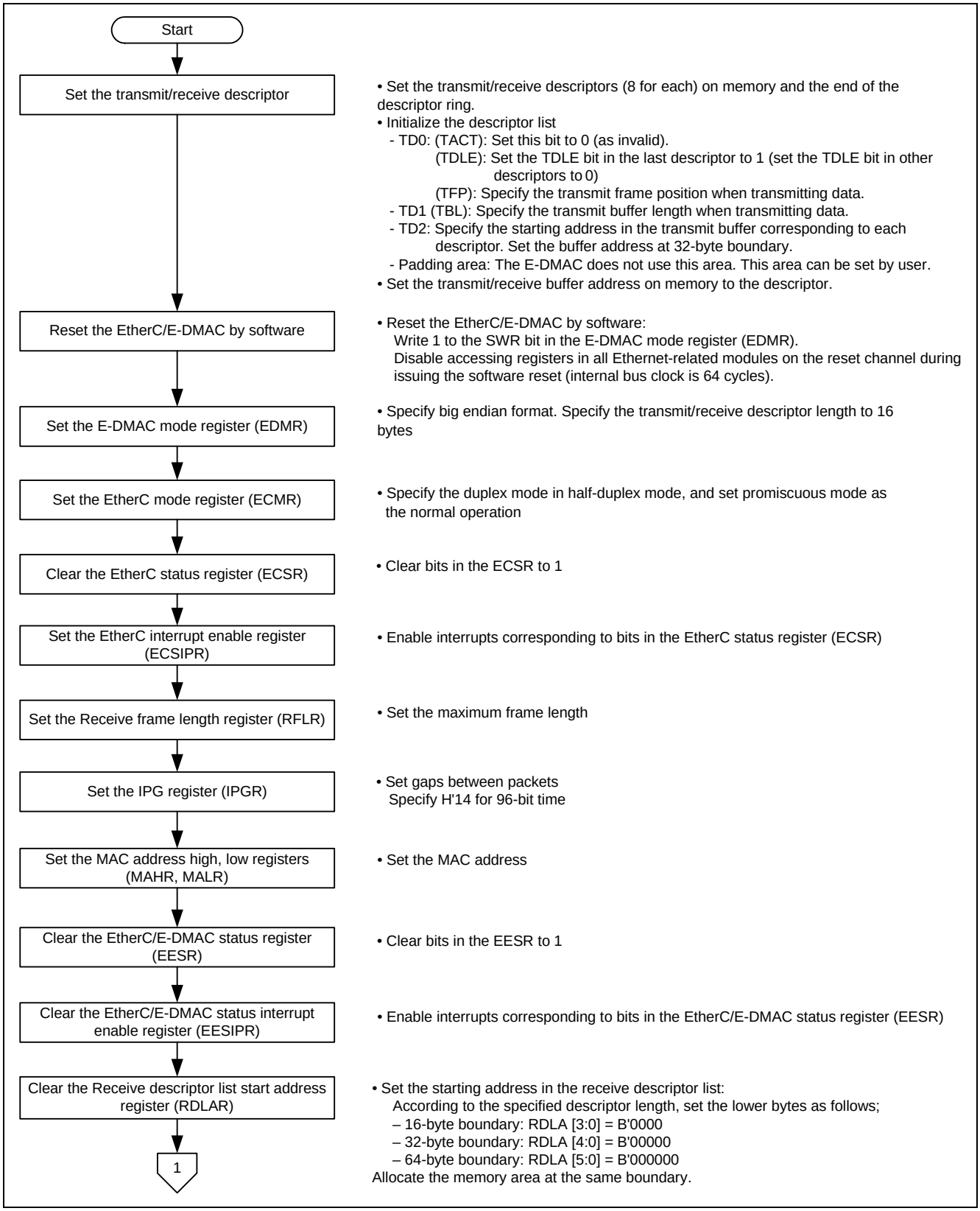


Figure 7 Transmitting Ethernet Frames (1/2)

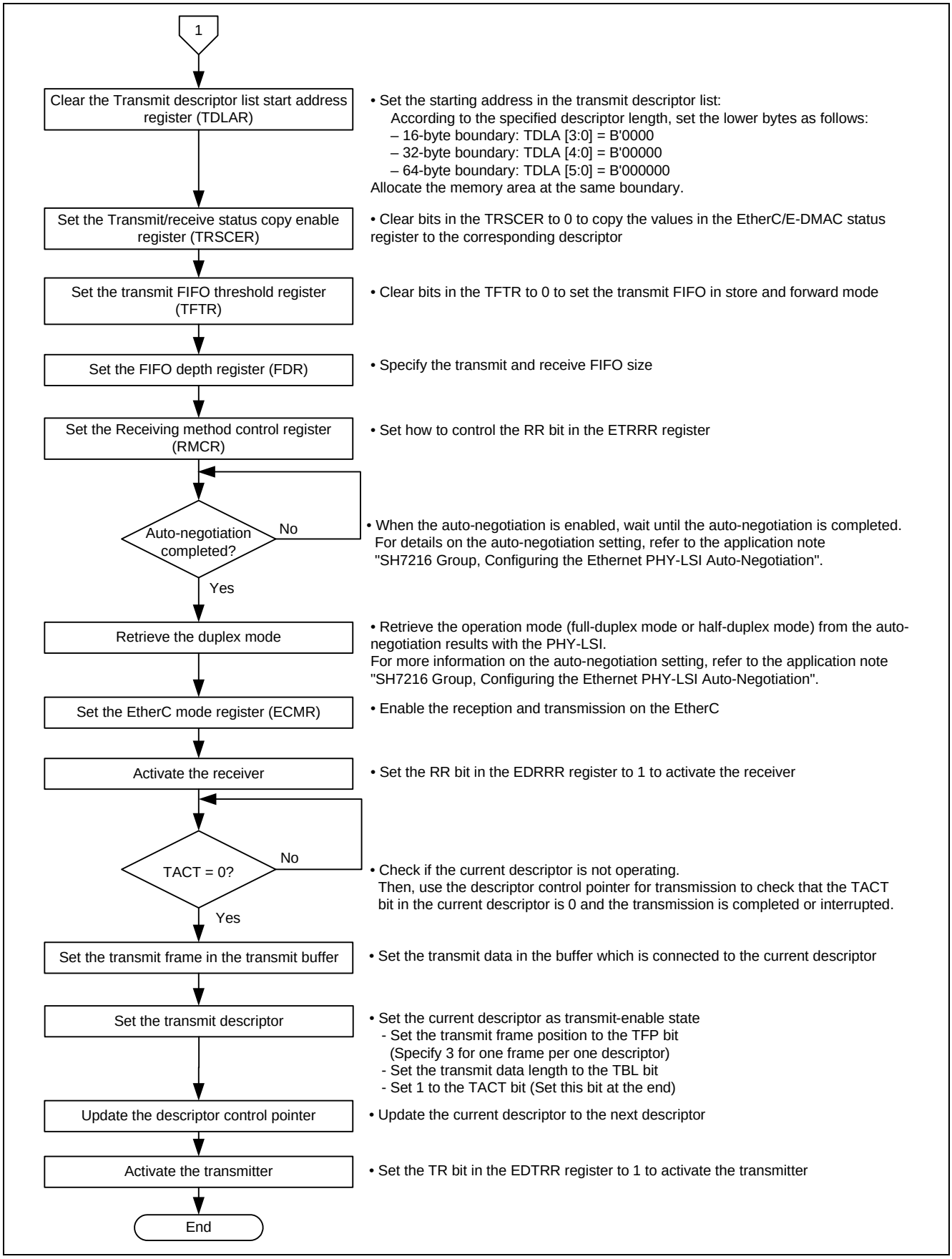


Figure 8 Transmitting Ethernet Frames (2/2)

2.2 Sample Program Operation

The sample program uses the EtherC and the E-DMAC to transmit 10 Ethernet frames to the host at the other end. It has transmit descriptors and 256-byte transmit buffers (one frame per multiple descriptors, total: 8). Also, it uses the transmit descriptor in the ring structure.

The sample program requires the portion of the Ethernet frame other than the preamble, SFD, and CRC data. Change the destination and source MAC addresses in the header to the address for the host to use. Note that the EtherC does not check if the source MAC address is correct.

Figure 9 shows the operation environment of the sample program. Figure 10 shows the Ethernet frame format.

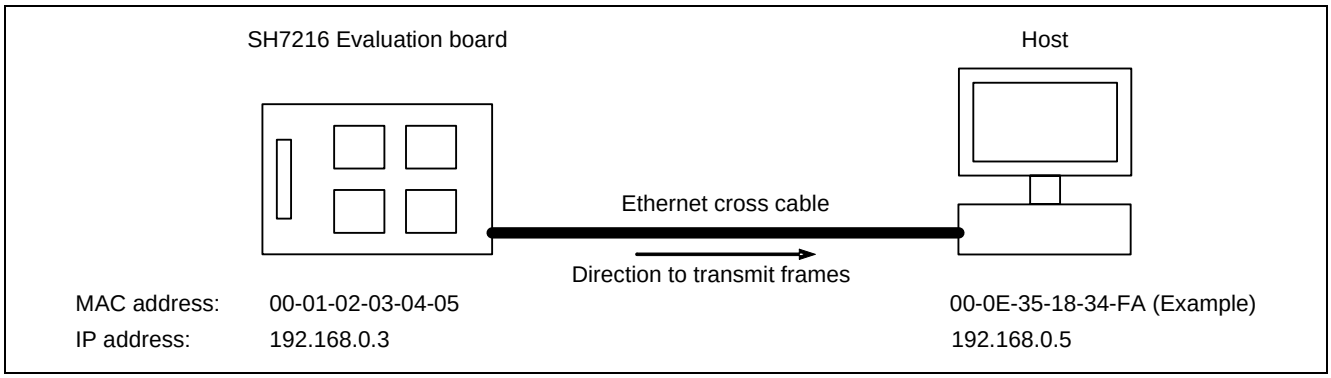


Figure 9 Sample Program Operation Environment

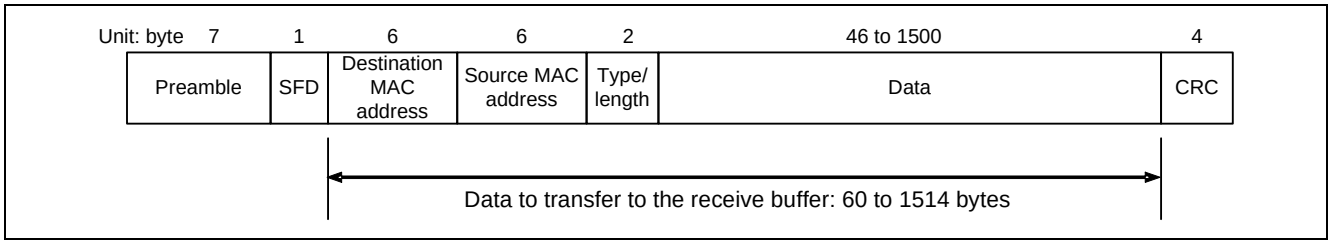


Figure 10 Ethernet Frame Format

2.3 Descriptor Definition in the Sample Program

The E-DMAC does not use the padding area in the descriptor, and that area can be used by user. The sample program sets the starting address in the next descriptor in the padding area to create the ring structure by software. Figure 11 shows the definition of the transmit descriptor structure in the sample program and example to use the transmit descriptor string.

Transmit descriptor structure definition

```
typedef struct Descriptor
{
        uint32_t            status;
        uint16_t            bufsize;
        uint16_t            size;
        int8_t              *buf_p;
        struct Descriptor   *next;
} ethfifo;
```

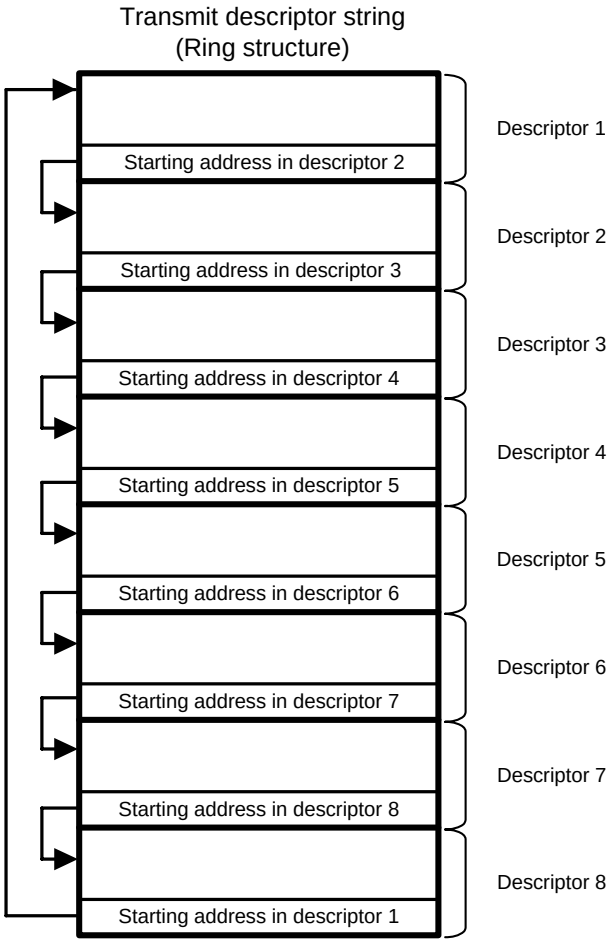


Figure 11 Transmit Descriptor Structure Definition and Example to Use the Transmit Descriptor String

2.4 Sample Program Flow Chart

Figure 12 to Figure 15 shows flow charts of the sample program. Flow charts to configure the EtherC/E-DMAC registers and descriptors include settings for receptions, however, the receive processing is not included.

For details on the function to retrieve the auto-negotiation result (phy_set_autonegotiate function), refer to the application note "SH7216 Group, Configuring the Ethernet PHY-LSI Auto-Negotiation".

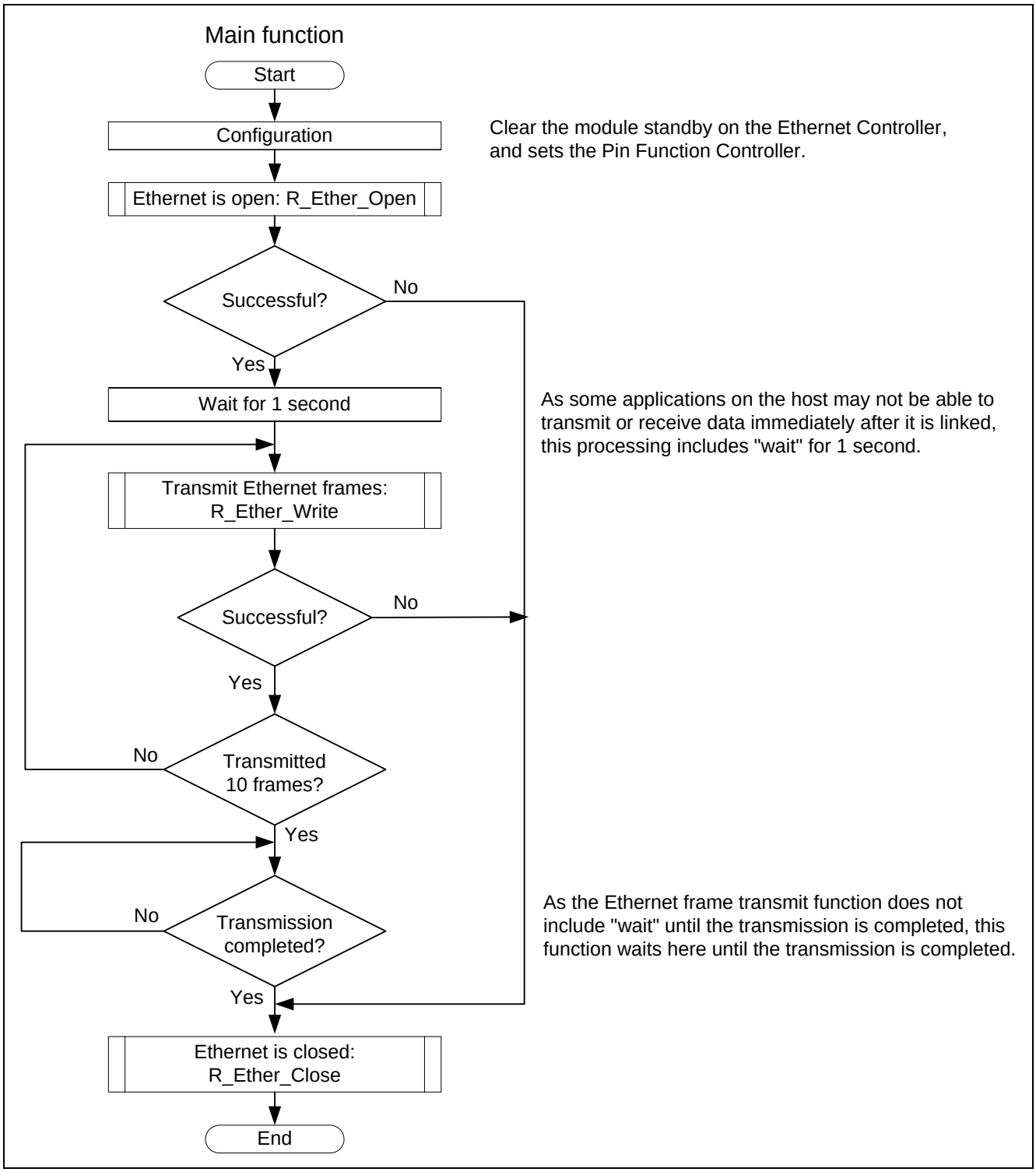


Figure 12 Sample Program Flow Chart (1/4)

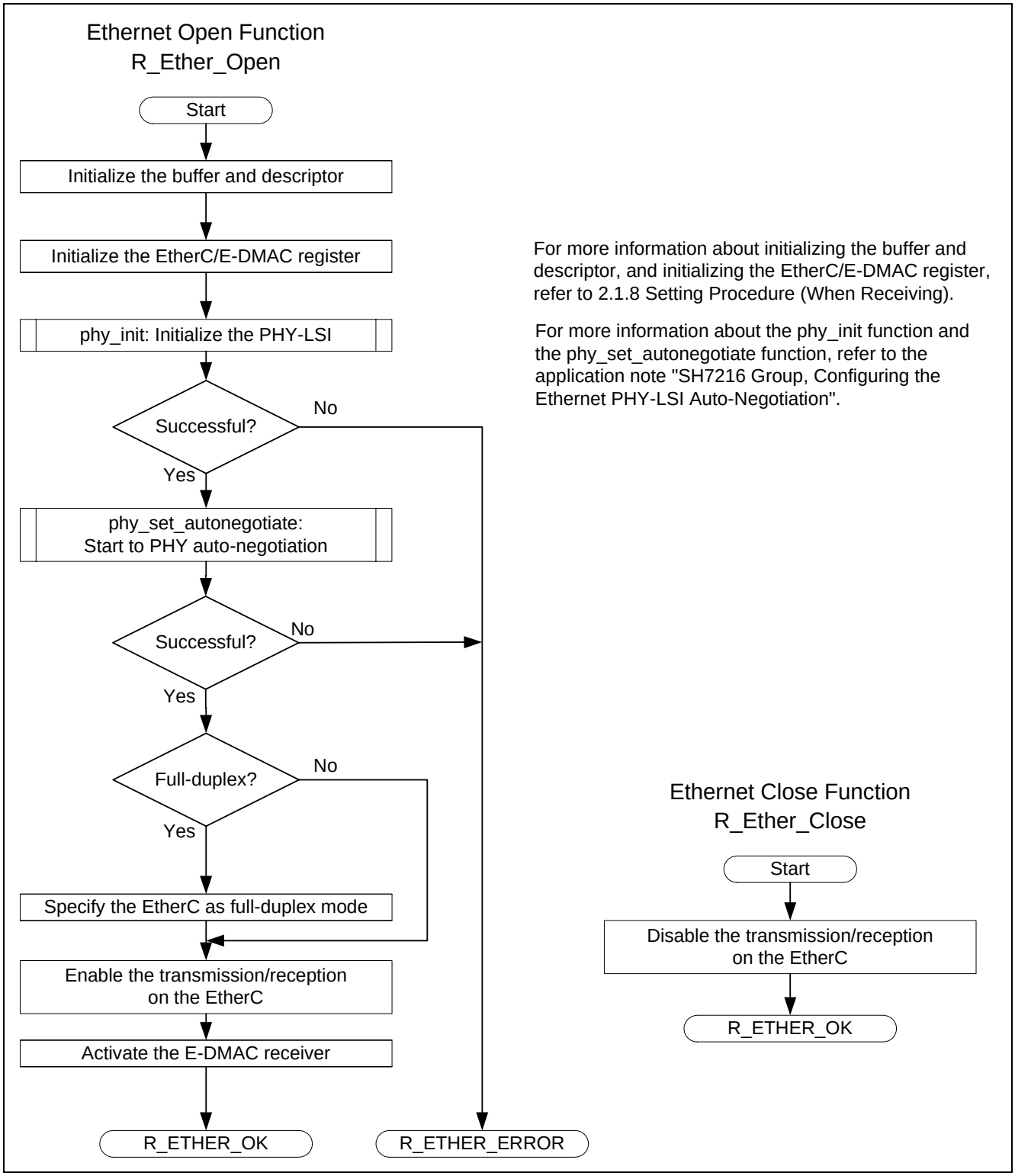


Figure 13 Sample Program Flow Chart (2/4)

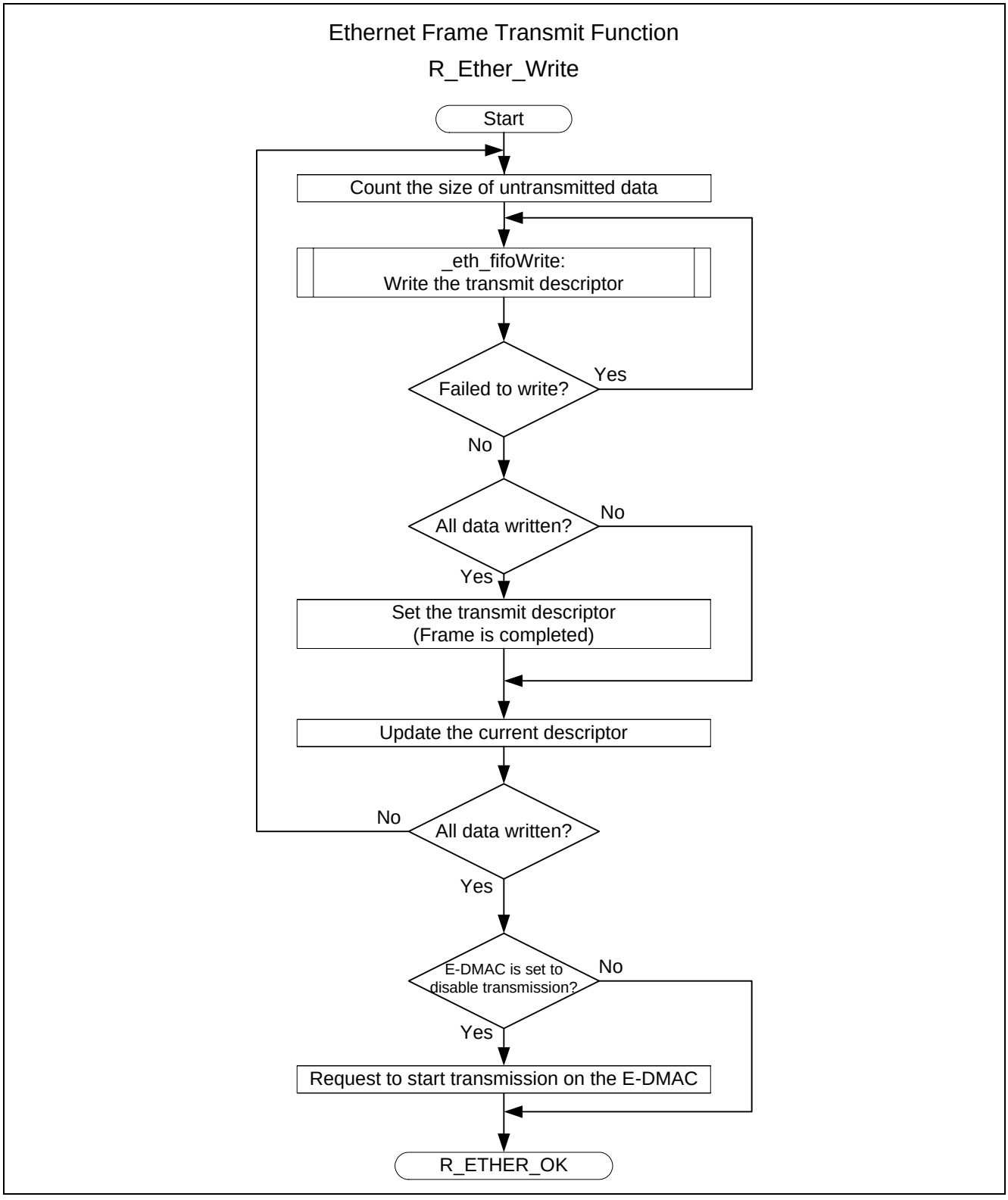


Figure 14 Sample Program Flow Chart (3/4)

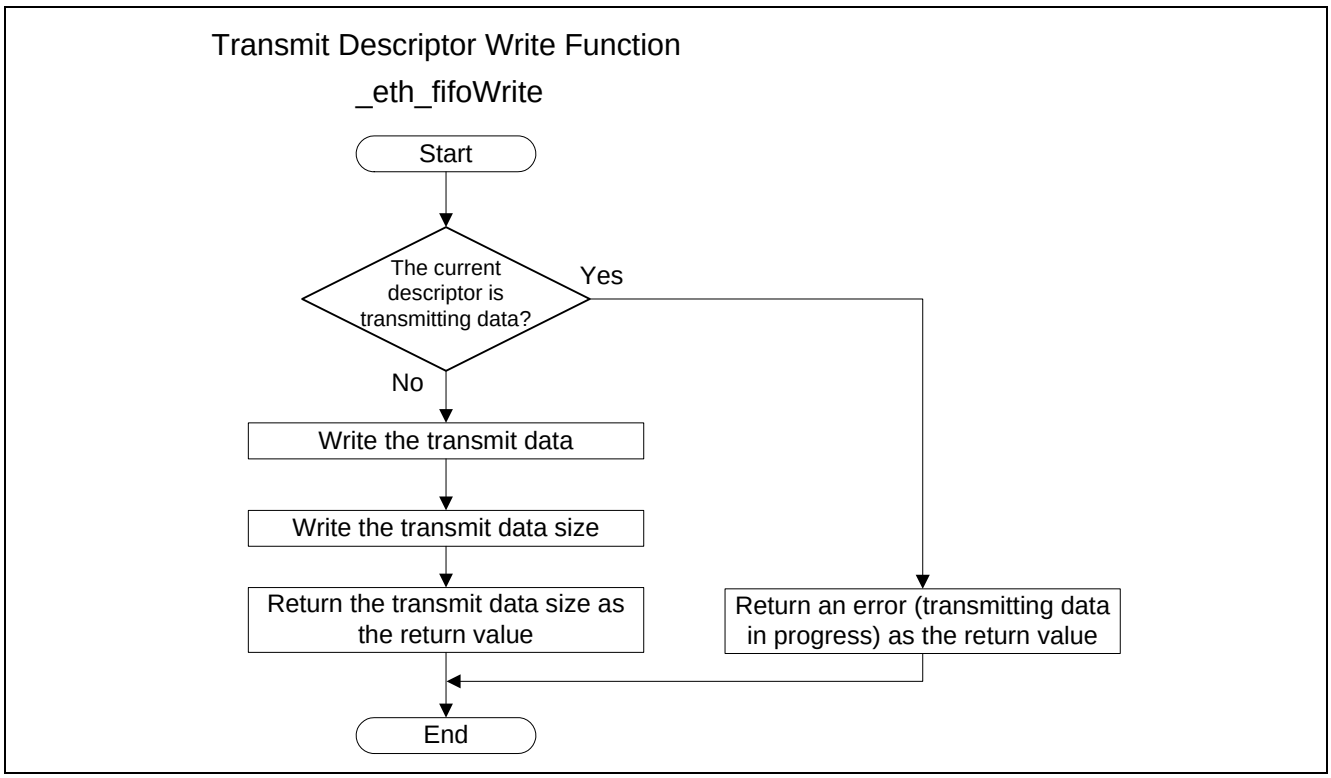


Figure 15 Sample Program Flow Chart (4/4)

3. Sample Program Listing

3.1 Sample Program Listing "main.c" (1/4)

```
1       /***********************************************************************
2       *   DISCLAIMER
3       *
4       *   This software is supplied by Renesas Electronics Corp. and is only
5       *   intended for use with Renesas products.  No other uses are authorized.
6       *
7       *   This software is owned by Renesas Electronics Corp. 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 CORP. 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) 2010 Renesas Electronics Corporation. All Rights Reserved.
29      *""FILE COMMENT""*********** Technical reference data *************************
30      *   System Name : SH7216 Sample Program
31      *   File Name   : main.c
32      *   Abstract    : Configuration to Transmit Ethernet Frames
33      *   Version     : 2.00.00
34      *   Device      : SH7216
35      *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
36      *               : C/C++ compiler package for the SuperH RISC engine family
37      *               :                             (Ver.9.03 Release00).
38      *   OS          : None
39      *   H/W Platform: R0K572167 (CPU board)
40      *   Description : Configures the MCU for the Ethernet transmission and transmits
41      *               : Ethernet frames.
42      ***********************************************************************
43      *   History     : Nov.18,2009 Ver.1.00.00
44      *               : Jul.23,2010 Ver.2.00.00 Comply with the Renesas API
45      *""FILE COMMENT END""*****************************************************/
```

3.2 Sample Program Listing "main.c" (2/4)

```
46    #include "iodefine.h"
47    #include "stdint.h"
48    #include "r_ether.h"
49    #include "phy.h"
50
51    /* ==== Prototype Declaration ==== */
52    void main(void);
53
54    /* ==== Variable Declaration ==== */
55    static uint8_t s_frame[] = {
56      0xff,0xff,0xff,0xff,0xff,0xff,  /* Destination MAC address                     */
57      0x00,0x01,0x02,0x03,0x04,0x05,  /* Source MAC address (00:01:02:03:04:05)      */
58      0x08,0x06,                      /* Type (ARP)                                  */
59      0x00,0x01,                      /*  +-- H/W type = Ethernet                    */
60      0x08,0x00,                      /*  +-- Protocol type = IP                     */
61      0x06,0x04,                      /*  +-- HW/protocol address length             */
62      0x00,0x01,                      /*  +-- OPCODE = request                       */
63      0x00,0x01,0x02,0x03,0x04,0x05,  /*  +-- Source MAC address (00:01:02:03:04:05) */
64      0xc0,0xa8,0x00,0x03,        /*  +-- Source IP address (192.168.0.3)        */
65      0x00,0x00,0x00,0x00,0x00,0x00,  /*  +-- Contact MAC address                    */
66      0xc0,0xa8,0x00,0x05,        /*  +-- Contact IP address (192.168.0.5)       */
67    };
68
69    extern volatile ethfifo txDesc[ENTRY];       /* Transmit descriptor */
70
```

3.3 Sample Program Listing "main.c" (3/4)

```
71     /*""FUNC COMMENT""*****************************************************************
72      * ID           :
73      * Outline      : Sample program main
74      *-------------------------------------------------------------------------------
75      * Include      : "iodefine.h", "stdint.h", "r_ether.h", and "phy.h"
76      *-------------------------------------------------------------------------------
77      * Declaration  : void main(void);
78      *-------------------------------------------------------------------------------
79      * Description  : Uses the internal Ethernet Controller (EtherC) and the Ethernet
80      *              : Controller Dynamic Memory Access Controller (E-DMAC) to transmit
81      *              : Ethernet frames. Ethernet PHY-LSI RTL8201CP (Realtek) is used
82      *              : in this application. Uses 8 transmit descriptors to transmit
83      *              : frames continuously.
84      *-------------------------------------------------------------------------------
85      * Argument     : void
86      *-------------------------------------------------------------------------------
87      * Return Value : void
88      *-------------------------------------------------------------------------------
89      * Note         : None
90      *""FUNC COMMENT END""*************************************************************/
91     void main(void)
92     {
93      int32_t    i, ret;
94      uint32_t ch = 0;
95      uint32_t tact_flag;
96      volatile int32_t w;
97
98      /* ==== Clears the module standby on the EtherC/E-DMAC ==== */
99      STB.CR4.BIT._ETHER = 0;
100     /* ==== Sets the PFC (For the EtherC) ==== */
101     PFC.PACRL4.BIT.PA12MD = 7; /* TX_CLK (input) */
102     PFC.PACRL3.WORD = 0x7777;  /* TX_EN,MII_TXD0,MII_TXD1,MII_TXD2 (output) */
103     PFC.PACRL2.BIT.PA7MD = 7;  /* MII_TXD3 (output) */
104     PFC.PACRL2.BIT.PA6MD = 7;  /* TX_ER (output) */
105     PFC.PDCRH4.WORD = 0x7777;  /* RX_DV,RX_ER,MII_RXD3,MII_RXD2 (input) */
106     PFC.PDCRH3.WORD = 0x7777;  /* MII_RXD1,MII_RXD0,RX_CLK,CRS (input) */
107     PFC.PDCRH2.WORD = 0x7777;  /* COL (input),WOL,EXOUT,MDC (input) */
108     PFC.PDCRH1.BIT.PD19MD = 7; /* LINKSTA (input) */
109     PFC.PDCRH1.BIT.PD18MD = 7; /* MDIO (input/output) */
110
```

3.4 Sample Program Listing "main.c" (4/4)

```
111      /* ==== Ethernet configuration ==== */
112      ret = R_Ether_Open(ch, &s_frame[6]);
113
114      if(R_ETHER_OK == ret){
115        /* ==== Waits until the application on the host is set up (1sec)@200 MHz ==== */
116        for( w=0; w < 0x00700000; w++ ){
117        }
118        /* ==== Starts transmitting 10 frames ==== */
119        for( i=0; i < 10; i++ ){
120            /* ---- Transmit frames ---- */
121            ret = R_Ether_Write(ch, s_frame, sizeof(s_frame));
122            if(ret != R_ETHER_OK){
123                break;
124            }
125        }
126        /* ==== Waits until the transmission is completed ==== */
127        for( i=0; i < ENTRY; i++ ){
128            do{
129                tact_flag = txDesc[i].status;
130                tact_flag &= ACT;
131            }while(ACT == tact_flag);
132        }
133      }
134
135      /* ==== Stops transmitting/receiving Ethernet frames ==== */
136      R_Ether_Close(ch);
137
138      while(1){
139        /* sleep */
140      }
141    }
142
143    /* End of file */
```

3.5 Sample Program Listing "r_ether.c" (1/9)

```
1   /***********************************************************************
2   *   DISCLAIMER
3   *
4   *   This software is supplied by Renesas Electronics Corp. and is only
5   *   intended for use with Renesas products.  No other uses are authorized.
6   *
7   *   This software is owned by Renesas Electronics Corp. 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 CORP. 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) 2009(2010). Renesas Electronics Corporation. All Rights Reserved.
29  *""FILE COMMENT""*********** Technical reference data *************************
30  *   System Name : SH7216 Sample Program
31  *   File Name   : r_ether.c
32  *   Version     : 2.00.00
33  *   Device      : SH7216
34  *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
35  *               : C/C++ compiler package for the SuperH RISC engine family
36  *               :                               (Ver.9.03 Release00).
37  *   OS          : None
38  *   H/W Platform: R0K572167 (CPU board)
39  *   Description : Ethernet module device driver
40  ***********************************************************************
41  *   History     : Jun.10.2009 Ver.1.00.00
42  *               : Jul.23,2010 Ver.2.00.00 Comply with the Renesas API
43  *""FILE COMMENT END""*************************************************************/
44  #include <machine.h>
45  #include <string.h>
46  #include "iodefine.h"
47  #include "stdint.h"
48  #include "r_ether.h"
49  #include "phy.h"
50
```

3.6 Sample Program Listing "r_ether.c" (2/9)

```
51    /* ==== Prototype Declaration ==== */
52    void    _eth_fifoInit(ethfifo p[], uint32_t status);
53    int32_t _eth_fifoWrite(ethfifo *p, int8_t buf[], int32_t size);
54    int32_t _eth_fifoRead(ethfifo *p, int8_t buf[]);
55
56    #pragma section  _RX_DESC
57    volatile ethfifo rxDesc[ENTRY];          /* Receive descriptor */
58    #pragma section  _TX_DESC
59    volatile ethfifo txDesc[ENTRY];          /* Transmit descriptor */
60    #pragma section
61
62    #pragma section  _RX_BUFF
63    int8_t rxbuf[ENTRY][BUFSIZE];        /* Receive data buffer */
64    #pragma section  _TX_BUFF
65    int8_t txbuf[ENTRY][BUFSIZE];        /* Transmit data buffer */
66    #pragma section
67
68    /* ==== Initializes the Ethernet device driver control structure ==== */
69    struct ei_device  le0 =
70    {
71      "eth0",      /* device name */
72      0,           /* open */
73      0,           /* Tx_act */
74      0,           /* Rx_act */
75      0,           /* txing */
76      0,           /* irq lock */
77      0,           /* dmaing */
78      0,           /* current receive descriptor */
79      0,           /* current transmit descriptor */
80      0,           /* save irq */
81      {
82        0,      /* rx packets */
83        0,      /* tx packets */
84        0,      /* rx errors */
85        0,      /* tx errors */
86        0,      /* rx dropped */
87        0,      /* tx dropped */
88        0,      /* multicast */
89        0,      /* collisions */
90
91        0,      /* rx length errors */
92        0,      /* rx over errors */
93        0,      /* rx CRC errors */
94        0,      /* rx frame errors */
95        0,      /* rx fifo errors */
96        0,      /* rx missed errors */
97
98        0,      /* tx aborted errors */
99        0,      /* tx carrier errors */
100       0,      /* tx fifo errors */
101       0,      /* tx heartbeat errors */
102       0       /* tx window errors */
```

3.7 Sample Program Listing "r_ether.c" (3/9)

```
103     },
104     0,          /* MAC 0 */
105     0,          /* MAC 1 */
106     0,          /* MAC 2 */
107     0,          /* MAC 3 */
108     0,          /* MAC 4 */
109     0       /* MAC 5 */
110   };
111
112   /*""FUNC COMMENT""*****************************************************************
113    * ID           :
114    * Outline      : Ethernet open
115    *-----------------------------------------------------------------------------
116    * Include      : "iodefine.h" , "phy.h", "r_ether.h" and "stdint.h"
117    *-----------------------------------------------------------------------------
118    * Declaration  : int32_t R_Ether_Open(uint32_t ch, uint8_t mac_addr[]);
119    *-----------------------------------------------------------------------------
120    * Description  : Initializes the EtherC, E-DMAC, PHY, and buffer memory.
121    *              : Initializes the MCU for the Ethernet and enables the MCU to
122    *              : transmit and receive Ethernet frames.
123    *              : When failed to initialize, it returns an error.
124    *-----------------------------------------------------------------------------
125    * Argument     : uint32_t ch;        I : Ethernet channel number
126    *              : uint8_t mac_addr[]; I : MAC address of such Ethernet channel
127    *-----------------------------------------------------------------------------
128    * Return Value : R_ETHER_OK;    Succeeded to initialize
129    *              : R_ETHER_ERROR; Failed to initialize
130    *-----------------------------------------------------------------------------
131    * Note         : None
132    *""FUNC COMMENT END""*************************************************************/
133   int32_t R_Ether_Open(uint32_t ch, uint8_t mac_addr[])
134   {
135     int32_t  i;
136     uint32_t mac;
137     uint16_t phydata;
138
139     ch = ch;                          /* Avoids the warning */
140
141     /* ==== Configures the Ethernet device driver ==== */
142     le0.open = 1;
143     /* ==== Sets the descriptor ==== */
144     _eth_fifoInit(rxDesc, (uint32_t)ACT);
145     _eth_fifoInit(txDesc, (uint32_t)0);
146     le0.rxcurrent = &rxDesc[0];
147     le0.txcurrent = &txDesc[0];
148     /* ==== Sets the MAC address ==== */
149     le0.mac_addr[0] = mac_addr[0];
150     le0.mac_addr[1] = mac_addr[1];
151     le0.mac_addr[2] = mac_addr[2];
152     le0.mac_addr[3] = mac_addr[3];
153     le0.mac_addr[4] = mac_addr[4];
154     le0.mac_addr[5] = mac_addr[5];
155
```

3.8 Sample Program Listing "r_ether.c" (4/9)

```
156     /* ==== Initializes the E-DMAC/EtherC ==== */
157     EDMAC.EDMR.BIT.SWR = 1;                 /* Enables the software reset */
158     for( i = 0 ; i < 0x00000100 ; i++ );    /* Waits until the E-DMAC/EtherC are initialized */
159                                             /* (B clock: 64 cycles) */
160     EDMAC.EDMR.LONG    = 0x00000000;        /* Sets the E-DMAC mode register */
161                                             /* (Big endian mode) */
162                                             /* (Transmit/receive descriptor length: 16 bytes) */
163     /* ==== Initializes the EtherC ==== */
164     EtherC.ECMR.LONG   = 0x00000000;        /* Sets the EtherC mode register */
165                                             /* (Sets the duplex mode as half-duplex) */
166                                             /* (Sets promiscuous mode as normal operation) */
167
168     EtherC.ECSR.LONG   = 0x00000037;        /* Clears all of the EtherC status */
169                                             /* (BFR, PSRTO, LCHNG, MPD, ICD) */
170     EtherC.ECSIPR.LONG = 0x00000020;        /* Disables the EtherC interrupt */
171          /* bit31~6 : Reserve : 0 ----- Reserved bits */
172          /* bit5    : BFSIPR  : 1 ----- Disables the continuous broadcast frame */
173          /*                             reception interrupt */
174          /* bit4    : PSRTOIP : 0 ----- Disables the PAUSE frame retransmit retry over */
175          /* bit3    : Reserve : 0 ----- Reserved bit */
176          /* bit2    : LCHNGIP : 0 ----- Disables the link signal change interrupt */
177          /* bit1    : MPDIP   : 0 ----- Disables the Magic Packet detection interrupt */
178          /* bit0    : ICDIP   : 0 ----- Disables the illegal carrier detection interrupt */
179
180     EtherC.RFLR.LONG   = 1518;              /* Sets the maximum receive frame length */
181     EtherC.IPGR.LONG   = 0x00000014;        /* Sets the gap between packets (96-bit time) */
182
183     /* ==== Sets the MAC address ==== */
184     mac = ((uint32_t)mac_addr[0] << 24) |
185     ((uint32_t)mac_addr[1] << 16) |
186     ((uint32_t)mac_addr[2] << 8 ) |
187     (uint32_t)mac_addr[3];
188     EtherC.MAHR = mac;
189
190     mac = ((uint32_t)mac_addr[4] << 8 ) |
191     (uint32_t)mac_addr[5];
192     EtherC.MALR.LONG = mac;
193
194     /* ==== Initializes the E-DMAC ==== */
195     EDMAC.EESR.LONG   = 0x47FF0F9F;         /* Initializes the EtherC/E-DMAC status register */
196     EDMAC.EESIPR.LONG = 0x00000000;         /* Initializes the EtherC/E-DMAC status */
197                                             /* interrupt enable register */
198     EDMAC.RDLAR       = le0.rxcurrent;      /* Sets the receive descriptor start address */
199     EDMAC.TDLAR       = le0.txcurrent;      /* Sets the transmit descriptor start address */
200     EDMAC.TRSCER.LONG = 0x00000000;         /* Brings the EtherC/E-DMAC status register */
201                                             /* value to the descriptor */
202     EDMAC.TFTR.LONG   = 0x00000000;         /* Sets store and forward mode */
203     EDMAC.FDR.LONG    = 0x00000000;         /* Sets the transmit/receive FIFO capacity */
204                                             /* (256 bytes) */
205     EDMAC.RMCR.LONG   = 0x00000001;         /* Sets to receive data continuously other */
206                                             /* than the receive descriptor is empty */
```

3.9 Sample Program Listing "r_ether.c" (5/9)

```
207
208     /* ==== Initializes the PHY ==== */
209     phydata = phy_init();
210
211     if(phydata == R_PHY_ERROR){
212       return R_ETHER_ERROR;
213     }
214
215     /* ==== Starts the PHY auto-negotiation ==== */
216     phydata = phy_set_autonegotiate();
217
218     /* ---- Determines whether to auto-negotiate or not ---- */
219     if(phydata == R_PHY_ERROR){            /* Failed to auto-negotiate */
220       return R_ETHER_ERROR;
221     }
222
223     /* ---- Detects the performance of the link partner ---- */
224     if(phydata & 0x0100){                  /* Detects PHY-LSI register 0 */
225                                         /* bit8 : DuplexMode : 1 ---- Supports */
226                                         /*                     full-duplex mode */
227       EtherC.ECMR.BIT.DM = 1;              /* Full-duplex communication */
228     }
229
230     /* ==== Enables the EtherC transmission/reception ==== */
231     EtherC.ECMR.BIT.RE = 1;
232     EtherC.ECMR.BIT.TE = 1;
233
234     /* ==== Enables the E-DMAC reception ==== */
235     EDMAC.EDRRR.LONG = 0x00000001;
236
237     return R_ETHER_OK;
238   }
239
```

3.10 Sample Program Listing "r_ether.c" (6/9)

```
240  /*""FUNC COMMENT""*****************************************************************
241   * ID           :
242   * Outline      : Ethernet close
243   *-------------------------------------------------------------------------------
244   * Include      : "iodefine.h" , "r_ether.h" and "stdint.h"
245   *-------------------------------------------------------------------------------
246   * Declaration  : int32_t R_Ether_Close(uint32_t ch);
247   *-------------------------------------------------------------------------------
248   * Description  : Stops the EtherC/E-DMAC.
249   *-------------------------------------------------------------------------------
250   * Argument     : uint32_t ch; I : Ethernet channel number
251   *-------------------------------------------------------------------------------
252   * Return Value : R_ETHER_OK; Disables the EtherC transmission/reception
253   *-------------------------------------------------------------------------------
254   * Note         : None
255   *""FUNC COMMENT END""*************************************************************/
256  int32_t R_Ether_Close(uint32_t ch)
257  {
258    ch = ch;                          /* Avoids the warning */
259
260    le0.open = 0;
261    EtherC.ECMR.LONG = 0x00000000;         /* Disables the EtherC transmission/reception */
262    le0.irqlock = 1;
263
264    return R_ETHER_OK;
265  }
266
267  /*""FUNC COMMENT""*****************************************************************
268   * ID           :
269   * Outline      : Transmit frames
270   *-------------------------------------------------------------------------------
271   * Include      : "iodefine.h" , "r_ether.h" and "stdint.h"
272   *-------------------------------------------------------------------------------
273   * Declaration  : int32_t R_Ether_Write(uint32_t ch, void *buf, uint32_t len);
274   *-------------------------------------------------------------------------------
275   * Description  : Copies the specified frame in the buffer registered in the
276   *              : transmit descriptor and transmits the frame.
277   *-------------------------------------------------------------------------------
278   * Argument     : uint32_t ch;  I : Ethernet channel number
279   *              : void *buf;    I : Transmit buffer pointer
280   *              : uint32_t len; I : Frame length
281   *-------------------------------------------------------------------------------
282   * Return Value : R_ETHER_OK ; Succeeded to transmit frames
283   *-------------------------------------------------------------------------------
284   * Note         : None
285   *""FUNC COMMENT END""*************************************************************/
286  int32_t R_Ether_Write(uint32_t ch, void *buf, uint32_t len)
287  {
288    int32_t xmit;                          /* Data size written in the transmit descriptor */
289    int32_t flag = FP1;                    /* Transmit frame position flag */
290                                      /* (default: starting frame) */
291    int8_t  *data = (int8_t *)buf;         /* Pointer to indicate the transmit data */
292
```

3.11 Sample Program Listing "r_ether.c" (7/9)

```
293     ch = ch;                                /* Avoids the warning */
294
295     /* ==== Transmits 1 frame ==== */
296     for( xmit = 0 ; len > 0 ; len -= xmit ){    /* Counts the size of untransmitted data */
297       /* ---- Writes data in the transmit descriptor ---- */
298       while( (xmit = _eth_fifoWrite(le0.txcurrent, data, (int32_t)len)) < 0 );
299
300       /* ---- When writing all data ---- */
301       if( xmit == len ){
302       flag |= FP0;                          /* Sets the transmit frame position flag */
303                                            /* (frame is completed) */
304       }
305
306       le0.txcurrent->status &= ~(FP1 | FP0);  /* Clear bits TFP1, and TFP0 to 0 */
307       le0.txcurrent->status |= (flag | ACT);  /* Brings the flag value to bits TFP1, TFP0 */
308                                                 /* Enables the TACT bit */
309
310       flag = 0;
311       le0.txcurrent = le0.txcurrent->next;
312       data += xmit;
313     }
314     le0.stat.tx_packets++;
315
316     /* ==== When E-DMAC transmission is disabled ==== */
317     if( EDMAC.EDTRR.LONG == 0x00000000 ){
318       EDMAC.EDTRR.LONG = 0x00000001;        /* Requests to start the E-DMAC transmission */
319     }
320
321     return R_ETHER_OK;
322   }
323
      (omitted)
```

3.12 Sample Program Listing "r_ether.c" (8/9)

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Initialize the FIFO
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration  : void _eth_fifoInit( ethfifo p[], uint32_t status );
 *-----------------------------------------------------------------------------
 * Description  : Initializes the E-DMAC descriptor.
 *-----------------------------------------------------------------------------
 * Argument     : ethfifo p[];     O : Pointer to the descriptor
 *              : uint32_t status; I : Descriptor default status
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
void _eth_fifoInit( ethfifo p[], uint32_t status )
{
  ethfifo  *current = 0;
  int32_t i, j;

  for( i = 0 ; i < ENTRY ; i++ ){
    current = &p[i];
    /* ==== Detects the descriptor status ==== */
    if( status == 0 ){
        current->buf_p = &txbuf[i][0];/* Determines to transmit when the ACT bit is 0 */
    }
    else{
        current->buf_p = &rxbuf[i][0];/* Determines to receive when the ACT bit is 1 */
    }

    /* ==== Clears the buffer ==== */
    for( j = 0 ; j < BUFSIZE ; j++ ){
        current->buf_p[j] = 0;
    }

    current->bufsize = BUFSIZE;
    current->size = 0;
    current->status = status;
    current->next = &p[i+1];
  }
  /* ==== Waits until the last FIFO entry is completed ==== */
  current->status |= DL;  /* Sets the current descriptor as the end of the descriptor ring */
  current->next = &p[0];
}
```

3.13 Sample Program Listing "r_ether.c" (9/9)

```
463    /*""FUNC COMMENT""*****************************************************************
464     * ID           :
465     * Outline      : Write to the transmit descriptor
466     *-------------------------------------------------------------------------------
467     * Include      :
468     *-------------------------------------------------------------------------------
469     * Declaration  : int32_t _eth_fifoWrite( ethfifo *p, int8_t buf[], int32_t size );
470     *-------------------------------------------------------------------------------
471     * Description  : Writes the data specified by the argument to the transmit
472     *              : descriptor.
473     *-------------------------------------------------------------------------------
474     * Argument     : ethfifo *p;   O ; Pointer to the transmit descriptor
475     *              : int8_t buf[]; O ; Pointer to the transmit data
476     *              : int32_t size; I : Transmit data size (bytes)
477     *-------------------------------------------------------------------------------
478     * Return Value : -1;    The current descriptor is transferring data
479     *              : 0 or bigger; Data size written in the transmit descriptor
480     *-------------------------------------------------------------------------------
481     * Note         : None
482     *""FUNC COMMENT END""*************************************************************/
483    int32_t _eth_fifoWrite( ethfifo *p, int8_t buf[], int32_t size )
484    {
485      int32_t i;
486      ethfifo  *current = p;
487
488      /* ==== The current descriptor is transmitting data ==== */
489      if( (current->status & ACT) != 0 ){
490        return( -1 );
491      }
492
493      for( i = 0 ; i < size; i++ ){
494        if( i >= BUFSIZE ){
495            break;
496        }
497        else{
498            /* ==== Writes the data in the transmit descriptor ==== */
499            current->buf_p[i] = buf[i];
500        }
501      }
502
503      current->bufsize = (uint16_t)i;
504
505      return i;
506    }
       (omitted)
570    /* End of File */
```

3.14 Sample Program Listing "r_ether.h" (1/3)

```
1   /***********************************************************************
2   *   DISCLAIMER
3   *
4   *   This software is supplied by Renesas Electronics Corp. and is only
5   *   intended for use with Renesas products.  No other uses are authorized.
6   *
7   *   This software is owned by Renesas Electronics Corp. 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 CORP. 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) 2009(2010). Renesas Electronics Corporation. All Rights Reserved.
29  *""FILE COMMENT""*********** Technical reference data *************************
30  *   System Name : SH7216 Sample Program
31  *   File Name   : r_ether.h
32  *   Version     : 2.00.00
33  *   Device      : SH7216
34  *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
35  *               : C/C++ compiler package for the SuperH RISC engine family
36  *               :                               (Ver.9.03 Release00).
37  *   OS          : None
38  *   H/W Platform: R0K572167 (CPU board)
39  *   Description : Ethernet module device driver
40  ***********************************************************************
41  *   History     : Jun.10.2009 Ver.1.00.00
42  *               : Jul.23,2010 Ver.2.00.00 Comply with the Renesas API
43  *""FILE COMMENT END""*****************************************************/
44  #ifndef ETH_H
45  #define ETH_H
46
```

3.15 Sample Program Listing "r_ether.h" (2/3)

```
47    /* ==== Type definition ==== */
48    typedef struct Descriptor
49    {
50      uint32_t     status;
51      uint16_t     bufsize;
52      uint16_t     size;
53      int8_t           *buf_p;
54      struct Descriptor *next;
55    } ethfifo;
56
57    /* ==== Macro definition ==== */
58    #define BUFSIZE   256
59    #define ENTRY     8
60
61    #define  ACT      0x80000000
62    #define  DL       0x40000000
63    #define  FP1      0x20000000
64    #define  FP0      0x10000000
65    #define  FE       0x08000000
66
67    #define  RFOVER   0x00000200
68    #define  RMAF     0x00000080
69    #define  RRF      0x00000010
70    #define  RTLF     0x00000008
71    #define  RTSF     0x00000004
72    #define  PRE      0x00000002
73    #define  CERF     0x00000001
74
75    #define  ITF      0x00000010
76    #define  CND      0x00000008
77    #define  DLC      0x00000004
78    #define  CD       0x00000002
79    #define  TRO      0x00000001
80
81    /* ==== Renesas Ethernet API return defines ==== */
82    #define R_ETHER_OK          0
83    #define R_ETHER_ERROR       -1
84    #define R_ETHER_HARD_ERROR  -3
85    #define R_ETHER_RECOVERABLE -4
86    #define R_ETHER_NO_DATA     -5
87
88    /* ==== Prototype Declaration ==== */
89    /* ==== Renesas Ethernet API prototypes ==== */
90    int32_t R_Ether_Open(uint32_t ch, uint8_t mac_addr[]);
91    int32_t R_Ether_Close(uint32_t ch);
92    int32_t R_Ether_Write(uint32_t ch, void *buf, uint32_t len);
93    int32_t R_Ether_Read(uint32_t ch, void *buf);
94
```

3.16 Sample Program Listing "r_ether.h" (3/3)

```
95    /* ==== Ethernet collected data ==== */
96    struct enet_stats
97    {
98      uint32_t rx_packets;
99      uint32_t tx_packets;
100     uint32_t rx_errors;
101     uint32_t tx_errors;
102     uint32_t rx_dropped;
103     uint32_t tx_dropped;
104     uint32_t multicast;
105     uint32_t collisions;
106
107     /* ---- Receive error ---- */
108     uint32_t rx_length_errors;
109     uint32_t rx_over_errors;
110     uint32_t rx_crc_errors;
111     uint32_t rx_frame_errors;
112     uint32_t rx_fifo_errors;
113     uint32_t rx_missed_errors;
114
115     /* ---- Transmit error ---- */
116     uint32_t tx_aborted_errors;
117     uint32_t tx_carrier_errors;
118     uint32_t tx_fifo_errors;
119     uint32_t tx_heartbeat_errors;
120     uint32_t tx_window_errors;
121   };
122
123   struct ei_device
124   {
125     const int8_t *name;     /* Device name */
126     uint8_t      open;
127     uint8_t      Tx_act;
128     uint8_t      Rx_act;
129     uint8_t      txing;
130     uint8_t      irqlock;
131     uint8_t      dmaing;
132     ethfifo      *rxcurrent;    /* Receive current descriptor */
133     ethfifo      *txcurrent;    /* Transmit current descriptor */
134     uint8_t      save_irq;
135     struct enet_stats stat;     /* Ethernet collected data */
136     uint8_t      mac_addr[6];   /* MAC address storage area */
137   };
138
139   #endif /* ETH_H  */
```

4. References

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

- Hardware Manual
  SH7214 Group, SH7216 Group Hardware User's 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	Mar.26.10	—	First edition issued
2.00	Sep.17.10	All pages	Updated to comply with the Renesas API

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