Date: Feb. 15, 2019

RENESAS TECHNICAL UPDATE

TOYOSU FORESIA, 3-2-24, Toyosu, Koto-ku, Tokyo 135-0061, Japan
Renesas Electronics Corporation

Product Category	MPU/MCU		Document No.	TN-H8*-A0443A/E	Rev.	1.00
Title	Usage Notes on Access to the EtherC and E-DMAC Registers		Information Category	Technical Notification		
Applicable Products	H8S/2472, H8S/2463, and H8S/2462 Group Products	Lot No. All lots	Reference Document	H8S/2472, H8S/2463, H8S/2462 Group Hardware Manual Rev. 2.00 (REJ09B0403-0200)		

This update is to inform you of usage notes on the Ethernet Controller (EtherC) and the Ethernet Controller Direct Memory Access Controller (E-DMAC) in the hardware manuals of the above applicable products.

1. List of Usage Notes

Table 1.1 lists the usage notes described in this technical update.

Table 1.1 List of Usage Notes

Classification	Applicable Condition	Phenomenon	Countermeasure	Affected Registers
Case 1	Writing to an EtherC register while transfer is being requested (EDTRR.TR = 1 or EDRRR.RR = 1)	Transfer of bus mastership to the E-DMAC coincided with attempted writing of a value to an EtherC register, so the value was not written to the affected register.	Write the same value until it is correctly written and confirm that it has actually been written by reading the register.	Table 2.1
Case 2	Writing to an E-DMAC register while transfer is being requested (EDTRR.TR = 1 or EDRRR.RR = 1)	Transfer of bus mastership to the E-DMAC coincided with attempted writing of a value to an E-DMAC register, so an incorrect value was temporarily written to the lower-order 16 bits of the affected register while bus mastership was being transferred to the E-DMAC.	1. Writing to any E-DMAC registers except EDTRR, EDRRR, or EESR is prohibited while transfer is being requested. 2. When writing a value to EDTRR.TR or EDRRR.RR while transfer is being requested, always read 0 before writing 1 to it. 3. When writing a value to EESR while transfer is being requested, write the value that you want to write to EESR to RMFCR, and then write the same value to EESR.	Table 2.2
Case 3	Using register-indirect addressing to read a value from an EtherC register immediately after access to an EtherC register	The value read from the higher-order 16 bits of the affected register was incorrect.	Execute a NOP or any other CPU instruction before a read instruction with register-indirect addressing.	Table 2.3
Case 4	Using register-indirect addressing to write a value to ECBRR immediately after a value was read from an EtherC or E-DMAC register (other than ECBRR)	The value written to ECBRR was undefined.	Execute a NOP or any other CPU instruction before a write instruction with register-indirect addressing.	Table 2.5
Case 5	Using register-indirect addressing to write a value to or read a value from a SCIF, SSU, LPC, USB, or PECI register immediately after a value was read from an EtherC or E-DMAC register (other than ECBRR)	The value written to the affected register or the value read from the affected register was undefined.	Execute a NOP or any other CPU instruction before a write or read instruction with register-indirect addressing.	Table 2.6

(c) 2019. Renesas Electronics Corporation. All rights reserved.

2. Usage Notes in Detail

2.1 Case 1: Transfer of Bus Mastership to the E-DMAC Coinciding with a Value being Written to an EtherC Register

(1) Phenomenon

When bus mastership was transferred to the E-DMAC while a value was being written to an EtherC register, the value was not written to the affected register.

(2) Affected Registers

Table 2.1 lists the registers affected by the phenomenon.

Table 2.1 Affected Registers in Case 1

No.	Affected Module	Affected Register	Affected: √	Remarks
1	Ethernet Controller (EtherC)	EtherC mode register (ECMR)	√	The effect arises when the TE and RE bits are being transferred.
2		EtherC status register (ECSR)	√	Flags to which 1 is written are cleared.
3		EtherC interrupt permission register (ECSIPR)	√	
4		PHY interface register (PIR)	√	The MDI bit is read-only.
5		MAC address high register (MAHR)	–	Writing is disabled during transfer.
6		MAC address low register (MALR)	–	
7		Receive frame length register (RFLR)	√	
8		PHY status register (PSR)	–	
9		Transmit retry over counter register (TROCR)	√	Writing leads to all bits being cleared to 0.
10		Delayed collision detect counter register (CDCR)	√	
11		Lost carrier counter register (LCCR)	√	
12		Carrier not detect counter register (CNDCR)	√	
13		CRC error frame counter register (CEFCR)	√	
14		Frame receive error counter register (FRECR)	√	
15		Too-short frame receive counter register (TSFRCR)	√	
16		Too-long frame receive counter register (TLFRCR)	√	
17		Residual-bit frame counter register (RFCR)	√	
18		Multicast address frame counter register (MAFCR)	√	
19		IPG register (IPGR)	–	Writing is disabled during transfer.
20		Automatic PAUSE frame set register (APR)	√	
21		Manual PAUSE frame set register (MPR)	√	Values read are undefined.
22		Automatic PAUSE frame retransmission count set register (TPAUSER)	√	

(3) Countermeasure

When writing a value to any of the registers listed in Table 2.1 while the transfer of a frame is being requested (EDTRR.TR = 1 or EDRRR.RR = 1), read the value in the register after writing it and check if the value has actually been written. If not, write the same value until it is correctly written. Since values read from the manual PAUSE frame set register (MPR) are undefined, use automatic PAUSE instead. When any of the bits in the EtherC status register (ECSR) is cleared, check that only the target bit has been cleared. Specify the value as 32-bit immediate data if any of the bits in ECSR is to be cleared with use of the C language.

Example) ETHER.ECSR.LONG = 0x00000002;

This measure is not necessary if a value is not to be written to an EtherC register while the transfer of a frame is being requested.

Figure 2.1 shows the flow of writing to the register as a workaround for this effect.

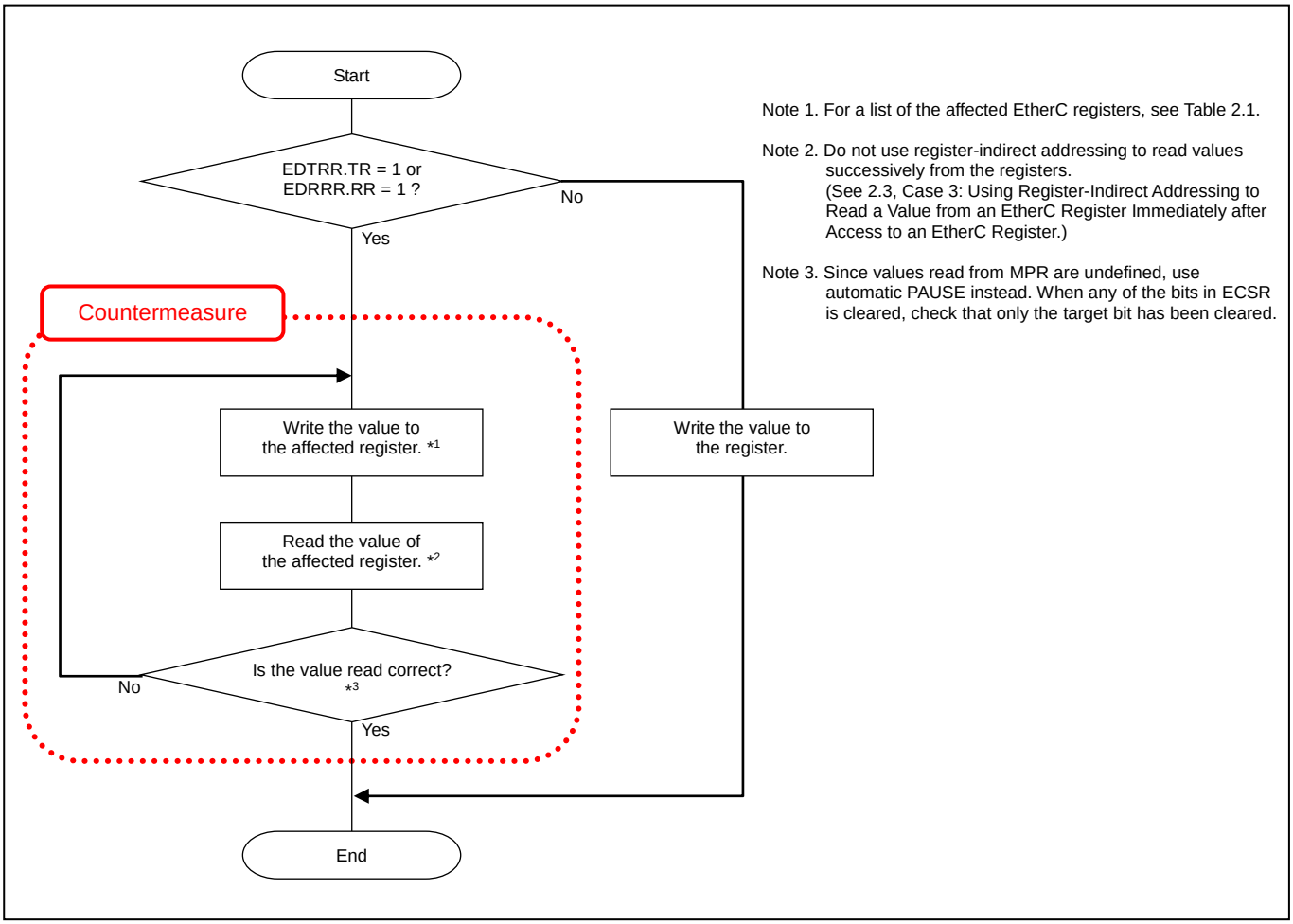

Figure 2.1 Flow of Writing to the Register as a Workaround for Case 1

2.2 Case 2: Transfer of Bus Mastership to the E-DMAC Coinciding with a Value being Written to an E-DMAC Register

(1) Phenomenon

When bus mastership was transferred to the E-DMAC while a value was being written to an E-DMAC register, an incorrect value was temporarily written to the lower-order 16 bits of the affected register. If the E-DMAC released bus mastership, the correct value was written to the affected register.

When bus mastership was transferred to the E-DMAC while a value was being written to the EtherC/E-DMAC status register (EESR), any of the bits might be cleared unintentionally. In this case, these bits remained cleared, even if the E-DMAC released bus mastership.

(2) Affected Registers

Table 2.2 lists the registers affected by the phenomenon.

Table 2.2 Affected Registers in Case 2

No.	Affected Module	Affected Register	Affected: √	Remarks
1	Ethernet Controller Direct Memory Access Controller (E-DMAC)	E-DMAC mode register (EDMR)	√	
2		E-DMAC transmit request register (EDTRR)	√	
3		E-DMAC receive request register (EDRRR)	√	
4		Transmit descriptor list address register (TDLAR)	√	
5		Receive descriptor list address register (RDLAR)	√	
6		EtherC/E-DMAC status register (EESR)	√	
7		EtherC/E-DMAC status interrupt permission register (EESIPR)	√	
8		Transmit/receive status copy enable register (TRSCER)	√	
9		Receive missed-frame counter register (RMFCR)	–	This register is not affected because it is a read-only register.
10		Transmit FIFO threshold register (TFTR)	√	
11		FIFO depth register (FDR)	√	
12		Receiving method control register (RMCR)	√	
13		Receiving-buffer write address register (RBWAR)	–	This register is not affected because it is a read-only register.
14		Receive descriptor fetch address register (RDFAR)	–	This register is not affected because it is a read-only register.
15		Transmit buffer read address register (TBRAR)	–	This register is not affected because it is a read-only register.
16		Transmit descriptor fetch address register (TDFAR)	–	This register is not affected because it is a read-only register.
17		Flow control FIFO threshold register (FCFTR)	√	
18		Bit rate setting register (ECBRR)	–	
19		Transmit interrupt register (TRIMD)	√	

(3) Countermeasure

The following measures should be taken.

1. Avoid writing to any E-DMAC registers other than the E-DMAC transmit request register (EDTRR), E-DMAC receive request register (EDRRR), and EtherC/E-DMAC status register (EESR) while the transfer of a frame is being requested (EDTRR.TR = 1 or EDRRR.RR = 1). Also avoid periodically overwriting an E-DMAC register with the same value while the transfer of a frame is being requested.
2. When writing a value to EDTRR.TR or EDRRR.RR while the transfer of a frame is being requested, always read the bit

to check that its value is 0 before writing 1 to it. Overwriting either of these bits with the value 1 that is its current value is prohibited. Writing 0 to these bits while the transfer of a frame is being requested is also prohibited.

3. When writing a value to EESR while the transfer of a frame is being requested, write the value that you want to write to EESR to the receive missed-frame counter register (RMFCR) beforehand (this does not change its value because RMFCR is read-only). After that, write the same value to EESR. If this processing proceeds within an interrupt processing routine, do not permit other interrupts. If not, prohibit interrupt processing before writing to RMFCR, and restore EESR to its previous state after having written the desired value to it. Figure 2.2 and Figure 2.3 show the procedures in cases where a value is to be written to EESR while the transfer of a frame is being requested.

These measures are not necessary if a value is written to an E-DMAC register while transfer of a frame is not being requested. In addition, in the case of writing a value to EESR while transfer is being requested, these measures are not necessary if interrupts related to the lower-order 16 bits of EESR are not enabled by the EtherC/E-DMAC status interrupt permission register (EESIPR).

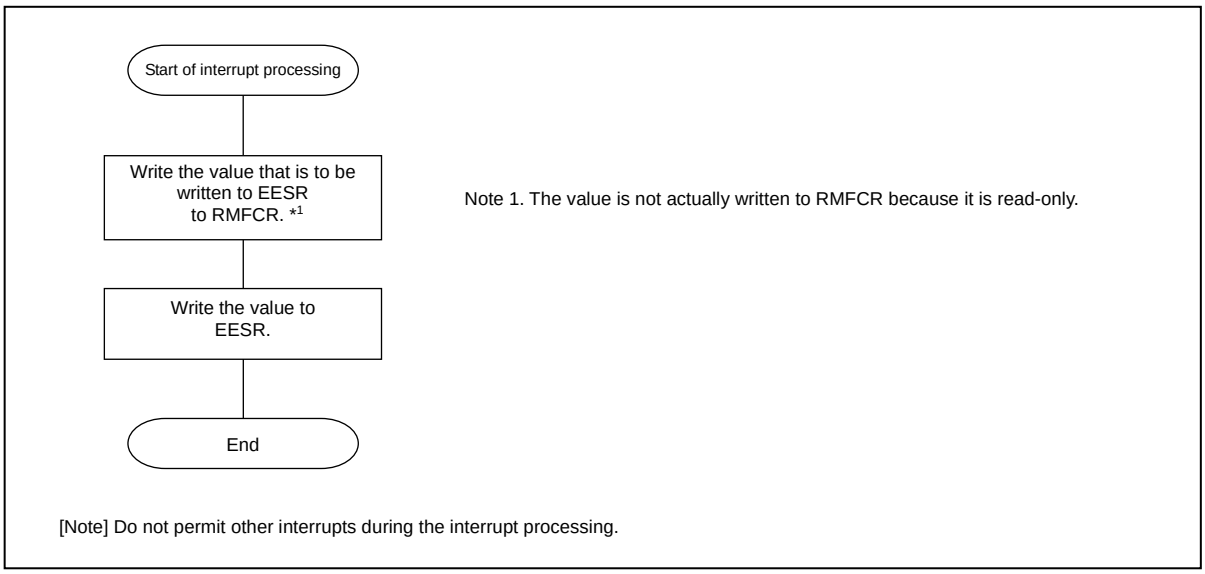

Figure 2.2 Procedure for Writing a Value to EESR while Transfer of a Frame is being Requested (When Done within Interrupt Processing)

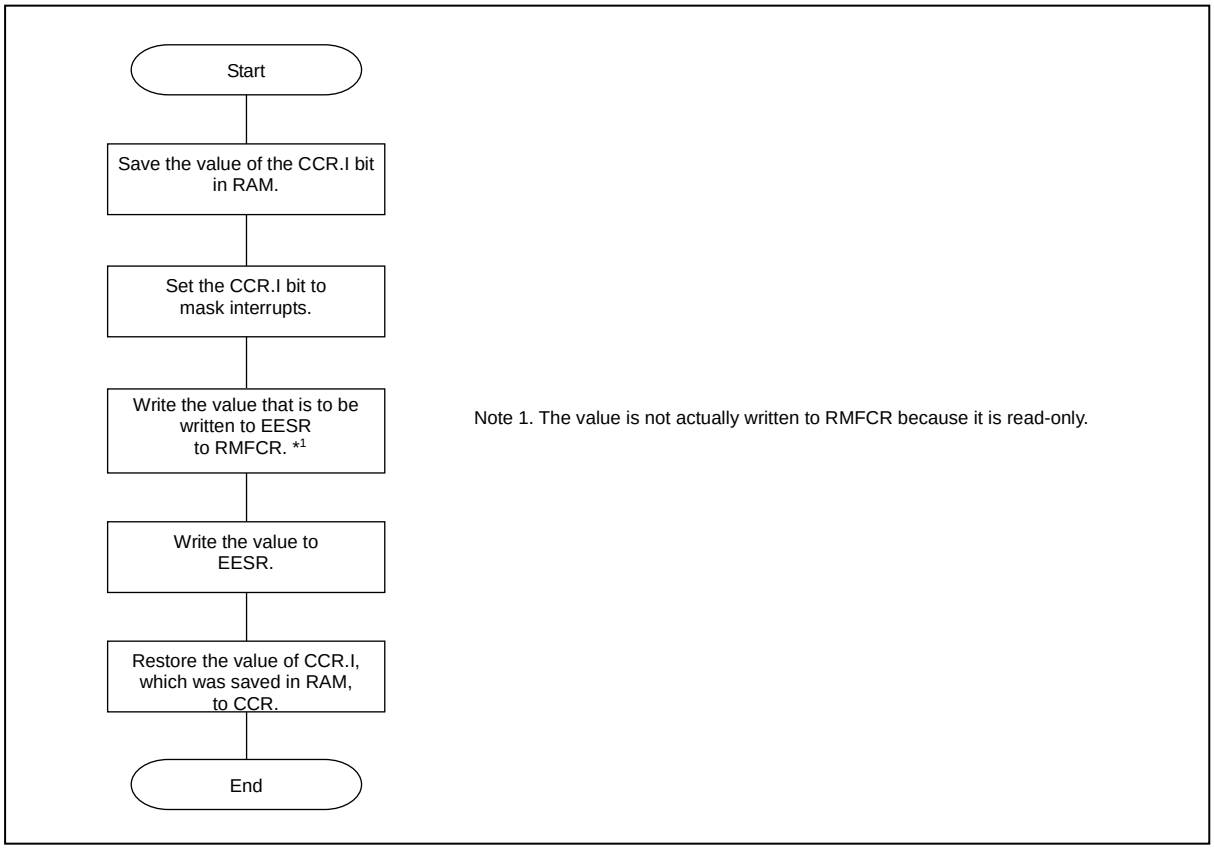

Figure 2.3 Procedure for Writing a Value to EESR while Transfer of a Frame is being Requested (Other than When Done within Interrupt Processing)

2.3 Case 3: Using Register-Indirect Addressing to Read a Value from an EtherC Register Immediately after Access to an EtherC Register

(1) Phenomenon

1. When register-indirect addressing is used to read the affected register immediately after an EtherC register was written, the higher-order 16 bits are read as H'0000.
   Registers in which the higher-order 16 bits are reserved are not affected by the phenomenon.
2. When register-indirect addressing is used to read the affected register immediately after an EtherC register was read, the higher-order 16 bits are read as the value from the EtherC register to have been read beforehand.

(2) Affected Registers

Table 2.3 lists the registers affected by the phenomenon. Table 2.4 lists the addressing modes and indicates which leads to the effect.

Table 2.3 Affected Registers in Case 3

No.	Affected Module	Affected Register	1. Affected Register in the Case of Reading Immediately after Writing: √	2. Affected Register in the Case of Reading Immediately after Reading: √
1	Ethernet Controller (EtherC)	EtherC mode register (ECMR)	√	√
2		EtherC status register (ECSR)	–	√
3		EtherC interrupt permission register (ECSIPR)	–	√
4		PHY interface register (PIR)	–	√
5		MAC address high register (MAHR)	√	√
6		MAC address low register (MALR)	–	√
7		Receive frame length register (RFLR)	–	√
8		PHY status register (PSR)	–	√
9		Transmit retry over counter register (TROCR)	√	√
10		Delayed collision detect counter register (CDCR)	√	√
11		Lost carrier counter register (LCCR)	√	√
12		Carrier not detect counter register (CNDCR)	√	√
13		CRC error frame counter register (CEFCR)	√	√
14		Frame receive error counter register (FRECR)	√	√
15		Too-short frame receive counter register (TSFRCR)	√	√
16		Too-long frame receive counter register (TLFRCR)	√	√
17		Residual-bit frame counter register (RFCR)	√	√
18		Multicast address frame counter register (MAFCR)	√	√
19		IPG register (IPGR)	–	√
20		Automatic PAUSE frame set register (APR)	–	√
21		Manual PAUSE frame set register (MPR)	–	√
22		Automatic PAUSE frame retransmission count set register (TPAUSER)	–	√

Table 2.4 Target Addressing Mode

No.	Addressing Mode	Symbol	Target: √
1	Register direct	Rn	–
2	Register indirect	@ERn	√
3	Register indirect with displacement	@(d:16,ERn)/@(d:32,ERn)	–
4	Register indirect with post-increment	@Ern+	–
5	Register indirect with pre-decrement	@–ERn	–
6	Absolute address	@aa:8/@aa:16/@aa:24/@aa:32	–
7	Immediate	#xx:8/#xx:16/#xx:32	–
8	Program-counter relative	@(d:8,PC)/@(d:16,PC)	–
9	Memory indirect	@@aa:8	–

(3) Countermeasure

When using register-indirect addressing to read a value from any of the registers listed in Table 2.3 by an instruction immediately after access (write or read) to an EtherC register, execute a NOP or any other CPU instruction before the register-indirect instruction. For example, instead of NOP, writing to or reading from registers or the RAM can be inserted. For how to insert NOP in the C language, refer to [Appendix].
This measure is not necessary with any addressing mode other than register-indirect addressing.

2.4 Case 4: Using Register-Indirect Addressing to Write a Value to the ECBRR Register Immediately after a Value was Read from an EtherC or E-DMAC Register (other than ECBRR)

(1) Phenomenon

When register-indirect addressing was used to write a value to the bit rate setting register (ECBRR) immediately after a value was read from an EtherC or E-DMAC register (other than ECBRR), the value written to ECBRR was undefined. However, reading from ECBRR in the same situation proceeds normally.

(2) Affected Registers

Table 2.5 lists the register affected by the phenomenon. The only affected register is ECBRR of E-DMAC.

Table 2.5 Affected Registers in Case 4

No.	Affected Module	Affected Register	Affected: √
1	Ethernet Controller Direct Memory Access Controller (E-DMAC)	E-DMAC mode register (EDMR)	–
2		E-DMAC transmit request register (EDTRR)	–
3		E-DMAC receive request register (EDRRR)	–
4		Transmit descriptor list address register (TDLAR)	–
5		Receive descriptor list address register (RDLAR)	–
6		EtherC/E-DMAC status register (EESR)	–
7		EtherC/E-DMAC status interrupt permission register (EESIPR)	–
8		Transmit/receive status copy enable register (TRSCER)	–
9		Receive missed-frame counter register (RMFCR)	–
10		Transmit FIFO threshold register (TFTR)	–
11		FIFO depth register (FDR)	–
12		Receiving method control register (RMCR)	–
13		Receiving-buffer write address register (RBWAR)	–
14		Receive descriptor fetch address register (RDFAR)	–
15		Transmit buffer read address register (TBRAR)	–
16		Transmit descriptor fetch address register (TDFAR)	–
17		Flow control FIFO threshold register (FCFTR)	–
18		Bit rate setting register (ECBRR)	√
19		Transmit interrupt register (TRIMD)	–

(3) Countermeasure

When using register-indirect addressing to write a value to ECBRR by an instruction immediately after a value was read from an EtherC or E-DMAC register (other than ECBRR), execute a NOP or any other CPU instruction before the register-indirect instruction. For example, instead of NOP, writing to or reading from registers or the RAM can be inserted. For how to insert NOP in the C language, refer to [Appendix].
This measure is not necessary with any addressing mode other than register-indirect addressing.

2.5 Case 5: Using Register-Indirect Addressing for Access to a Register Immediately after a Value was Read from an EtherC or E-DMAC Register (other than ECBRR)

(1) Phenomenon

When register-indirect addressing was used to write a value to the affected register immediately after data was read from an EtherC or E-DMAC register (other than ECBRR), the value written to the affected register was undefined.
When register-indirect addressing was used to read a value from the affected register immediately after a value was read from an EtherC or E-DMAC register (other than ECBRR), the value read from the affected register was undefined.

(2) Affected Registers

Table 2.6 lists the registers affected by the phenomenon. Since all the registers of the SCIF, SSU, LPC, USB, and PECI modules are affected, they are all listed, with the exception of those of PECI. This phenomenon does not affect other modules.

Table 2.6 Affected Registers in Case 5 (1)

No.	Affected Module	Affected Register	Affected: √	Remarks
1	Serial Communication Interface with FIFO (SCIF)	Host interface control register 5 (HICR5)	√	
2		Sub-chip module stop control register BL (SUBMSTPBL)	√	
3		Receive buffer register (FRBR)	√	
4		Transmitter holding register (FTHR)	√	
5		Divisor latch L (FDLL)	√	
6		Interrupt enable register (FIER)	√	
7		Divisor latch H (FDLH)	√	
8		Interrupt identification register (FIIR)	√	
9		FIFO control register (FFCR)	√	
10		Line control register (FLCR)	√	
11		Modem control register (FMCR)	√	
12		Line status register (FLSR)	√	
13		Modem status register (FMSR)	√	
14		Scratch pad register (FSCR)	√	
15		SCIF control register (SCIFCR)	√	
16		SCIF address register H (SCIFADRH)	√	
17		SCIF address register L (SCIFADRL)	√	
18		SERIRQ control register 4 (SIRQCR4)	√	
19	Synchronous Serial Communication Unit (SSU)	SS control register H (SSCRH)	√	
20		SS control register L (SSCRL)	√	
21		SS mode register (SSMR)	√	
22		SS enable register (SSER)	√	
23		SS status register (SSSR)	√	
24		SS control register 2 (SSCR2)	√	
25		SS transmit data register 0 (SSTDR0)	√	
26		SS transmit data register 1 (SSTDR1)	√	
27		SS transmit data register 2 (SSTDR2)	√	
28		SS transmit data register 3 (SSTDR3)	√	
29		SS receive data register 0 (SSRDR0)	√	
30		SS receive data register 1 (SSRDR1)	√	
31		SS receive data register 2 (SSRDR2)	√	
32		SS receive data register 3 (SSRDR3)	√	
33		SS shift register (SSTRSR)	√	

Table 2.6 Affected Registers in Case 5 (2)

No.	Affected Module	Affected Register	Affected: √	Remarks
1	LPC Interface (LPC)	Host interface control register 0 (HICR0)	√	
2		Host interface control register 1 (HICR1)	√	
3		Host interface control register 2 (HICR2)	√	
4		Host interface control register 3 (HICR3)	√	
5		Host interface control register 4 (HICR4)	√	
6		Host interface control register 5 (HICR5)	√	
7		Pin function control register (PINFNCR)	√	
8		LPC channel 1, 2 address register H, L (LADR12H, LADR12L)	√	
9		LPC channel 3 address register H, L (LADR3H, LADR3L)	√	
10		Input data register 1 (IDR1)	√	
11		Input data register 2 (IDR2)	√	
12		Input data register 3 (IDR3)	√	
13		Output data register 1 (ODR1)	√	
14		Output data register 2 (ODR2)	√	
15		Output data register 3 (ODR3)	√	
16		Status register 1 (STR1)	√	
17		Status register 2 (STR2)	√	
18		Status register 3 (STR3)	√	
19		Bidirectional data registers 0 to 15 (TWR0 to TWR15)	√	
20		SERIRQ control register 0 (SIRQCR0)	√	
21		SERIRQ control register 1 (SIRQCR1)	√	
22		SERIRQ control register 2 (SIRQCR2)	√	
23		SERIRQ control register 3 (SIRQCR3)	√	
24		SERIRQ control register 4 (SIRQCR4)	√	
25		SERIRQ control register 5 (SIRQCR5)	√	
26		Host interface select register (HISEL)	√	
27		SCIF address register H(SCIFADRH)	√	
28		SCIF address register L (SCIFADRL)	√	
29		SMIC flag register (SMICFLG)	√	
30		SMIC control/status register (SMICCSR)	√	
31		SMIC data register (SMICDTR)	√	
32		SMIC interrupt register 0 (SMICIR0)	√	
33		SMIC interrupt register 1 (SMICIR1)	√	
34		BT status register 0 (BTSR0)	√	
35		BT status register 1 (BTSR1)	√	
36		BT control/status register 0 (BTCSR0)	√	
37		BT control/status register 1 (BTCSR1)	√	
38		BT control register (BTCR)	√	
39		BT data buffer (BTDTR)	√	
40		BT interrupt mask register (BTIMSR)	√	
41		FIFO valid size register 0 (BTFVSR0)	√	
42		FIFO valid size register 1 (BTFVSR1)	√	

Table 2.6 Affected Registers in Case 5 (3)

No.	Affected Module	Affected Register	Affected: √	Remarks
1	USB Function Module (USB)	Interrupt flag register 0 (IFR0)	√	
2		Interrupt flag register 1 (IFR1)	√	
3		Interrupt flag register 2 (IFR2)	√	
4		Interrupt select register 0 (ISR0)	√	
5		Interrupt select register 1 (ISR1)	√	
6		Interrupt select register 2 (ISR2)	√	
7		Interrupt enable register 0 (IER0)	√	
8		Interrupt enable register 1 (IER1)	√	
9		Interrupt enable register 2 (IER2)	√	
10		EP0i data register (EPDR0i)	√	
11		EP0o data register (EPDR0o)	√	
12		EP0s data register (EPDR0s)	√	
13		EP1 data register (EPDR1)	√	
14		EP2 data register (EPDR2)	√	
15		EP3 data register (EPDR3)	√	
16		EP0o receive data size register (EPSZ0o)	√	
17		EP1 receive data size register (EPSZ1)	√	
18		Trigger register (TRG)	√	
19		Data status register (DASTS)	√	
20		FIFO clear register (FCLR)	√	
21		DTC transfer setting register (DMA)	√	
22		Endpoint stall register (EPSTL)	√	
23		Configuration value register (CVR)	√	
24		Control register (CTLR)	√	
25		Endpoint information register (EPIR)	√	
26		Transceiver test register 0 (TRNTREG0)	√	
27		Transceiver test register 1 (TRNTREG1)	√	
28	Platform Environment Control Interface (PECI)	All registers in this module	√	We will disclose registers on condition of entry to an agreement of confidentiality.

(3) Countermeasure

When using register-indirect addressing for access (write/read) to any of the registers listed in Table 2.6 by an instruction immediately after a value was read from an EtherC or E-DMAC register (other than ECBRR), execute a NOP or any other CPU instruction before the register-indirect instruction. For example, instead of NOP, writing to or reading from registers or the RAM can be inserted. For how to insert NOP in the C language, refer to [Appendix].

This measure is not necessary with any addressing mode other than register-indirect addressing.

Fin

[Appendix] Example of Programming in the C Language

Figure a shows how to insert a NOP when the C language is used. Figure b shows an example of a program statement in which NOP cannot be inserted and a countermeasure for it.

```
#include “machine.h”
*snip*
        while (ETHER.ECSIPR.LONG != 0x00000017)
        {
                ETHER.ECSIPR.LONG = 0x00000017;
                nop();
        }
```

Figure a Inserting NOP in the C Language

1) Example of Program Statement in which NOP cannot be Inserted

C Language Statement

```
if ( ETHER.LCCR > ETHER.CNDCR )
{
        *snip*
}
```

Example of Compiling Result (Disassemble)

```
MOV.L   #H'00FFF928,ER0
MOV.L   #H'00FFF92C,ER1
MOV.L   @ER0,ER0
MOV.L   @ER1,ER1
CMP.L   ER1,ER0
BLS     @H'xxxx
```

When reading of EtherC registers in the if statement is deployed by using register-indirect addressing, a NOP cannot be inserted into code in the C language.

2) Example Countermeasure

C Language Statement

```
a = ETHER.LCCR;
nop();      /* NOP is inserted considering the
               possibility of consecutive access being
               deployed by using register-indirect
               addressing.*/
b = ETHER.CNDCR
if ( a > b )
{
        *snip*
}
```

Example of Compiling Result (Disassemble)

```
MOV.L   #H'00FFF928,ER0
MOV.L   #H'00FFF92C,ER1
MOV.L   @ER0,ER2
MOV.L   ER2,@H'00FF0C20:32
NOP     ;NOP inserted in C language
MOV.L   @ER1,ER2
MOV.L   ER2,@H'00FF0C24:32
MOV.L   @H'00FF0C20:32,ER1
MOV.L   @H'00FF0C24:32,ER2
CMP.L   ER2,ER1
BLS     @H'xxxx
```

Consecutive access to EtherC registers in the if statement is inhibited by saving their values in variables.

Note: The result of compiling changes through optimizing the code.
Since the above result of compiling is not necessarily obtained, you need to check the code generated in your environment.

Figure b Example of Program Statement in which NOP cannot be Inserted and Example Countermeasure