[Notes]

RX Family

Ethernet Module Using Firmware Integration Technology,

RX Driver Package

R20TS0307EJ0100
Rev.1.00
May 16, 2018

Outline

When using the product in the title, note the following points.

1. Link-up processing by the “R_ETHER_LinkProcess” function when auto-negotiation is not completed
2. Calling the “R_ETHER_Read_ZC2” function or the “R_ETHER_Read” function from the interrupt function
3. Receiving an error frame when executing the “R_ETHER_Read_ZC2” function

1. Link-Up Processing by the “R_ETHER_LinkProcess” Function When Auto-Negotiation is Not Completed

1.1 Applicable Products

This function is provided in the r_ether_rx_v*.**.zip file (*.** is a revision number) included in the products in (1) and (2).

(1) RX family Ethernet module using Firmware Integration Technology (referred to as “Ethernet FIT module” hereafter)

The relevant revisions and documents are as shown in "Table 1.1, Applicable Ethernet FIT modules".

Table 1.1 Applicable Ethernet FIT modules

Revision of the Ethernet FIT module	Document number
Rev.1.00	R01AN2009EJ0100
Rev.1.01	R01AN2009EJ0101
Rev.1.02	R01AN2009EJ0102
Rev.1.10	R01AN2009EJ0110
Rev.1.11	R01AN2009EJ0111
Rev.1.12	R01AN2009EJ0112
Rev.1.13	R01AN2009EJ0113
Rev.1.14	R01AN2009EJ0114

(2) RX Driver Package

The Ethernet FIT module in (1) is also included in the following RX Driver Package products. The product names and revisions of the relevant RX Driver Package products and the revisions of the included Ethernet FIT module are as shown in "Table 1.2, Products which include the Ethernet FIT module".

Table 1.2 Products which include the Ethernet FIT module

Product name of the RX Driver Package	Revision of the RX Driver Package	Document number	Revision of the included Ethernet FIT module
RX64M Group RX Driver Package User's Manual	Rev.1.01	R01AN2460EJ0101	Rev.1.00
RX64M, RX71M Group RX Driver Package Ver.1.02	Rev.1.04	R01AN2606EJ0104	Rev.1.02
RX Family RX Driver Package Ver.1.10	Rev.1.10	R01AN3345EJ0100	Rev.1.10
RX Family RX Driver Package Ver.1.11	Rev.1.11	R01AN3467EJ0111	Rev.1.11
RX Family RX Driver Package Ver.1.12	Rev.1.12	R01AN3651EJ0112	Rev.1.12
RX Family RX Driver Package Ver.1.13	Rev.1.13	R01AN3859EJ0113	Rev.1.13
RX Family RX Driver Package Ver.1.14	Rev.1.14	R01AN4191EJ0114	Rev.1.14

1.2 Applicable MCUs

RX63N, RX65N, RX64M, and RX71M groups

1.3 Details

The MCU may not be able to send or receive Ethernet frames after the "R_ETHER_LinkProcess" function performs link-up processing.

1.4 Conditions

This problem occurs if PHY auto-negotiation is not completed after the link-up processing by the "R_ETHER_LinkProcess" function.

- Example

  If PHY auto-negotiation is not completed, the "ether_do_link" function does not enable the ETHERC transmit function or receive function, preventing the MCU from sending or receiving Ethernet frames successfully.

```
 void R_ETHER_LinkProcess (uint32_t channel)
 {
--------------------------- (omitted) --------------------------
        ether_configure_mac(channel, mac_addr_buf[channel],
            NO_USE_MAGIC_PACKET_DETECT);
        ether_do_link(channel, NO_USE_MAGIC_PACKET_DETECT);


        transfer_enable_flag[channel] = ETHER_FLAG_ON;


        if ((NULL != cb_func.pcb_func) && (FIT_NO_FUNC !=
            cb_func.pcb_func))
        {
            cb_arg.channel = channel;
            cb_arg.event_id = ETHER_CB_EVENT_ID_LINK_ON;
            (*cb_func.pcb_func)((void *) &cb_arg);
        }
--------------------------- (omitted) --------------------------
                  If PHY auto-negotiation is not completed, the ETHERC transmit or receive
                  function is disabled.
}
```

1.5 Workaround

Make the changes in (1) and (2) for the "R_ETHER_LinkProcess" function.

(1) Make changes so that the transmit/receive enable flag for the Ethernet FIT module is set before the "ether_do_link" function is called.

(2) PHY auto-negotiation may not be completed after link-up processing. As a countermeasure, make changes to perform link-up processing when the "R_ETHER_LinkProcess" function is called again.

The details of the modifications are as follows. Modify the processing in blue to the processing in red.

Before modification:

```
 void R_ETHER_LinkProcess (uint32_t channel)
 {
--------------------------- (omitted) --------------------------
 #if (ETHER_CFG_USE_LINKSTA == 1)
--------------------------- (omitted) --------------------------
        /*
         * ETHERC and EDMAC are set after ETHERC and EDMAC are reset in
            software
         * and sending and receiving is permitted.
         */
        ether_configure_mac(channel, mac_addr_buf[channel],
            NO_USE_MAGIC_PACKET_DETECT);
        ether_do_link(channel, NO_USE_MAGIC_PACKET_DETECT);


        transfer_enable_flag[channel] = ETHER_FLAG_ON;


        if ((NULL != cb_func.pcb_func) && (FIT_NO_FUNC !=
            cb_func.pcb_func))
        {
            cb_arg.channel = channel;
            cb_arg.event_id = ETHER_CB_EVENT_ID_LINK_ON;
            (*cb_func.pcb_func)((void *) &cb_arg);
        }
--------------------------- (omitted) --------------------------
#elif (ETHER_CFG_USE_LINKSTA == 0)
--------------------------- (omitted) --------------------------
        /*
         * ETHERC and EDMAC are set after ETHERC and EDMAC are reset in
            software
         * and sending and receiving is permitted.
         */
        ether_configure_mac(channel, mac_addr_buf[channel],
            NO_USE_MAGIC_PACKET_DETECT);
        ether_do_link(channel, NO_USE_MAGIC_PACKET_DETECT);


        transfer_enable_flag[channel] = ETHER_FLAG_ON;


        if ((NULL != cb_func.pcb_func) && (FIT_NO_FUNC !=
            cb_func.pcb_func))
        {
```

```
            cb_arg.channel = channel;
            cb_arg.event_id = ETHER_CB_EVENT_ID_LINK_ON;
            (*cb_func.pcb_func)((void *) &cb_arg);
        }
-------------------------- (omitted) --------------------------
}
```

After modification:

```
  void R_ETHER_LinkProcess (uint32_t channel)
  {
-------------------------- (omitted) --------------------------
  #if (ETHER_CFG_USE_LINKSTA == 1)
-------------------------- (omitted) --------------------------
            transfer_enable_flag[channel] = ETHER_FLAG_ON;
            /*
             * ETHERC and EDMAC are set after ETHERC and EDMAC are reset in
                 software
             * and sending and receiving is permitted.
             */
            ether_configure_mac(channel, mac_addr_buf[channel],
             NO_USE_MAGIC_PACKET_DETECT);
            ret = ether_do_link(channel, NO_USE_MAGIC_PACKET_DETECT);
            if (ETHER_SUCCESS == ret)
            {
                if ((NULL != cb_func.pcb_func) && (FIT_NO_FUNC !=
                cb_func.pcb_func))
                {
                    cb_arg.channel = channel;
                    cb_arg.event_id = ETHER_CB_EVENT_ID_LINK_ON;
                    (*cb_func.pcb_func)((void *) &cb_arg);
                }
            }
            else
            {
                /* When PHY auto-negotiation is not completed */
                transfer_enable_flag[channel] = ETHER_FLAG_OFF;
                lchng_flag[channel] = ETHER_FLAG_ON_LINK_ON;
            }
-------------------------- (omitted) --------------------------
```

Workaround (1):
Insert the above line to set the transmit/receive enable flag for the Ethernet FIT module before the “ether_do_link” function is called.

Workaround (2):
Make changes to perform link-up processing when the “R_ETHER_LinkProcess” function is called again if PHY auto-negotiation is not completed.

```
#elif (ETHER_CFG_USE_LINKSTA == 0)
-------------------------- (omitted) --------------------------
        transfer_enable_flag[channel] = ETHER_FLAG_ON;

       /*
        * ETHERC and EDMAC are set after ETHERC and EDMAC are reset in
          software
        * and sending and receiving is permitted.
        */
        ether_configure_mac(channel, mac_addr_buf[channel],
            NO_USE_MAGIC_PACKET_DETECT);
        ret = ether_do_link(channel, NO_USE_MAGIC_PACKET_DETECT);
        if (ETHER_SUCCESS == ret)
        {
            if ((NULL != cb_func.pcb_func) && (FIT_NO_FUNC !=
            cb_func.pcb_func))
            {
                cb_arg.channel = channel;
                cb_arg.event_id = ETHER_CB_EVENT_ID_LINK_ON;
                (*cb_func.pcb_func)((void *) &cb_arg);
            }
        }
        else
        {
            /* When PHY auto-negotiation is not completed */
            transfer_enable_flag[channel] = ETHER_FLAG_OFF;
            lchng_flag[channel] = ETHER_FLAG_ON_LINK_ON;
        }
}
```

Workaround (1):
Insert the above line to set the transmit/receive enable flag for the Ethernet FIT module before the “ether_do_link” function is called.

Workaround (2):
Make changes to perform link-up processing when the “R_ETHER_LinkProcess” function is called again if PHY auto-negotiation is not completed.

1.6 Schedule for Fixing the Problem

(1) Ethernet FIT module

This problem will be fixed in Rev.1.15, which will be the next release.

(2) RX Driver Package

The Ethernet FIT module Rev.1.15 modified in accord with this note will be included in Ver.1.15 of the RX family RX Driver Package, which will be the next release.

2. Calling the “R_ETHER_Read_ZC2” Function or the “R_ETHER_Read” Function from the Interrupt Function

2.1 Applicable Products

These functions are provided in the r_ether_rx_v*.**.zip file (*.** is a revision number) included in the products in (1) and (2).

(1) RX family Ethernet module using Firmware Integration Technology (referred to as “Ethernet FIT module” hereafter)

The relevant revisions and documents are as shown in "Table 2.1, Applicable Ethernet FIT modules".

Table 2.1 Applicable Ethernet FIT modules

Revision of the Ethernet FIT module	Document number
Rev.1.00	R01AN2009EJ0100
Rev.1.01	R01AN2009EJ0101
Rev.1.02	R01AN2009EJ0102
Rev.1.10	R01AN2009EJ0110
Rev.1.11	R01AN2009EJ0111
Rev.1.12	R01AN2009EJ0112
Rev.1.13	R01AN2009EJ0113
Rev.1.14	R01AN2009EJ0114

(2) RX Driver Package

The Ethernet FIT module in (1) is included in the following RX Driver Package products. The product names and revisions of the relevant RX Driver Package products and the revisions of the included Ethernet FIT module are as shown in "Table 2.2, Products which include the Ethernet FIT module".

Table 2.2 Products which include the Ethernet FIT module

Product name of the RX Driver Package	Revision of the RX Driver Package	Document number	Revision of the included Ethernet FIT module
RX64M Group RX Driver Package User’s Manual	Rev.1.01	R01AN2460EJ0101	Rev.1.00
RX64M, RX71M Group RX Driver Package Ver.1.02	Rev.1.04	R01AN2606EJ0104	Rev.1.02
RX Family RX Driver Package Ver.1.10	Rev.1.10	R01AN3345EJ0100	Rev.1.10
RX Family RX Driver Package Ver.1.11	Rev.1.11	R01AN3467EJ0111	Rev.1.11
RX Family RX Driver Package Ver.1.12	Rev.1.12	R01AN3651EJ0112	Rev.1.12
RX Family RX Driver Package Ver.1.13	Rev.1.13	R01AN3859EJ0113	Rev.1.13
RX Family RX Driver Package Ver.1.14	Rev.1.14	R01AN4191EJ0114	Rev.1.14

2.2 Applicable MCUs

RX63N, RX65N, RX64M, and RX71M groups

2.3 Details

If the “R_ETHER_Read_ZC2” function or the “R_ETHER_Read” function is called from the interrupt function, the user program may not be able to receive Ethernet frames successfully.

2.4 Conditions

This problem occurs when the “R_ETHER_LinkProcess” function is executed if ETHERC transmit/reception is enabled in the “ether_do_link” function and then reception of Ethernet frames is completed before the transmit/receive enable flag for the Ethernet FIT module is set.

- Example

The MCU cannot receive Ethernet frames successfully when the “R_ETHER_Read_ZC2” function or the “R_ETHER_Read” function is called from the interrupt function that is called because of an interrupt (generated in the section indicated by red dotted lines).

```
 void R_ETHER_LinkProcess (uint32_t channel)
 {
-------------------------- (omitted) --------------------------
        ether_configure_mac(channel, mac_addr_buf[channel],
            NO_USE_MAGIC_PACKET_DETECT);
        ether_do_link(channel, NO_USE_MAGIC_PACKET_DETECT);
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
        transfer_enable_flag[channel] = ETHER_FLAG_ON;


        if ((NULL != cb_func.pcb_func) && (FIT_NO_FUNC !=
            cb_func.pcb_func))
        {
            cb_arg.channel = channel;
            cb_arg.event_id = ETHER_CB_EVENT_ID_LINK_ON;
            (*cb_func.pcb_func)((void *) &cb_arg);
        }
-------------------------- (omitted) --------------------------
 }
```

2.5 Workaround

Make changes to set the transmit/receive enable flag for the Ethernet FIT module before the “ether_do_link” function is called. This workaround is the same as the one in 1.5 Workaround (1). For details, see section 1.5.

2.6 Schedule for Fixing the Problem

(1) Ethernet FIT module

This problem will be fixed in Rev.1.15, which will be the next release.

(2) RX Driver Package

The Ethernet FIT module Rev.1.15 modified in accord with this note will be included in Ver.1.15 of the RX family RX Driver Package, which will be the next release.

3. Receiving an Error Frame When Executing the “R_ETHER_Read_ZC2” Function

3.1 Applicable Products

This function is provided in the r_ether_rx_v*.**.zip file (*.** is a revision number) included in the products in (1) and (2).

(1) RX family Ethernet module using Firmware Integration Technology (referred to as “Ethernet FIT module” hereafter)

The relevant revisions and documents are as shown in "Table 3.1, Applicable Ethernet FIT modules".

Table 3.1 Applicable Ethernet FIT modules

Revision of the Ethernet FIT module	Document number
Rev.1.00	R01AN2009EJ0100
Rev.1.01	R01AN2009EJ0101
Rev.1.02	R01AN2009EJ0102
Rev.1.10	R01AN2009EJ0110
Rev.1.11	R01AN2009EJ0111
Rev.1.12	R01AN2009EJ0112
Rev.1.13	R01AN2009EJ0113
Rev.1.14	R01AN2009EJ0114

(2) RX Driver Package

The Ethernet FIT module in (1) is included in the following RX Driver Package products. The product names and revisions of the relevant RX Driver Package products and the revisions of the included Ethernet FIT module are as shown in "Table 3.2, Products which include the Ethernet FIT module".

Table 3.2 Products which include the Ethernet FIT module

Product name of the RX Driver Package	Revision of the RX Driver Package	Document number	Revision of the included Ethernet FIT module
RX64M Group RX Driver Package User’s Manual	Rev.1.01	R01AN2460EJ0101	Rev.1.00
RX64M, RX71M Group RX Driver Package Ver.1.02	Rev.1.04	R01AN2606EJ0104	Rev.1.02
RX Family RX Driver Package Ver.1.10	Rev.1.10	R01AN3345EJ0100	Rev.1.10
RX Family RX Driver Package Ver.1.11	Rev.1.11	R01AN3467EJ0111	Rev.1.11
RX Family RX Driver Package Ver.1.12	Rev.1.12	R01AN3651EJ0112	Rev.1.12
RX Family RX Driver Package Ver.1.13	Rev.1.13	R01AN3859EJ0113	Rev.1.13
RX Family RX Driver Package Ver.1.14	Rev.1.14	R01AN4191EJ0114	Rev.1.14

3.2 Applicable MCUs

RX63N, RX65N, RX64M, and RX71M groups

3.3 Details

The processing of the “R_ETHER_Read_ZC2” function may not be completed successfully.

3.4 Conditions

This problem occurs if conditions (1) and (2) below are satisfied while the “R_ETHER_Read_ZC2” function is executed.

(1) The “R_ETHER_LinkProcess” function that is being executed inside the interrupt function is performing link-down processing.

(2) The received Ethernet frames contain an error frame.

➢ Example

Because the “R_ETHER_Read_ZC2_BufRelease” function, which is a lower-level function to the “R_ETHER_Read_ZC2” function, returns an error, a loop is performed until next link-up. For this reason, the processing of the “R_ETHER_Read_ZC2” function is not completed normally.

```
int32_t R_ETHER_Read_ZC2 (uint32_t channel, void **pbuf)
{
--------------------------- (omitted) --------------------------
        while (ETHER_SUCCESS != complete_flag)
        {
            if (RACT != (papp_rx_desc[channel]->status & RACT))
            {
                if (ETHER_MC_FILTER_ON == mc_filter_flag[channel])
                {
                    if (RFS7_RMAF == (papp_rx_desc[channel]->status &
                          RFS7_RMAF))
                    {
                        R_ETHER_Read_ZC2_BufRelease(channel);

                        ret = ETHER_ERR_MC_FRAME;
                        complete_flag = ETHER_SUCCESS;
                    }
                }
                if (ETHER_ERR_MC_FRAME != ret)
                {
                    if (RFE == (papp_rx_desc[channel]->status & RFE))
                    {
                        R_ETHER_Read_ZC2_BufRelease(channel);
                    }
                    else
```

If the link is down, the processing is not completed because the “R_ETHER_Read_ZC2_BufRelease” function returns an error.

```
                    {
                        (*pbuf) = (void *) papp_rx_desc[channel]->buf_p;

                        num_recvd = papp_rx_desc[channel]->size;
                        ret = num_recvd;
                        complete_flag = ETHER_SUCCESS;
                    }
                }
            }
            else
            {
                ret = ETHER_NO_DATA;
                complete_flag = ETHER_SUCCESS;
            }
        }
--------------------------- (omitted) --------------------------
} /* End of function R_ETHER_Read_ZC2() */
```

3.5 Workaround

Modify the “R_ETHER_Read_ZC2” function so that it ends when the “R_ETHER_Read_ZC2_BufRelease” function returns an error.

The details of the modification are as follows.

(1) Add the processing in red.

Before modification:

```
int32_t R_ETHER_Read_ZC2 (uint32_t channel, void **pbuf)
{
    int32_t num_recvd;
    int32_t ret;
    int32_t complete_flag;
```

After modification:

```
int32_t R_ETHER_Read_ZC2 (uint32_t channel, void **pbuf)
{
    int32_t num_recvd;
    int32_t ret;
    int32_t complete_flag;
    int32_t ret2;
```

(2) Modify the processing in blue to the processing in red.

Before modification:

```
int32_t R_ETHER_Read_ZC2 (uint32_t channel, void **pbuf)
{
--------------------------- (omitted) --------------------------
               if (ETHER_MC_FILTER_ON == mc_filter_flag[channel])
               {
                   if (RFS7_RMAF == (papp_rx_desc[channel]->status &
                         RFS7_RMAF))
                   {
                       R_ETHER_Read_ZC2_BufRelease(channel);

                       ret = ETHER_ERR_MC_FRAME;
                       complete_flag = ETHER_SUCCESS;
                   }
               }

               if (ETHER_ERR_MC_FRAME != ret)
               {
                   if (RFE == (papp_rx_desc[channel]->status & RFE))
                   {
                       R_ETHER_Read_ZC2_BufRelease(channel);
                   }
                   else
                   {
                       (*pbuf) = (void *) papp_rx_desc[channel]->buf_p;

                       num_recvd = papp_rx_desc[channel]->size;
                       ret = num_recvd;
                       complete_flag = ETHER_SUCCESS;
                   }
--------------------------- (omitted) --------------------------
} /* End of function R_ETHER_Read_ZC2() */
```

After modification:

```
int32_t R_ETHER_Read_ZC2 (uint32_t channel, void **pbuf)
{
-------------------------- (omitted) --------------------------
               if (ETHER_MC_FILTER_ON == mc_filter_flag[channel])
               {
                   if (RFS7_RMAF == (papp_rx_desc[channel]->status &
                         RFS7_RMAF))
                   {
                       ret2 = R_ETHER_Read_ZC2_BufRelease(channel);
                       if (ETHER_SUCCESS != ret2)
                       {
                           return ret2;
                       }

                       ret = ETHER_ERR_MC_FRAME;
                       complete_flag = ETHER_SUCCESS;
                   }
               }

               if (ETHER_ERR_MC_FRAME != ret)
               {
                   if (RFE == (papp_rx_desc[channel]->status & RFE))
                   {
                       ret2 = R_ETHER_Read_ZC2_BufRelease(channel);
                       if (ETHER_SUCCESS != ret2)
                       {
                           return ret2;
                       }
                   }
                   else
                   {
                       (*pbuf) = (void *) papp_rx_desc[channel]->buf_p;

                       num_recvd = papp_rx_desc[channel]->size;
                       ret = num_recvd;
                       complete_flag = ETHER_SUCCESS;
                   }
-------------------------- (omitted) --------------------------
} /* End of function R_ETHER_Read_ZC2() */
```

3.6 Schedule for Fixing the Problem

(1) Ethernet FIT module

This problem will be fixed in Rev.1.15, which will be the next release.

(2) RX Driver Package

The Ethernet FIT module Rev.1.15 modified in accord with this note will be included in Ver.1.15 of the RX family RX Driver Package, which will be the next release.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	May 16, 2018	-	First edition issued

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

■Inquiry
https://www.renesas.com/contact/