[Notes]

RX Family

Memory Access Driver Interface Firmware Integration Technology,
RX Driver Package

R20TS0636EJ0100
Rev.1.00
Dec. 01, 2020

Overview

When using any of the products in the title, note the following point.

1. Caution regarding a problem in RSPI Data Transfer (when both IAR compiler and big endian are used)

1. Caution regarding a problem in RSPI Data Transfer (when both IAR compiler and big endian are used)

1.1 Applicable Product

(1) Memory access driver interface module using Firmware Integration Technology (MEMDRV FIT module)

The applicable revision numbers and document numbers are as follows.

Table 1.1 MEMDRV FIT Module Applicable Products

Revision of the MEMDRV FIT module	Document number
Rev.1.02	R01AN4548EJ0102

(2) RX Driver Package

The MEMDRV FIT module in (1) is also included in the RX Driver Package products.
The product names and revision numbers of the applicable RX Driver Package and the revision numbers and document of the included MEMDRV FIT module are as follows.

Table 1.2 Products Which Include the MEMDRV FIT Module

Product name of the RX Driver Package	Revision of the RX Driver Package	Document number	Revision of the included MEMDRV FIT module
RX Driver Package for RX Family, Ver.1.26	Rev.1.26	R01AN5401EJ0126	Rev.1.02
RX Driver Package for RX Family, Ver.1.25	Rev.1.25	R01AN5371EJ0125	Rev.1.02
RX Driver Package for RX Family, Ver.1.24	Rev.1.24	R01AN5267EJ0124	Rev.1.02

1.2 Applicable MCUs

RX family

1.3 Description and Conditions

If all the following conditions apply, data is not transferred properly.

(a) IAR compiler is used.

(b) Big endian is used.

(c) Either R_MEMDRV_TxData() or R_MEMDRV_RxData() is used as the RSPI software transfer mode.

(d) Data of 4 or more bytes is transferred.

1.4 Detailed Description

When the transfer data size specified by the data length is 4 bytes or more, R_MEMDRV_TxData() and R_MEMDRV_RxData(), whose transfer data unit is 32-bit, convert and divide the data length into 4-byte units.

However, if the above (a) through (d) in 1.3 apply, data with transfer size of 4 bytes or more cannot be properly divided due to the faults in the following processing, causing this phenomenon.

- Data transfer command determination processing
- Data division processing

The following shows a source code for determining the data transfer command and dividing the transfer size inside the data transfer functions(Note).

Note: The data transfer functions are as follows.
- r_memdrv_rspi_write_data() called by R_MEMDRV_TxData()
- r_memdrv_rspi_read_data() called by R_MEMDRV_RxData()

■ For r_memdrv_rspi_write_data() source code:

For r_memdrv_rspi_read_data(), replace the description as follows.
Read r_memdrv_rspi_write_data() as r_memdrv_rspi_read_data().

r_memdrv_rspi_write_data() (Lines 2667 to 2677 in r_memdrv_rspi.c)

```
#if RSPI_LITTLE_ENDIAN == 1
    if (MEMDRV_TRNS_DATA_CMD == cmd.word[0])
    {
        count = count >> 2;
    }
#else
    if (MEMDRV_TRNS_DATA_CMD == cmd.word[1])
    {
        count = count >> 2;
    }
#endif
```

■ Source Code Description

For r_memdrv_rspi_read_data(), replace the description as follows.
Read r_memdrv_rspi_write_data() as r_memdrv_rspi_read_data().
Read R_RSPI_Write() as R_RSPI_Read().

To perform RSPI 32-bit software transfer (R_RSPI_Write()), the program divides the transfer data into 4-byte units. Here, the following equation holds:

−Len = Len1 * 4 + Len2 (Len2 >= 0 and Len2 < 32)

Len (bits): Transfer data length

Len1: Data frame length during 32-bit data transfer

Len2: Data length when data of less than 32 bits (including the divided data at the end) is transferred

The variable "count" indicated in red has the following two meanings in r_memdrv_rspi_write_data() and performs conversions accordingly in r_memdrv_rspi_write_data().

−Before conversion: The variable indicates the data length during data transfer.
(The second argument of r_memdrv_rspi_write_data(). Corresponds to (Len1 *4) or Len2.)

−After conversion: The variable indicates the number of transfer data frames.
(The fourth argument of R_RSPI_Write() that is called inside r_memdrv_rspi_write_data())

When the transfer data is four bytes or more (which corresponds to Len1 * 4), because data is transferred in 32-bit units, the ”count” variable, which indicates the data length, is divided by 4 (shifted right by 2 bits), and the result is reassigned to the ”count” variable as the number of transfer data frames.

If the transfer data is 3 bytes or less (corresponding to Len2), data is properly transferred (without the need to divide the data).

1.5 Workaround

Modify the code as shown in red in r_memdrv_rspi_write_data() and r_memdrv_rspi_read_data().

−Before modification

memdrv_rspi_write_data()　(Lines 2667 to 2677 in r_memdrv_rspi.c)

```
#if RSPI_LITTLE_ENDIAN == 1
    if (MEMDRV_TRNS_DATA_CMD == cmd.word[0])
    {
        count = count >> 2;
    }
#else
    if (MEMDRV_TRNS_DATA_CMD == cmd.word[1])
    {
        count = count >> 2;
    }
#endif
```

r_memdrv_rspi_read_data()　(Lines 2727 to 2737 in r_memdrv_rspi.c)

```
#if RSPI_LITTLE_ENDIAN == 1
    if (MEMDRV_TRNS_DATA_CMD == cmd.word[0])
    {
        count = count >> 2;
    }
#else
    if (MEMDRV_TRNS_DATA_CMD == cmd.word[1])
    {
        count = count >> 2;
    }
#endif
```

−After modification

memdrv_rspi_write_data()　(Lines 2667 to 2677 in r_memdrv_rspi.c)

```
#if RSPI_LITTLE_ENDIAN == 1
    if (MEMDRV_TRNS_DATA_CMD == cmd.word[0])
    {
        count = count >> 2;
    }
#else
    #if defined (__ICCRX__)
    uint16_t cmd_big_endian = 0;
    cmd_big_endian = (MEMDRV_TRNS_DATA_CMD >> 8);
    cmd_big_endian |= (MEMDRV_TRNS_DATA_CMD << 8);
    if (cmd_big_endian == cmd.word[0])
    {
        count = count >> 2;
    }
    #else
    if (MEMDRV_TRNS_DATA_CMD == cmd.word[1])
    {
        count = count >> 2;
    }
    #endif
#endif
```

r_memdrv_rspi_read_data() (Lines 2727 to 2737 in r_memdrv_rspi.c)

```
#if RSPI_LITTLE_ENDIAN == 1
    if (MEMDRV_TRNS_DATA_CMD == cmd.word[0])
    {
        count = count >> 2;
    }
#else
    #if defined (__ICCRX__)
    uint16_t cmd_big_endian = 0;
    cmd_big_endian = (MEMDRV_TRNS_DATA_CMD >> 8);
    cmd_big_endian |= (MEMDRV_TRNS_DATA_CMD << 8);
    if (cmd_big_endian == cmd.word[0])
    {
        count = count >> 2;
    }
    #else
    if (MEMDRV_TRNS_DATA_CMD == cmd.word[1])
    {
        count = count >> 2;
    }
    #endif
#endif
```

1.6 Schedule for fixing the problem

The problem has already been rectified in MEMDRV FIT Rev.1.03.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Dec.01.20	-	First edition issued

Corporate Headquarters

TOYOSU FORESIA, 3- 2- 24 Toyosu,
Koto-ku, Tokyo 135- 0061, Japan
www.renesas.com

Trademarks

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/

TS Colophon 4.1