【注意事項】

R20TS0636JJ0100
Rev.1.00
2020.12.01 号

RX ファミリ
メモリアクセス用ドライバインタフェース Firmware Integration Technology, RX Driver Package

概要

タイトルに記載している製品の使用上の注意事項を連絡します。

1. RSPI データ転送の不具合（IAR コンパイラかつビッグエンディアン使用時）に関する注意事項

1. RSPI データ転送の不具合（IAR コンパイラかつビッグエンディアン使用時）に関する注意事項

1.1 該当製品

(1) メモリアクセス用ドライバインタフェースモジュール Firmware Integration Technology（MEMDRV FIT モジュール）

該当するリビジョンおよびドキュメントは、以下のとおりです。

表 1.1 MEMDRV FIT モジュール該当製品一覧

MEMDRV FIT モジュールのリビジョン	資料番号
Rev.1.02	R01AN4548JJ0102

(2) RX Driver Package

(1)の MEMDRV FIT モジュールは、RX Driver Package にも同梱されています。
該当する RX Driver Package の製品名、リビジョン、および同梱している MEMDRV FIT モジュールのリビジョンおよびドキュメントは、以下のとおりです。

表 1.2 MEMDRV FIT モジュール同梱製品一覧

RX Driver Package の製品名	RX Driver Package のリビジョン	資料番号	同梱している MEMDRV FIT モジュールのリビジョン
RX ファミリ RX Driver Package Ver.1.26	Rev.1.26	R01AN5401JJ0126	Rev.1.02
RX ファミリ RX Driver Package Ver.1.25	Rev.1.25	R01AN5371JJ0125	Rev.1.02
RX ファミリ RX Driver Package Ver.1.24	Rev.1.24	R01AN5267JJ0124	Rev.1.02

1.2 該当デバイス

RX ファミリ

1.3 内容および発生条件

以下の条件をすべて満たす場合、意図したとおりのデータを送受信できません。

(a) IAR コンパイラを使用する

(b) ビッグエンディアンを使用する

(c) RSPI ソフトウェア転送モードで R_MEMDRV_TxData()または R_MEMDRV_RxData()を使用する

(d) 4 バイト以上のデータを転送する

1.4 内容の詳細説明

R_MEMDRV_TxData()および R_MEMDRV_RxData()は、32 ビット転送を実行するため、データ長で指定された転送サイズが 4 バイト以上の場合は、4 バイト単位で分割したデータ長に変換します。

しかし、1.3 項の(a)～(d)を満たしている場合、以下の処理に誤りがあるため、転送サイズが 4 バイト以上の時にデータを正しく分割できません。このため、本現象が発生します。

・データ転送コマンドの判定処理

・データ分割処理

以下は、データ転送を実行する関数(注)において、データ転送コマンドを判定して転送サイズを分割する処理を実行するためのソースコードです。

注：データ転送を実行する関数
・R_MEMDRV_TxData()から呼び出される r_memdrv_rspi_write_data()
・R_MEMDRV_RxData()から呼び出される r_memdrv_rspi_read_data()

■ ソースコード：r_memdrv_rspi_write_data() の場合
r_memdrv_rspi_read_data() の場合は、以下のとおり読み替えてください。
r_memdrv_rspi_write_data() → r_memdrv_rspi_read_data()

r_memdrv_rspi_write_data() （r_memdrv_rspi.c の 2667~2677 行目）

```
#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
```

■　ソースコードの説明

r_memdrv_rspi_read_data()の場合は、以下のとおり読み替えてください。
r_memdrv_rspi_write_data() → r_memdrv_rspi_read_data()
R_RSPI_Write() → R_RSPI_Read()

RSPI の 32 ビットソフトウェア転送（R_RSPI_Write()）を実行するために、転送データを 4 バイト単位で分割します。このとき、以下の等式が成り立ちます。

・Len = Len1 * 4 + Len2　　（Len2 >= 0 かつ Len2 < 32）

Len（bits）：転送データ長

Len1：32 ビット転送時のデータフレーム長

Len2：32 ビット未満のデータ（末尾の分割データを含む）転送時のデータ長

赤字の箇所の変数”count”は、r_memdrv_rspi_write_data()において以下の 2 つの意味があり、r_memdrv_rspi_write_data()内で変換操作が行われています。

・変換前：転送時のデータ長
（r_memdrv_rspi_write_data()の第 2 引数。（Len1 *4）または Len2 に相当。）

・変換後：転送データフレーム数
（r_memdrv_rspi_write_data()内で実行される R_RSPI_Write()の第 4 引数。）

転送データが 4 バイト以上（Len1 * 4 に相当）の場合、32 ビット転送なので、データ長を示す変数”count”を 4 で割り（2 ビット右シフト）、その計算結果を転送データフレーム数として変数”count”に再代入しています。

転送データが 3 バイト以下（Len2 に相当）の場合は分割する必要がないため、転送データの不具合は発生しません。

1.5　回避策

r_memdrv_rspi_write_data()および r_memdrv_rspi_read_data()を以下の赤字どおりに修正してください。

・修正前

memdrv_rspi_write_data()　（r_memdrv_rspi.c の 2667~2677 行目）

```
#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()　（r_memdrv_rspi.c の 2727~2737 行目）

```
#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
```

・修正後

memdrv_rspi_write_data()　（r_memdrv_rspi.c の 2667~2677 行目）

```
#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()　（r_memdrv_rspi.c の 2727~2737 行目）

```
#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　恒久対策

MEMDRV FIT Rev.1.03 で改修済みです。

以上

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Dec.01.20	-	新規発行

本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。

過去のニュース内容は発行当時の情報をもとにしており、現時点では変更された情報や無効な情報が含まれている場合があります。

ニュース本文中の URL を予告なしに変更または中止することがありますので、あらかじめご承知ください。

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/