

【注意事項】

R20TS0720JJ0100

Rev.1.00

2021.07.01 号

RX Driver Package,

RX ファミリ RSPI モジュール Firmware Integration Technology

概要

RX Driver Package および RX ファミリ RSPI モジュール Firmware Integration Technology（以下、RSPI FIT モジュール）の使用上の注意事項を連絡します。

1. “R_RSPI_Write” 関数、“R_RSPI_Read” 関数または “R_RSPI_WriteRead” 関数を呼び出す場合の注意事項

1. “R_RSPI_Write” 関数、“R_RSPI_Read” 関数または “R_RSPI_WriteRead” 関数を呼び出す場合の注意事項

1.1 該当製品

- (1) RX ファミリ RSPI モジュール Firmware Integration Technology

該当するリビジョンとドキュメントは「表 1.1 RSPI FIT モジュール該当製品一覧」のとおりです。

表 1.1 RSPI FIT モジュール該当製品一覧

RSPI FIT モジュールのリビジョン	資料番号
Rev.1.00	R01AN1817JJ0100
Rev.1.20	R01AN1817JJ0120
Rev.1.30	R01AN1817JJ0130
Rev.1.40	R01AN1817JJ0140
Rev.1.50	R01AN1817JJ0150
Rev.1.60	R01AN1817JJ0160
Rev.1.70	R01AN1817JJ0170
Rev.1.80	R01AN1817JJ0180
Rev.2.00	R01AN1817JJ0200
Rev.2.01	R01AN1817JJ0201
Rev.2.02	R01AN1817JJ0202
Rev.2.03	R01AN1817JJ0203
Rev.2.04	R01AN1817JJ0204
Rev.2.05	R01AN1817JJ0205
Rev.3.00	R01AN1817JJ0300

上記 RSPI FIT モジュールを同梱^{注1}している RX Driver Package の下記製品も対象になります。

注 1: r_rspi_rx_v*.**.zip（*.**はリビジョン番号）として RSPI FIT モジュールが同梱されています。

(2) RX Driver Package

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 RSPI FIT モジュールのリビジョンは、「表 1.2 RSPI FIT モジュール同梱製品一覧」のとおりです。

表 1.2 RSPI 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 RSPI FIT モジュールの リビジョン
RX110, RX111, RX113, RX231 グループ RX Driver Package Ver.1.01	Rev.1.01	R01AN2670JJ0101	Rev.1.40
RX110, RX111, RX113, RX130, RX231, RX23T グループ RX Driver Package Ver.1.02	Rev.1.02	R01AN3159JJ0102	Rev.1.40
RX110, RX111, RX113, RX130, RX230, RX231, RX23T, RX24T グループ RX Driver Package Ver.1.03	Rev.1.03	R01AN3233JJ0103	Rev.1.40
RX64M, RX71M Group RX Driver Package Ver.1.02	Rev.1.04	R01AN2606JJ0104	Rev.1.30
RX ファミリ RX Driver Package Ver.1.11	Rev.1.11	R01AN3467JJ0111	Rev.1.50
RX ファミリ RX Driver Package Ver.1.12	Rev.1.12	R01AN3651JJ0112	Rev.1.60
RX ファミリ RX Driver Package Ver.1.13	Rev.1.13	R01AN3859JJ0113	Rev.1.70
RX ファミリ RX Driver Package Ver.1.14	Rev.1.14	R01AN4191JJ0114	Rev.1.70
RX ファミリ RX Driver Package Ver.1.15	Rev.1.15	R01AN4372JJ0115	Rev.1.70
RX ファミリ RX Driver Package Ver.1.16	Rev.1.16	R01AN4471JJ0116	Rev.1.80
RX ファミリ RX Driver Package Ver.1.17	Rev.1.17	R01AN4572JJ0117	Rev.1.80
RX ファミリ RX Driver Package Ver.1.18	Rev.1.18	R01AN4659JJ0118	Rev.1.80
RX ファミリ RX Driver Package Ver.1.19	Rev.1.19	R01AN4677JJ0119	Rev.2.00
RX ファミリ RX Driver Package Ver.1.20	Rev.1.20	R01AN4794JJ0120	Rev.2.01
RX ファミリ RX Driver Package Ver.1.21	Rev.1.21	R01AN4843JJ0121	Rev.2.02
RX ファミリ RX Driver Package Ver.1.22	Rev.1.22	R01AN4873JJ0122	Rev.2.03
RX ファミリ RX Driver Package Ver.1.23	Rev.1.23	R01AN4976JJ0123	Rev.2.03
RX ファミリ RX Driver Package Ver.1.24	Rev.1.24	R01AN5267JJ0124	Rev.2.04
RX ファミリ RX Driver Package Ver.1.25	Rev.1.25	R01AN5371JJ0125	Rev.2.04

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 RSPi 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.2.05
RX ファミリ RX Driver Package Ver.1.27	Rev.1.27	R01AN5600JJ0127	Rev.3.00
RX ファミリ RX Driver Package Ver.1.28	Rev.1.28	R01AN5761JJ0128	Rev.3.00
RX ファミリ RX Driver Package Ver.1.29	Rev.1.29	R01AN5826JJ0129	Rev.3.00
RX ファミリ RX Driver Package Ver.1.30	Rev.1.30	R01AN5882JJ0130	Rev.3.00

1.2 該当デバイス

RX110、RX111、RX113、RX130、RX231、RX23E-A、RX23W、RX23T、RX24T、RX24U、RX64M、RX65N、RX651、RX66T、RX66N、RX71M、RX72T、RX72M、RX72N、RX210^{注1}、RX62N^{注1}、RX62T^{注1}、RX63N^{注1}およびRX631^{注1}グループ

注 1: RSPI FIT モジュール Rev.2.01 以降では RX210、RX62N、RX62T、RX63N および RX631 グループはサポートされません。

1.3 内容

RSPI FIT の設定をマスタモード動作に設定した状態で、“R_RSPI_Write” 関数、“R_RSPI_Read” 関数または“R_RSPI_WriteRead” 関数を呼び出した場合、シリアル転送の最終データを受信後、SSL ネゲート遅延及び次アクセス遅延を待たずに RSPI 機能が停止します。

1.4 発生条件

マスタモード動作に設定した状態で“R_RSPI_Write” 関数、“R_RSPI_Read” 関数または“R_RSPI_WriteRead” 関数を呼び出した場合、本現象が発生します。

1.5 回避策

Rev.2.00 より前の RSPI FIT モジュールのソースコード “r_rsapi_rx.c” の”rsapi_tx_rx_common”関数処理、または Rev.2.00 以降の RSPI FIT モジュールのソースコード “r_rsapi_rx.c” の”rsapi_rx_common”関数処理を変更してください。

以下に詳細を記します。赤字の内容を追加してください。

修正前(Rev.2.00 より前):

```
/* Check for last data. */
if(rx_count == g_rsapi_tcb[channel].xfr_length)
{
    /* Last data was transferred. */
    (*g_rsapi_channels[channel]).SPCR.BIT.SPRIE = 0; /* Disable SPRI
interrupt. */
    (*g_rsapi_channels[channel]).SPCR.BIT.SPE = 0; /* Disable RSPI. */
    if (0 == (*g_rsapi_channels[channel]).SPCR.BIT.SPE)
    {
        nop();
    }

    #if RSPI_CFG_REQUIRE_LOCK == 1
    /* Release lock for this channel. */
    R_BSP_HardwareUnlock((mcu_lock_t)(BSP_LOCK_RSPIO + channel));
    #endif

    /* Transfer complete. Call the user callback function passing
pointer to the result structure. */
    if((FIT_NO_FUNC != g_rsapi_handles[channel].pcallback) && (NULL !=
g_rsapi_handles[channel].pcallback))
    {
        g_rsapi_cb_data[channel].handle = &(g_rsapi_handles[channel]);
        g_rsapi_cb_data[channel].event_code = RSPI_EVT_TRANSFER_COMPLETE;
g_rsapi_handles[channel].pcallback((void*)&(g_rsapi_cb_data[channel]));
    }
}
```

修正後(Rev.2.00 より前):

```

/* Check for last data. */
if(rx_count == g_rspi_tcb[channel].xfr_length)
{
    /* Last data was transferred. */
    (*g_rspi_channels[channel]).SPCR.BIT.SPRIE = 0; /* Disable SPRI
interrupt. */

    while(1==(*g_rspi_channels[channel]).SPSR.BIT.IDLNF)
    {
        nop();
    }

    (*g_rspi_channels[channel]).SPCR.BIT.SPE = 0; /* Disable RSPI. */
    if (0 == (*g_rspi_channels[channel]).SPCR.BIT.SPE)
    {
        nop();
    }

    #if RSPI_CFG_REQUIRE_LOCK == 1
    /* Release lock for this channel. */
    R_BSP_HardwareUnlock((mcu_lock_t)(BSP_LOCK_RSPI0 + channel));
    #endif

    /* Transfer complete. Call the user callback function passing
pointer to the result structure. */
    if((FIT_NO_FUNC != g_rspi_handles[channel].pcallback) && (NULL !=
g_rspi_handles[channel].pcallback))
    {
        g_rspi_cb_data[channel].handle = &(g_rspi_handles[channel]);
        g_rspi_cb_data[channel].event_code = RSPI_EVT_TRANSFER_COMPLETE;

        g_rspi_handles[channel].pcallback((void*)&(g_rspi_cb_data[channel]));
    }
}

```

修正前(Rev.2.00 以降):

```

/* Check for last data. */
if (rx_count == g_rspi_tcb[channel].xfr_length)
{
    /* Last data was transferred. */
#ifdef RSPI_CFG_HIGH_SPEED_READ == 1
        (*g_rspi_channels[channel]).SPCR.BIT.SPTIE = 0; // Disable SPTI
        interrupt.
#endif
        (*g_rspi_channels[channel]).SPCR.BIT.SPRIE = 0; // Disable SPRI
        interrupt.
        (*g_rspi_channels[channel]).SPCR.BIT.SPE = 0; // Disable RSPI.
        if (0 == (*g_rspi_channels[channel]).SPCR.BIT.SPE)
        {
            nop();
        }
#ifdef RSPI_CFG_REQUIRE_LOCK == 1
        /* Release lock for this channel. */
        R_BSP_HardwareUnlock((mcu_lock_t)(BSP_LOCK_RSPI0 + channel));
#endif

        /* Transfer complete. Call the user callback function passing
        pointer to the result structure. */
        if ((FIT_NO_FUNC != g_rspi_handles[channel].pcallback) && (NULL !=
        g_rspi_handles[channel].pcallback))
        {
            g_rspi_cb_data[channel].handle = &(g_rspi_handles[channel]);
            g_rspi_cb_data[channel].event_code = RSPI_EVT_TRANSFER_COMPLETE;

            g_rspi_handles[channel].pcallback((void*)&(g_rspi_cb_data[channel]));
        }
    }
}

```

修正後(Rev.2.00 以降):

```

/* Check for last data. */
if (rx_count == g_rspi_tcb[channel].xfr_length)
{
    /* Last data was transferred. */
#ifdef RSPI_CFG_HIGH_SPEED_READ == 1
        (*g_rspi_channels[channel]).SPCR.BIT.SPTIE = 0; // Disable SPTI
        interrupt.
#endif
        (*g_rspi_channels[channel]).SPCR.BIT.SPRIE = 0; // Disable SPRI
        interrupt.
        while(1==(*g_rspi_channels[channel]).SPSR.BIT.IDLNF)
        {
            nop();
        }
        (*g_rspi_channels[channel]).SPCR.BIT.SPE = 0; // Disable RSPI.
        if (0 == (*g_rspi_channels[channel]).SPCR.BIT.SPE)
        {
            nop();
        }

#ifdef RSPI_CFG_REQUIRE_LOCK == 1
        /* Release lock for this channel. */
        R_BSP_HardwareUnlock((mcu_lock_t) (BSP_LOCK_RSPI0 + channel));
#endif

        /* Transfer complete. Call the user callback function passing
        pointer to the result structure. */
        if ((FIT_NO_FUNC != g_rspi_handles[channel].pcallback) && (NULL !=
        g_rspi_handles[channel].pcallback))
        {
            g_rspi_cb_data[channel].handle = &(g_rspi_handles[channel]);
            g_rspi_cb_data[channel].event_code = RSPI_EVT_TRANSFER_COMPLETE;
            g_rspi_handles[channel].pcallback((void*)&(g_rspi_cb_data[channel]));
        }
    }
}

```

1.6 恒久対策

(1) RSPI FIT モジュール

今後のバージョンで改修予定です。

(2) RX Driver Package

今後リリースされる RX ファミリ RX Driver Package にて、本注意事項を改修した RSPI FIT モジュールを同梱する予定です。

以上

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Jul.01.21	-	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。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/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。