RENESAS ツールニュース

RENESAS TOOL NEWS 2012年06月01日：120601/tn10

Renesas Peripheral Driver Library および Peripheral Driver Generator ご使用上のお願い --RX630グループマイコンのIEBus使用時の注意事項--

Renesas Peripheral Driver Libraryおよび Peripheral Driver Generatorの 使用上の注意事項を連絡します。

- RX630グループマイコンのIEBus使用時の注意事項

---

1. 該当製品

- RX630 Group Renesas Peripheral Driver Library V.1.00
- Peripheral Driver Generator V.2.04以降

2. 内容

該当製品を使用して生成されたコード中で、IEBusでのデータ送受信完了後、
以下のいずれかのコールバック関数が呼び出される場合は、IEBus送受信
ステータスレジスタのクリアが必要になります。

(1) 該当Renesas Peripheral Driver Libraryの以下のコールバック関数を使用する場合
- R_IEB_MasterSend関数
- R_IEB_MasterReceive関数
- R_IEB_SlaveMonitor関数

(2) 該当Peripheral Driver Generatorで以下の関数を設定した場合
- IEBのコールバック関数

IEBus送受信ステータスレジスタのクリアを行わない場合、
ステータスレジスタをクリアするまでIEBus割り込みが発生し続け、
その度、コールバック関数が呼び出されます。

2.1 発生条件

(1) Renesas Peripheral Driver Libraryの場合

R_IEB_MasterSend関数、R_IEB_MasterReceive関数、または
R_IEB_SlaveMonitor関数の引数にコールバック関数を設定している。

(2) Peripheral Driver Generatorの場合
以下のいずれかの設定を行っている。
- IEBの「マスタ送信方法」リストから「全データの送信完了、エラー検出を関数呼び出しで通知する」を選択。
- IEBの「マスタ受信方法」リストから「全データの受信完了、エラー検出を関数呼び出しで通知する」を選択。
- IEBの「スレーブモニタ方法」リストから「全データの受信完了、スレーブリード要求、エラー検出を関数呼び出しで通知する」を選択。

2.2 発生例

(1) Renesas Peripheral Driver Libraryの場合

```
------------------------------------------------------------------
// スレーブの送信データをセット
R_IEB_SlaveWrite(
   0,
   iebus_data,
   iebus_data_length
);

// IEBusのR_IEB_SlaveMonitor関数にコールバック関数を設定
/* Monitor channel 0, using polling */
R_IEB_SlaveMonitor(
   0,
   iebus_data,
   &iebus_data_length,
   IEB_CALL_BACK_FUNC
);

// コールバック関数でステータスフラグをクリアしない
Void IEB_CALL_BACK_FUNC(){

}
------------------------------------------------------------------
```

(2) Peripheral Driver Generatorの場合

(a) IEBのデータを下のとおり入力する。
- 「マスタ送信方法」リストから「全データの送信完了、エラー検出を関数呼び出しで通知する」を選択。
- マスタ送信の「通知関数名」に Ieb0MasterTrFunc を入力。

(b) 以下のプログラムを実行する。

```
------------------------------------------------------------------
 void SetupFunction(void)
 {
```

```
    uint8_t tx[]="ABCDEFG";
    R_PG_IEB_Set_C0();    // IEBの設定
    R_PG_IEB_MasterSendData_C0( 1, tx, 5);    // マスタ送信
  }

  void Ieb0MasterTrFunc(void){
    Function();    // 送信完了時処理 (送信ステータスレジスタを
                   // クリアしない)
  }
  ------------------------------------------------------------
```

3. 回避策

コールバック関数内で、R_IEB_GetStatus関数を用いて以下の処理を行なってください。

(1) 送信完了フラグのクリア

- R_IEB_GetStatus関数をコールし、送信完了フラグ (第3引数のbit5) が"1" (送信完了) であるかを確認する。
- 送信完了フラグ (第3引数のbit5) が"1"である場合、以下の処理を実施する。

```
IEB.IETSR.BYTE = 0x6Fu;    // 送信ステータスフラグのクリア
```

(2) 受信完了フラグのクリア

- R_IEB_GetStatus関数をコールし、送信完了フラグ (第3引数のbit5) が"1" (送信完了) であるかを確認する。

```
IEB.IERSR.BYTE = 0xFFu;    //受信ステータスフラグのクリア
```

3.1 Renesas Peripheral Driver Libraryの回避例

```
------------------------------------------------------------
// スレーブの送信データをセット
R_IEB_SlaveWrite(
    0,
    iebus_data,
    iebus_data_length
);

// IEBusのR_IEB_SlaveMonitor関数にコールバック関数を設定
/* Monitor channel 0, using call back function */
R_IEB_SlaveMonitor(
    0,
    iebus_data,
    &iebus_data_length,
    IEB_CALL_BACK_FUNC
);
```

```
// IEB割り込みで発生するコールバック関数内でR_IEB_GetStatus関数で
// 送受信の完了を確認しステータスフラグをクリアする
Void IEB_CALL_BACK_FUNC(){
   R_IEB_GetStatus(
      0,
      &StatusValue,
      &SendstatusValue,
      &RcvstatusValue,
      PDL_NO_PTR,
      PDL_NO_PTR
   );

// 送信完了フラグが"1"か確認
   if((SendstatusValue & 0x20) != 0x00){
      IEB.IETSR.BYTE = 0x6Fu;    // 送信ステータスフラグのクリア
   }

// 受信完了フラグが"1"か確認
   if((RcvstatusValue & 0x20) != 0x00){
      // R_IEB_MasterReceiveの第4引数 (本例では&iebus_data_length)
      // の受信lengthが"0"でないことを確認
      if(iebus_data_length != 0x00){
         IEB.IERSR.BYTE = 0xFFu;    //受信ステータスフラグのクリア
      }
   }
}
-----------------------------------------------------------------
```

3.2 Peripheral Driver Generatorの回避例

(1) IEBの「マスタ送信方法」リストから「全データの送信完了、エラー検出を関数呼び出しで通知する」を選択し、マスタ送信の「通知関数名」に Ieb0MasterTrFunc を入力した場合の回避例

```
-----------------------------------------------------------------
// I/Oレジスタ定義ファイルのインクルード
#include "iodefine_Renesas Peripheral Driver Library.h"

// マスタ送信通知関数
void Ieb0MasterTrFunc(void){
   bool complete;
//送信完了フラグの取得
   R_PG_IEB_GetTransmitStatus_C0(
      0,
      &complete,
      0,
      0,
```

```
        0,
        0
    );
    if(complete){
    // 送信ステータスレジスタのクリア
        IEB.IETSR.BYTE = 0x6Fu;
    }
    Function();    // 送信完了時処理
}
 ---------------------------------------------------------------
```

(2) IEBの「マスタ受信方法」リストから 「全データの受信完了、エラー検出を関数呼び出しで通知する」を選択し、マスタ受信の「通知関数名」にIeb0MasterReFuncを入力した場合の回避例

```
---------------------------------------------------------------
// マスタ受信通知関数
// I/Oレジスタ定義ファイルのインクルード
#include "iodefine_Renesas Peripheral Driver Library.h"

void Ieb0MasterReFunc(void){
    bool complete;
    uint8_t count;
// 受信完了フラグの取得
    R_PG_IEB_GetReceiveStatus_C0(
        0,
        0,
        &complete,
        0,
        0,
        0,
        0,
        0
    );
    R_PG_IEB_GetReceivedDataCount_C0(&count);
    if((complete)&&(count)){
    // 受信ステータスレジスタのクリア
        IEB.IERSR.BYTE = 0xFFu;
    }
    Function();    // 受信完了時処理
}
---------------------------------------------------------------
```

(3) 「スレーブモニタ方法」リストから「全データの受信完了、スレーブリード要求、エラー検出を関数呼び出しで通知する」を選択し、スレーブモニタの「通知関数名」に Ieb0SlaveFunc を入力した場合の回避例

```
------------------------------------------------------------
// I/Oレジスタ定義ファイルのインクルード
#include "iodefine_Renesas Peripheral Driver Library.h"

// スレーブモニタ通知関数
void Ieb0SlaveFunc(void){
// 送信完了フラグの取得
   bool tx_complete;
   bool rx_complete;
   uint8_t count;
   R_PG_IEB_GetTransmitStatus_C0(
      0,
      &tx_complete,
      0,
      0,
      0,
      0
   );
   if(tx_complete){
   // 送信ステータスレジスタのクリア
      IEB.IETSR.BYTE = 0x6Fu;
   }
   // 受信完了フラグの取得
   R_PG_IEB_GetReceiveStatus_C0(
      0,
      0,
      &rx_complete,
      0,
      0,
      0,
      0,
      0
   );
   R_PG_IEB_GetReceivedDataCount_C0(&count);
   if((rx_complete)&&(count)){
   // 受信ステータスレジスタのクリア
      IEB.IERSR.BYTE = 0xFFu;
   }
   Function();    // イベント検出時処理
}
------------------------------------------------------------
```

4. 恒久対策

今後のバージョンで改修する予定です。
