[Notes]
RX Family
SCI Module Firmware Integration Technology,
RX Driver Package

Outline

When using the products in the title, note the following point.
1. Notes on possible junk byte being transmitted while using asynchronous mode

1. Notes on possible junk byte being sent out while using asynchronous mode

1.1 Applicable Products

(1) SCI module Firmware Integration Technology (SCI FIT module)

The applicable revision numbers and document numbers are as follows.

Table 1.1 SCI FIT module applicable products

<table>
<thead>
<tr>
<th>Revision number of the SCI FIT module</th>
<th>Document number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rev.3.70</td>
<td>R01AN1815EJ0370</td>
</tr>
<tr>
<td>Rev.3.60</td>
<td>R01AN1815EJ0360</td>
</tr>
<tr>
<td>Rev.3.50</td>
<td>R01AN1815EJ0350</td>
</tr>
<tr>
<td>Rev.3.40</td>
<td>R01AN1815EJ0340</td>
</tr>
<tr>
<td>Rev.3.30</td>
<td>R01AN1815EJ0330</td>
</tr>
<tr>
<td>Rev.3.21</td>
<td>R01AN1815EJ0321</td>
</tr>
<tr>
<td>Rev.3.20</td>
<td>R01AN1815EJ0320</td>
</tr>
<tr>
<td>Rev.3.10</td>
<td>R01AN1815EJ0310</td>
</tr>
<tr>
<td>Rev.3.00</td>
<td>R01AN1815EJ0300</td>
</tr>
<tr>
<td>Rev.2.20</td>
<td>R01AN1815EJ0220</td>
</tr>
<tr>
<td>Rev.2.11</td>
<td>R01AN1815EJ0211</td>
</tr>
<tr>
<td>Rev.2.10</td>
<td>R01AN1815EJ0210</td>
</tr>
<tr>
<td>Rev.2.01</td>
<td>R01AN1815EJ0201</td>
</tr>
<tr>
<td>Rev.2.00</td>
<td>R01AN1815EJ0200</td>
</tr>
<tr>
<td>Rev.1.90</td>
<td>R01AN1815EJ0190</td>
</tr>
<tr>
<td>Rev.1.80</td>
<td>R01AN1815EJ0180</td>
</tr>
<tr>
<td>Rev.1.70</td>
<td>R01AN1815EU0170</td>
</tr>
</tbody>
</table>

(2) RX Driver Package

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

Table 1.2 Products which include the SCI FIT module

<table>
<thead>
<tr>
<th>RX Driver Package product name</th>
<th>RX Driver Package revision number</th>
<th>Document number</th>
<th>Revision number of the included SCI FIT module</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX Family RX Driver Package Ver.1.30</td>
<td>Rev.1.30</td>
<td>R01AN5882xx0130</td>
<td>Rev.3.70</td>
</tr>
<tr>
<td>RX Family RX Driver Package Ver.1.29</td>
<td>Rev.1.29</td>
<td>R01AN5826xx0129</td>
<td>Rev.3.70</td>
</tr>
</tbody>
</table>
1.2 Applicable Devices

RX110, RX111, RX113, and RX130, RX13T groups
RX230, RX231, RX23E-A, RX23W, RX23T, RX24T, and RX24U groups
RX64M, RX65N, RX66N, and RX66T groups
RX71M, RX72T, RX72M, and RX72N groups

1.3 Details and Conditions

We have identified a potential error where asynchronous R_SCI_Send() and R_SCI_Receive() are called in a program. Under certain condition, an additional garbage byte will be transmitted.

We have also identified the root cause as the IENx bit is corrupted due to non-atomic macros DISABLE_RXI_INT and ENABLE_RXI_INT; which are called in an asynchronous receive routine.
We use RX65N TXI5/RXI5 as example below:

Interrupt request enable bit for RXI5 and TXI5 are IER0A.IEN4 and IER0A.IEN5 respectively. Located in the same register means potential of corruption (due to read-modify-write) if the operation on the register is not atomic

IER0A register (RX65N):

<table>
<thead>
<tr>
<th>b7</th>
<th>b6</th>
<th>b5</th>
<th>b4</th>
<th>b3</th>
<th>b2</th>
<th>b1</th>
<th>b0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEN7</td>
<td>IEN6</td>
<td>IEN5</td>
<td>IEN4</td>
<td>IEN3</td>
<td>IEN2</td>
<td>IEN1</td>
<td>IEN0</td>
</tr>
</tbody>
</table>

IER0A.IEN4 is used to enable or disable RXI5
IER0A.IEN5 is used to enable or disable TXI5

In asynchronous R_SCI_Receive() routine, the following macros will be called:

DISABLE_RXI_INT: IEN4 will be cleared to ‘0’
ENABLE_RXI_INT: IEN4 will be set to ‘1’

Note that these two macros are not atomic (verified with disassembly code)

```c
static sci_err_t sci_receive_async_data(sci_hdl_t const hdl, uint8_t *p_dst, uint16_t const length)
{
    ...
    DISABLE_RXI_INT;     // this is not atomic! ❶
    byteq_err = R_BYTEQ_Get(hdl->u_rx_data.que, p_dst++);
    ENABLE_RXI_INT;     // this is not atomic ❷
    ...
}
```

In asynchronous R_SCI_Send() routine, the following macro will be called when the last byte is copied to TDR register:

DISABLE_TXI_INT: IEN5 will be cleared to ‘0’

```c
void txi_handler(sci_hdl_t const hdl) ❸
{
    ...
    DISABLE_TXI_INT;  // this will be called when last byte of data is copied to TDR
        // Clear IEN5 to ‘0’
    ...
}
```

The following explains how IEN5 can be corrupted, and how it results to additional garbage data being transmitted:
void main(void)
{
    ...
    R_SCI_Send(…);   // txi_handler() will be triggered n times if n bytes are to be sent  
    R_SCI_Receive(…); // txi_handler() could be triggered while R_SCI_Receive() is in progress  
    ...
}

- When all n bytes are transmitted, expected value of IEN5 is ‘0’
- But if either ❶ or ❷ is interrupted by the last (nth) txi_handler() (❸), IEN5 could
  unexpectedly become ‘1’ due to read-modify-write at the end of non-atomic ❶ ❷

When this happens, txi_handler() will be triggered once more, even though all data has been
sent. This results in an additional garbage byte being transmitted

1.4 Workaround
Make the following macros atomic (note that RX65N is used as example):

Before modification

```
#define ENABLE_RXI_INT      (*hdl->rom->icu_rxi |= hdl->rom->rxi_en_mask)
#define DISABLE_RXI_INT     (*hdl->rom->icu_rxi &= (uint8_t)~hdl->rom->rxi_en_mask)
```

After modification

```
#define ENABLE_RXI_INT      (R_BSP_BIT_SET(hdl->rom->icu_rxi, hdl->rom->rxi_bit_num))
#define DISABLE_RXI_INT     (R_BSP_BIT_CLEAR(hdl->rom->icu_rxi, hdl->rom->rxi_bit_num))
```

1.5 Schedule for Fixing the Problem
This problem will be fixed in Rev.3.80.
Revision History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Description</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Sep.16.21</td>
<td>-</td>
<td></td>
<td>First edition issued</td>
</tr>
</tbody>
</table>

Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.

The past news contents have been based on information at the time of publication. Now changed or invalid information may be included.

The URL in the Tool News also may be subject to change or become invalid without prior notice.

Corporate Headquarters

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

www.renesas.com

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/

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.