Abstract

This document describes an example of using communication (single-master transmission/reception) via the I²C bus interface (hereinafter called "RIIC") of the RZ/A1H to implement read and write access to the EEPROM.

Products

RZ/A1H

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.
Contents

1. Specifications ................................................................................................................................. 4

2. Operation Confirmation Conditions ............................................................................................. 5

3. Reference Application Notes ......................................................................................................... 5

4. Hardware ....................................................................................................................................... 6
   4.1 Hardware Configuration .................................................................................................................. 6
   4.2 EEPROM Specifications ............................................................................................................... 6
   4.3 Pins Used .................................................................................................................................... 7

5. Software ....................................................................................................................................... 8
   5.1 Operation Overview ....................................................................................................................... 8
      5.1.1 Writing to EEPROM .................................................................................................................. 8
      5.1.2 Reading from EEPROM ........................................................................................................... 9
      5.1.3 Acknowledge Polling ................................................................................................................ 10
      5.1.4 Operation Sequence ................................................................................................................ 11
   5.2 Peripheral Function Settings and Memory Allocation Used in Sample Code ................................ 12
      5.2.1 Peripheral Function Settings .................................................................................................. 12
      5.2.2 Section Assignment of Sample Code ...................................................................................... 13
   5.3 Interrupts ................................................................................................................................... 16
   5.4 Fixed-Width Integers ................................................................................................................... 16
   5.5 Constants .................................................................................................................................. 17
   5.6 Variables .................................................................................................................................... 18
   5.7 Functions .................................................................................................................................. 19
   5.8 Function Specifications ............................................................................................................... 23
   5.9 Flowcharts ................................................................................................................................ 48
      5.9.1 Main Function .......................................................................................................................... 48
      5.9.2 Sample Code Main Function .................................................................................................. 49
      5.9.3 RIIC Sample Code Main Function ......................................................................................... 50
      5.9.4 Sample Function for Reading/Writing Data From/To EEPROM ............................................ 51
      5.9.5 Initial Setting Function for PORT ............................................................................................ 52
      5.9.6 Sample Function for Writing to EEPROM ........................................................................... 53
      5.9.7 Sample Function for Reading from EEPROM ........................................................................ 54
      5.9.8 Sample Function to Wait for Writing Completion Inside EEPROM ..................................... 56
      5.9.9 RIIC Channel 2 Receive Data Full Interrupt Function ............................................................. 57
      5.9.10 RIIC Channel 2 Transmit Data Empty Interrupt Function .................................................. 57
      5.9.11 RIIC Channel 2 Transmit-End Interrupt Function .................................................................. 57
      5.9.12 RIIC Initial Setting Function ................................................................................................ 58
      5.9.13 RIIC Condition Issuance Request Function .......................................................................... 59
      5.9.14 RIIC Slave Address Transmit Function .................................................................................. 63
      5.9.15 RIIC Transmit Function ........................................................................................................ 65
      5.9.16 RIIC Receive Function ......................................................................................................... 67
      5.9.17 RIIC Dummy Read Function .................................................................................................. 74
      5.9.18 Checking Function of RIIC Stop Condition Detection .......................................................... 75
      5.9.19 Clear Function of RIIC NACK Receive Flag ........................................................................... 76

Sample Function to Wait for Writing Completion Inside EEPROM .................................... 56
Sample Function for Writing to EEPROM ........................................................................... 53
Initial Setting Function for PORT .......................................................................................... 52
Sample Function for Writing to EEPROM ........................................................................... 53
Sample Function for Reading from EEPROM ........................................................................ 54
Sample Function to Wait for Writing Completion Inside EEPROM ..................................... 56
RIIC Channel 2 Receive Data Full Interrupt Function ............................................................. 57
RIIC Channel 2 Transmit Data Empty Interrupt Function .................................................. 57
RIIC Channel 2 Transmit-End Interrupt Function .................................................................. 57
RIIC Initial Setting Function ........................................................................................................ 58
RIIC Condition Issuance Request Function ............................................................................ 59
RIIC Slave Address Transmit Function ...................................................................................... 63
RIIC Transmit Function ............................................................................................................ 65
RIIC Receive Function ................................................................................................................. 67
RIIC Dummy Read Function ......................................................................................................... 74
Checking Function of RIIC Stop Condition Detection .......................................................... 75
Clear Function of RIIC NACK Receive Flag ............................................................................ 76
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.9.20</td>
<td>RIIC Transmit Data Empty Interrupt Notification Function ........................................... 77</td>
</tr>
<tr>
<td>5.9.21</td>
<td>RIIC Transmit-End Interrupt Notification Function ....................................................... 78</td>
</tr>
<tr>
<td>5.9.22</td>
<td>RIIC Receive Data Full Interrupt Notification Function .................................................. 79</td>
</tr>
<tr>
<td>5.9.23</td>
<td>RIIC Channel 2 Initial Setting Function ........................................................................... 80</td>
</tr>
<tr>
<td>5.9.24</td>
<td>Initialization Function of RIIC Channel 2 Receive Data Full Notification Information .... 82</td>
</tr>
<tr>
<td>5.9.25</td>
<td>Initialization Function of RIIC Channel 2 Transmit Data Empty Notification Information .. 82</td>
</tr>
<tr>
<td>5.9.26</td>
<td>Initialization Function of RIIC Transmit-End Notification Information .............................. 83</td>
</tr>
<tr>
<td>5.9.27</td>
<td>Setting Function to Satisfy the Conditions for RIIC Channel 2 Receive Data Full Notification Information ............................................................................................................. 84</td>
</tr>
<tr>
<td>5.9.28</td>
<td>Setting Function to Satisfy the Conditions for RIIC Channel 2 Transmit Data Empty Notification Information ............................................................................................................. 84</td>
</tr>
<tr>
<td>5.9.29</td>
<td>Setting Function to Satisfy the Conditions for RIIC Channel 2 Transmit-End Notification Information ............................................................................................................. 85</td>
</tr>
<tr>
<td>5.9.30</td>
<td>Wait Function for Satisfaction of the Conditions for RIIC Channel 2 Receive Data Full Notification Information ............................................................................................................. 86</td>
</tr>
<tr>
<td>5.9.31</td>
<td>Wait Function for Satisfaction of the Conditions for RIIC Channel 2 Transmit Data Empty Notification Information ............................................................................................................. 86</td>
</tr>
<tr>
<td>5.9.32</td>
<td>Wait Function for Satisfaction of the Conditions for RIIC Channel 2 Transmit-End Notification Information ............................................................................................................. 87</td>
</tr>
<tr>
<td>5.9.33</td>
<td>Wait Function for RIIC Channel 2 Bus Free and Bus Busy ..................................................... 88</td>
</tr>
<tr>
<td>5.9.34</td>
<td>Wait Function for RIIC Channel 2 Stop Condition Detection ................................................ 88</td>
</tr>
<tr>
<td>5.10</td>
<td>Running Sample Code ........................................................................................................... 89</td>
</tr>
</tbody>
</table>

6. Sample Code ......................................................................................................................... 90

7. Reference Documents ........................................................................................................... 90
1. Specifications

The RIIC is used to write/read data to/from the EEPROM. The RZ/A1H operates as the master device and the EEPROM as the slave device.

In this application note, the interrupt controller, the serial communication interface with FIFO, the general I/O port, and the power-down mode are abbreviated as INTC, SCIF, PORT, and STB respectively.

Table 1.1 lists the Peripheral Functions and Their Applications and Figure 1.1 shows the Operation Overview.

### Table 1.1 Peripheral Functions and Their Applications

<table>
<thead>
<tr>
<th>Peripheral Function</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIIC</td>
<td>Used to read/write data from/to the EEPROM using master transmit operation and master receive operation with the I²C bus interface method taking the RIIC channel 2 of the RZ/A1H as the master device and the EEPROM as the slave device.</td>
</tr>
<tr>
<td>STB</td>
<td>Used to supply a clock to the RIIC channel 2.</td>
</tr>
<tr>
<td>PORT</td>
<td>Used to switch the multiplexed pins of the RIIC channel 2.</td>
</tr>
<tr>
<td>INTC</td>
<td>Used for the RIIC interrupt control.</td>
</tr>
<tr>
<td>SCIF</td>
<td>Used for communication between the SCIF channel 2 and the host PC.</td>
</tr>
</tbody>
</table>

Figure 1.1 Operation Overview
2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

<table>
<thead>
<tr>
<th>Table 2.1 Operation Confirmation Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Item</strong></td>
</tr>
<tr>
<td>MCU Used</td>
</tr>
</tbody>
</table>
| Operating Frequency* | CPU clock (I\(\Phi\)): 400 MHz  
Image processing clock (G\(\Phi\)): 266.67 MHz  
Internal bus clock (G\(\Phi\)): 133.33 MHz  
Peripheral clock 1 (P1\(\Phi\)): 66.67 MHz  
Peripheral clock 0 (P0\(\Phi\)): 33.33 MHz |
| Operating voltage | Power supply voltage (I/O): 3.3 V  
Power supply voltage (internal): 1.18 V |
| Integrated development environment | ARM® integrated development environment  
ARM Development Studio 5 (DS-5™) Version 5.16 |
| C compiler | ARM C/C++ Compiler/Linker/Assembler Ver.5.03 [Build 102]  
Compiler Option (excluding additional directory path):  
-O3 -Ospace --cpu=Cortex-A9 --littleend --arm --apcs=/interwork  
--no_unaligned_access --fpu=vfpv3_fp16 -g --asm |
| Operating Mode | Boot mode 0  
(CS0 space 16-bit boot) |
| Terminal software communication settings | Communication speed: 115200bps  
Data length: 8 bits  
Parity: None  
Stop bit length: 1 bit  
Flow control: None |
| Board Used | GENMAI board  
RTK772100BC00000BR (R7S72100 CPU board) |
| Device Used (Functions used on the board) | NOR flash memory (Connected to CS0 and CS1 spaces)  
Manufacturer: Spansion Inc.  
Product No.: S29GL512S10TFI01  
EEPROM  
Manufacturer: Renesas Electronics Corporation  
Product No.: R1EX24128ASAS0A  
Serial interface (9-pin D-Sub connector)  
LED1 |

Note: * The operating frequency used in clock mode 0 (Clock input of 13.33MHz from EXTAL pin)

3. Reference Application Notes

For additional information associated with this document, refer to the following application notes.

- RZ/A1H Group I/O definition header file <iodefine.h> (R01AN1860EJ)
- RZ/A1H Group Example of Initialization (R01AN1864EJ)
4. Hardware

4.1 Hardware Configuration

Figure 4.1 shows a Connection Example in which the RZ/A1H is used as the master device and the EEPROM (R1EX24128ASAS0A) as the slave device. In the connection example, the pins are processed such that the device address code of the EEPROM is "B'000".

![Connection Example Diagram](image)

**Figure 4.1 Connection Example**

4.2 EEPROM Specifications

Table 4.1 lists the EEPROM Specifications.

<table>
<thead>
<tr>
<th>Item</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product No.</td>
<td>R1EX24128ASAS0A</td>
</tr>
<tr>
<td>Capacity</td>
<td>16KB (16K words x 8 bits)</td>
</tr>
<tr>
<td>Slave address</td>
<td>H'A0</td>
</tr>
<tr>
<td></td>
<td>Device code (bit7 to bit4): B'1010</td>
</tr>
<tr>
<td></td>
<td>Device address code (bit3 to bit1): B'000</td>
</tr>
<tr>
<td></td>
<td>(Set by pins A2 to A0. Connected to GND on board.)</td>
</tr>
<tr>
<td></td>
<td>R/W# bit (bit0): Write when &quot;0&quot;, read when &quot;1&quot;</td>
</tr>
<tr>
<td>Write operation</td>
<td>Page write function support</td>
</tr>
<tr>
<td></td>
<td>(Support automatic incrementing for addresses (A5 to A0) within pages of up</td>
</tr>
<tr>
<td></td>
<td>to 64 bytes)</td>
</tr>
<tr>
<td>Read operation</td>
<td>Sequential read function support</td>
</tr>
<tr>
<td></td>
<td>(Enable continuous data output with incrementing the address when ACK is input</td>
</tr>
<tr>
<td></td>
<td>after output of each data block)</td>
</tr>
</tbody>
</table>
4.3 Pins Used

Table 4.2 lists the Pins Used and Their Functions.

Table 4.2 Pins Used and Their Functions

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Input/output</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A25 to A1</td>
<td>Output</td>
<td>Address signal output to the NOR flash memory</td>
</tr>
<tr>
<td>D15 to D0</td>
<td>Input/output</td>
<td>Data signal input/output of the NOR flash memory</td>
</tr>
<tr>
<td>CS0#</td>
<td>Output</td>
<td>Device select signal output to the NOR flash memory connected to CS0 space</td>
</tr>
<tr>
<td>RD#</td>
<td>Output</td>
<td>Read control signal output to the NOR flash memory</td>
</tr>
<tr>
<td>WE0#</td>
<td>Output</td>
<td>Write enable control signal output to the NOR flash memory</td>
</tr>
<tr>
<td>MD_BOOT1</td>
<td>Input</td>
<td>Selection of boot mode</td>
</tr>
<tr>
<td>MD_BOOT0</td>
<td>Input</td>
<td>MD_BOOT1: &quot;L&quot;, MD_BOOT0: &quot;L&quot; (Set to boot mode 0)</td>
</tr>
<tr>
<td>P4_10</td>
<td>Output</td>
<td>LED ON/OFF</td>
</tr>
<tr>
<td>RIIC2SCL</td>
<td>Input/output</td>
<td>Serial clock signal input/output to the EEPROM</td>
</tr>
<tr>
<td>RIIC2SDA</td>
<td>Input/output</td>
<td>Serial data signal input/output to the EEPROM</td>
</tr>
<tr>
<td>RxD2</td>
<td>Input</td>
<td>Serial receive data signal</td>
</tr>
<tr>
<td>TxD2</td>
<td>Output</td>
<td>Serial transmit data signal</td>
</tr>
</tbody>
</table>

Note: The symbol # indicates negative logic (or active low).
5. Software

5.1 Operation Overview

The software provides initial settings to the peripheral function modules (RIIC, PORT, STB, and INTC) to enable sending and receiving of data on RIIC channel 2 at a bit rate of 100 kbps, and the master transmit operation on RIIC channel 2 is used to write 64 bytes of data to the EEPROM. The 64 bytes of data that has been written is then read from the EEPROM using master transmit and master receive operation.

The page write function is used when writing to the EEPROM. The sequential read function is used when reading data from the EEPROM. Acknowledge polling is performed after writing to the EEPROM and before reading from it to ensure that the data has been written to the EEPROM.

When transmitting the slave address, "B'1010" is used as the device code and "B'000" as the device address code of the EEPROM.

Note: The transfer speed of the RIIC is calculated from the settings of the RIIC-related registers. For details of calculating the transfer speed, see RZ/A1H Group User's Manual.

5.1.1 Writing to EEPROM

The master transmit mode is used to perform 64-byte page writes to the EEPROM.

A start condition is issued on RIIC channel 2, and next the slave address (device code and device address code) is transmitted. Following the slave address, "0" is transmitted as the R/W# bit, indicating a write operation. Then, a 2-byte memory address (A13 to A0) and 64 bytes of data to be written to the EEPROM are transmitted one after another. The 2-byte memory address transmitted at this time specifies the address in the EEPROM where the data is to be written.

After transmission of the 64 bytes of data finishes, the stop condition is issued and the bus is released. In the sample code, "H'0000" is used as the memory address to which the data is written.

Figure 5.1 shows the EEPROM Write Operation Example (Page Write).
5.1.2 Reading from EEPROM

The master transmit mode and the master receive mode are used to perform a sequential read of 64 bytes of data from the EEPROM.

Master transmit is used to issue a start condition on RIIC channel 2, and next the slave address (device code and device address code) is transmitted. Following the slave address, "0" is transmitted as the R/W# bit, indicating a write operation. Then, a 2-byte memory address is transmitted. The 2-byte memory address transmitted at this time specifies the address in the EEPROM from which data is to be read.

Next, the restart condition is issued, and the slave address is transmitted once again. Following the slave address, "1" is transmitted as the R/W# bit, indicating a read operation, and a transition to master receive mode occurs. After transmission of the R/W# bit, the clock is output on the I2C bus, and read data from the EEPROM is received in succession. The RIIC transmits an ACK for each byte of received data as the read data is being received, with the exception of the final byte of data, after which a NACK is transmitted. After transmission of the NACK, the stop condition is issued and the bus is released. In the sample code, "H'0000" is used as the memory address from which data is read.

Figure 5.2 shows the EEPROM Read Operation Example (Sequential Read).

![Figure 5.2 EEPROM Read Operation Example (Sequential Read)](image-url)
5.1.3 Acknowledge Polling

After writing to the EEPROM and before reading from it, the sample code uses acknowledge polling to determine whether the data has been written inside the EEPROM. In acknowledge polling, the start condition is issued, and then the slave address (device code and device address code) and the R/W# bit with a value of "0" are transmitted. If at this point the writing data is still in progress inside the EEPROM, the EEPROM transmits a NACK. If the writing of data has finished, the EEPROM transmits an ACK.

The sample code continues the polling processing until an ACK is transmitted by the EEPROM.

Figure 5.3 shows the EEPROM Acknowledge Polling Operation Example.
5.1.4 Operation Sequence

Figure 5.4 shows the Operation Sequence of Sample Code from writing data to the EEPROM to reading data from it.

**Write to EEPROM operation**
- Issue start condition
- Transmit slave address
- Transmit memory address to write to
- Transmit write data
- Issue stop condition

Sets the PORT multiplexed pin functions used in RIIC transmission/reception.
Sets the STB to enable the RIIC channel 2.
Sets the bit rate to 98 kbps on RIIC channel 2.
Sets the INTC to enable interrupts on RIIC channel 2.

The RIIC transitions to master transmit mode automatically.

Transmit data = "H'A0":
Device code + device address code + "R/W# bit = 0"

1st byte of transmit data = "H'00": Transmit address (upper 8 bits)
2nd byte of transmit data = "H'00": Transmit address (lower 8 bits)

Writes 64 bytes of data to the EEPROM.

The RIIC transitions to slave receive mode automatically.

**Read from EEPROM operation**
- Issue start condition
- Transmit slave address
- Transmit memory address to read from
- Issue restart condition
- Transmit slave address
- Perform dummy read
- Receive read data
- Issue stop condition

The RIIC transitions to master transmit mode automatically.

Transmit data = "H'A0":
Device code + device address code + "R/W# bit = 0"

1st byte of transmit data = "H'00": Receive address (upper 8 bits)
2nd byte of transmit data = "H'00": Receive address (lower 8 bits)

Transmit data = "H'A1":
Device code + device address code + "R/W# bit = 1"

The RIIC transitions to master receive mode automatically.

Outputs clock to I2C bus and starts RIIC receive operation.

Reads 64 bytes of data from the EEPROM.

The RIIC transitions to slave receive mode automatically.

**Acknowledge polling operation**
- Issue start condition
- Transmit slave address
- Issue restart condition
- Transmit slave address
- Issue stop condition

The RIIC transitions to master transmit mode automatically.

Transmit data = "H'A0":
Device code + device address code + "R/W# bit = 0"

Repeats until NACK receive flag is cleared to "0" (ACK transmitted by EEPROM).

The RIIC transitions to slave receive mode automatically.

**End**
5.2 Peripheral Function Settings and Memory Allocation Used in Sample Code

5.2.1 Peripheral Function Settings
Table 5.1 lists the Peripheral Function Settings.

Table 5.1 Peripheral Function Settings

<table>
<thead>
<tr>
<th>Module</th>
<th>Setting</th>
</tr>
</thead>
</table>
| RIIC | Channel 2  
Master/slave: Single master  
Address format: 7-bit address format  
Transfer speed: 100kbps  
(IIC\text{f} set to P0/\text{f}/8 when P0\text{f} = 33.33 MHz, SCL clock high width to 16 clock cycles, SCL clock low width to 14 clock cycles, resulting in a transfer speed of 100 kbps*)  
Extra SCL clock cycle output function: Disabled  
Internal reference clock (IIC\text{f}): P0/\text{f}/8  
SCL synchronous circuit enable: Enabled  
Digital noise filter: Enabled  
Noise filter stage count: 4 stages (eliminates noise with frequency of 4 × IIC\text{f} or below)  
Transfer suspension during NACK receive: Enabled  
NACK transmit arbitration-lost detection: Disabled  
Master arbitration-lost detection: Enabled  
Timeout detection: Disabled  
Receive data-full interrupt (INTIICRI2): Enabled  
Transmit-data-empty interrupt (INTIICTI2): Enabled  
Transmit-end interrupt (INTIICTEI2): Controller disable or enable |
| STB | Clock is supplied to RIIC channel 2. |
| PORT | PORT1 multiplexed pin function settings  
• P1_4: RIIC2SCL  
• P1_5: RIIC2SDA |
| INTC | Registration and execution of handlers for RIIC channel 2 receive data-full interrupt (interrupt ID: 206), RIIC channel 2 transmit data-empty interrupt (interrupt ID: 207), and RIIC channel 2 transmit-end interrupt (interrupt ID: 205) |
| SCIF | Sets the channel 2 in asynchronous communication mode.  
• Data length: 8 bits  
• Stop bits: 1 bit  
• Parity: None  
Sets the clock source without frequency dividing and the bit rate value at 17. Sets the bit rate to be 115200bps when P1\text{f} is 66.67MHz.  
Difference is 0.46%. |

Note: * The RIIC transfer speed is 100 kbps (duty ratio: 0.524) when a four-stage digital noise filter is used. See RZ/A1H Group User's Manual for the formula used to calculate the transfer speed.
5.2.2 Section Assignment of Sample Code

Table 5.2 and Table 5.3 list the Sections Used in the sample code, and Figure 5.5 shows the Section Assignment of the sample code in initial state (load view) and the one after the scatter loading function is used (execution view).

For details about sections and the scatter loading function, refer to "Image structure and generation" in ARM Compiler toolchain: Using the Linker.

Table 5.2 Sections Used (1/2)

<table>
<thead>
<tr>
<th>Area Name</th>
<th>Description</th>
<th>Type</th>
<th>Load Area</th>
<th>Execution Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>VECTOR_TABLE</td>
<td>Exception processing vector table</td>
<td>Code</td>
<td>FLASH</td>
<td>FLASH</td>
</tr>
<tr>
<td>RESET_HANDLER</td>
<td>Program code area of reset handler processing</td>
<td>Code</td>
<td>FLASH</td>
<td>FLASH</td>
</tr>
<tr>
<td></td>
<td>This area consists of the following sections.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• INITCA9CACHE (L1 cache setting)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• INIT_TTB (MMU setting)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• RESET_HANDLER (Reset handler)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CODE_BASIC_SETUP</td>
<td>Program code area to optimize operating frequency and flash memory</td>
<td>Code</td>
<td>FLASH</td>
<td>FLASH</td>
</tr>
<tr>
<td>InRoot</td>
<td>This area consists of the sections located in the root area such as C standard library.</td>
<td>Code and RO Data</td>
<td>FLASH</td>
<td>FLASH</td>
</tr>
<tr>
<td>CODE_FPU_INIT</td>
<td>Program code area for NEON and VFP initializations</td>
<td>Code</td>
<td>FLASH</td>
<td>FLASH</td>
</tr>
<tr>
<td>CODE_RESET</td>
<td>Program code area for hardware initialization</td>
<td>Code</td>
<td>FLASH</td>
<td>FLASH</td>
</tr>
<tr>
<td>CODE_IO_REGRW</td>
<td>Program code area for read/write functions of I/O register</td>
<td>Code</td>
<td>FLASH</td>
<td>LRAM</td>
</tr>
<tr>
<td>CODE</td>
<td>Program code area for defaults</td>
<td>Code</td>
<td>FLASH</td>
<td>FLASH</td>
</tr>
<tr>
<td></td>
<td>All the Code type sections which do not define section names with C source are assigned in this area.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONST</td>
<td>Constant data area for defaults</td>
<td>RO Data</td>
<td>FLASH</td>
<td>FLASH</td>
</tr>
<tr>
<td></td>
<td>All the RO Data type sections which do not define section names with C source are assigned in this area.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 5.3  Sections Used (2/2)

<table>
<thead>
<tr>
<th>Area Name</th>
<th>Description</th>
<th>Type</th>
<th>Load Area</th>
<th>Execution Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>VECTOR_MIRROR_TABLE</td>
<td>Exception processing vector table (Section to transfer data to large-capacity on-chip RAM)</td>
<td>Code</td>
<td>FLASH</td>
<td>LRAM</td>
</tr>
<tr>
<td>CODE_HANDLER_JMPTBL</td>
<td>Program code area for user-defined functions of IRQ interrupt handler</td>
<td>Code</td>
<td>FLASH</td>
<td>LRAM</td>
</tr>
<tr>
<td>CODE_HANDLER</td>
<td>Program code area of IRQ interrupt handler</td>
<td>Code</td>
<td>FLASH</td>
<td>LRAM</td>
</tr>
<tr>
<td>CODE_CACHE_OPERATION</td>
<td>Program code area for setting the L1 and L2 caches (see Note3)</td>
<td>Code</td>
<td>FLASH</td>
<td>LRAM</td>
</tr>
<tr>
<td>DATA_HANDLER_JMPTBL</td>
<td>Registration table data area for user-defined functions of IRQ interrupt handler</td>
<td>RW Data</td>
<td>FLASH</td>
<td>LRAM</td>
</tr>
<tr>
<td>ARM_LIB_STACK</td>
<td>Application stack area</td>
<td>ZI Data</td>
<td>-</td>
<td>LRAM</td>
</tr>
<tr>
<td>IRQ_STACK</td>
<td>IRQ mode stack area</td>
<td>ZI Data</td>
<td>-</td>
<td>LRAM</td>
</tr>
<tr>
<td>FIQ_STACK</td>
<td>FIQ mode stack area</td>
<td>ZI Data</td>
<td>-</td>
<td>LRAM</td>
</tr>
<tr>
<td>SVC_STACK</td>
<td>Supervisor (SVC) mode stack area</td>
<td>ZI Data</td>
<td>-</td>
<td>LRAM</td>
</tr>
<tr>
<td>ABT_STACK</td>
<td>Abort (ABT) mode stack area</td>
<td>ZI Data</td>
<td>-</td>
<td>LRAM</td>
</tr>
<tr>
<td>TTB</td>
<td>MMU translation table area</td>
<td>ZI Data</td>
<td>-</td>
<td>LRAM</td>
</tr>
<tr>
<td>ARM_LIB_HEAP</td>
<td>Application heap area</td>
<td>ZI Data</td>
<td>-</td>
<td>LRAM</td>
</tr>
<tr>
<td>DATA</td>
<td>Data area with initial value for defaults</td>
<td>RW Data</td>
<td>FLASH</td>
<td>LRAM</td>
</tr>
<tr>
<td></td>
<td>All the RW Data type sections which do not define section names with C source are assigned in this area.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSS</td>
<td>Data area without initial value for defaults</td>
<td>ZI Data</td>
<td>-</td>
<td>LRAM</td>
</tr>
<tr>
<td></td>
<td>All the ZI Data type sections which do not define section names with C source area assigned in this area.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSS_RIIC_SAMPLE</td>
<td>Data area without initial value used by the RIIC sample code</td>
<td>ZI Data</td>
<td>-</td>
<td>LRAM</td>
</tr>
<tr>
<td></td>
<td>Defined to be allocated to a disabled area of the L1 cache.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes: 1. In the listing of load areas and run areas in the table, "FLASH" indicates the NOR flash memory area and "LRAM" the large-capacity on-chip RAM area.
2. Basically, the section names are the same as the area names, but the RESET_HANDLER, InRoot, CODE_FPU_INIT, CODE_RESET, CODE, CONST, CODE_HANDLER, DATA, and BSS areas each comprise multiple sections. For details on areas and sections, refer to the ARM compiler toolchain manual.
3. This section should be placed in the cache-disabled area.
RZ/A1H group address space (Virtual address)

H'FFFF FFFF

H'6000 0000

H'6000 0000

H'600F FFFF

H'603F FFFF

H'6030 0000

H'6030 0000

H'602F FFFF

Cache-disabled space in large-capacity on-chip RAM (10MB)

CS0 space (64 MB)

H'4000 0000

H'2000 0000

H'1C00 0000

H'1800 0000

H'1000 0000

H'0C00 0000

H'0800 0000

H'0400 0000

H'0000 0000

H'0C00 0000

H'0800 0000

H'0400 0000

H'0000 0000

Large-capacity on-chip RAM (10MB)

H'20A0 0000

H'1000 0000

H'0C00 0000

H'0800 0000

H'0400 0000

H'0000 0000

H'0C00 0000

H'0800 0000

H'0400 0000

H'0000 0000

H'209F FFFF

H'6000 0000

H'6050 0000

H'609F FFFF

Section arrangement (load view)

Section arrangement (execution view)

Memory allocation after scatter loading

Transfer cache operation process to cache-disabled area in large-capacity on-chip RAM.

Clear to 0

Initialize data that have initial values

Obtain areas for stacks and the like

Initialize data that have initial values

Transfer program code that requires high-speed processing to the on-chip RAM

Transfer the exception handling vector to the on-chip RAM

Figure 5.5 Section Assignment
5.3 Interrupts
Table 5.4 lists the Interrupts Used in Sample Code.

<table>
<thead>
<tr>
<th>Interrupt Source (Interrupt ID)</th>
<th>Priority</th>
<th>Processing Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTIICRI2 (206)</td>
<td>9</td>
<td>Notifies that 1 byte of data has been received from the EEPROM.</td>
</tr>
<tr>
<td>INTIICTI2 (207)</td>
<td>9</td>
<td>Notifies that the RIIC2DRT register is in empty state.</td>
</tr>
<tr>
<td>INTIICTEI2 (205)</td>
<td>9</td>
<td>Notifies that 1 byte of data has been transmitted to the EEPROM.</td>
</tr>
</tbody>
</table>

5.4 Fixed-Width Integers
Table 5.5 lists the Fixed-Width Integers Used in Sample Code.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char_t</td>
<td>8-bit character</td>
</tr>
<tr>
<td>bool_t</td>
<td>Boolean type, value: true (1) or false (0)</td>
</tr>
<tr>
<td>int_t</td>
<td>High-speed integer, signed 32-bit integer in this sample code</td>
</tr>
<tr>
<td>int8_t</td>
<td>8-bit integer, signed (Defined by standard library)</td>
</tr>
<tr>
<td>int16_t</td>
<td>16-bit integer, signed (Defined by standard library)</td>
</tr>
<tr>
<td>int32_t</td>
<td>32-bit integer, signed (Defined by standard library)</td>
</tr>
<tr>
<td>int64_t</td>
<td>64-bit integer, signed (Defined by standard library)</td>
</tr>
<tr>
<td>uint8_t</td>
<td>8-bit integer, unsigned (Defined by standard library)</td>
</tr>
<tr>
<td>uint16_t</td>
<td>16-bit integer, unsigned (Defined by standard library)</td>
</tr>
<tr>
<td>uint32_t</td>
<td>32-bit integer, unsigned (Defined by standard library)</td>
</tr>
<tr>
<td>uint64_t</td>
<td>64-bit integer, unsigned (Defined by standard library)</td>
</tr>
<tr>
<td>float32_t</td>
<td>32-bit floating point (Defined by standard library when specifying &quot;<strong>ARM_NEON</strong>&quot;)</td>
</tr>
<tr>
<td>float64_t</td>
<td>64-bit floating point (Defined by standard library when specifying &quot;<strong>ARM_NEON</strong>&quot;)</td>
</tr>
<tr>
<td>float128_t</td>
<td>128-bit floating point</td>
</tr>
</tbody>
</table>
### 5.5 Constants
Table 5.6 lists the Constants Used in Sample Code.

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAMPLE_RIIC_DEVICE_CODE</td>
<td>(0xA0)</td>
<td>EEPROM device code</td>
</tr>
<tr>
<td>SAMPLE_RIIC_DEVICE_ADDR</td>
<td>(0x00)</td>
<td>EEPROM device address code</td>
</tr>
<tr>
<td>SAMPLE_RIIC_DATA_SIZE</td>
<td>(64)</td>
<td>Data size to be read/written</td>
</tr>
<tr>
<td>SAMPLE_RIIC_RWCODE_W</td>
<td>(0)</td>
<td>Set R/W# bit to &quot;0&quot;.</td>
</tr>
<tr>
<td>SAMPLE_RIIC_RWCODE_R</td>
<td>(1)</td>
<td>Set R/W# bit to &quot;1&quot;.</td>
</tr>
<tr>
<td>RIIC_CKS_DIVISION_1</td>
<td>(0)</td>
<td>Set internal reference clock (IICf) to P0f/1.</td>
</tr>
<tr>
<td>RIIC_CKS_DIVISION_2</td>
<td>(1)</td>
<td>Set internal reference clock (IICf) to P0f/2.</td>
</tr>
<tr>
<td>RIIC_CKS_DIVISION_4</td>
<td>(2)</td>
<td>Set internal reference clock (IICf) to P0f/4.</td>
</tr>
<tr>
<td>RIIC_CKS_DIVISION_8</td>
<td>(3)</td>
<td>Set internal reference clock (IICf) to P0f/8.</td>
</tr>
<tr>
<td>RIIC_CKS_DIVISION_16</td>
<td>(4)</td>
<td>Set internal reference clock (IICf) to P0f/16.</td>
</tr>
<tr>
<td>RIIC_CKS_DIVISION_32</td>
<td>(5)</td>
<td>Set internal reference clock (IICf) to P0f/32.</td>
</tr>
<tr>
<td>RIIC_CKS_DIVISION_64</td>
<td>(6)</td>
<td>Set internal reference clock (IICf) to P0f/64.</td>
</tr>
<tr>
<td>RIIC_CKS_DIVISION_128</td>
<td>(7)</td>
<td>Set internal reference clock (IICf) to P0f/128.</td>
</tr>
<tr>
<td>RIIC_TX_MODE_START</td>
<td>(0)</td>
<td>Start condition issuance request in transmit mode</td>
</tr>
<tr>
<td>RIIC_TX_MODE_RESTART</td>
<td>(1)</td>
<td>Restart condition issuance request in transmit mode</td>
</tr>
<tr>
<td>RIIC_TX_MODE_STOP</td>
<td>(2)</td>
<td>Stop condition issuance request in transmit mode</td>
</tr>
<tr>
<td>RIIC_TEND_WAIT_TRANSMIT</td>
<td>(0)</td>
<td>Select wait operation after slave address transmission Continuation of transmit mode</td>
</tr>
<tr>
<td>RIIC_TEND_WAIT_RECEIVE</td>
<td>(1)</td>
<td>Select wait operation after slave address transmission Transition to receive mode.</td>
</tr>
<tr>
<td>RIIC_BUS_MASTERSHIP_WAIT_FREE</td>
<td>(0)</td>
<td>Bus free wait mode</td>
</tr>
<tr>
<td>RIIC_BUS_MASTERSHIP_WAIT_BUSY</td>
<td>(1)</td>
<td>Bus busy wait mode</td>
</tr>
<tr>
<td>DEVDRV_ERROR_RIIC_NACK</td>
<td>(-2)</td>
<td>Notifies the NACK reception</td>
</tr>
</tbody>
</table>
5.6 Variables

Table 5.6 lists the Constants Used in Sample Code.

Table 5.7 static Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Contents</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>static volatile</td>
<td>riic_receive_full_flg</td>
<td>RIIC channel 2 receive data full interrupt</td>
<td>Userdef_RIIC2_InitReceiveFull</td>
</tr>
<tr>
<td>uint8_t</td>
<td></td>
<td>notification flag</td>
<td>Userdef_RIIC2_SetReceiveFull</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Userdef_RIIC2_WaitReceiveFull</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Userdef_RIIC2_WaitTransmitEnd</td>
</tr>
<tr>
<td>static volatile</td>
<td>riic_transmit_empty_flg</td>
<td>RIIC channel 2 transmit data empty interrupt</td>
<td>Userdef_RIIC2_InitTransmitEmpty</td>
</tr>
<tr>
<td>uint8_t</td>
<td></td>
<td>notification flag</td>
<td>Userdef_RIIC2_SetTransmitEmpty</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Userdef_RIIC2_WaitTransmitEnd</td>
</tr>
<tr>
<td>static volatile</td>
<td>riic_transmit_end_flg</td>
<td>RIIC channel 2 transmit-end interrupt</td>
<td>Userdef_RIIC2_InitTransmitEnd</td>
</tr>
<tr>
<td>uint8_t</td>
<td></td>
<td>notification flag</td>
<td>Userdef_RIIC2_SetTransmitEnd</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Userdef_RIIC2_WaitTransmitEnd</td>
</tr>
<tr>
<td>static uint8_t</td>
<td>riic_trans_data[SAMPLE_RIIC_DATA_SIZE]</td>
<td>Transmit data buffer*</td>
<td>Sample_RIIC_EepAccess</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>static uint8_t</td>
<td>riic_receive_data[SAMPLE_RIIC_DATA_SIZE]</td>
<td>Receive data buffer*</td>
<td>Sample_RIIC_EepAccess</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: * Both riic_trans_data and riic_receive_data are defined as the BSS_RIIC_SAMPLE section and allocated to a mirror space (addresses H’6000 0000 to H’609F FFFF) in the large-capacity on-chip RAM. This space disables the L1 cache by setting the MMU.

Figure 5.6 shows the Memory Allocation of Transmit Data Buffer and Receive Data Buffer.
5.7 Functions

The sample code consists of the interface functions (API functions) to perform RIIC transmission/reception and the like, the user-defined functions (functions called by the API functions) which should be prepared by the user corresponding to the system applied, and the sample functions required to operate the sample code such as implementing read/write access to the EEPROM.

Table 5.8 and Table 5.9 list the Sample Functions and the API Functions respectively. Table 5.10 to Table 5.12 list the User-defined Functions.

Table 5.8 Sample Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>Main processing</td>
</tr>
<tr>
<td>Sample_Main</td>
<td>Sample code main processing</td>
</tr>
<tr>
<td>Sample_RIIC_Main</td>
<td>RIIC sample code main processing</td>
</tr>
<tr>
<td>Sample_RIIC_EepAccess</td>
<td>Sample processing for reading/writing data from/to EEPROM</td>
</tr>
<tr>
<td>RIIC_PORT_Init</td>
<td>Initial setting of PORT used in RIIC transmission/reception</td>
</tr>
<tr>
<td>Sample_RIIC_EepWrite</td>
<td>Sample processing for writing to EEPROM</td>
</tr>
<tr>
<td>Sample_RIIC_EepRead</td>
<td>Sample processing for reading from EEPROM</td>
</tr>
<tr>
<td>Sample_RIIC_EepAckPolling</td>
<td>Sample processing to wait for writing completion inside EEPROM</td>
</tr>
<tr>
<td>Sample_RIIC_Ti2_Interrupt</td>
<td>RIIC channel 2 transmit data empty interrupt processing</td>
</tr>
<tr>
<td>Sample_Ri2_Interrupt</td>
<td>RIIC channel 2 receive data full interrupt processing</td>
</tr>
</tbody>
</table>

Table 5.9 API Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_RIIC_Init</td>
<td>RIIC initial settings</td>
</tr>
<tr>
<td>R_RIIC_SendCond</td>
<td>RIIC condition issuance request</td>
</tr>
<tr>
<td>R_RIIC_WriteSlaveAddr</td>
<td>RIIC slave address transmission</td>
</tr>
<tr>
<td>R_RIIC_Write</td>
<td>RIIC transmission</td>
</tr>
<tr>
<td>R_RIIC_Read</td>
<td>RIIC reception</td>
</tr>
<tr>
<td>R_RIIC_ReadDummy</td>
<td>RIIC dummy read</td>
</tr>
<tr>
<td>R_RIIC_DetectStop</td>
<td>Checking RIIC stop condition detection</td>
</tr>
<tr>
<td>R_RIIC_ClearNack</td>
<td>Clearing RIIC NACK receive flag</td>
</tr>
<tr>
<td>R_RIIC_TiInterrupt</td>
<td>RIIC transmit data empty interrupt notification</td>
</tr>
<tr>
<td>R_RIIC_TeiInterrupt</td>
<td>RIIC transmit-end interrupt notification</td>
</tr>
<tr>
<td>R_RIIC_Ri2Interrupt</td>
<td>RIIC receive data full interrupt notification</td>
</tr>
</tbody>
</table>
### Table 5.10 User-Defined Functions (1/3)

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Userdef_RIIC0_Init</td>
<td>RIIC channel 0 initial settings</td>
</tr>
<tr>
<td></td>
<td>(In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC1_Init</td>
<td>RIIC channel 1 initial settings</td>
</tr>
<tr>
<td></td>
<td>(In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC2_Init</td>
<td>RIIC channel 2 initial settings</td>
</tr>
<tr>
<td>Userdef_RIIC3_Init</td>
<td>RIIC channel 3 initial settings</td>
</tr>
<tr>
<td></td>
<td>(In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC0_InitReceiveFull</td>
<td>Initialization of RIIC channel 0 receive data full notification</td>
</tr>
<tr>
<td></td>
<td>(In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC1_InitReceiveFull</td>
<td>Initialization of RIIC channel 1 receive data full notification</td>
</tr>
<tr>
<td></td>
<td>(In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC2_InitReceiveFull</td>
<td>Initialization of RIIC channel 2 receive data full notification</td>
</tr>
<tr>
<td></td>
<td>(In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC3_InitReceiveFull</td>
<td>Initialization of RIIC channel 3 receive data full notification</td>
</tr>
<tr>
<td></td>
<td>(In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC0_InitTransmitEmpty</td>
<td>Initialization of RIIC channel 0 transmit data empty notification</td>
</tr>
<tr>
<td></td>
<td>(In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC1_InitTransmitEmpty</td>
<td>Initialization of RIIC channel 1 transmit data empty notification</td>
</tr>
<tr>
<td></td>
<td>(In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC2_InitTransmitEmpty</td>
<td>Initialization of RIIC channel 2 transmit data empty notification</td>
</tr>
<tr>
<td></td>
<td>(In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC3_InitTransmitEmpty</td>
<td>Initialization of RIIC channel 3 transmit data empty notification</td>
</tr>
<tr>
<td></td>
<td>(In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC0_SetReceiveFull</td>
<td>Setting to satisfy the conditions for RIIC channel 0 receive data full</td>
</tr>
<tr>
<td></td>
<td>notification information</td>
</tr>
<tr>
<td></td>
<td>(In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC1_SetReceiveFull</td>
<td>Setting to satisfy the conditions for RIIC channel 1 receive data full</td>
</tr>
<tr>
<td></td>
<td>notification information</td>
</tr>
<tr>
<td></td>
<td>(In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC2_SetReceiveFull</td>
<td>Setting to satisfy the conditions for RIIC channel 2 receive data full</td>
</tr>
<tr>
<td></td>
<td>notification information</td>
</tr>
<tr>
<td></td>
<td>(In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC3_SetReceiveFull</td>
<td>Setting to satisfy the conditions for RIIC channel 3 receive data full</td>
</tr>
<tr>
<td></td>
<td>notification information</td>
</tr>
<tr>
<td></td>
<td>(In the sample code, this function returns without any processing.)</td>
</tr>
</tbody>
</table>
### Table 5.11 User-Defined Functions (2/3)

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
</tr>
</thead>
</table>
| Userdef_RIIC0_SetTransmitEmpty       | Setting to satisfy the conditions for RIIC channel 0 transmit data empty notification information  
  (In the sample code, this function returns without any processing.) |
| Userdef_RIIC1_SetTransmitEmpty       | Setting to satisfy the conditions for RIIC channel 1 transmit data empty notification information  
  (In the sample code, this function returns without any processing.) |
| Userdef_RIIC2_SetTransmitEmpty       | Setting to satisfy the conditions for RIIC channel 2 transmit data empty notification information |
| Userdef_RIIC3_SetTransmitEmpty       | Setting to satisfy the conditions for RIIC channel 3 transmit data empty notification information  
  (In the sample code, this function returns without any processing.) |
| Userdef_RIIC0_SetTransmitEnd         | Setting to satisfy the conditions for RIIC channel 0 transmit-end notification information  
  (In the sample code, this function returns without any processing.) |
| Userdef_RIIC1_SetTransmitEnd         | Setting to satisfy the conditions for RIIC channel 1 transmit-end notification information  
  (In the sample code, this function returns without any processing.) |
| Userdef_RIIC2_SetTransmitEnd         | Setting to satisfy the conditions for RIIC channel 2 transmit-end notification information |
| Userdef_RIIC3_SetTransmitEnd         | Setting to satisfy the conditions for RIIC channel 3 transmit-end notification information  
  (In the sample code, this function returns without any processing.) |
| Userdef_RIIC0_WaitReceiveFull        | Waiting for satisfaction of the conditions for RIIC channel 0 receive data full notification information  
  (In the sample code, this function returns without any processing.) |
| Userdef_RIIC1_WaitReceiveFull        | Waiting for satisfaction of the conditions for RIIC channel 1 receive data full notification information  
  (In the sample code, this function returns without any processing.) |
| Userdef_RIIC2_WaitReceiveFull        | Waiting for satisfaction of the conditions for RIIC channel 2 receive data full notification information |
| Userdef_RIIC3_WaitReceiveFull        | Waiting for satisfaction of the conditions for RIIC channel 3 receive data full notification information  
  (In the sample code, this function returns without any processing.) |
| Userdef_RIIC0_WaitTransmitEmpty      | Waiting for satisfaction of the conditions for RIIC channel 0 transmit data empty notification information  
  (In the sample code, this function returns without any processing.) |
| Userdef_RIIC1_WaitTransmitEmpty      | Waiting for satisfaction of the conditions for RIIC channel 1 transmit data empty notification information  
  (In the sample code, this function returns without any processing.) |
| Userdef_RIIC2_WaitTransmitEmpty      | Waiting for satisfaction of the conditions for RIIC channel 2 transmit data empty notification information |
| Userdef_RIIC3_WaitTransmitEmpty      | Waiting for satisfaction of the conditions for RIIC channel 3 transmit data empty notification information  
  (In the sample code, this function returns without any processing.) |
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Userdef_RIIC0_WaitTransmitEnd</td>
<td>Waiting for satisfaction of the conditions for RIIC channel 0 transmit-end notification information (In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC1_WaitTransmitEnd</td>
<td>Waiting for satisfaction of the conditions for RIIC channel 1 transmit-end notification information (In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC2_WaitTransmitEnd</td>
<td>Waiting for satisfaction of the conditions for RIIC channel 2 transmit-end notification information</td>
</tr>
<tr>
<td>Userdef_RIIC3_WaitTransmitEnd</td>
<td>Waiting for satisfaction of the conditions for RIIC channel 3 transmit-end notification information (In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC0_WaitBusMastership</td>
<td>Waiting for RIIC channel 0 bus free and bus busy (In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC1_WaitBusMastership</td>
<td>Waiting for RIIC channel 1 bus free and bus busy (In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC2_WaitBusMastership</td>
<td>Waiting for RIIC channel 2 bus free and bus busy</td>
</tr>
<tr>
<td>Userdef_RIIC3_WaitBusMastership</td>
<td>Waiting for RIIC channel 3 bus free and bus busy (In the sample code, this function returns without any processing.)</td>
</tr>
<tr>
<td>Userdef_RIIC0_WaitStop</td>
<td>Waiting for RIIC channel 0 stop condition to be detected (In the sample code, this function returns without any processing)</td>
</tr>
<tr>
<td>Userdef_RIIC1_WaitStop</td>
<td>Waiting for RIIC channel 1 stop condition to be detected (In the sample code, this function returns without any processing)</td>
</tr>
<tr>
<td>Userdef_RIIC2_WaitStop</td>
<td>Waiting for RIIC channel 2 stop condition to be detected</td>
</tr>
<tr>
<td>Userdef_RIIC3_WaitStop</td>
<td>Waiting for RIIC channel 3 stop condition to be detected (In the sample code, this function returns without any processing)</td>
</tr>
</tbody>
</table>
5.8 Function Specifications

The following tables list the sample code function specifications.

**main**

<table>
<thead>
<tr>
<th>Outline</th>
<th>Main processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>int_t main(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Displays the sample code information to the terminal running on the host PC connected to the GENMAI board via the serial interface.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>0</td>
</tr>
</tbody>
</table>

**Sample_Main**

<table>
<thead>
<tr>
<th>Outline</th>
<th>Sample code main processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>void Sample_Main(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Waits for character input from the terminal running on the host PC connected to the GENMAI board via the serial interface. Activates the RIIC sample code when &quot;RIIC&quot; + &quot;Enter&quot; key is input.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

**Sample_RIIC_Main**

<table>
<thead>
<tr>
<th>Outline</th>
<th>RIIC sample code main processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>int32_t Sample_RIIC_Main(int32_t argc, char_t **argv)</td>
</tr>
<tr>
<td>Description</td>
<td>Waits for character input from the terminal running on the host PC connected to the GENMAI board via the serial interface. The sample code is executed to enable the read/write accesses to the EEPROM when &quot;1&quot; + &quot;Enter&quot; key is input.</td>
</tr>
<tr>
<td>Arguments</td>
<td>int32_t argc : Number of command arguments input from the terminal char_t **argv : Pointer to the command input from the terminal</td>
</tr>
<tr>
<td>Return Value</td>
<td>COMMAND_EXIT: Termination of RIIC sample code</td>
</tr>
</tbody>
</table>
### Sample_RIIC_EepAccess

**Outline**
Sample processing for reading/writing data from/to EEPROM

**Declaration**
```c
int32_t Sample_RIIC_EepAccess(int32_t argc, char_t **argv)
```

**Description**
In the sample code, the transfer speed is set to 100kbps, and 64 bytes of data allocated to the large-capacity on-chip RAM space is written to the EEPROM. Then waits until the data has been written into the EEPROM using the acknowledge polling processing.

Reads the 64 bytes of data from the EEPROM and writes it into the large-capacity on-chip RAM space.

**Arguments**
- `int32_t argc` : Number of command arguments input from the terminal
  - Not used in this function.
- `char_t **argv` : Pointer to the command input from the terminal
  - Not used in this function.

**Return Value**
- `COMMAND_SUCCESS` : Success of RIIC sample code processing
- `COMMAND_ERROR` : Failure of RIIC sample code processing

### RIIC_PORT_Init

**Outline**
Initial setting of PORT used in RIIC transmission/reception

**Declaration**
```c
static void RIIC_PORT_Init(void)
```

**Description**
Initializes the PORT used in the RIIC transmission and reception.

In the sample code, the PORT is initialized to set the P1_4 pin and P1_5 pin to RIIC2SCL function and RIIC2SDA function respectively.

**Arguments**
- None

**Return Value**
- None
Sample_RIIC_EepWrite

Outline
Sample processing for writing to EEPROM

Declaration
int32_t Sample_RIIC_EepWrite(uint32_t channel, uint8_t d_code, uint8_t d_adr,
                               uint16_t w_adr, uint32_t w_byte, const uint8_t * w_buffer)

Description
Writes the memory specified by the argument *w_buffer for the byte count specified
by the argument w_byte to the EEPROM indicated by the device code specified by
the argument d_code, the device address code specified by the argument d adr, and
the memory address specified by the argument w_adr using the RIIC of the channel
specified by the argument channel.

In the sample code, the data is transmitted by the channel 2 RIIC master transmit
operation with the EEPROM page write function.

Arguments
- uint32_t channel : RIIC channel (0 to 3)
- uint8_t d_code : Device code
- uint8_t d_adr : Device address code (3 bits)
- uint16_t w_adr : Memory address
- uint32_t w_byte : Byte count to be written to EEPROM
- uint8_t * w_buffer : Store buffer of the data written to EEPROM

Return Value
- DEVDRV_SUCCESS : Success of RIIC transmit processing
- DEVDRV_ERROR : Failure of RIIC transmit processing
- DEVDRV_ERROR_RIIC_NACK : Failure of RIIC transmit processing
due to NACK reception

Note
The upper 5 bits of argument d adr have no effect.

Sample_RIIC_EepRead

Outline
Sample processing for reading from EEPROM

Declaration
int32_t Sample_RIIC_EepRead(uint32_t channel, uint8_t d_code, uint8_t d_adr,
                              uint16_t r_adr, uint32_t r_byte, uint8_t * r_buffer)

Description
Reads the data from the EEPROM to the device code specified by the argument
d_code, the device address code specified by the argument d adr, and the memory
address specified by the argument r_adr using the RIIC channel specified by the
argument channel. Stores the data for the byte count specified by the argument
r_byte to the area indicated by the argument r_buffer.

In the sample code, the data is read from the EEPROM by using the sequential read
function and is received by using the RIIC master receive operation on channel 2.

Arguments
- uint32_t channel : RIIC channel (0 to 3)
- uint8_t d_code : Device code
- uint8_t d_adr : Device address code (3 bits)
- uint16_t r_adr : Memory address
- uint32_t r_byte : Byte count to be read from EEPROM
- uint8_t * r_buffer : Store buffer of the data read from EEPROM

Return Value
- DEVDRV_SUCCESS : Success of RIIC receive processing
- DEVDRV_ERROR : Failure of RIIC receive processing
- DEVDRV_ERROR_RIIC_NACK : Failure of RIIC receive processing
due to NACK reception

Note
The upper 5 bits of argument d adr have no effect.
### Sample_RIIC_EepAckPolling

**Outline**  
Sample processing to wait for writing completion inside EEPROM

**Declaration**  
`void Sample_RIIC_EepAckPolling(uint32_t channel, uint8_t d_code, uint8_t d_adr)`

**Description**  
Waits until the data has been written to the EEPROM indicated by the device code specified by the argument d_code and the device address code specified by the argument d_adr with the EEPROM acknowledge polling function using the RIIC channel specified by the argument channel.

In the sample code, the processing is repeated until the ACK is received from the EEPROM using the RIIC master transmit operation on channel 2.

**Arguments**
- `uint32_t channel`: RIIC channel (0 to 3)
- `uint8_t d_code`: Device code
- `uint8_t d_adr`: Device address code (3 bits)

**Return Value**  
None

**Note**  
The upper 5 bits of argument d_adr have no effect.

### Sample_RIIC_Ti2_Interrupt

**Outline**  
RIIC channel 2 transmit data empty interrupt processing

**Declaration**  
`void Sample_RIIC_Ti2_Interrupt(uint32_t int_sense)`

**Description**  
This is an interrupt handler to be executed when the RIIC channel transmit data empty interrupt is accepted, which notifies that the said interrupt has been generated.

In the sample code, this function is registered as the INTC handler by using the user-defined function Userdef_RIIC2_Init.

The API function R_RIIC_TiInterrupt is called by this function.

**Arguments**
- `uint32_t int_sense`: Interrupt detection method (not used)
  - INTC_LEVELSENSITIVE: Level sense
  - INTC_EDGETRIGGER: Edge trigger

**Return Value**  
None

### Sample_RIIC_Tei2_Interrupt

**Outline**  
RIIC channel 2 transmit-end interrupt processing

**Declaration**  
`void Sample_RIIC_Tei2_Interrupt(uint32_t int_sense)`

**Description**  
This is an interrupt handler to be executed when the RIIC channel 2 transmit-end interrupt is accepted, which notifies that the said interrupt has been generated.

In the sample code, this function is registered as the INTC handler by using the user-defined function Userdef_RIIC2_Init.

The API function R_RIIC_TeiInterrupt is called by this function.

**Arguments**
- `uint32_t int_sense`: Interrupt detection method (not used)
  - INTC_LEVELSENSITIVE: Level sense
  - INTC_EDGETRIGGER: Edge trigger

**Return Value**  
None
### Sample_RIIC_Ri2_Interrupt

**Outline**
RIIC channel 2 receive data full interrupt processing

**Declaration**
void Sample_RIIC_Ri2_Interrupt(uint32_t int_sense)

**Description**
This is an interrupt handler to be executed when the RIIC channel 2 receive data full interrupt is accepted, which notifies that the said interrupt has been generated.

In the sample code, this function is registered as the INTC handler by using the user-defined function Userdef_RIIC2_Init.

The API function R_RIIC_RiInterrupt is called by this function.

**Arguments**
- `uint32_t int_sense`: Interrupt detection method (not used)
  - INTC_LEVEL_SENSITIVE: Level sense
  - INTC_EDGE_TRIGGER: Edge trigger

**Return Value**
None

### R_RIIC_Init

**Outline**
RIIC initial settings

**Declaration**
int32_t R_RIIC_Init(uint32_t channel)

**Description**
Initializes the RIIC specified by the argument channel by using the user-defined function Userdef_RIICn_Init (n=0 to 3).

**Arguments**
- `uint32_t channel`: RIIC channel (0 to 3)

**Return Value**
- DEVDRV_SUCCESS: Success of RIIC initialization
- DEVDRV_ERROR: Failure of RIIC initialization

### R_RIIC_SendCond

**Outline**
RIIC condition issuance request

**Declaration**
int32_t R_RIIC_SendCond(uint32_t channel, uint32_t mode)

**Description**
Issues a request to generate the condition specified by the argument mode to the RIIC of the channel specified by the argument channel.

When the start condition issuance and the restart condition issuance are requested, the information for the receive data full notification, the transmit data empty notification, and the transmit end notification are initialized by using the user-defined functions (Userdef_RIICn_InitReceiveFull, Userdef_RIICn_InitTransmitEmpty, and Userdef_RIICn_InitTransmitEnd). (n=0 to 3)

After requesting the stop condition issuance, check the stop condition detection by using the API function R_RIIC_DetectStop.

**Arguments**
- `uint32_t channel`: RIIC channel (0 to 3)
- `uint32_t mode`: Condition modes to be issued
  - RIIC_TX_MODE_START: Start condition
  - RIIC_TX_MODE_RESTART: Restart condition
  - RIIC_TX_MODE_STOP: Stop condition

**Return Value**
- DEVDRV_SUCCESS: Success of RIIC condition issuance request
- DEVDRV_ERROR: Failure of RIIC condition issuance request
Outline
RIIC slave address transmission

Declaration
int32_t R_RIIC_WriteSlaveAddr(uint32_t channel, uint16_t addr, uint32_t mode, uint32_t addrbit)

Description
Transmits the slave address (1 byte) specified by the argument addr using the RIIC master transmit operation of the channel specified by the argument channel.
When transmitting the slave address with the R/# bit "0" (write operation) and transmitting the one with "1" (read operation), RIIC_TEND_WAIT_TRANSMIT and RIIC_TEND_WAIT_RECEIVE should be specified as the argument mode respectively.
First of all, checks the NACKF bit in the RIICnSR2 register (n=0 to 3). If no slave device is recognized by the NACKF bit, DEVDRV_ERROR_RIIC_NACK is returned.
Waits until the conditions for the transmit data empty notification information are satisfied by using the user-defined function Userdef_RIICn_WaitTransmitEmpty.
After initializing the transmit data empty notification information using the user-defined function Userdef_RIICn_InitTransmitEmpty (n=0 to 3), transmits the slave address. Then, waits until the conditions for the transmit end notification information are satisfied and initializes the information using the user-defined function Userdef_RIICn_InitTransmitEnd (n=0 to 3).
Finally, checks the ACK response with the NACKF bit. If the ACK response cannot be confirmed, DEVDRV_ERROR_RIIC_NACK is returned by this function.

Arguments
uint32_t channel : RIIC channel (0 to 3)
uint16_t addr : Slave address to be transmitted
bit7 to bit1 : Slave address
bit0 : R/W# bit
(Write when "0" is set, read when "1" is set.)
uint32_t mode : Mode after transmitting slave address
RIIC_TEND_WAIT_TRANSMIT : Continue transmission mode
RIIC_TEND_WAIT_RECEIVE : Transit to reception mode
uint32_t addrbit : Reserved (not used)

Return Value
DEVDRV_SUCCESS : Success of RIIC transmission
DEVDRV_ERROR : Failure of RIIC transmission
DEVDRV_ERROR_RIIC_NACK : Failure of RIIC transmission due to NACK reception

Note
This function should be called after requesting issuance of the start condition by using the API function R_RIIC_WriteCond.
R_RIIC_Write

Outline
RIIC transmission

Declaration
int32_t R_RIIC_Write(uint32_t channel, const uint8_t * buffer, uint32_t byte)

Description
Transmits the data specified by the argument *buffer for the bytes specified by the argument byte by using the RIIC master transmit operation of the channel specified by the argument channel. In the sample code, this function is applied to transmit the memory address and the data to be written.

Disables the transmit-end interrupt. After checking the NACK receive flag, waits until the conditions for the transmit data empty notification information area satisfied by the user-defined function Userdef_RIICn_WaitTransmitEmpty. After initializing the information by using the user-defined function Userdef_RIICn_InitTransmitEmpty (n=0 to 3), transmits the data. This transmit operation continues until the specified byte counts are reached.

The data transmission is suspended when the NACK is received, and DEVDRV_ERROR_RIIC_NACK is returned.

After writing the data for the specified byte count to the I2C bus transmit data register (RIICnDRT), enables the transmit-end interrupt. Waits until the conditions for the transmit end notification information are satisfied by the user-defined function Userdef_RIICn_WaitTransmitEnd. Then initializes the information by using the user-defined function Userdef_RIICn_InitTransmitEnd (n=0 to 3).

Finally, checks the NACK receive flag. When the NACK is received, DEVDRV_ERROR_RIIC_NACK is returned by this function.

Arguments
- uint32_t channel : RIIC channel (0 to 3)
- uint8_t * buffer : Transmit data
- uint32_t byte : Byte count of transmit data

Return Value
- DEVDRV_SUCCESS : Success of RIIC transmission
- DEVDRV_ERROR : Failure of RIIC transmission
- DEVDRV_ERROR_RIIC_NACK : Failure of RIIC transmission due to NACK reception
## R_RIIC_Read

<table>
<thead>
<tr>
<th>Outline</th>
<th>RIIC reception</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>int32_t R_RIIC_Read(uint32_t channel, uint8_t * buffer, uint32_t byte)</td>
</tr>
<tr>
<td>Description</td>
<td>Receives the data for the bytes specified by the argument byte by using the RIIC master receive operation of the channel specified by the argument channel, and stores the data to the area specified by the argument buffer. Waits until the conditions for the receive data full notification information are satisfied by the user-defined function Userdef_RIICn_WaitReceiveFull. After initializing the information by using the Userdef_RIICn_InitReceiveFull function (n=0 to 3), the data is received. When the data at the second byte from last is received, 1 is set to the WAIT bit in the I2C bus mode register 3 (RIICnMR3) before the I2C bus receive data register (RIICnDRR) is read. This holds the period between the ninth clock and the first clock of the SCL at the low level until the I2C bus receive data register (RIICnDRR) is read when the last byte is received, so the state is such that issuing a stop condition is possible. When a value 1 or 2 is specified for the argument byte, 1 is set to the WAIT bit in the I2C bus mode register 3 (RIICnMR3) before a dummy read of the I2C bus receive data register (RIICnDRR), to output the SCL clock, is performed. When the data at the byte before last is received, 1 is set to the ACKBT bit in the I2C bus mode register 3 (RIICnMR3) and 1 is sent to the acknowledge bit in reception of the last byte to make the NACK response. When the last byte is received, the stop condition issuance is required. Finally checks the stop condition detection using the user-defined function Userdef_RIICn_WaitStop (n=0 to 3) to terminate this function.</td>
</tr>
</tbody>
</table>

| Arguments        | 
|------------------|-------------------|
| channel          | RIIC channel (0 to 3) |
| buffer           | Receive data      |
| byte             | Byte count of receive data |

| Return Value     | 
|------------------|-------------------|
| DEVDRV_SUCCESS   | Success of RIIC reception |
| DEVDRV_ERROR     | Failure of RIIC reception |

## R_RIIC_ReadDummy

<table>
<thead>
<tr>
<th>Outline</th>
<th>RIIC dummy read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>int32_t R_RIIC_ReadDummy(uint32_t channel)</td>
</tr>
<tr>
<td>Description</td>
<td>Performs a dummy read for the RIIC receive data register (RIICnDRR) of the channel specified by the argument channel (n=0 to 3) and outputs the clock to the I2C bus to start the receive operation.</td>
</tr>
</tbody>
</table>

| Arguments        | 
|------------------|-------------------|
| channel          | RIIC channel (0 to 3) |

| Return Value     | 
|------------------|-------------------|
| DEVDRV_SUCCESS   | Success of RIIC reception |
| DEVDRV_ERROR     | Failure of RIIC reception |
### R_RIIC_DetectStop

**Outline**
Checking RIIC stop condition detection

**Declaration**
static void RIIC_DetectStop(uint32_t channel)

**Description**
Checks the RIIC stop condition detection of the channel specified by the argument channel. Waits until the stop condition is detected by using the user-defined function Userdef_RIICn_WaitStop (n=0 to 3). Clears the NACK receive flag (NACKF) and the stop condition detection flag (STOP) for the next transfer operation.

**Arguments**
- uint32_t channel : RIIC channel (0 to 3)

**Return Value**
- DEVDRV_SUCCESS : Success in checking RIIC stop condition detection
- DEVDRV_ERROR : Failure in checking RIIC stop condition detection

### R_RIIC_ClearNack

**Outline**
Clearing RIIC NACK receive flag

**Declaration**
int32_t R_RIIC_ClearNack(uint32_t channel)

**Description**
Clears the RIIC NACK receive flag (NACKF) of the channel specified by the argument channel. In the sample code, the NACKF is cleared when the NACK is received within the acknowledge polling processing. This function is called when the polling processing is continued.

**Arguments**
- uint32_t channel : RIIC channel (0 to 3)

**Return Value**
- DEVDRV_SUCCESS : Success in clearing RIIC NACK receive flag
- DEVDRV_ERROR : Failure in clearing RIIC NACK receive flag
### R_RIIC_TiInterrupt

**Outline**
RIIC transmit data empty interrupt notification

**Declaration**
int32_t R_RIIC_TiInterrupt(uint32_t channel)

**Description**
Executes the processing to satisfy the conditions for the RIIC transmit data empty notification information of the channel specified by the argument channel.

The interrupt issuance can be notified by calling this function from the RIIC transmit data empty interrupt handler processing. The processing to satisfy the conditions of the transmit data empty notification information should be executed by using the user-defined function Userdef_RIICn_SetTransmitEmpty (n=0 to 3).

**Arguments**
- uint32_t channel: RIIC channel (0 to 3)

**Return Value**
- DEVDRV_SUCCESS: Success of RIIC transmit data empty interrupt notification
- DEVDRV_ERROR: Failure of RIIC transmit data empty interrupt notification

**Note**
Edge sense is used as the source of the RIIC transmit data empty interrupt, so it is not necessary to clear the interrupt source flag.

### R_RIIC_TeiInterrupt

**Outline**
RIIC transmit-end interrupt notification

**Declaration**
int32_t R_RIIC_TeiInterrupt(uint32_t channel)

**Description**
Executes the processing to satisfy the conditions for the RIIC transmit-end notification information of the channel specified by the argument channel, and clears the interrupt source flag (TEND bit).

The interrupt issuance can be notified by calling this function from the RIIC transmit-end interrupt handler processing.

The processing to satisfy the conditions for the transmit-end notification information should be executed by using the user-defined function Userdef_RIICn_SetTransmitEnd (n=0 to 3).

**Arguments**
- uint32_t channel: RIIC channel (0 to 3)

**Return Value**
- DEVDRV_SUCCESS: Success of RIIC transmit-end interrupt notification
- DEVDRV_ERROR: Failure of RIIC transmission-end interrupt notification

### R_RIIC_RiInterrupt

**Outline**
RIIC receive data full interrupt notification

**Declaration**
int32_t R_RIIC_RiInterrupt(uint32_t channel)

**Description**
Executes the processing to satisfy the conditions for the RIIC receive data full notification information of the channel specified by the argument channel.

The interrupt issuance can be notified by calling this function from the RIIC receive data full interrupt handler processing. The processing to satisfy the conditions for the receive data full notification information should be executed by using the user-defined function Userdef_RIICn_SetReceiveFull (n=0 to 3).

**Arguments**
- uint32_t channel: RIIC channel (0 to 3)

**Return Value**
- DEVDRV_SUCCESS: Success of RIIC receive data full interrupt notification
- DEVDRV_ERROR: Failure of RIIC receive data full interrupt notification

**Note**
Edge sense is used as the source of the RIIC receive data full interrupt, so it is not necessary to clear the interrupt source flag.
Userdef_RIIC0_Init

Outline  RIIC channel 0 initial settings
Declaration void Userdef_RIIC0_Init(void)
Description This is a user-defined function. The RIIC initial settings and the settings for the
interrupts (receive data full interrupt, transmit data empty interrupt, and transmit-end
interrupt) are required when the RIIC channel 0 is used.
In the sample code, this function returns without any processing.
Arguments None
Return Value None

Userdef_RIIC1_Init

Outline  RIIC channel 1 initial settings
Declaration void Userdef_RIIC1_Init(void)
Description This is a user-defined function. The RIIC initial settings and the settings for the
interrupts (receive data full interrupt, transmit data empty interrupt, and transmit-end
interrupt) are required when the RIIC channel 1 is used.
In the sample code, this function returns without any processing.
Arguments None
Return Value None

Userdef_RIIC2_Init

Outline  RIIC channel 2 initial settings
Declaration void Userdef_RIIC2_Init(void)
Description This is a user-defined function. The initial settings are required to use the RIIC
channel 2.
In the sample code, the STB initial setting is executed to supply a clock to the RIIC
channel 2. The RIIC initial settings such as RIIC internal reference clock, digital noise
filter stages, SCL clock High/Low width are executed to enable the transfer at the
speed of 100kbps.
The settings to use the receive data full interrupt, the transmit data empty interrupt,
and the transmit-end interrupt and the INTC setting are also executed.
Arguments None
Return Value None

Userdef_RIIC3_Init

Outline  RIIC channel 3 initial settings
Declaration void Userdef_RIIC3_Init(void)
Description This is a user-defined function. The RIIC initial settings and the settings for the
interrupts (receive data full interrupt, transmit data empty interrupt, and transmit-end
interrupt) are required when the RIIC channel 3 is used.
In the sample code, this function returns without any processing.
Arguments None
Return Value None
Userdef_RIIC0_InitReceiveFull

Outline: Initialization of RIIC channel 0 receive data full notification information

Declaration: void Userdef_RIIC0_InitReceiveFull(void)

Description: This is a user-defined function.
Execute this function to initialize the RIIC channel 0 receive data full notification information when the RIIC channel 0 is used.
In the sample code, this function returns without any processing.

Arguments: None
Return Value: None

Userdef_RIIC1_InitReceiveFull

Outline: Initialization of RIIC channel 1 receive data full notification information

Declaration: void Userdef_RIIC1_InitReceiveFull(void)

Description: This is a user-defined function.
Execute this function to initialize the RIIC channel 1 receive data full notification information when the RIIC channel 1 is used.
In the sample code, this function returns without any processing.

Arguments: None
Return Value: None

Userdef_RIIC2_InitReceiveFull

Outline: Initialization of RIIC channel 2 receive data full notification information

Declaration: void Userdef_RIIC2_InitReceiveFull(void)

Description: This is a user-defined function.
Execute this function to initialize the RIIC channel 2 receive data full notification information.
In the sample code, "0" is set to the software flag to notify that the RIIC channel 2 receive data full interrupt has been generated.

Arguments: None
Return Value: None

Userdef_RIIC3_InitReceiveFull

Outline: Initialization of RIIC channel 3 receive data full notification information

Declaration: void Userdef_RIIC3_InitReceiveFull(void)

Description: This is a user-defined function.
Execute this function to initialize the RIIC channel 3 receive data full notification information when the RIIC channel 3 is used.
In the sample code, this function returns without any processing.

Arguments: None
Return Value: None
### Userdef_RIIC0_InitTransmitEmpty

**Outline**
Initialization of RIIC channel 0 transmit data empty notification information

**Declaration**
```c
void Userdef_RIIC0_InitTransmitEmpty(void)
```

**Description**
This is a user-defined function.

Execute this function to initialize the RIIC channel 0 transmit data empty notification information when the RIC channel 0 is used.

In the sample code, this function returns without any processing.

**Arguments**
None

**Return Value**
None

### Userdef_RIIC1_InitTransmitEmpty

**Outline**
Initialization of RIIC channel 1 transmit data empty notification information

**Declaration**
```c
void Userdef_RIIC1_InitTransmitEmpty(void)
```

**Description**
This is a user-defined function.

Execute this function to initialize the RIIC channel 1 transmit data empty notification information when the RIC channel 1 is used.

In the sample code, this function returns without any processing.

**Arguments**
None

**Return Value**
None

### Userdef_RIIC2_InitTransmitEmpty

**Outline**
Initialization of RIIC channel 2 transmit data empty notification information

**Declaration**
```c
void Userdef_RIIC2_InitTransmitEmpty(void)
```

**Description**
This is a user-defined function.

Execute this function to initialize the RIIC channel 2 transmit data empty notification information.

In the sample code, "0" is set to the software flag to notify that the RIIC channel 2 data empty interrupt has been generated.

**Arguments**
None

**Return Value**
None

### Userdef_RIIC3_InitTransmitEmpty

**Outline**
Initialization of RIIC channel 3 transmit data empty notification information

**Declaration**
```c
void Userdef_RIIC3_InitTransmitEmpty(void)
```

**Description**
This is a user-defined function.

Execute this function to initialize the RIIC channel 3 transmit data empty notification information when the RIC channel 3 is used.

In the sample code, this function returns without any processing.

**Arguments**
None

**Return Value**
None
Userdef_RIIC0_InitTransmitEnd

Outline
Initialization of RIIC channel 0 transmit-end notification information

Declaration
void Userdef_RIIC0_InitTransmitEnd(uint32_t mode)

Description
This is a user-defined function.
When the RIIC channel 0 is used, execute this function to initialize the RIIC channel 0 transmit-end notification information if the argument mode specifies RIIC_TEND_WAIT_TRANSMIT, or initialize the RIIC channel 0 receive data full notification information if the argument mode specifies RIIC_TEND_WAIT_RECEIVE.
In the sample code, this function returns without any processing.

Arguments
uint32_t mode : Continuation of transmission mode or transition to reception mode
  RIIC_TEND_WAIT_TRANSMIT : Continuation of transmission mode
  RIIC_TEND_WAIT_RECEIVE : Transition to reception mode

Return Value
None

Userdef_RIIC1_InitTransmitEnd

Outline
Initialization of RIIC channel 1 transmit-end notification information

Declaration
void Userdef_RIIC1_InitTransmitEnd(uint32_t mode)

Description
This is a user-defined function.
When the RIIC channel 1 is used, execute this function to initialize the RIIC channel 1 transmit-end notification information if the argument mode specifies RIIC_TEND_WAIT_TRANSMIT, or initialize the RIIC channel 1 receive data full notification information if the argument mode specifies RIIC_TEND_WAIT_RECEIVE.
In the sample code, this function returns without any processing.

Arguments
uint32_t mode : Continuation of transmission mode or transition to reception mode
  RIIC_TEND_WAIT_TRANSMIT : Continuation of transmission mode
  RIIC_TEND_WAIT_RECEIVE : Transition to reception mode

Return Value
None
Userdef_RIIC2_InitTransmitEnd

Outline
Initialization of RIIC channel 2 transmit-end notification information

Declaration
void Userdef_RIIC2_InitTransmitEnd(uint32_t mode)

Description
This is a user-defined function. Execute this function to initialize the RIIC channel 2 transmit-end notification information. In the sample code, "0" is set to the software flag to notify that the RIIC channel 2 transmit-end interrupt has been generated if the argument mode specifies RIIC_TEND_WAIT_TRANSMIT. When the said argument specifies RIIC_TEND_WAIT_RECEIVE, "0" is set to the software flag to notify that the RIIC channel 2 receive data full interrupt has been generated.

Arguments
uint32_t mode : Continuation of transmission mode or transition to reception mode

   RIIC_TEND_WAIT_TRANSMIT : Continuation of transmission mode

   RIIC_TEND_WAIT_RECEIVE : Transition to reception mode

Return Value
None

Userdef_RIIC3_InitTransmitEnd

Outline
Initialization of RIIC channel 3 transmit-end notification information

Declaration
void Userdef_RIIC3_InitTransmitEnd(uint32_t mode)

Description
This is a user-defined function. When the RIIC channel 3 is used, execute this function to initialize the RIIC channel 3 transmit-end notification information if the argument mode specifies RIIC_TEND_WAIT_TRANSMIT, or initialize the RIIC channel 3 receive data full notification information if the argument mode specifies RIIC_TEND_WAIT_RECEIVE. In the sample code, this function returns without any processing.

Arguments
uint32_t mode : Continuation of transmission mode or transition to reception mode

   RIIC_TEND_WAIT_TRANSMIT : Continuation of transmission mode

   RIIC_TEND_WAIT_RECEIVE : Transition to reception mode

Return Value
None
<table>
<thead>
<tr>
<th>Function</th>
<th>Outline</th>
<th>Declaration</th>
<th>Description</th>
<th>Arguments</th>
<th>Return Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Userdef_RIIC0_SetReceiveFull</td>
<td>Setting to satisfy the conditions for RIIC channel 0 receive data full notification information</td>
<td>void Userdef_RIIC0_SetReceiveFull(void)</td>
<td>Execute this function to satisfy the conditions for the RIIC channel 0 receive data full notification information when the RIIC channel 0 is used.</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Userdef_RIIC1_SetReceiveFull</td>
<td>Setting to satisfy the conditions for RIIC channel 1 receive data full notification information</td>
<td>void Userdef_RIIC1_SetReceiveFull(void)</td>
<td>Execute this function to satisfy the conditions for the RIIC channel 1 receive data full notification information when the RIIC channel 1 is used.</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Userdef_RIIC2_SetReceiveFull</td>
<td>Setting to satisfy the conditions for RIIC channel 2 receive data full notification information</td>
<td>void Userdef_RIIC2_SetReceiveFull(void)</td>
<td>Execute this function to satisfy the conditions for the RIIC channel 2 receive data full notification information. In the sample code, &quot;1&quot; is set to the software flag to notify that the RIIC channel 2 receive data full interrupt has been generated.</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Userdef_RIIC3_SetReceiveFull</td>
<td>Setting to satisfy the conditions for RIIC channel 3 receive data full notification information</td>
<td>void Userdef_RIIC3_SetReceiveFull(void)</td>
<td>Execute this function to satisfy the conditions for the RIIC channel 3 receive data full notification information when the RIIC channel 3 is used.</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Userdef_RIIC0_SetTransmitEmpty

Outline
Setting to satisfy the conditions for RIIC channel 0 transmit data empty notification information

Declaration
void Userdef_RIIC0_SetTransmitEmpty(void)

Description
This is a user-defined function.
Execute this function to satisfy the conditions for the RIIC channel 0 transmit data empty notification information when the RIIC channel 0 is used.
In the sample code, this function returns without any processing.

Arguments
None

Return Value
None

Userdef_RIIC1_SetTransmitEmpty

Outline
Setting to satisfy the conditions for RIIC channel 1 transmit data empty notification information

Declaration
void Userdef_RIIC1_SetTransmitEmpty(void)

Description
This is a user-defined function.
Execute this function to satisfy the conditions for the RIIC channel 1 transmit data empty notification information when the RIIC channel 1 is used.
In the sample code, this function returns without any processing.

Arguments
None

Return Value
None

Userdef_RIIC2_SetTransmitEmpty

Outline
Setting to satisfy the conditions for RIIC channel 2 transmit data empty notification information

Declaration
void Userdef_RIIC2_SetTransmitEmpty(void)

Description
This is a user-defined function.
Execute this function to satisfy the conditions for the RIIC channel 2 transmit data empty notification information.
In the sample code, "1" is set to the software flag to notify that the RIIC channel 2 transmit data empty interrupt has been generated.

Arguments
None

Return Value
None

Userdef_RIIC3_SetTransmitEmpty

Outline
Setting to satisfy the conditions for RIIC channel 3 transmit data empty notification information

Declaration
void Userdef_RIIC3_SetTransmitEmpty(void)

Description
This is a user-defined function.
Execute this function to satisfy the conditions for the RIIC channel 3 transmit data empty notification information when the RIIC channel 3 is used.
In the sample code, this function returns without any processing.

Arguments
None

Return Value
None
<table>
<thead>
<tr>
<th>Function</th>
<th>Outline</th>
<th>Declaration</th>
<th>Description</th>
<th>Arguments</th>
<th>Return Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Userdef_RIIC0_SetTransmitEnd</td>
<td>Setting to satisfy the conditions for RIIC channel 0 transmit-end notification information</td>
<td>void Userdef_RIIC0_SetTransmitEnd(void)</td>
<td>Execute this function to satisfy the conditions for the RIIC channel 0 transmit-end notification information when the RIIC channel 0 is used. In the sample code, this function returns without any processing.</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Userdef_RIIC1_SetTransmitEnd</td>
<td>Setting to satisfy the conditions for RIIC channel 1 transmit-end notification information</td>
<td>void Userdef_RIIC1_SetTransmitEnd(void)</td>
<td>Execute this function to satisfy the conditions for the RIIC channel 1 transmit-end notification information when the RIIC channel 1 is used. In the sample code, this function returns without any processing.</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Userdef_RIIC2_SetTransmitEnd</td>
<td>Setting to satisfy the conditions for RIIC channel 2 transmit-end notification information</td>
<td>void Userdef_RIIC2_SetTransmitEnd(void)</td>
<td>Execute this function to satisfy the conditions for the RIIC channel 2 transmit-end notification information. In the sample code, &quot;1&quot; is set to the software flag to notify that the RIIC channel 2 transmit-end interrupt has been generated.</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Userdef_RIIC3_SetTransmitEnd</td>
<td>Setting to satisfy the conditions for RIIC channel 3 transmit-end notification information</td>
<td>void Userdef_RIIC3_SetTransmitEnd(void)</td>
<td>Execute this function to satisfy the conditions for the RIIC channel 3 transmit-end notification information when the RIIC channel 3 is used. In the sample code, this function returns without any processing.</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
**Userdef_RIIC0_WaitReceiveFull**

**Outline**
Waiting for satisfaction of the conditions for RIIC channel 0 receive data full notification information

**Declaration**
void Userdef_RIIC0_WaitReceiveFull(void)

**Description**
This is a user-defined function.
Execute this function to wait until the conditions for the RIIC channel 0 receive data full notification information to be satisfied when the RIIC channel 0 is used.
In the sample code, this function returns without any processing.

**Arguments**
None

**Return Value**
None

**Userdef_RIIC1_WaitReceiveFull**

**Outline**
Waiting for satisfaction of the conditions for RIIC channel 1 receive data full notification information

**Declaration**
void Userdef_RIIC1_WaitReceiveFull(void)

**Description**
This is a user-defined function.
Execute this function to wait until the conditions for the RIIC channel 1 receive data full notification information to be satisfied when the RIIC channel 1 is used.
In the sample code, this function returns without any processing.

**Arguments**
None

**Return Value**
None

**Userdef_RIIC2_WaitReceiveFull**

**Outline**
Waiting for satisfaction of the conditions for RIIC channel 2 receive data full notification information

**Declaration**
void Userdef_RIIC2_WaitReceiveFull(void)

**Description**
This is a user-defined function.
Execute this function to wait until the conditions for the RIIC channel 2 receive data full notification information to be satisfied.
In the sample code, waits until "1" is set to the software flag to notify that the RIIC channel 2 receive data full interrupt has been generated.

**Arguments**
None

**Return Value**
None

**Userdef_RIIC3_WaitReceiveFull**

**Outline**
Waiting for satisfaction of the conditions for RIIC channel 3 receive data full notification information

**Declaration**
void Userdef_RIIC3_WaitReceiveFull(void)

**Description**
This is a user-defined function.
Execute this function to wait until the conditions for the RIIC channel 3 receive data full notification information to be satisfied when the RIIC channel 3 is used.
In the sample code, this function returns without any processing.

**Arguments**
None

**Return Value**
None
**Userdef_RIIC0_WaitTransmitEmpty**

<table>
<thead>
<tr>
<th>Outline</th>
<th>Waiting for satisfaction of the conditions for RIIC channel 0 transmit data empty notification information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>void Userdef_RIIC0_WaitTransmitEmpty(void)</td>
</tr>
<tr>
<td>Description</td>
<td>This is a user-defined function. Execute this function to wait until the conditions for the RIIC channel 0 transmit data empty notification information to be satisfied when the RIIC channel 0 is used. In the sample code, this function returns without any processing.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

**Userdef_RIIC1_WaitTransmitEmpty**

<table>
<thead>
<tr>
<th>Outline</th>
<th>Waiting for satisfaction of the conditions for RIIC channel 1 transmit data empty notification information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>void Userdef_RIIC1_WaitTransmitEmpty(void)</td>
</tr>
<tr>
<td>Description</td>
<td>This is a user-defined function. Execute this function to wait until the conditions for the RIIC channel 1 transmit data empty notification information to be satisfied when the RIIC channel 1 is used. In the sample code, this function returns without any processing.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

**Userdef_RIIC2_WaitTransmitEmpty**

<table>
<thead>
<tr>
<th>Outline</th>
<th>Waiting for satisfaction of the conditions for RIIC channel 2 transmit data empty notification information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>void Userdef_RIIC2_WaitTransmitEmpty(void)</td>
</tr>
<tr>
<td>Description</td>
<td>This is a user-defined function. Execute this function to wait until the conditions for the RIIC channel 2 transmit data empty notification information to be satisfied when the RIIC channel 2 is used. In the sample code, waits until &quot;1&quot; is set to the software flag to notify that the RIIC channel 2 transmit data empty interrupt has been generated.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

**Userdef_RIIC3_WaitTransmitEmpty**

<table>
<thead>
<tr>
<th>Outline</th>
<th>Waiting for satisfaction of the conditions for RIIC channel 3 transmit data empty notification information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>void Userdef_RIIC3_WaitTransmitEmpty(void)</td>
</tr>
<tr>
<td>Description</td>
<td>This is a user-defined function. Execute this function to wait until the conditions for the RIIC channel 3 transmit data empty notification information to be satisfied when the RIIC channel 3 is used. In the sample code, this function returns without any processing.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>
Userdef_RIIC0_WaitTransmitEnd

Outline
Waiting for satisfaction of the conditions for RIIC channel 0 transmit-end notification information

Declaration
void Userdef_RIIC0_WaitTransmitEnd(uint32_t mode)

Description
This is a user-defined function.
Execute this function to wait until the conditions for the RIIC channel 0 transmit-end notification information to be satisfied when the RIIC channel 0 is used.
In the sample code, this function returns without any processing.

Arguments
uint32_t mode : Selection for wait operation after transmission
  RIIC_TEND_WAIT_TRANSMIT : Continuation of transmission mode
  RIIC_TEND_WAIT_RECEIVE : Transition to reception mode

Return Value
None

Userdef_RIIC1_WaitTransmitEnd

Outline
Waiting for satisfaction of the conditions for RIIC channel 1 transmit-end notification information

Declaration
void Userdef_RIIC1_WaitTransmitEnd(uint32_t mode)

Description
This is a user-defined function.
Execute this function to wait until the conditions for the RIIC channel 1 transmit-end notification information to be satisfied when the RIIC channel 1 is used.
In the sample code, this function returns without any processing.

Arguments
uint32_t mode : Selection for wait operation after transmission
  RIIC_TEND_WAIT_TRANSMIT : Continuation of transmission mode
  RIIC_TEND_WAIT_RECEIVE : Transition to reception mode

Return Value
None
**Userdef_RIIC2_WaitTransmitEnd**

<table>
<thead>
<tr>
<th>Outline</th>
<th>Waiting for satisfaction of the conditions for RIIC channel 2 transmit-end notification information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>void Userdef_RIIC2_WaitTransmitEnd(uint32_t mode)</td>
</tr>
<tr>
<td>Description</td>
<td>This is a user-defined function. Execute this function to wait until the conditions for the RIIC channel 2 transmit-end notification information to be satisfied. In the sample code, waits until &quot;1&quot; is set to the software flag to notify that the RIIC channel 2 transmit-end interrupt has been generated if the argument mode specifies RIIC_TEND_WAIT_TRANSMIT. When the said argument specifies RIIC_TEND_WAIT_RECEIVE, waits until &quot;1&quot; is set to the software flag to notify that the RIIC channel 2 receive data full interrupt has been generated.</td>
</tr>
<tr>
<td>Arguments</td>
<td></td>
</tr>
<tr>
<td>uint32_t mode</td>
<td>Selection for wait operation after transmission</td>
</tr>
<tr>
<td></td>
<td>RIIC_TEND_WAIT_TRANSMIT : Continuation of transmission mode</td>
</tr>
<tr>
<td></td>
<td>RIIC_TEND_WAIT_RECEIVE : Transition to reception mode</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
<tr>
<td>Note</td>
<td>When transmitting a slave address with the R/W# bit cleared to &quot;0&quot; (write operation), and data transmission has finished, processing to wait for transmit-end occurs when the value of the argument mode is RIIC_TEND_WAIT_TRANSMIT because the transmit-end flag (TEND) is set to &quot;1&quot;. When transmitting a slave address with the R/W# bit set to &quot;1&quot; (read operation), there is a transition from master transmit mode to master receive mode, and processing to wait for receive data full occurs when the value of the argument mode is RIIC_TEND_WAIT_RECEIVE because the receive data-full flag (RDRF) is set to &quot;1&quot;.</td>
</tr>
</tbody>
</table>

**Userdef_RIIC3_WaitTransmitEnd**

<table>
<thead>
<tr>
<th>Outline</th>
<th>Waiting for satisfaction of the conditions for RIIC channel 3 transmit-end notification information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>void Userdef_RIIC3_WaitTransmitEnd(uint32_t mode)</td>
</tr>
<tr>
<td>Description</td>
<td>This is a user-defined function. Execute this function to wait until the conditions for the RIIC channel 3 transmit-end notification information to be satisfied when the RIIC channel 3 is used. In the sample code, this function returns without any processing.</td>
</tr>
<tr>
<td>Arguments</td>
<td></td>
</tr>
<tr>
<td>uint32_t mode</td>
<td>Selection for wait operation after transmission</td>
</tr>
<tr>
<td></td>
<td>RIIC_TEND_WAIT_TRANSMIT : Continuation of transmission mode</td>
</tr>
<tr>
<td></td>
<td>RIIC_TEND_WAIT_RECEIVE : Transition to reception mode</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>


### Userdef_RIIC0_WaitBusMastership

<table>
<thead>
<tr>
<th><strong>Outline</strong></th>
<th>Waiting for RIIC channel 0 bus free and bus busy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Declaration</strong></td>
<td>void Userdef_RIIC0_WaitBusMastership(uint32_t mode)</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>This is a user-defined function. Execute this function to wait for the RIIC channel 0 bus free and bus busy when the RIIC channel 0 is used. In the sample code, this function returns without any processing.</td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
<td>uint32_t mode : Mode selection for bus free or bus busy</td>
</tr>
<tr>
<td></td>
<td>RIIC_BUS_MASTERSHIP_WAIT_FREE : Wait for bus free</td>
</tr>
<tr>
<td></td>
<td>RIIC_BUS_MASTERSHIP_WAIT_BUSY : Wait for bus busy</td>
</tr>
<tr>
<td><strong>Return Value</strong></td>
<td>None</td>
</tr>
</tbody>
</table>

### Userdef_RIIC1_WaitBusMastership

<table>
<thead>
<tr>
<th><strong>Outline</strong></th>
<th>Waiting for RIIC channel 1 bus free and bus busy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Declaration</strong></td>
<td>void Userdef_RIIC1_WaitBusMastership(uint32_t mode)</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>This is a user-defined function. Execute this function to wait for the RIIC channel 1 bus free and bus busy when the RIIC channel 1 is used. In the sample code, this function returns without any processing.</td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
<td>uint32_t mode : Mode selection for bus free or bus busy</td>
</tr>
<tr>
<td></td>
<td>RIIC_BUS_MASTERSHIP_WAIT_FREE : Wait for bus free</td>
</tr>
<tr>
<td></td>
<td>RIIC_BUS_MASTERSHIP_WAIT_BUSY : Wait for bus busy</td>
</tr>
<tr>
<td><strong>Return Value</strong></td>
<td>None</td>
</tr>
</tbody>
</table>

### Userdef_RIIC2_WaitBusMastership

<table>
<thead>
<tr>
<th><strong>Outline</strong></th>
<th>Waiting for RIIC channel 2 bus free and bus busy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Declaration</strong></td>
<td>void Userdef_RIIC2_WaitBusMastership(uint32_t mode)</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>This is a user-defined function. Execute this function to wait for the RIIC channel 2 bus free and bus busy. In the sample code, waits until &quot;0&quot; is set to the BBSY bit in the RIIC2CR2 register if the argument mode specifies RIIC_BUS_MASTERSHIP_WAIT_FREE. When the said argument specifies RIIC_BUS_MASTERSHIP_WAIT_BUSY, waits until &quot;1&quot; is set to the BBSY bit in the RIIC2CR2 register.</td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
<td>uint32_t mode : Mode selection for bus free or bus busy</td>
</tr>
<tr>
<td></td>
<td>RIIC_BUS_MASTERSHIP_WAIT_FREE : Wait for bus free</td>
</tr>
<tr>
<td></td>
<td>RIIC_BUS_MASTERSHIP_WAIT_BUSY : Wait for bus busy</td>
</tr>
<tr>
<td><strong>Return Value</strong></td>
<td>None</td>
</tr>
</tbody>
</table>
### Userdef_RIIC3_WaitBusMastership

**Outline**  Waiting for RIIC channel 3 bus free and bus busy

**Declaration**  
```c
void Userdef_RIIC3_WaitBusMastership(uint32_t mode)
```

**Description**  
This is a user-defined function.
- Execute this function to wait for the RIIC channel 3 bus free and bus busy when the RIIC channel 3 is used.
- In the sample code, this function returns without any processing.

**Arguments**  
- `uint32_t mode`: Mode selection for bus free or bus busy
  - `RIIC_BUS_MASTERSHIP_WAIT_FREE`: Wait for bus free
  - `RIIC_BUS_MASTERSHIP_WAIT_BUSY`: Wait for bus busy

**Return Value**  None

### Userdef_RIIC0_WaitStop

**Outline**  Waiting for RIIC channel 0 stop condition to be detected

**Declaration**  
```c
void Userdef_RIIC0_WaitStop(void)
```

**Description**  
This is a user-defined function.
- Execute this function to wait until the RIIC channel 0 stop condition to be detected when the RIIC channel 0 is used.
- In the sample code, this function returns without any processing.

**Arguments**  None

**Return Value**  None

### Userdef_RIIC1_WaitStop

**Outline**  Waiting for RIIC channel 1 stop condition to be detected

**Declaration**  
```c
void Userdef_RIIC1_WaitStop(void)
```

**Description**  
This is a user-defined function.
- Execute this function to wait until the RIIC channel 1 stop condition to be detected when the RIIC channel 1 is used.
- In the sample code, this function returns without any processing.

**Arguments**  None

**Return Value**  None
### Userdef_RIIC2_WaitStop

**Outline**
Waiting for RIIC channel 2 stop condition to be detected

**Declaration**
```c
void Userdef_RIIC2_WaitStop(void)
```

**Description**
This is a user-defined function. Execute this function to wait until the RIIC channel 2 stop condition to be detected. In the sample code, waits until "1" is set to the STOP bit in the RIIC2SR2 register.

**Arguments**
None

**Return Value**
None

### Userdef_RIIC3_WaitStop

**Outline**
Waiting for RIIC channel 3 stop condition to be detected

**Declaration**
```c
void Userdef_RIIC3_WaitStop(void)
```

**Description**
This is a user-defined function. Execute this function to wait until the RIIC channel 3 stop condition to be detected when the RIIC channel 3 is used. In the sample code, this function returns without any processing.

**Arguments**
None

**Return Value**
None
5.9 Flowcharts

5.9.1 Main Function

Figure 5.7 shows the flowchart of Main Processing.

```
main
  | Output to terminal
  | printf()
  | OSTM0-related settings
  | Peripheral function sample code startup function
  | Sample_Main()
  | return (0)
  
Outputs the sample code version information to the terminal program running on the host PC connected via the serial interface.

Blinks the LED at 500ms intervals using the OSTM channel 0 interrupt. Refer to the RZ/A1H Group Example of Initialization application note for details.

Branches to processing that waits to receive a command from the terminal. The sample code for the appropriate peripheral function is executed according to the command received.
```

Figure 5.7 Main Processing
5.9.2 Sample Code Main Function

Figure 5.8 shows the flowchart of the Sample Code Main Function. This function waits for the character input from the terminal running on the host PC.

The RIIC sample code is executed when "RIIC" + "Enter" key is input.

```
Sample_Main

Output to terminal
printf()

Acquisition of main processing command list
Sample_GetCmdList()

Registration of main processing command list
CommandSetCmdList()

Wait for command input
gets()

Analyze and run command
CommandExe()

*EXIT* input?

*EXIT* was input
Reacquisition of main processing command list
Sample_GetCmdList()

*EXIT* was not input
Reregistration of main processing command list
CommandSetCmdList()
```

Figure 5.8 Sample Code Main Function
5.9.3 RIIC Sample Code Main Function

Figure 5.9 shows the flowchart of RIIC Sample Main Function. This function waits for the character input from the terminal running on the host PC and branches to the RIIC sample code processing according to the input command. When "1" + "Enter" key is input, the sample code is executed to write/read the 64 bytes of data to/from the EEPROM at transfer speed of 100kbps.

Outputs the RIIC sample program version information to the terminal program running on the host PC.

Obtains the RIIC sample processing command list. The menu list for launching the RIIC sample code is obtained in this sample code.

Registers the RIIC sample processing command list. The menu list for launching the RIIC sample code is registered in this sample code.

Waits for command input from the terminal and stores it in the command buffer.

Analyzes and executes the contents of the command buffer. In the sample code, this function branches to one of the following RIIC sample code processing routines, according to the input command:
- "1": Branches to the Sample_RIIC_EepAccess function.
- "HELP": Displays a list of the available commands.

"EXIT" input?

"EXIT" was not input

"EXIT" was input

return (COMMAND_EXIT)
5.9.4 Sample Function for Reading/Writing Data From/To EEPROM

Figure 5.10 shows the flowchart of Sample Function for Reading/Writing Data From/To EEPROM. In the sample code, the transfer speed is set to 100kbps to write 64 bytes of data to the EEPROM and then to read the data from it.

```
Sample_RIIC_EepAccess
  Transmit buffer initialization
  Receive buffer initialization
  PORT initial settings
    RIIC_PORT_Init()

  RIIC initial settings
    R_RIIC_Init()

  Transmit data to EEPROM
    Sample_RIIC_EepWrite()

  Wait for writing completion
    to EEPROM
    Sample_RIIC_EepAckPolling()

  Receive data from EEPROM
    Sample_RIIC_EepRead()

  return (COMMAND_SUCCESS)
```

Figure 5.10   Sample Function for Reading/Writing Data From/To EEPROM

Generates the transmit data to the EEPROM (data incremented from "0" to "63") and stores in the transmit buffer (trans_data[]).

Initializes the store buffer (receive_data[]) for data received from the EEPROM to "0".

Provides initial settings for the PORT used in RIIC transmission/reception.

Provides initial settings for RIIC channel 2 such that the transfer speed is 100 kbps when P0f = 33.33 MHz.

Transmits 64 bytes of data from transmit buffer trans_data[] to the EEPROM, starting at memory address H'0000. (The device code is set to "B'1010" and the device address code to "B'000" as the slave address for transmission. The device address code is set to "B'000" by means of EEPROM pins A2 to A0 on the board.)

Uses the EEPROM acknowledge polling function to wait until the data has been written to the EEPROM internally.

Starting from EEPROM memory address H'0000, receives 64 bytes of data and stores it in the receive buffer receive_data[].
5.9.5 Initial Setting Function for PORT

Figure 5.11 shows the flowchart of Initial Setting Function for PORT. In the sample code, PORT1 pin functions are set so that P1_4 is assigned as RIIC2SCL and P1_5 is assigned as RIIC2SDA.

Figure 5.11 Initial Setting Function for PORT

Set PORT1 multiplexed pin functions

RIIC_PORT_Init

Port P1-related registers
PIBC14 bit ← 0, PIBC15 bit ← 0
PBDC14 bit ← 0, PBDC15 bit ← 0
PM14 bit ← 1, PM15 bit ← 1
PMC14 bit ← 0, PMC15 bit ← 0
PIPC14 bit ← 0, PIPC15 bit ← 0
PBDC14 bit ← 1, PBDC15 bit ← 1
PFCE14 bit ← 0, PFCE15 bit ← 0
PFCAE14 bit ← 0, PFCAE15 bit ← 0
PIPC14 bit ← 1, PIPC15 bit ← 1
PMC14 bit ← 1, PMC15 bit ← 1

To enable transmission and reception on RIIC channel 2, multiplex pin functions set to use the P1_4 pin and the P1_5 pin as RIIC2SCL and RIIC2SDA respectively.
### 5.9.6 Sample Function for Writing to EEPROM

Figure 5.12 shows the flowchart of Sample Function for Writing to EEPROM. The master transmit mode is used to perform a page write for 64 bytes of data to the EEPROM. The RiIC transitions to master transmit mode automatically when the start condition is issued.

#### Sample_RIIC_EepWrite

Uses master transmit mode on the RiIC to perform a page write to the EEPROM.

\[
\text{w_saddr} \leftarrow (\text{d_code} | \text{d_adr} << 1) | \text{R/W# bit (=0)}
\]

Generates the slave address (device code and device address code) from the values of arguments d_code and d_addr. Clears the R/W# bit to "0" and stores the result in the transmit data buffer.

\[
\text{maddr}[0] \leftarrow (\text{uint8_t})(\text{w_adr} >> 8), \text{maddr}[1] \leftarrow (\text{uint8_t})(\text{w_adr} & 0xff)
\]

Generates the 14-bit EEPROM memory address from the value of the argument w_adr, and stores it in the transmit data buffer.

#### Request start condition issuance

\[\text{R_RIIC_SendCond()}\]

Issues the start condition on RiIC channel 2. The RiIC automatically transitions to master transmit mode.

#### Transmit slave address

\[\text{R_RIIC_WriteSlaveAddr()}\]

Transmits the slave address to the EEPROM on RiIC channel 2. Sets the argument to w_saddr.

#### Set EEPROM memory address

\[\text{maddr}[0] \leftarrow (\text{uint8_t})(\text{w_adr} >> 8), \text{maddr}[1] \leftarrow (\text{uint8_t})(\text{w_adr} & 0xff)\]

Generates the 14-bit EEPROM memory address from the value of the argument w_adr, and stores it in the transmit data buffer.

#### Set data for slave address transmission for EEPROM write operation

\[\text{w_saddr} \leftarrow (\text{d_code} | \text{d_adr} << 1) | \text{R/W# bit (=0)}\]

Generates the slave address (device code and device address code) from the values of arguments d_code and d_addr. Clears the R/W# bit to "0" and stores the result in the transmit data buffer.

#### Transmit memory address

\[\text{R_RIIC_Write()}\]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

#### Transmit write data

\[\text{R_RIIC_Write()}\]

Writes the number of bytes specified by the argument w_byte of the data stored in the location specified by the argument *w_buffer to the EEPROM on RiIC channel 2. The page write for 64 bytes of data is performed in the sample code.

#### Request stop condition issuance

\[\text{R_RIIC_SendCond()}\]

Issues the stop condition on RiIC channel 2.

#### Confirm detection of stop condition

\[\text{R_RIIC_DetectStop()}\]

Confirms detection of the stop condition on RiIC channel 2. The RiIC automatically transitions to slave receive mode.

#### Return value of function indicates error?

Return value indicates error?

Return value indicates no error?

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the memory address on RiIC channel 2.

1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]
5.9.7 Sample Function for Reading from EEPROM

Figure 5.13 and Figure 5.14 show the flowcharts of Sample Function for Reading from EEPROM.

The format combining master transmit mode and master receive mode is used to perform a sequential read for the 64-byte data from the EEPROM. The RIIC automatically transitions to the master transmit mode at the start condition issuance and then automatically transitions to the master receive mode when the R/W# bit of the slave address is 1.

```
Sample_RIIC_EepRead

Set data for EEPROM slave address transmission

Set EEPROM memory address

Set data for slave address transmission for EEPROM read operation

Request start condition issuance
  R_RIIC_SendCond()

Transmit memory address
  R_RIIC_Write() 

   Return value indicates error
   Return value indicates no error

   Transmit memory address
     R_RIIC_Write()

   Return value indicates error
   Return value indicates no error

Request restart condition issuance
  R_RIIC_SendCond()

Transmit slave address
  R_RIIC_WriteSlaveAddr()

   Return value indicates an error
   Return value indicates no error

Transmit slave address
  R_RIIC_WriteSlaveAddr()

Return value indicates error
Return value indicates no error

Uses a format combining master transmit and master receive to perform a sequential read data from the EEPROM.

w_saddr ← ( d_code | d_addr << 1 | R/W# bit (=0) )
Generates the slave address (device code and device address code) from the values of arguments d_code and d_addr. Clears the R/W# bit to “0” and stores the result in the transmit data buffer.

maddr[0] ← (uint8_t)( r_adr >> 8 ), maddr[1] ← (uint8_t)( r_adr & 0xff )
Generates the 14-bit EEPROM memory address from the value of the argument w_addr, and stores it in the transmit data buffer.

r_saddr ← ( d_code | d_addr << 1 | R/W# bit (=1) )
Generates the slave address (device code and device address code) from the values of arguments d_code and d_addr. Clears the R/W# bit to “1” and stores the result in the transmit data buffer.

Transmits the slave address to the EEPROM on RIIC channel 2. Sets the argument to w_saddr. (The R/W# bit is transmitted with a value of “0”.)

Transmits the memory address on RIIC channel 2.
1st byte of argument: maddr[0]
2nd byte of argument: maddr[1]

Transmits the slave address to the EEPROM on RIIC channel 2. Sets the argument to r_saddr. (The R/W# bit is transmitted with a value of “1”.)
The RIIC automatically transitions from master transmit mode to master receive mode.

A

Figure 5.13 Sample Function for Reading from EEPROM (1/2)
Figure 5.14 Sample Function for Reading from EEPROM (2/2)
5.9.8 Sample Function to Wait for Writing Completion Inside EEPROM

Figure 5.15 shows the flowchart of Sample Function to Wait for Writing Completion Inside EEPROM.

After writing to the EEPROM and before reading from it, the acknowledge polling is performed to confirm that the writing of data has been completed inside the EEPROM. The RIIC automatically transitions to the master transmit mode at start condition issuance.

```
RIIC_EepAckPolling

Set data for slave address transmission for EEPROM write operation

Request start condition issuance
R_RIIC_SendCond()

Transmit slave address
R_RIIC_WriteSlaveAddr()

Return value indicates an error?
Return value indicates no error

Clear NACK receive flag
R_RIIC_ClearNack()

Request restart condition issuance
R_RIIC_SendCond()

Issues the stop condition on RIIC channel 2.
The RIIC automatically transitions to slave receive mode.

w_saddr ← (d_code | d_adr << 1 | R/W# bit (=0))
Generates the slave address (device code and device address code) from the values of arguments d_code and d_adr, clears the R/W# bit to “0”, and stores the result in the transmit data buffer.

Issues the start condition on RIIC channel 2.
The RIIC automatically transitions to master transmit mode.

Transmits the slave address to the EEPROM on RIIC channel 2.
Sets the argument to w_saddr.

Return value of function indicates an error?
Return value indicates no error

Clears the NACK receive flag on RIIC channel 2.

Issues the restart condition on RIIC channel 2.

Issues the stop condition on RIIC channel 2.

Confirms detection of the stop condition on RIIC channel 2.
The RIIC automatically transitions to slave receive mode.
```

Figure 5.15 Sample Function to Wait for Writing Completion Inside EEPROM
5.9.9  RIIC Channel 2 Receive Data Full Interrupt Function

Figure 5.16 shows the flowchart of RIIC Channel 2 Receive Data Full Interrupt Function.

In the sample code, this function is registered as the INTC handler by using the user-defined function Userdef_RIIC2_Init.

```
Sample_RIIC_Ri2_Interrupt

RIIC receive data full
interrupt notification
R_RIIC_RiInterrupt()

return
```

**Figure 5.16  RIIC Channel 2 Receive Data Full Interrupt Function**

5.9.10  RIIC Channel 2 Transmit Data Empty Interrupt Function

Figure 5.17 shows the flowchart of RIIC Channel 2 Transmit Data Empty Interrupt Function.

In the sample code, this function is registered as the INTC handler by using the user-defined function Userdef_RIIC2_Init.

```
Sample_RIIC_Ti2_Interrupt

RIIC transmit data empty
interrupt notification
R_RIIC_TiInterrupt()

return
```

**Figure 5.17  RIIC Channel 2 Transmit Data Empty Interrupt Function**

5.9.11  RIIC Channel 2 Transmit-End Interrupt Function

Figure 5.18 shows the flowchart of RIIC Channel 2 Transmit-End Interrupt Function.

In the sample code, this function is registered as the INTC handler by using the user-defined function Userdef_RIIC2_Init.

```
Sample_RIIC_Tei2_Interrupt

RIIC transmit-end
interrupt notification
R_RIIC_TeiInterrupt()

return
```

**Figure 5.18  RIIC Channel 2 Transmit-End Interrupt Function**
5.9.12 RIIC Initial Setting Function

Figure 5.19 shows the flowchart of RIIC Initial Setting Function.

R_RIIC_Init

Argument error in function?

channel = 0

RIIC channel 0 initial settings
Userdef_RIIC0_Init()

channel = 1

RIIC channel 1 initial settings
Userdef_RIIC1_Init()

channel = 2

RIIC channel 2 initial settings
Userdef_RIIC2_Init()

channel = 3

RIIC channel 3 initial settings
Userdef_RIIC3_Init()

default

return (DEVDRV_SUCCESS)

Error in specification for channel

return (DEVDRV_ERROR)

Provides RIIC initial settings.
When a value from 0 to 3 is specified for the argument channel, the user-defined function corresponding to the channel is called.
In the sample code, initial settings for the RIIC channel 2 and its interrupt are provided by using the user-defined function Userdef_RIIC2_Init.

Figure 5.19 RIIC Initial Setting Function
5.9.13 RIIC Condition Issuance Request Function

Figure 5.20 to Figure 5.23 show the flowcharts of RIIC Condition Issuance Request Function.

In order to issue the start condition, waits until the RIIC enters the bus-free state and initializes the receive data full, transmit data empty, and transmit-end notification information.

In the sample code, the user-defined function Userdef_RIIC2_WaitBusMastership is used to wait until the BBSY bit in the RIICnCR2 register is cleared to "0". The notification information for receive data full, transmit data empty, and transmit-end is initialized, and each flag notifying that an interrupt has been generated is cleared by using the user-defined functions Userdef_RIIC2_InitReceiveFull, Userdef_RIIC2_InitTransmitEmpty, and Userdef_RIIC2_InitTransmitEnd respectively.

User-defined functions have been prepared for RIIC channels, but the following functions are not used in the sample code.

Userdef_RIIC0_WaitBusMastership
Userdef_RIIC0_InitReceiveFull
Userdef_RIIC0_InitTransmitEmpty
Userdef_RIIC0_InitTransmitEnd
Userdef_RIIC1_WaitBusMastership
Userdef_RIIC1_InitReceiveFull
Userdef_RIIC1_InitTransmitEmpty
Userdef_RIIC1_InitTransmitEnd
Userdef_RIIC3_WaitBusMastership
Userdef_RIIC3_InitReceiveFull
Userdef_RIIC3_InitTransmitEmpty
Userdef_RIIC3_InitTransmitEnd

Figure 5.20 RIIC Condition Issuance Request Function (1/4)
channel = 2
Wait for RIIC channel 2 bus free state
Userdef_RIIC2_WaitBusMastership()

Initialize RIIC channel 2 receive
data full notification information
Userdef_RIIC2_InitReceiveFull()

Initialize RIIC channel 2 transmit
data empty notification information
Userdef_RIIC2_InitTransmitEmpty()

channel = 3
Wait for RIIC channel 3 bus free state
Userdef_RIIC3_WaitBusMastership()

Initialize RIIC channel 3 receive
data full notification information
Userdef_RIIC3_InitReceiveFull()

Initialize RIIC channel 3 transmit
data empty notification information
Userdef_RIIC3_InitTransmitEmpty()

default
Request start condition issuance
RIICnCR2 register
ST bit ← 1 : Request start condition issuance

Figure 5.21 RIIC Condition Issuance Request Function (2/4)
Request restart condition issuance

Channel?

channel = 0

Wait for RIIC channel 0 bus busy state
Userdef_RIIC0_WaitBusMastership()

Initialize RIIC channel 0 receive data full notification information
Userdef_RIIC0_InitReceiveFull()

Initialize RIIC channel 0 transmit data empty notification information
Userdef_RIIC0_InitTransmitEmpty()

Initialize RIIC channel 0 transmit-end notification information
Userdef_RIIC0_InitTransmitEnd()

channel = 1

Wait for RIIC channel 1 bus busy state
Userdef_RIIC1_WaitBusMastership()

Initialize RIIC channel 1 receive data full notification information
Userdef_RIIC1_InitReceiveFull()

Initialize RIIC channel 1 transmit data empty notification information
Userdef_RIIC1_InitTransmitEmpty()

Initialize RIIC channel 1 transmit-end notification information
Userdef_RIIC1_InitTransmitEnd()

In order to issue the restart condition, waits until the RIIC enters the bus free state and initializes the receive data full, transmit data empty, and transmit-end notification information.

In the sample code, the user-defined function Userdef_RIIC2_WaitBusMastership is used to wait until the BBSY bit in the RIICnCR2 register is cleared to "0". The notification information for receive data full, transmit data empty, and transmit-end is initialized and each flag notifying that an interrupt has been generated is cleared by using the user-defined functions. The user-defined functions Userdef_RIIC2_InitReceiveFull, Userdef_RIIC2_InitTransmitEmpty, and Userdef_RIIC2_InitTransmitEnd respectively.

User-defined functions have been prepared for the RIIC channels, but the following functions are not used in the sample code.
Userdef_RIIC0_WaitBusMastership
Userdef_RIIC0_InitReceiveFull
Userdef_RIIC0_InitTransmitEmpty
Userdef_RIIC0_InitTransmitEnd
Userdef_RIIC1_WaitBusMastership
Userdef_RIIC1_InitReceiveFull
Userdef_RIIC1_InitTransmitEmpty
Userdef_RIIC1_InitTransmitEnd
Userdef_RIIC3_WaitBusMastership
Userdef_RIIC3_InitReceiveFull
Userdef_RIIC3_InitTransmitEmpty
Userdef_RIIC3_InitTransmitEnd

Figure 5.22  RIIC Condition Issuance Request Function (3/4)
channel = 2
Wait for RIIC channel 2 bus free state
Userdef_RIIC2_WaitBusMastership()

Initialize RIIC channel 2 receive
data full notification information
Userdef_RIIC2_InitReceiveFull()

Initialize RIIC channel 2 transmit
data empty notification information
Userdef_RIIC2_InitTransmitEmpty()

Initialize RIIC channel 2 transmit-end
notification information
Userdef_RIIC2_InitTransmitEnd()

channel = 3
Wait for RIIC channel 3 bus free state
Userdef_RIIC3_WaitBusMastership()

Initialize RIIC channel 3 receive
data full notification information
Userdef_RIIC3_InitReceiveFull()

Initialize RIIC channel 3 transmit
data empty notification information
Userdef_RIIC3_InitTransmitEmpty()

Initialize RIIC channel 3 transmit-end
notification information
Userdef_RIIC3_InitTransmitEnd()

default

Request restart condition issuance
RIICnCR2 register
RS bit ← 1 : Request restart condition issuance

Request stop condition issuance
RIICnSR2 register
STOP bit ← 0 : Clear stop condition detection flag
RIICnCR2 register
SP bit ← 1 : Request stop condition issuance

default

return (DEVDRV_SUCCESS)

Figure 5.23  RIIC Condition Issuance Request Function (4/4)
5.9.14 RIIC Slave Address Transmit Function

Figure 5.24 and Figure 5.25 show the flowcharts of RIIC Slave Address Transmit Function.

![Flowchart of RIIC Slave Address Transmit Function](image)

channel = 0
Wait for satisfaction of the conditions for RIIC channel 0 transmit data empty notification information
Userdef_RIIC0_WaitTransmitEmpty()
Initialize RIIC channel 0 transmit data empty notification information
Userdef_RIIC0_InitTransmitEmpty()

channel = 1
Wait for satisfaction of the conditions for RIIC channel 1 transmit data empty notification information
Userdef_RIIC1_WaitTransmitEmpty()
Initialize RIIC channel 1 transmit data empty notification information
Userdef_RIIC1_InitTransmitEmpty()

channel = 2
Wait for satisfaction of the conditions for RIIC channel 2 transmit data empty notification information
Userdef_RIIC2_WaitTransmitEmpty()
Initialize RIIC channel 2 transmit data empty notification information
Userdef_RIIC2_InitTransmitEmpty()

channel = 3
Wait for satisfaction of the conditions for RIIC channel 3 transmit data empty notification information
Userdef_RIIC3_WaitTransmitEmpty()
Initialize RIIC channel 3 transmit data empty notification information
Userdef_RIIC3_InitTransmitEmpty()

default

Set 1 byte of transmit data
A

RIICnDRT register ← addr
Writes the value of the argument addr to RIICnDRT register.

Figure 5.24 RIIC Slave Address Transmit Function (1/2)
Waits until the conditions for the RIIC transmit-end notification information to be satisfied, and initializes the notification information.

In the sample code, the user-defined function Userdef_RIIC2_WaitTransmitEnd is used to wait until the RIIC channel 2 transmit-end notification flag is set, then the notification flag is cleared by using the user-defined function Userdef_RIIC2_InitTransmitEnd.

The transmit-end notification flag is cleared when a start condition or a restart condition is issued and set by the transmit-end interrupt handler.

User-defined functions have been prepared for the RIIC channels, but the following functions are not used in the sample code:

- Userdef_RIIC0_WaitTransmitEnd
- Userdef_RIIC1_WaitTransmitEnd
- Userdef_RIIC3_WaitTransmitEnd
- Userdef_RIIC0_InitTransmitEnd
- Userdef_RIIC1_InitTransmitEnd
- Userdef_RIIC3_InitTransmitEnd

Figure 5.25   RIIC Slave Address Transmit Function (2/2)
5.9.15 RIIC Transmit Function

Figure 5.26 and Figure 5.27 show the flowcharts of RIIC Transmit Function.

```c
R_RIIC_Write
Argument error in function?
return (DEVDRV_ERROR)

RIICnIER register
TEIE bit ← 0 : Disables the transmit end interrupt requests.
After TEIE bit is cleared, read TEIE bit (dummy read).

Disable
transmit end interrupt request

NACK not received?
(NACKF = 0?)
NACK received
return
(DEVDRV_ERROR_RIIC_NACK)

Channel?
channel = 0
Wait for satisfaction of the conditions for
RIIC channel 0 transmit data empty
notification information
Userdef_RIIC0_WaitTransmitEmpty()
Initialize RIIC channel 0 transmit
data empty notification information
Userdef_RIIC0_InitTransmitEmpty()

channel = 1
Wait for satisfaction of the conditions for
RIIC channel 1 transmit data empty
notification information
Userdef_RIIC1_WaitTransmitEmpty()
Initialize RIIC channel 1 transmit
data empty notification information
Userdef_RIIC1_InitTransmitEmpty()

channel = 2
Wait for satisfaction of the conditions for
RIIC channel 2 transmit data empty
notification information
Userdef_RIIC2_WaitTransmitEmpty()
Initialize RIIC channel 2 transmit
data empty notification information
Userdef_RIIC2_InitTransmitEmpty()

channel = 3
Wait for satisfaction of the conditions for
RIIC channel 3 transmit data empty
notification information
Userdef_RIIC3_WaitTransmitEmpty()
Initialize RIIC channel 3 transmit
data empty notification information
Userdef_RIIC3_InitTransmitEmpty()

default
Set 1 byte of transmit data
RIICnDRT register ← data
Writes the value of the argument data to RIICnDRT register.

All data transmitted?
Transmission of number of bytes specified
by argument byte finished
```

Figure 5.26 RIIC Transmit Function (1/2)
Waits until the conditions for the RIIC transmit-end notification information to be satisfied, and initializes the notification information.

In the sample code, the user-defined function Userdef_RIIC2_WaitTransmitEnd is used to wait until the RIIC channel 2 transmit-end notification flag is set, then the notification flag is cleared by using the user-defined function Userdef_RIIC2_InitTransmitEnd.

The transmit-end notification flag is cleared when a start condition or a restart condition is issued and set by the transmit-end interrupt handler.

User-defined functions have been prepared for the RIIC channels, but the following functions are not used in the sample code.

Userdef_RIIC0_WaitTransmitEnd
Userdef_RIIC1_WaitTransmitEnd
Userdef_RIIC3_WaitTransmitEnd
Userdef_RIIC0_InitTransmitEnd
Userdef_RIIC1_InitTransmitEnd
Userdef_RIIC3_InitTransmitEnd

Figure 5.27  RIIC Transmit Function (2/2)
5.9.16 RIIC Receive Function

Figure 5.28 to Figure 5.34 show the flowcharts of RIIC Receive Function.

```
R_RIIC_Read
    Argument error in function?
        Error in specification for channel
            return (DEVDRV_ERROR)
    Specifies byte count = 1 or 2 byte?
        More than 2 byte
            Set to hold SCL low from 9th to 1st clock cycle
                RIICnMR3 register
                    WAIT bit ← 1
                    : Enables NACK output when final byte is received, causes SCL to be
                    fixed at low-level at falling edge of 9th clock cycle when final byte is
                    received, and enables issuance of the stop condition.
            Specifies byte count = 1 byte?
                Set to send "NACK" to acknowledge bit
                    RIICnMR3 register
                        ACKWP bit ← 1
                        ACKBT bit ← 1
                        : Enables writing to ACKBT.
                        : Setting for transmitting NACK when final byte is received.
                RIICnDRR register
                    Reading the contents of the RIICnDRR register causes the SCL clock to be output and receive operation starts.
```

Figure 5.28 RIIC Receive Function (1/7)
channel = 0
Wait for satisfaction of the conditions for RIIC channel 0 receive data full notification information
Userdef_RIIC0_WaitReceiveFull()
Initialize RIIC channel 0 receive data full notification information
Userdef_RIIC0_InitReceiveFull()

channel = 1
Wait for satisfaction of the conditions for RIIC channel 1 receive data full notification information
Userdef_RIIC1_WaitReceiveFull()
Initialize RIIC channel 1 receive data full notification information
Userdef_RIIC1_InitReceiveFull()

channel = 2
Wait for satisfaction of the conditions for RIIC channel 2 receive data full notification information
Userdef_RIIC2_WaitReceiveFull()
Initialize RIIC channel 2 receive data full notification information
Userdef_RIIC2_InitReceiveFull()

channel = 3
Wait for satisfaction of the conditions for RIIC channel 3 receive data full notification information
Userdef_RIIC3_WaitReceiveFull()
Initialize RIIC channel 3 receive data full notification information
Userdef_RIIC3_InitReceiveFull()

default

Set to send “NACK” to acknowledge bit
RIICnMR3 register
ACKWP bit ← 1 : Enables writing to ACKBT.
ACKBT bit ← 1 : Setting for transmitting NACK when final byte is received.

*buffer ← RIICnDRR register
After the RIICnDRR register is read, the value is written to the pointer indicated by the argument buffer.

Figure 5.29  RIIC Receive Function (2/7)
Figure 5.30  RIIC Receive Function (3/7)
Set to hold SCL low from 9th to 1st clock cycle

Acquire 1 byte of receive data

Processing when 3 bytes of receive data remain

channel = 0

Wait for satisfaction of the conditions for RIIC channel 0 receive data full notification information
Userdef_RIIC0_WaitReceiveFull()

Initialize RIIC channel 0 receive data full notification information
Userdef_RIIC0_InitReceiveFull()

channel = 1

Wait for satisfaction of the conditions for RIIC channel 1 receive data full notification information
Userdef_RIIC1_WaitReceiveFull()

Initialize RIIC channel 1 receive data full notification information
Userdef_RIIC1_InitReceiveFull()

channel = 2

Wait for satisfaction of the conditions for RIIC channel 2 receive data full notification information
Userdef_RIIC2_WaitReceiveFull()

Initialize RIIC channel 2 receive data full notification information
Userdef_RIIC2_InitReceiveFull()

channel = 3

Wait for satisfaction of the conditions for RIIC channel 3 receive data full notification information
Userdef_RIIC3_WaitReceiveFull()

Initialize RIIC channel 3 receive data full notification information
Userdef_RIIC3_InitReceiveFull()

default

Wait until the conditions for the RIIC receive data full notification information to be satisfied, and initializes the notification information.

In the sample code, the user-defined function Userdef_RIIC2_WaitReceiveFull is used to wait until the RIIC channel 2 receive data full notification flag is set, then the notification flag is cleared by using the user-defined function Userdef_RIIC2_InitReceiveFull.

The receive data full notification flag is cleared when a start condition or a restart condition is issued and set by the receive data full interrupt processing.

User-defined functions have been prepared for the RIIC channels, but following functions are not used in the sample code.
Userdef_RIIC0_WaitReceiveFull
Userdef_RIIC1_WaitReceiveFull
Userdef_RIIC3_WaitReceiveFull
Userdef_RIIC0_InitReceiveFull
Userdef_RIIC1_InitReceiveFull
Userdef_RIIC3_InitReceiveFull

*buffer ← RIICnDRR register
After the RIICnDRR register is read, the value is written to the pointer indicated by the argument buffer.

Figure 5.31 RIIC Receive Function (4/7)
Figure 5.32 RIIC Receive Function (5/7)
Figure 5.33  RIIC Receive Function (6/7)
return (DEVDRV_SUCCESS)

channel = 0

Wait for RIIC channel 0 stop condition detection flag to be set
Userdef_RIIC0_WaitStop()

channel = 1

Wait for RIIC channel 1 stop condition detection flag to be set
Userdef_RIIC1_WaitStop()

channel = 2

Wait for RIIC channel 2 stop condition detection flag to be set
Userdef_RIIC2_WaitStop()

channel = 3

Wait for RIIC channel 3 stop condition detection flag to be set
Userdef_RIIC3_WaitStop()

default

Initialize flags for next communication
RIICnSR2 register
NACKF bit ← 0 : Clears the NACK receive flag.
STOP bit ← 0 : Clears the STOP condition detect flag.

RIICnMR3 register
WAIT bit ← 0 : Cancels fixing SCL output low-level from the 9th to the 1st clock cycle.

Waits for the RIIC stop condition detection flag (STOP bit) to be set.

In the sample code, the user-defined function Userdef_RIIC2_WaitStop is used to wait for the RIIC channel 2 stop condition detection flag (STOP bit) to be set.

User-defined functions have been prepared for the RIIC channels, but the following functions are not used in the sample code:
Userdef_RIIC0_WaitStop
Userdef_RIIC1_WaitStop
Userdef_RIIC3_WaitStop

Cancel holding SCL low from 9th to 1st clock cycle

Figure 5.34  RIIC Receive Function (7/7)
5.9.17 RIIC Dummy Read Function
Figure 5.35 shows the flowchart of RIIC Dummy Read Function.

Figure 5.35  RIIC Dummy Read Function
5.9.18 Checking Function of RIIC Stop Condition Detection

Figure 5.36 shows the flowchart of Checking Function of RIIC Stop Condition Detection.

Figure 5.36  Checking Function of RIIC Stop Condition Detection

R_RIIC_DetectStop

Error in specification for channel

Argument error in function?

return (DEVDRV_ERROR)

Channel?

channel = 0

Wait for RIIC channel 0 stop condition detection flag to be set
Userdef_RIIC0_WaitStop()

channel = 1

Wait for RIIC channel 1 stop condition detection flag to be set
Userdef_RIIC1_WaitStop()

channel = 2

Wait for RIIC channel 2 stop condition detection flag to be set
Userdef_RIIC2_WaitStop()

channel = 3

Wait for RIIC channel 3 stop condition detection flag to be set
Userdef_RIIC3_WaitStop()

default

Initialize flags for next communication

RIIcnSR2 register
NACKF bit ← 0 : Clears the NACK receive flag.
STOP bit ← 0 : Clears the STOP condition detection flag.

return (DEVDRV_SUCCESS)
5.9.19 Clear Function of RIIC NACK Receive Flag

Figure 5.37 shows the flowchart of Clear Function of RIIC NACK Receive Flag.

![Flowchart of Clear Function of RIIC NACK Receive Flag](image)

Figure 5.37 Clear Function of RIIC NACK Receive Flag
5.9.20 RIIC Transmit Data Empty Interrupt Notification Function

Figure 5.38 shows the flowchart of RIIC Transmit Data Empty Interrupt Notification Function.

The Edge sense is used as the source of the RIIC transmit data empty interrupt, therefore the source flag is not cleared.

Figure 5.38   RIIC Transmit Data Empty Interrupt Notification Function
5.9.21 RIIC Transmit-End Interrupt Notification Function

Figure 5.39 shows the flowchart of RIIC Transmit-End Interrupt Notification Function.

```
R_RIIC_TeiInterrupt

Function argument error?

 Illegal specification for channel

   return (DEVDRV_ERROR)

 Channel?

 channel = 0

 Set to satisfy the conditions for RIIC channel 0 transmit-end notification information
 Userdef_RIIC0_SetTransmitEnd()

 channel = 1

 Set to satisfy the conditions for RIIC channel 1 transmit-end notification information
 Userdef_RIIC1_SetTransmitEnd()

 channel = 2

 Set to satisfy the conditions for RIIC channel 2 transmit-end notification information
 Userdef_RIIC2_SetTransmitEnd()

 channel = 3

 Set to satisfy the conditions for RIIC channel 3 transmit-end notification information
 Userdef_RIIC3_SetTransmitEnd()

 default

 Clear transmit-end flag

 RIIcnSR2 register
 TEND bit ← 0 : Clears the transmit-end flag.
 After TEND bit is cleared, another read (dummy read) of TEND bit is performed.

 return (DEVDRV_SUCCESS)
```

In the sample code, the user-defined function Userdef_RIIC2_SetTransmitEnd is used to set the flag to notify that the RIIC channel 2 transmit-end interrupt has been generated. The transmit data-empty notification flag is cleared when a start condition or a restart condition is issued after waiting for the conditions for the transmit-end notification information to be satisfied during RIIC transmission.

User-defined functions have been prepared for RIIC channels, but the following functions are not used in the sample code.

- Userdef_RIIC0_SetTransmitEnd
- Userdef_RIIC1_SetTransmitEnd
- Userdef_RIIC3_SetTransmitEnd

Figure 5.39   RIIC Transmit-End Interrupt Notification Function
5.9.22 RIIC Receive Data Full Interrupt Notification Function

Figure 5.40 shows the flowchart of RIIC Receive Data Full Interrupt Notification Function.

The Edge sense is used as the source of the RIIC receive data full interrupt, therefore the source flag is not cleared.

![Flowchart of RIIC Receive Data Full Interrupt Notification Function]

Figure 5.40   RIIC Receive Data Full Interrupt Notification Function

Provides settings to satisfy the conditions for the RIIC receive data full notification information.

In the sample code, the user-defined function Userdef_RIIC2_SetReceiveFull is used to set the flag to notify that the RIIC channel 2 receive data full interrupt has been generated. The receive data full notification flag is cleared when a start condition or a restart condition is issued after waiting for the conditions for the receive data full notification information to be satisfied during RIIC reception.

User-defined functions have been prepared for RIIC channels, but the following functions are not used in the sample code.

Userdef_RIIC0_SetReceiveFull
Userdef_RIIC1_SetReceiveFull
Userdef_RIIC3_SetReceiveFull
5.9.23 RIIC Channel 2 Initial Setting Function

Figure 5.41 and Figure 5.42 show the flowcharts of RIIC Channel 2 Initial Setting Function.

The API functions are used to register the RIIC channel 2 interrupt function as the INTC handler, to set the interrupt priority level, and to enable the interrupts. Refer to the RZ/A1H group Example of Initialization application note for more details about the INTC interrupt API functions R_INTC_RegistIntFunc, R_INTC_SetPriority, and R_INTC_Enable.

User-defined functions for other channels of the RIIC are prepared, but these are not used in the sample code. When other channel except channel 2 is applied, the function of the channel used should be executed to initialize the receive data full notification information.

<table>
<thead>
<tr>
<th>Userdef_RIIC2_Init</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancel module standby for RIIC channel 2</td>
<td>STBCR9 register MSTP95 bit ← 0 : Supplies a clock to RIIC channel 2.</td>
</tr>
<tr>
<td>Reset RIIC channel 2</td>
<td>RIIC2CR1 register ICE bit ← 0 : Sets the RIIC2SCL and RIIC2SDA pins to the non-driven state.</td>
</tr>
<tr>
<td>Select internal reference clock</td>
<td>RIIC2CR1 register IICRST bit ← 1 : Resets all registers and internal state of the RIIC (RIIC reset).</td>
</tr>
<tr>
<td>Set SCL clock low width and high width</td>
<td>IICRST bit ← 1 : Resets RIIC channel 2. Sets the SCL and SDA pins to the non-driven state (internal reset).</td>
</tr>
<tr>
<td>Set digital noise filter stage count</td>
<td>RIIC2MR1 register</td>
</tr>
<tr>
<td>Enable to abort transfer in NACK reception and use digital noise filter</td>
<td>RIIC2BRL register BRL bit ← 14 : Sets 14 × IICf clock cycles as the as the SCL clock low width.</td>
</tr>
<tr>
<td>Enable interrupts on RIIC channel 2</td>
<td>RIIC2BRH register</td>
</tr>
<tr>
<td>Releasel internal reset</td>
<td>RIIC2MR3 register NF[1:0] bits ← B’11 : Sets the filter stage count to four stages.</td>
</tr>
</tbody>
</table>

Note: The RIIC transfer speed and duty ratio can be calculated from the internal reference clock, digital noise filter stage count, and SCL clock low and high widths. In the sample code, the transfer speed and the duty ratio are set to be 100 kbps 0.524 respectively.

Figure 5.41 RIIC Channel 2 Initial Setting Function (1/2)
Enable INTIICCRI2 interrupt
R_INTC_Enable()

Register INTIICCRI2 interrupt handler
R_INTC_RegistIntFunc()

Set INTIICCRI2 interrupt priority level
R_INTC_SetPriority()

Enables the RIIC channel 2 receive data full interrupt.

Enables the RIIC channel 2 transmit data empty interrupt.

Enables the RIIC channel 2 transmit-end interrupt.

Register INTIICCTI2 interrupt handler
R_INTC_RegistIntFunc()

Set INTIICCTI2 interrupt priority level
R_INTC_SetPriority()

Registers the RIIC channel 2 transmit data empty interrupt handler.
In the sample code, the handler function Sample_RIIC_Ti2_Interrupt to notify the transmit data-empty state is registered.

Sets the priority level of the RIIC channel 2 transmit data empty interrupt.
In the sample code, the interrupt priority level is set to "9".

Sets the priority level of the RIIC channel 2 transmit-end interrupt.
In the sample code, the interrupt priority level is set to "9".

Sets the priority level of the RIIC channel 2 transmit-end interrupt.
In the sample code, the interrupt priority level is set to "9".

Figure 5.42 RIIC Channel 2 Initial Setting Function (2/2)
5.9.24 Initialization Function of RIIC Channel 2 Receive Data Full Notification Information

Figure 5.43 shows the flowchart of Initialization Function of RIIC Channel 2 Receive Data Full Notification Information.

User-defined functions for other channels of the RIIC are prepared, but these are not used in the sample code. When other channel except channel 2 is applied, the function of the channel used should be executed to initialize the receive data full notification information.

```c
Userdef_RIIC2_InitReceiveFull

Clear RIIC register channel 2 receive data full notification flag

Clears the flag to notify that the RIIC channel 2 receive data full interrupt has been generated.

return
```

Figure 5.43 Initialization Function of RIIC Channel 2 Receive Data Full Notification Information

5.9.25 Initialization Function of RIIC Channel 2 Transmit Data Empty Notification Information

Figure 5.44 shows the flowchart of Initialization Function of RIIC Channel 2 Transmit Data Empty Notification Information.

User-defined functions for other channels of the RIIC are prepared, but these are not used in the sample code. When other channel except channel 2 is applied, the function of the channel used should be executed to initialized the transmit data empty notification information.

```c
Userdef_RIIC2_InitTransmitEmpty

Clear RIIC channel 2 transmit data empty notification flag

Clears the flag to notify that the RIIC channel 2 transmit data empty interrupt has been generated.

return
```

Figure 5.44 Initialization Function of RIIC Channel 2 Transmit Data Empty Notification Information
5.9.26 Initialization Function of RIIC Transmit-End Notification Information

Figure 5.45 shows the flowchart of Initialization Function of RIIC Transmit-End Notification Information.

When transmission finishes of a slave address with the R/W# bit cleared to "0" (write operation), the transmit-end flag (TEND) is set to "1", and the sample code sets the transmit-end notification information flag to "1". This function is then used to clear the transmit-end notification information flag to "0". When transmission finishes of a slave address with the R/W# bit set to "1" (read operation), the receive data-full flag (RDRF) is set to "1", and the sample code sets the receive data-full notification information flag to "1". This function is then used to clear the receive data-full notification information flag to "0".

User-defined functions for other channels of the RIIC are prepared, but these are not used in the sample code. When other channel except channel 2 is applied, the function of the channel used should be executed to initialized the transmit data empty notification information.

Figure 5.45 Initialization Function of RIIC Transmit-End Notification Information
5.9.27 Setting Function to Satisfy the Conditions for RIIC Channel 2 Receive Data Full Notification Information

Figure 5.46 shows the flowchart of Setting Function to Satisfy the Conditions for RIIC Channel 2 Receive Data Full Notification Information.

User-defined functions for other channels of the RIIC are prepared, but these are not used in the sample code. When other channel except channel 2 is applied, the function of the channel used should be executed to satisfy the conditions for the receive data full notification information.

```
Userdef_RIIC2_SetReceiveFull

Sets the RIIC channel 2 receive data full notification flag

Sets the flag to notify that the RIIC channel 2 receive data full interrupt has been generated.

return
```

Figure 5.46 Setting Function to Satisfy the Conditions for RIIC Channel 2 Receive Data Full Notification Information

5.9.28 Setting Function to Satisfy the Conditions for RIIC Channel 2 Transmit Data Empty Notification Information

Figure 5.47 shows the flowchart of Setting Function to Satisfy the Conditions for RIIC Channel 2 Transmit Data Empty Notification Information.

User-defined functions for other channels of the RIIC are prepared, but these are not used in the sample code. When other channel except channel 2 is applied, the function of the channel used should be executed to satisfy the conditions for the transmit data empty notification information.

```
Userdef_RIIC2_SetTransmitEmpty

Sets the RIIC channel 2 transmit data empty notification flag

Sets the flag to notify that the RIIC channel 2 transmit data empty interrupt has been generated.

return
```

Figure 5.47 Setting Function to Satisfy the Conditions for RIIC Channel 2 Transmit Data Empty Notification Information
5.9.29 Setting Function to Satisfy the Conditions for RIIC Channel 2 Transmit-End Notification Information

Figure 5.48 shows the flowchart of Setting Function to Satisfy the Conditions for RIIC Channel 2 Transmit-End Notification Information.

User-defined functions for other channels of the RIIC are prepared, but these are not used in the sample code. When other channel except channel 2 is applied, the function of the channel used should be executed to satisfy the conditions for the transmit-end notification information.

Userdef_RIIC2_SetTransmitEnd

Sets the flag to notify that the RIIC channel 2 transmit-end interrupt has been generated.

Figure 5.48 Setting Function to Satisfy the Conditions for RIIC Channel 2 Transmit-End Notification Information
5.9.30 Wait Function for Satisfaction of the Conditions for RIIC Channel 2 Receive Data Full Notification Information

Figure 5.49 shows the flowchart of Wait Function for Satisfaction of the Conditions for RIIC Channel 2 Receive Data Full Notification Information.

User-defined functions for other channels of the RIIC are prepared, but these are not used in the sample code. When other channel except channel 2 is applied, the function of the channel used should be executed to wait until the conditions for the receive data full notification information to be satisfied.

```
Userdef_RIIC2_WaitReceiveFull

Wait for RIIC channel 2 receive data full notification flag to be set

Waits for the flag to notify that the RIIC channel 2 receive data full interrupt has been generated to be set.

return
```

Figure 5.49 Wait Function for Satisfaction of the Conditions for RIIC Channel 2 Receive Data Full Notification Information

5.9.31 Wait Function for Satisfaction of the Conditions for RIIC Channel 2 Transmit Data Empty Notification Information

Figure 5.50 shows the flowchart of Wait Function for Satisfaction of the Conditions for RIIC Channel 2 Transmit Data Empty Notification Information.

User-defined functions for other channels of the RIIC are prepared, but these are not used in the sample code. When other channel except channel 2 is applied, the function of the channel used should be executed to wait until the conditions for the transmit data empty notification information to be satisfied.

```
Userdef_RIIC2_WaitTransmitEmpty

Wait for RIIC channel 2 transmit data empty notification flag to be set

Waits for the flag to notify that the RIIC channel 2 transmit data empty interrupt has been generated to be set.

return
```

Figure 5.50 Wait Function for Satisfaction of the Conditions for RIIC Channel 2 Transmit Data Empty Notification Information
5.9.32 Wait Function for Satisfaction of the Conditions for RIIC Channel 2 Transmit-End Notification Information

Figure 5.51 shows the flowchart of Wait Function for Satisfaction of the Conditions for RIIC Channel 2 Transmit-End Notification Information.

When transmission finishes of a slave address with the R/W# bit cleared to "0" (write operation), the transmit-end flag (TEND) is set to "1", and the sample code sets the transmit-end notification information flag to "1". This function is then used to wait for the transmit-end notification information flag to be set to "1". When transmission finishes of a slave address with the R/W# bit set to "1" (read operation), the receive data-full flag (RDRF) is set to "1", and the sample code sets the receive data-full notification information flag to "1". This function is then used to wait for the receive data-full notification information flag to be set to "1".

User-defined functions for other channels of the RIIC are prepared, but these are not used in the sample code. When other channel except channel 2 is applied, the function of the channel used should be executed to wait until the conditions for the transmit-end notification information to be satisfied.

Figure 5.51   Wait Function for Satisfaction of the Conditions for RIIC Channel 2 Transmit-End Notification Information
5.9.33 Wait Function for RIIC Channel 2 Bus Free and Bus Busy
Figure 5.52 shows the flowchart of Wait Function for RIIC Channel 2 Bus Free and Bus Busy. User-defined functions for other channels of the RIIC are prepared, but these are not used in the sample code. When other channel except channel 2 is applied, the function of the channel used should be executed to wait for the bus free and bus busy state.

```
Userdef_RIIC2_WaitBusMastership

Mode?

Wait for bus free mode

Wait for bus free

Wait for bus busy mode

Wait for bus busy

RIIC2CR2 register
Reads the BBSY bit.
Wait for bus free: Loops until the BBSY bit is cleared to "0".
Wait for bus busy: Loops until the BBSY bit is set to "1".

return
```

Figure 5.52 Wait Function for RIIC Channel 2 Bus Free and Bus Busy

5.9.34 Wait Function for RIIC Channel 2 Stop Condition Detection
Figure 5.53 shows the flowchart of Wait Function for RIIC Channel 2 Stop Condition Detection. User-defined functions for other channels of the RIIC are prepared, but these are not used in the sample code. When other channel except channel 2 is applied, the function of the channel used should be executed to wait until the stop condition to be detected.

```
Userdef_RIIC2_WaitStop

Wait for stop condition detection

RIIC2SR2 register
Reads the STOP bit.
Loops until the STOP bit is set to "1" (stop condition detected).

return
```

Figure 5.53 Wait Function for RIIC Channel 2 Stop Condition Detection
### 5.10 Running Sample Code

The sample code is operated by entering commands in the terminal program running on the host PC connected to the GENMAI board via the serial interface.

After supplying power to the GENMAI board, the message (1) in Figure 5.54 is output. To run the RIIC sample code, input "RIIC" + "Enter" key subsequent to the "SAMPLE>" prompt. When the message (2) in Figure 5.54 is output. Input "1" + "Enter" key subsequent to the "RIIC SAMPLE>" prompt to run the RIIC sample code.

By inputting "HELP" + "Enter" key, the sample code information (3) is displayed. "EXIT" + "Enter" key terminates the RIIC sample code operation.

Ver.X.XX and Ver.Y.YY shown in Figure 5.54 indicate the main processing version of the sample code and the RIIC sample code version respectively.

<table>
<thead>
<tr>
<th>Display messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>RZ/A1H CPU Board Sample Program. Ver.X.XX</td>
</tr>
<tr>
<td>Copyright (C) 2015 Renesas Electronics Corporation. All rights reserved.</td>
</tr>
<tr>
<td>select sample program.</td>
</tr>
<tr>
<td>SAMPLE&gt;</td>
</tr>
<tr>
<td>RIIC SAMPLE&gt; help</td>
</tr>
<tr>
<td>1 : EEPROM access</td>
</tr>
<tr>
<td>- Bit rate : 100kbps</td>
</tr>
<tr>
<td>EXIT : Exit RIIC sample</td>
</tr>
<tr>
<td>RIIC SAMPLE&gt;</td>
</tr>
</tbody>
</table>

**Figure 5.54 Terminal Display at RIIC Sample Code Startup**
6. Sample Code
Sample code can be downloaded from the Renesas Electronics website.

7. Reference Documents
User's Manual: Hardware
RZ/A1H Group User's Manual: Hardware
The latest version can be downloaded from the Renesas Electronics website.

R7S72100 RTK772100BC0000BR (GENMAI) User's Manual
The latest version can be downloaded from the Renesas Electronics website.

The latest version can be downloaded from the ARM website.

ARM Generic Interrupt Controller Architecture Specification Architecture version 1.0
The latest version can be downloaded from the ARM website.

ARM Cortex™-A9 (Revision: r3p0) Technical Reference Manual
The latest version can be downloaded from the ARM website.

ARM CoreLink™ Level 2 Cache Controller L2C-310 (Revision: r3p2) Technical Reference Manual
The latest version can be downloaded from the ARM website.

Technical Update/Technical News
The latest information can be downloaded from the Renesas Electronics website.

ARM software development tool (ARM Compiler toolchain, ARM DS-5, etc.) can be downloaded from the ARM website
The latest version can be downloaded from the ARM website.
Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

All trademarks and registered trademarks are the property of their respective owners.
<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rev.1.00</td>
<td>Jul. 11, 2014</td>
<td>-</td>
<td>First edition issued</td>
</tr>
<tr>
<td>Rev.1.01</td>
<td>Aug. 28, 2015</td>
<td>P13</td>
<td>Reflected the update contents of &quot;RZ/A1H Group Example of Initialization Rev.1.01&quot; application note.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P14</td>
<td>• Table 5.2  Sections Used (1/2) Changed the execution area for the CODE_IO_REGRW section in the table from &quot;FLASH&quot; to &quot;LRAM&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P15</td>
<td>• Table 5.3  Sections Used (2/2) Added the CODE_CACHE_OPERATION section to the table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Figure 5.5   Section Assignment Provided information due to the CODE_IO_REGRW section arrangement being changed and the CODE_CACHE_OPERATION section being added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In the processing of the R_RIIC_Write function, added operations to control to disable or enable the transmit end interrupt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P29</td>
<td>• 5.8 Function Specifications R_RIIC_Write Added explanations of operations to control to disable or enable the transmit end interrupt to the description of the function.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P65</td>
<td>• Figure 5.26  RIIC Transmit Function (1/2) Added an operation to disable the transmit end interrupt requests before transmission is started.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P66</td>
<td>• Figure 5.27  RIIC Transmit Function (2/2) Added an operation to enable the transmit end interrupt requests after the last data for transmission is written to the RIIcnDRT register.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In the processing of the R_RIIC_Read function, added operations to control the WAIT bit when a value 2 is specified for the argument byte.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P30</td>
<td>• 5.8 Function Specifications R_RIIC_Read Added explanations of operations when a value 1 or 2 is specified for the argument byte to the description of the function.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P67</td>
<td>• Figure 5.28  RIIC Receive Function Added operations in the case when a value 2 is specified for the argument byte.</td>
</tr>
</tbody>
</table>
General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins
   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.
   • The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   • The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   • The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable.
   When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   • When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products
   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.
   • The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.
Notice

1. Descriptions of circuits, software, and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics uses 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.

3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied, or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.

5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High-Quality.” The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below:

   - “Standard”: Computers, office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools, personal electronic equipment; and industrial robots etc.
   - “High-Quality”: Transport equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.

Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.

6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.

7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.

8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that require the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such law and regulations.

10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.

11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.

12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product” means any product developed or manufactured by or for Renesas Electronics.