To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

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

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)
Send any inquiries to http://www.renesas.com/inquiry.
Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.

2. 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.

3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.

4. 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.

5. When exporting the 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 laws and regulations. You should not use Renesas Electronics products or the 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. 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.

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

7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depend on the product’s quality grade, as indicated below. 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 categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. 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 an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

- “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.
- “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
- “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.

8. 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.

9. 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 them 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 any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.

10. 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 regulate 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.

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(s)” means any product developed or manufactured by or for Renesas Electronics.
H8S Family
Master Transmission/Slave Reception Example Using IEBus Controller

Introduction
Unit A performs master transmission and slave reception between other units using the IEBus controller included in the H8S/2258F.

Target Device
H8S/2258F

Contents
1. Specifications ......................................................................................................................... 2
2. Principles of Operation ......................................................................................................... 7
3. Hardware Configuration ...................................................................................................... 18
4. Description of Software ..................................................................................................... 19
5. Flowcharts ........................................................................................................................ 26
6. Program Listings ............................................................................................................... 38
1. Specifications

1.1 Overview

The configuration of the IEBus connections is shown in figure 1. This application note describes a software example for master transmission and slave reception in which unit A uses the IEBus controller of the H8S/2258F and units B and C are virtual units. Note that the IEBus interface of virtual units B and C is controlled using functions equivalent to those of the IEBus controller of the H8S/2258F. Furthermore, unit A uses the HA12187FP as a driver and transceiver for the IEBus interface.

Figure 1  Configuration of IEBus Connections

1.2 Usage Conditions of IEBus

The IEBus controller for the units operates under the following conditions.

(1) System clock: 12.58 MHz
(2) Communications mode: Mode 1
(3) Control bits in IEMCR: Always fixed at H'0F (used for master transmission and slave reception only)
(4) Number of retransmissions in case of arbitration loss: 3
(5) Local addresses: Unit A = H'AAA
  Unit B = H'BBB
  Unit C = H'B0
1.3 Master Transmission/Slave Reception Operation

The IEBus controller of the H8S/2258F uses the data transfer controller (DTC) to transmit and receive data during master transmission and slave reception. However, during master transmission the first byte of data is written to the transmit buffer by software processing, without using the DTC. For details, see section 2.3, Error Handling During Master Transmission.

![Data Transmission and Reception Using DTC](image-url)

**Figure 2 Data Transmission and Reception Using DTC**
1.4 Specifications of Communication Operations

Unit A uses master transmission and slave reception to perform the communication operations described below. Note that each unit may begin master transmission arbitrarily. Consequently, CSMA/CD access control is used as the basis for determining which unit has exclusive control of the bus in cases where more than one unit attempts to initiate master transmission simultaneously.

1.4.1 Master Transmission

Three frames of data are transmitted to each unit.

(1) Normal communication (broadcast bit = 1, slave address = H'BBB) is used to transmit 32 bytes of data to unit B. The 32 bytes of data transmitted consists of H'11.

(2) Simultaneous broadcast communication (broadcast bit = 0, slave address = H'FFF) is used to transmit 32 bytes of data to units B and C. The 32 bytes of data transmitted consists of H'22.

(3) Group broadcast communication (broadcast bit = 0, slave address = H'BBB) is used to transmit 32 bytes of data to units B and C. The 32 bytes of data transmitted consists of H'33.

Figure 3 Master Transmission Operation of Unit A
1.4.2 Slave Reception

Three frames of data to be received are transmitted from each unit.

(1) 32 bytes of data transmitted from unit B using normal communication (broadcast bit = 1, slave address = H'AAA) is received. The data values are random.
(2) 32 bytes of data transmitted from unit B using simultaneous broadcast communication (broadcast bit = 0, slave address = H'FFF) is received. The data values are random.
(3) 32 bytes of data transmitted from unit C using group broadcast communication (broadcast bit = 0, slave address = H'AAA) is received. The data values are random.

![Figure 4 Slave Reception Operation of Unit A](image-url)
1.5 Specifications of Communication Error Handling

Unit A performs the following error handling operations.

(1) Errors during Master Transmission

The IEBus controller of the H8S/2258F detects the errors listed below. When an error occurs, the frame that was being transmitted is discarded and retransmitted. Retransmission is repeated until transmission completes normally.

- Arbitration Loss (AL)
  Occurs when arbitration is lost three times.

- Underrun Error (UE)
  Occurs during data transmission when no data is waiting in the transmit buffer.

- Timing Error (TTME)
  Occurs when data is not transferred according to the timing specified by the IEBus protocol.

- Transmit Frame Over Maximum Number of Transfer Bytes (RO)
  Occurs when data equivalent to the maximum byte length defined by the communications mode has been transmitted but the transmission has not completed.

- Acknowledge Error (ACK)
  Condition 1: A NAK was received before transmission of the data field.
  Condition 2: During transmission, data equivalent to the maximum byte length defined by the communications mode has been transmitted but no ACK has been received.

(2) Errors during Slave Reception

The IEBus controller of the H8S/2258F detects the errors listed below. When an error occurs, the frame that was being received is discarded. After discarding the frame, the controller returns to ready-to-receive state for the next frame.

- Overrun Error (OVE)
  Occurs when the next data is received before the data in the receive buffer has been read.

- Timing Error (RTME)
  Occurs when data is not transferred according to the timing specified by the IEBus protocol.

- Receive Frame Over Maximum Number of Transfer Bytes (DLE)
  Occurs when data equivalent to the maximum byte length defined by the communications mode has been received but reception has not completed.

- Parity Error (PE)
  Occurs when a parity error cannot be cleared even after reception of a data field has proceeded up to the maximum byte length defined by the communications mode.
2. Principles of Operation

The principles of operation for master transmission and slave reception are illustrated below.

2.1 Master Transmission Operation

The principles of operation of the IEBus controller during master transmission are illustrated in figure 5, and a flowchart of master transmission operation is shown in figure 6. The DTC is used to transmit data fields. However, during master transmission the first byte of data is written to the transmit buffer by software processing, without using the DTC. For an explanation of the reasons for this, see section 2.3, Error Handling during Master Transmission.

---

**Figure 5  Principles of Operation during Master Transmission**

- **IECMR**
- **IEFLG**
- **CMX**
- **MRQ**
- **IETSU**
- **TxRDY**
- **TxS**
- **TxF**
- **Interrupt**
- **IETxI(TxRDY) (To DTC)**
- **IETxI(TxRDY) (To CPU)**
- **IETS1 (To CPU)**

<table>
<thead>
<tr>
<th>Communication frame</th>
<th>MA</th>
<th>SA</th>
<th>CF</th>
<th>LF</th>
<th>D</th>
</tr>
</thead>
</table>

| Registers of IEBus controller | IEAR1, IEAR2 | IESA1, IESA2 | CTL3 to CTL0 of IEMCR | IETBFL | IETBR (transmit buffer) |

S: Start bit, broadcast bit
MA: Master address field
SA: Slave address field
CF: Control field
LF: Message length field
D1, D2,..., Dn-1, Dn: Data fields
Master transmission operation

Initialize IEBus controller

Determine new transmit frame

1. Prepare for master transmission
   • Write first byte of transmission data to IETBR
   • Clear RxRDY flag
   • Set slave address in IESA1 and IESA2
   • Set control field in IEMCR
   • Set message length in IETBFL
   • Set broadcast/normal mode in SS bit of IEMCR

DTC settings

2. Set address where 2nd byte of transmission data is stored in SAR
   • Set value of transmit message length - 1 in CRA

3. Set master transmission request command in IECMR

Transmit start interrupt (TxS)

4. Enable ready-to-transmit interrupt (TxRDY)
   • Activate DTC with TxRDY startup source

5. Transfer transmission data to IETBR using DTC

Transmit error interrupt (TxE)

• Master transmission error handling

Communication error generated?

NO

Transfer by DTC completed?

NO

YES

Ready-to-transmit interrupt (TxRDY)

• Halt ready-to-transmit interrupt (TxRDY)
• Halt DTC operation

Transmission completion interrupt (TxF)

• Recognition that transmission completed normally

End transmission of 1 frame

Figure 6  Flowchart of Master Transmission Operation
2.2 Slave Reception Operation

The principles of operation of the IEBus controller during slave reception are illustrated in figure 7, and a flowchart of slave reception operation is shown in figure 8. The DTC is used to receive data fields.

![Diagram of Slave Reception](image)

Figure 7 Principles of Operation during Slave Reception

- S: Start bit, broadcast bit
- MA: Master address field
- SA: Slave address field
- CF: Control field
- LF: Message length field
- D1, D2, .... Dn - 1, Dn: Data fields
Slave reception operation

1. Initialize IEBus controller

   Receive start interrupt (RxS)
   - Store received master address (IEMA1, IEMA2)
   - Store received control field (IERBFL)
   - Store received message length (IERBFL)
   - Verify broadcast or normal communication from RSS and GG bits in IEFGL

   DTC settings
   - Set address for storing received data in DAR
   - Set value of receive message length in CRA
   - Activate DTC with RxRDY startup source

2. Transfer received data to storage destination using DTC

3. Communication error generated?

   Receive error interrupt (RxE)
   - Receive error handling

4. Transfer by DTC completed?

   YES

5. Ready-to-receive interrupt (RxRDY)
   - Halt ready-to-receive interrupt (RxRDY)
   - Halt DTC operation

6. Receive completion interrupt (RxF)
   - Recognition that reception completed normally

End reception of 1 frame

Figure 8 Flowchart of Slave Reception Operation
2.3 Error Handling during Master Transmission

Operation when a timing error is generated is illustrated in figure 9. When a timing error or other error occurs during data transmission ([1]), in some cases the DTC will already have sent the next transmission data to the transmit buffer and the TxRDY flag, which is the startup source of the DTC, will already have been cleared ([2]).

Performing a retransmission in this state would cause the data remaining in the transmit buffer (the data from the previous frame) to be transmitted as the first byte of the data field ([3]).

In this application note, during master transmission the first byte of data in the data field is written to the transmit buffer by software processing, without using the DTC, to avoid the problem described above. The second and subsequent bytes are then transferred using the DTC.

In this sample task the settings for the DTC’s SAR (DTC transfer source address register) and CRA (DTC transfer count register A) are as follows:

- Address in internal memory for storing second and subsequent bytes of data: SAR
- Number of data bytes specified in messages minus 1: CRA

If an error occurs during data transmission, the IEBus controller returns to ready-to-transmit state from standby state. Furthermore, DTC transfer processing is halted when the following errors occur.
(1) Handling of Arbitration Loss (AL) Errors

The handling of arbitration loss errors is illustrated in figure 10.
[1] When arbitration loss occurs three times the IEBus controller sets the AL flag and transitions to the wait state.
[2] A TxE interrupt is generated. The arbitration loss error is detected within this interrupt.
[3] The AL flag is cleared within the interrupt.

---

(2) Handling of Underrun Errors (UE)

The handling of underrun errors is illustrated in figure 11.
[1] When an underrun error occurs the IEBus controller sets the UE flag and transitions to the wait state.
[2] A TxE interrupt is generated. The underrun error is detected within this interrupt.
[3] The UE flag is cleared within the interrupt.
(3) Handling of Timing Errors (TTME)

The handling of timing errors is illustrated in figure 12.

1. When a timing error occurs, the IEBus controller sets the TTME flag and transitions to the wait state.
2. A TxE interrupt is generated. The timing error is detected within this interrupt.
3. The TTME flag is cleared within the interrupt.
4. Retransmission takes place.

![Figure 12 Handling of Timing Errors](image-url)

(4) Handling of Transmit Frame Over Maximum Number of Transfer Bytes (RO) and Acknowledge (ACK) Errors

The handling of transmit frame over maximum number of transfer bytes and acknowledge errors is illustrated in figure 13. Note that no acknowledge errors are generated during broadcast communication.

1. When a transmit frame over maximum number of transfer bytes error or an acknowledge error occurs, the IEBus controller sets the RO or ACK flag, respectively, and transitions to the wait state.
2. A TxE interrupt is generated. Within this interrupt, error is detected by monitoring both the RO and ACK flags, taking into account the possibility of a transmit frame over maximum number of transfer bytes error or an acknowledge error occurring independently (either an acknowledge error preceding transmission of the data field, or a transmit frame over maximum number of transfer bytes error occurring alone).
3. The RO and ACK flags are cleared within the interrupt.
4. Retransmission takes place.

![Figure 13 Handling of Transmit Frame over Maximum Number of Transfer Bytes and Acknowledge Errors](image-url)
2.4 Error Handling During Slave Reception

When an error occurs during slave reception, the frame that was being transmitted is discarded and the IEBus controller returns to ready-to-transmit/receive state from standby state. Furthermore, DTC transfer processing is halted when the following errors occur.

(1) Handling of Overrun Errors (OVE)

The handling of overrun errors is illustrated in figure 14.

- **Condition 1: Normal Communication [1] to [5]**
  1. An overrun error occurs. The IEBus controller sets the OVE flag.
  2. An RxE interrupt is generated. Within this interrupt it is determined whether the communications mode is normal or broadcast, and, if an overrun error is detected and the communications mode is normal, the OVE flag is not cleared until a receive frame over maximum number of transfer bytes error (DLE) has been generated. The IEBus controller transmits a NAK. (The received frame is discarded.)
  3. When the receive frame over maximum number of transfer bytes error is generated the IEBus controller sets the DLE flag and transitions to the wait state.
  4. An RxE interrupt is generated. The error state is detected within the interrupt and the DLE, OVE, and RxRDY flags are cleared.
  5. The IEBus controller returns to ready-to-transmit/receive state.

- **Condition 2: Broadcast Communication [6] to [8]**
  6. When an overrun error occurs the IEBus controller sets the OVE flag and transitions to the wait state.
  7. An RxE interrupt is generated. Within this interrupt it is determined whether the communications mode is normal or broadcast, and, if an overrun error is detected and the communications mode is broadcast, the OVE flag is cleared. Furthermore, the RxRDY flag is cleared.
  8. The IEBus controller returns to ready-to-transmit/receive state.
Figure 14 Handling of Overrun Errors
(2) Handling of Timing Errors (RTME)

The handling of timing errors is illustrated in figure 15.

[1] When a timing error occurs the IEBus controller sets the RTME flag and transitions to the wait state.

[2] An RxE interrupt is generated. The timing error is detected within this interrupt. Furthermore, the RTME flag is cleared.


---

**Figure 15 Handling of Timing Errors**

---

(3) Handling of Parity Errors (PE) and Receive Frame Over Maximum Number of Transfer Bytes Errors (DLE)

The handling of parity errors and receive frame over maximum number of transfer bytes errors is illustrated in figure 16.

[1] When a parity error or a receive frame over maximum number of transfer bytes error occurs the IEBus controller sets the PE or the DLE flag, respectively, and transitions to the wait state.

[2] An RxE interrupt is generated. The parity error or a receive frame over maximum number of transfer bytes error is detected within this interrupt. Furthermore, the PE or the DLE flag is cleared. Within the interrupt the error is detected by monitoring both the PE and the DLE flags, taking into account the possibility of a parity error or a receive frame over maximum number of transfer bytes error occurring independently (a parity error when in broadcast communications mode, for example).


---

**Figure 16 Handling of Parity Errors and Receive Frame over Maximum Number of Transfer Bytes Errors**
2.5 Processing of IEBus Runaway State (IRA)

The processing of IEBus control runaway state is illustrated in figure 17. When IEBus control runaway occurs the processing shown in figure 17 is performed within the IRA interrupt.

IRA interrupt

Enable module stop mode for IEBus controller (reset IEBus controller)

Cancel module stop mode for IEBus controller

Reset IEBus controller (restore ready-to-transmit/receive state)

Interrupt end

Figure 17 Handling of IEBus Runaway Status
3. Hardware Configuration

3.1 Example of Connections with IEBus Driver/Transceiver

An example of connections between the H8S/2258F and the HA12187FP IEBus driver/transceiver is shown in figure 18.

**Figure 18 Example of Connections with HA12187FP IEBus Driver/Transceiver**
4. Description of Software

The software used for master transmission/slave reception is described in this section.

4.1 DTC Settings

(1) DTC Register Information During Master Transmission

The DTC vector table and memory mapping used during master transmission are shown in figure 19. DTC register information, in the order MRA, SAR, MRB, DAR, CRA, and CRB, is set in internal RAM beginning at address H'FFEC00. The DTC register settings mapped to memory during master transmission are listed in table 1.

![Figure 19 DTC Vector Table and Memory Mapping During Master Transmission](image)

<table>
<thead>
<tr>
<th>DTC Register</th>
<th>Settings</th>
</tr>
</thead>
</table>
| MRA          | • DAR is fixed  
               • SAR is incremented following transfers  
               • Normal mode  
               • Byte transfer mode |
| SAR          | • Address in memory where 2nd byte of transmission data is stored |
| MRB          | • No DTC chain transfer  
               • Interrupts to CPU disabled unless transfer counter value is 0 |
| DAR          | • Address of the IEBus transmit buffer register (IETBR) |
| CRA          | • Value of number of transfer bytes set in the IEBus transmit message length register (IETBFL) minus 1 |
| CRB          | • Not used |
(2) DTC Register Information during Slave Reception

The DTC vector table and memory mapping used during slave reception are shown in figure 20. DTC register information, in the order MRA, SAR, MRB, DAR, CRA, and CRB, is set in the internal RAM beginning at address H'FFEC0C. The DTC register settings mapped to memory during slave reception are listed in table 2.

![Figure 20 DTC Vector Table and Memory Mapping During Slave Reception](image)

### Table 2 DTC Register Settings During Slave Reception

<table>
<thead>
<tr>
<th>DTC Register</th>
<th>Settings</th>
</tr>
</thead>
</table>
| MRA          | • SAR is fixed  
               • DAR is incremented following transfers  
               • Normal mode  
               • Byte transfer mode  |
| SAR          | • Address of the IEBus receive buffer register (IERBR)  |
| MRB          | • No DTC chain transfer  
               • Interrupts to CPU disabled unless transfer counter value is 0  |
| DAR          | • Address in memory where received data is stored  |
| CRA          | • Value of the IEBus receive message length register (IERBFL)  |
| CRB          | • Not used  |
4.2  Send/Receive Data Settings

(1) Transmit Data during Master Transmission

Figure 21 illustrates the transmit frames used for master transmission. During master transmission the frames shown in figure 21 are sent in sequence, [1] to [3]. The transmit frames are mapped in software by transdata (label).

<table>
<thead>
<tr>
<th>Frame</th>
<th>B</th>
<th>MA</th>
<th>SA</th>
<th>CF</th>
<th>LF</th>
<th>D (32 bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1] :</td>
<td>1</td>
<td>H'AAA</td>
<td>H'BBB</td>
<td>H'F</td>
<td>H'20</td>
<td>H'11 ...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[2] :</td>
<td>0</td>
<td>H'AAA</td>
<td>H'FFF</td>
<td>H'F</td>
<td>H'20</td>
<td>H'22 ...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[3] :</td>
<td>0</td>
<td>H'AAA</td>
<td>H'BBB</td>
<td>H'F</td>
<td>H'20</td>
<td>H'33 ...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

B: Broadcast bit  
MA: Master address field  
SA: Slave address field  
CF: Control field  
LF: Message length field  
D: Data field

Figure 21  Transmit Frames for Master Transmission
(2) Storing Received Data during Slave Reception

Figure 22 illustrates the way received data is stored during slave reception. During slave reception the communications mode is recognized as normal, general broadcast, or group broadcast, and the received data is stored in the appropriate destinations (in the internal RAM).

B: Broadcast bit
MA: Master address field
SA: Slave address field
CF: Control field
LF: Message length field
D: Data field

Figure 22  Storing Received Data during Slave Reception
4.3 Software Configuration

(1) Modules

Table 3  Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power-on reset routine</td>
<td>preset</td>
<td>Performs power-on reset processing</td>
</tr>
<tr>
<td>Internal I/O initialization routine</td>
<td>io_int</td>
<td>Initializes I/O functions, the IEBus controller, and the DTC</td>
</tr>
<tr>
<td>DTC initialization routine</td>
<td>dtc_int</td>
<td>Initializes DTC related registers in internal RAM</td>
</tr>
<tr>
<td>IEBus controller initialization routine</td>
<td>iebus_int</td>
<td>Initializes the IEBus controller</td>
</tr>
<tr>
<td>Transmit/receive main routine</td>
<td>iebus_main</td>
<td>Manages the number of transmit and receive frames. Controls retransmission when transmit errors occur</td>
</tr>
<tr>
<td>Start master transmission routine</td>
<td>master_trans</td>
<td>Controls the start of master transmission</td>
</tr>
<tr>
<td>Set up master transmission routine</td>
<td>mtrans_setup</td>
<td>Sets up transmit frames for master transmission</td>
</tr>
<tr>
<td>Input command routine</td>
<td>iebuscmd_input</td>
<td>Inputs commands to the IEBus controller</td>
</tr>
<tr>
<td>Ready-to-transmit interrupt handler</td>
<td>txrdyi</td>
<td>Recognizes the end of transmission data transfer by the DTC</td>
</tr>
<tr>
<td>txsti interrupt handler</td>
<td>txsti</td>
<td>Controls the following interrupts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- IEBus controller runaway interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Transmit start interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Transmit error interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Transmit end interrupt</td>
</tr>
<tr>
<td>IEBus controller runaway interrupt handler</td>
<td>ira_sub</td>
<td>Performs error processing when IEBus controller runaway occurs</td>
</tr>
<tr>
<td>Transmit start interrupt handler</td>
<td>txs_sub</td>
<td>Enables transfer of transmission data by the DTC</td>
</tr>
<tr>
<td>Transmit error interrupt handler</td>
<td>txe_sub</td>
<td>Detects transmit errors and performs error handling</td>
</tr>
<tr>
<td>Transmit end interrupt handler</td>
<td>txf_sub</td>
<td>Counts the number of frames transmitted</td>
</tr>
<tr>
<td>Ready-to-receive interrupt handler</td>
<td>rxrdyi</td>
<td>Recognizes the end of received data transfer by the DTC</td>
</tr>
<tr>
<td>rxsti interrupt handler</td>
<td>rxsti</td>
<td>Controls the following interrupts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Receive start interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Receive error interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Receive end interrupt</td>
</tr>
<tr>
<td>Receive start interrupt handler</td>
<td>rxs_sub</td>
<td>Performs setup for reception and enables transmission of data by the DTC</td>
</tr>
<tr>
<td>Set up slave reception routine</td>
<td>sreceiv_setup</td>
<td>Performs setup for slave reception</td>
</tr>
<tr>
<td>Receive end interrupt handler</td>
<td>rxf_sub</td>
<td>Counts the number of frames received</td>
</tr>
<tr>
<td>Receive error interrupt handler</td>
<td>rxe_sub</td>
<td>Detects receive errors and performs error handling</td>
</tr>
</tbody>
</table>
(2) Module Configuration

- **Main processing**
  - preset
  - ie_bus_int
  - ie_bus_main
- **iebuscmd_input**
- **mtrans_setup**
- **ire_trans**
- **ireceiv_setup**

- **Ready-to-transmit interrupt**
  - txrdyi
- **txsti interrupt**
  - txsti
  - ira_sub
  - txs_sub
  - txe_sub
  - txf_sub

- **Ready-to-receive interrupt**
  - rxrdyi
- **rxsti interrupt**
  - rxsti
  - rxs_sub
  - rxe_sub
  - rxf_sub

Figure 23  Module Configuration
### Table 4 Arguments

<table>
<thead>
<tr>
<th>Label</th>
<th>Function</th>
<th>Data Length</th>
<th>Modules Used by</th>
<th>I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEBUS_CMD</td>
<td>Stores IEBus controller commands</td>
<td>Byte</td>
<td>master_trans</td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>iebuscmd_input</td>
<td>Input</td>
</tr>
<tr>
<td>TRANS_FRAMECNT</td>
<td>Stores the number of frames transmitted</td>
<td>Byte</td>
<td>txf_sub</td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>mtrans_setup</td>
<td>Input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>iebus_main</td>
<td>Input</td>
</tr>
<tr>
<td>RECEIV_FRAMECNT</td>
<td>Stores the number of frames received</td>
<td>Byte</td>
<td>rxf_sub</td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>iebus_sub</td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>iebus_main</td>
<td>Input</td>
</tr>
<tr>
<td>IEBUS_CONDITION</td>
<td>Indicates the transmit/receive state of the IEBus controller</td>
<td>Byte</td>
<td>rxe_sub</td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td>0: Master transmission, slave reception complete</td>
<td></td>
<td>rxf_sub</td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td>1: Master transmission in progress</td>
<td></td>
<td>rsx_sub</td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td>2: Slave reception in progress</td>
<td></td>
<td>txe_sub</td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>txf_sub</td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>txe_sub</td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>iebus_sub</td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>iebus_main</td>
<td>Input</td>
</tr>
<tr>
<td>IEBUS_ERROR</td>
<td>Indicates IEBus controller transmission errors</td>
<td>Byte</td>
<td>rxe_sub</td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td>Slave reception</td>
<td></td>
<td>iebus_main</td>
<td>Input</td>
</tr>
<tr>
<td></td>
<td>0: No error</td>
<td></td>
<td>ira_sub</td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td>1: Timing error</td>
<td></td>
<td>txe_sub</td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td>2: Transfer byte count error</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3: Parity error</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4: Overrun error</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Master transmission</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5: Arbitration error</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6: Underrun error</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7: Timing error</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8: Transfer byte over</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9: ACK error</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5. Flowcharts

```plaintext
preset
  Set stack pointer
  Operation mode latch
    io_int
      Clear interrupt mask (Clear I bit in CCR to 0)
        iebus_main
          Infinite loop
            io_int
              Clear work area in internal RAM to 0
                Set the PG2 pin in port G to output (TX pin)
                  dtc_int
                    iebus_int
                      RTS
```
dtc_int

Cancel module stop mode for DTC  
(Clear MSTPC6 bit in MSTPCR to 0)

Start initialization of DTC registers in internal RAM  
Set T_MRA  
Settings  
• DAR is fixed  
• SAR is incremented following transfers  
• Normal mode  
• Byte transfer mode

Set T_MRB  
Settings  
• No DTC chain transfer  
• Interrupts to CPU disabled unless transfer counter value is 0

Set T_DAR  
Settings  
• Set IETBR address in T_DAR

Set R_MRA  
Settings  
• SAR is fixed  
• DAR is incremented following transfers  
• Normal mode  
• Byte transfer mode

Set R_MRB  
Settings  
• No DTC chain transfer  
• Interrupts to CPU disabled unless transfer counter value is 0

Set R_SAR  
Settings  
• Set IERBR address in R_SAR

RTS
### iebus_int

Cancel module stop mode for IEBus controller
(Clear MSTPC3 bit in MSTPCRC to 0)

Set IECTR
Settings
- Enable receive operation
- Set I/O of transmit and receive pins to negative logic
- Use 1/2 system clock
- No underrun error when transmitting last byte

Set IEMCR
Settings
- Set to 3 number of retransmissions in case of arbitration loss

Set IEAR1
Settings
- Set communications mode to mode 1
- Set lower 4 bits of local address

Set IEAR2
Settings
- Set upper 8 bits of local address

Set IEIER
Settings
- Enable receive start, receive end, and receive error interrupts

Set IEIET
Settings
- Enable transmit start, runaway, transmit end, and transmit error interrupts

Set IECTR
Settings
- Set PG pins as IEBus transmit/receive pins

### RTS
3 frames transmitted?
(TRANS_FRAMECNT=3?)

NO

master_trans

YES

Master transmission in progress?
(IEBUS_CONDITION=1?)

NO

NO

Slave reception in progress?
(IEBUS_CONDITION=2?)

NO

Slave reception complete?
(IEBUS_CONDITION=0?)

NO

Communication error generated?
(IEBUS_CONDITION=0?)

YES

YES

Clear error
(Clear IEBUS_CONDITION to 0)

3 frames received?
(RECEIV_FRAMECNT=3?)

YES

NO

NO

Slave reception in progress?
(IEBUS_CONDITION=2?)

YES

NO

Slave reception complete?
(IEBUS_CONDITION=0?)

YES

Communication error generated?
(IEBUS_CONDITION=0?)

YES

Clear error
(Clear IEBUS_CONDITION to 0)

RTS
Set up master transmission request command
H'02→IEBUS_CMD

Set control field of transmit frame
(Control field→IEMCR)

Set message length of transmit frame
(Message length→IETBFL)

Set CRA of DTC
(Message length - 1→T_CRA)

Set 1st byte of transmission data
(Transmission data→IETBR
Clear TxRDY in IETSR to 0)

Set SAR of DTC
(Address where 2nd byte of transmission data is stored→T_SAR)

Search for transmit frame in transdata area and identify start address where transmit frame is stored

Is transmit frame broadcast mode?

Set slave address of transmission destination
(Slave address→IESA1, IESA2)

Set transmit frame to normal mode
(Set SS bit in IEMCR to 1)

Set transmit frame to broadcast mode
(Clear SS bit in IEMCR to 0)

Command input enabled?
(CMX bit in IEFLG is 0?)

Set command to IEBus
(IEBUS_CMD→IECMR)

RTS

RTS

Set control field of transmit frame
(Control field→IEMCR)

Set message length of transmit frame
(Message length→IETBFL)

Set CRA of DTC
(Message length - 1→T_CRA)

Set 1st byte of transmission data
(Transmission data→IETBR
Clear TxRDY in IETSR to 0)

Set SAR of DTC
(Address where 2nd byte of transmission data is stored→T_SAR)
End data transfer using DTC
Disable ready-to-transmit interrupt
(Clear TxRDYE bit in IIEIET to 0)

Runaway interrupt?
(IR flag in IETS = 1?)

Transmit start interrupt?
(TxS flag in IETS = 1?)

Transmit end interrupt?
(TxF flag in IETS = 1?)

NO

YES

NO

YES

NO

YES

NO
Clear interrupts source
(Clear IRA flag in IETSR)

Disable ready-to-transmit interrupts
(Clear TxRDYE bit in IEIET to 0
Clear DTCEG5 in DTCERG to 0)

Disable ready-to-receive interrupts
Halt DTC operation
(Clear RxRDYE bit in IEIER to 0
Clear DTCEG6 in DTCERG to 0)

Reset IEBus controller
Transition to module stop mode
(Set MSTPC3 in MSTPCRC to 1)

Set error code H'0A
(H'0A→IEBUS_ERROR)

Transmit/receive complete
(H'00→IEBUS_CONDITION)

Clear interrupts source
(Clear TxS flag in IETSR)

Enable ready-to-transmit interrupts
Start DTC operation
(Set TxRDYE bit in IEIER to 1
Set DTCEG5 in DTCERG to 1)

Recognize start of transmission
(H'01→IEBUS_CONDITION)

Clear interrupts source
(Clear TxF flag in IETSR)

Increment number of transmitted frames
(Incremented value →TRANS_FRAMECNT)

Set transmit end
(0→IEBUS_CONDITION)
Clear interrupts source
(Clear TxE flag in IETSR)

Disable ready-to-transmit interrupts
Halt DTC operation
(Clear TxRDYE bit in IEIET to 0
Clear DTCEG5 in DTCERG to 0)

Arbitration loss error?
(AL flag in IETEF = 1?)

NO

Underrun error?
(UE flag in IETEF = 1?)

NO

Timing error?
(TTME flag in IETEF = 1?)

NO

Clear transfer byte over
(Clear R0 flag in IETEF to 0)

Set error code (H'08)

NO

ACK error?
(ACK flag in IETEF = 1?)

NO

Set error code (H'09)

Clear error
(Clear ACK flag to 0)

YES

Set error code to IEBUS_ERROR

RTS

YES

Set error code (H'05)

Clear error
(Clear AL flag to 0)

Set error code (H'06)

Clear error
(Clear UE flag to 0)

Set error code (H'07)

Clear error
(Clear TTME flag to 0)
End data transfer using DTC
;Stop receive data ready interrupt
(Clear RxRDYE bit in IERIER to 0)

Clear interrupt source
(Clear RxRDYE flag in IERSR to 0)

Receive start interrupt?
(RxS flag in IERSR = 1?)

Receive end interrupt?
(RxF flag in IERSR = 1?)

YES

YES

NO

NO
rxs_sub

Clear interrupt source
(Clear RxS flag in IERSR to 0)

srecciv_setup

Enable ready-to-receive interrupt
Start DTC operation
(Set RxRDYE bit in IEIER to 1
Set DTCEG6 in DTCERG to 1)

Recognize start of transmission
(H’02→IEBUS_CONDITION)

RTS

rxf_sub

Clear interrupt source
(Clear RxF flag in IERSR to 0)

Increment number of received frames
(Incremented value
→RECEIV_FRAMECNT)

Set receive end
(0→IEBUS_CONDITION)

RTS
sreceiv_setup

Are received frames normal mode? (RSS flag in IEFLG = 1?)

YES

Set receive control field in destination for storing received frames (IERCTL → storage destination)

Set receive master address in destination for storing received frames (IEMA1, IEMA2 → storage destination)

Set receive message length in destination for storing received frames (IERFBL → storage destination)

Set CRA of DTC (IERBFL → R_CRA)

Set SAR of DTC (Address of storage destination for received data → T_SAR)

NO

All received frames broadcast mode? (GG flag in IEFLG = 1?)

YES

Set destination for storing received frames to BROAD_BUFF

NO

Set destination for storing received frames to GROUP_BUFF

Set destination for storing received frames to NORMAL_BUFF

YES

RTS
rxe_sub

Clear interrupt source (Clear RxE flag in IETSR to 0)

Disable ready-to-receive interrupt
Halt DTC operation
(Clear RxRDYE bit in IEIER to 0)
(Clear DTCEG6 in DTCERG to 0)

Timing error? (RTME flag in IEREF = 1?)

NO

Transfer byte error? (DLE flag in IEREF = 1?)

YES

Set error code (H'02)

Clear max. transfer byte exceeded flag
(Clear DLE flag in IEREF to 0)

Parity error? (PE flag in IEREF = 1?)

YES

Set error code (H'03)

NO

Overrun error? (OVE flag in IEREF = 1?)

YES

Set error code (H'04)

NO

Clear error
(Clear PE flag to 0)

Clear error
(Clear OVE flag to 0)
(Clear RxRDY flag to 0)

Set error code in IEBUS_ERROR

RTS
6. Program Listings

```assembly
;---------------------------------------------------------------------
;--Master transmission/slave reception example using internal IEBus --
;-- controller of H8S/2258F--
;--IEBus usage conditions--
;--1. System clock: 12.58 MHz--
;--2. Communications mode: Mode 1--
;--3. Local address: H'AAA--
;--4. Number of retransmissions in case of arbitration loss: 3--
;--5. Data transmission and reception performed by DTC--
;--6. _TX and _RX pins set to negative logic I/O--
;---------------------------------------------------------------------
.cpu        2000A:24
.include    "2258FIE.H"

I1  ;******************************************************
I1  ;**2258F_IEBus.H**
I1  ;**Defines registers related to the IEBus controller**
I1  ;******************************************************
I1  ;*System control related*
I1  MDCR        .equ    H'00FFFDE7 ;Mode control register
I1
I1  ;*Module stop control registers A to C*
I1  MSTPCRA     .equ    H'00FFFDE8
I1  MSTPA6      .equ    6 ;DTC module stop bit
I1  MSTPCRB     .equ    H'00FFFDE9
I1  MSTPCRC     .equ    H'00FFDEA
I1  MSTPC3      .equ    3 ;IEBus module stop bit
I1
I1  ;*DTC enable registers*
I1  DTCERG      .equ    H'00FFFE1C
I1  DTCG6       .equ    6 ;IEBus RxRDY source
I1  DTCG5       .equ    5 ;IEBus TxRDY source
I1  DTVECR      .equ    H'00FFFE1F
I1
I1  ;*Port G registers*
I1  PGDDR       .equ    H'00FFFE3F
I1  PGDR        .equ    H'00FFFPF0F
I1
I1  ;*IEBus controller registers*
I1  IECTR       .equ    H'00FF800 ;IEBus control register
I1  IEE         .equ    7
I1  IOL         .equ    6
I1  DEE         .equ    5
I1  CKS         .equ    4
I1  RE          .equ    3
I1  LUEE        .equ    2
I1  IECMR       .equ    H'00FF801 ;IEBus command register
I1  IEMCR       .equ    H'00FF802 ;IEBus master control register
I1  SS          .equ    7
I1  IEAR1       .equ    H'00FF803 ;IEBus local address register 1
I1  STE         .equ    0
I1  IEAR2       .equ    H'00FF804 ;IEBus local address register 2
I1  IESA1       .equ    H'00FF805 ;IEBus slave address setting register 1
```
<table>
<thead>
<tr>
<th>Line</th>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>IESA2</td>
<td>.equ H'00FF806 ;IEBus slave address setting register 2</td>
</tr>
<tr>
<td>41</td>
<td>IETBF1</td>
<td>.equ H'00FF807 ;IEBus transmit message length register</td>
</tr>
<tr>
<td>42</td>
<td>IETBRA</td>
<td>.equ H'00FF808 ;IEBus transmit buffer register</td>
</tr>
<tr>
<td>43</td>
<td>IEMA1</td>
<td>.equ H'00FF809 ;IEBus receive master address register 1</td>
</tr>
<tr>
<td>44</td>
<td>IEMA2</td>
<td>.equ H'00FF80A ;IEBus receive master address register 2</td>
</tr>
<tr>
<td>45</td>
<td>IERCTL</td>
<td>.equ H'00FF80B ;IEBus receive control field register</td>
</tr>
<tr>
<td>46</td>
<td>IERBF1</td>
<td>.equ H'00FF80C ;IEBus receive message length register</td>
</tr>
<tr>
<td>47</td>
<td>IERBR</td>
<td>.equ H'00FF80D ;IEBus receive buffer register</td>
</tr>
<tr>
<td>48</td>
<td>IELA1</td>
<td>.equ H'00FF80E ;IEBus lock address register 1</td>
</tr>
<tr>
<td>49</td>
<td>IELA2</td>
<td>.equ H'00FF80F ;IEBus lock address register 2</td>
</tr>
<tr>
<td>50</td>
<td>IEFLG</td>
<td>.equ H'00FF810 ;IEBus general flag register</td>
</tr>
<tr>
<td>51</td>
<td>CMX</td>
<td>.equ 7</td>
</tr>
<tr>
<td>52</td>
<td>MRQ</td>
<td>.equ 6</td>
</tr>
<tr>
<td>53</td>
<td>SRQ</td>
<td>.equ 5</td>
</tr>
<tr>
<td>54</td>
<td>SRE</td>
<td>.equ 4</td>
</tr>
<tr>
<td>55</td>
<td>LCK</td>
<td>.equ 3</td>
</tr>
<tr>
<td>56</td>
<td>RSS</td>
<td>.equ 1</td>
</tr>
<tr>
<td>57</td>
<td>GG</td>
<td>.equ 0</td>
</tr>
<tr>
<td>58</td>
<td>IETSRA</td>
<td>.equ H'00FF811 ;IEBus transmit/runaway status register</td>
</tr>
<tr>
<td>59</td>
<td>TxRDY</td>
<td>.equ 7</td>
</tr>
<tr>
<td>60</td>
<td>IRA</td>
<td>.equ 3</td>
</tr>
<tr>
<td>61</td>
<td>TXS</td>
<td>.equ 2</td>
</tr>
<tr>
<td>62</td>
<td>TXF</td>
<td>.equ 1</td>
</tr>
<tr>
<td>63</td>
<td>TXE</td>
<td>.equ 0</td>
</tr>
<tr>
<td>64</td>
<td>IETSIE</td>
<td>.equ H'00FF812 ;IEBus transmit/runaway interrupt enable register</td>
</tr>
<tr>
<td>65</td>
<td>TxRDYE</td>
<td>.equ 7</td>
</tr>
<tr>
<td>66</td>
<td>IRAE</td>
<td>.equ 3</td>
</tr>
<tr>
<td>67</td>
<td>TXSE</td>
<td>.equ 2</td>
</tr>
<tr>
<td>68</td>
<td>TXFE</td>
<td>.equ 1</td>
</tr>
<tr>
<td>69</td>
<td>TXEE</td>
<td>.equ 0</td>
</tr>
<tr>
<td>70</td>
<td>IETEF</td>
<td>.equ H'00FF813 ;IEBus transmit error flag register</td>
</tr>
<tr>
<td>71</td>
<td>AL</td>
<td>.equ 4</td>
</tr>
<tr>
<td>72</td>
<td>UE</td>
<td>.equ 3</td>
</tr>
<tr>
<td>73</td>
<td>TTME</td>
<td>.equ 2</td>
</tr>
<tr>
<td>74</td>
<td>RO</td>
<td>.equ 1</td>
</tr>
<tr>
<td>75</td>
<td>ACK</td>
<td>.equ 0</td>
</tr>
<tr>
<td>76</td>
<td>IERSR</td>
<td>.equ H'00FF814 ;IEBus receive status register</td>
</tr>
<tr>
<td>77</td>
<td>RxRDY</td>
<td>.equ 7</td>
</tr>
<tr>
<td>78</td>
<td>RXS</td>
<td>.equ 2</td>
</tr>
<tr>
<td>79</td>
<td>RXF</td>
<td>.equ 1</td>
</tr>
<tr>
<td>80</td>
<td>RXE</td>
<td>.equ 0</td>
</tr>
<tr>
<td>81</td>
<td>IERIER</td>
<td>.equ H'00FF815 ;IEBus receive interrupt enable register</td>
</tr>
<tr>
<td>82</td>
<td>RxRDYE</td>
<td>.equ 7</td>
</tr>
<tr>
<td>83</td>
<td>RXSE</td>
<td>.equ 2</td>
</tr>
<tr>
<td>84</td>
<td>RXFE</td>
<td>.equ 1</td>
</tr>
<tr>
<td>85</td>
<td>RXEE</td>
<td>.equ 0</td>
</tr>
<tr>
<td>86</td>
<td>IEREF</td>
<td>.equ H'00FF816 ;IEBus receive error flag register</td>
</tr>
<tr>
<td>87</td>
<td>OVE</td>
<td>.equ 3</td>
</tr>
<tr>
<td>88</td>
<td>RTME</td>
<td>.equ 2</td>
</tr>
<tr>
<td>89</td>
<td>DLE</td>
<td>.equ 1</td>
</tr>
<tr>
<td>90</td>
<td>PE</td>
<td>.equ 0</td>
</tr>
</tbody>
</table>

;--Vector table

;---------------------------------------------------------------------

;--Master Transmission/Slave Reception Example Using IEBus Controller

REJ06B0488-0100/Rev.1.00 March 2005 Page 39 of 53
H8S Family

Master Transmission/Slave Reception Example Using IEBus Controller

17 ;---------------------------------------------------------------------
18              .section    VECT,code,locate=0
19              .data.l     preset          ;Power-on reset
20              .org       H'1A0
21              .data.l     rxsti           ;RxSTI interrupt
22              .data.l     rxrdyi          ;RxRDYI interrupt
23              .data.l     txrdyi          ;TxRDYI interrupt
24              .data.l     txsti           ;TxSTI interrupt
25 ;---------------------------------------------------------------------
26 ;--DTC vector table                                                 --
27 ;---------------------------------------------------------------------
28        .org        H'4D2
29        .data.w     (R_MRA - H'FF0000)   ;Start DTC with RxRDYI interrupt
30        .org        H'4D4
31        .data.w     (T_MRA - H'FF0000)   ;Start DTC with TxRDYI interrupt
32
33        .section    PROG,code,locate=H'500
34 ;---------------------------------------------------------------------
35     ;-preset                                                            --
36     ;-Power-on reset processing                                         --
37     ;-[Input]: None                                                     --
38     ;-[Output]: None                                                    --
39     ;---------------------------------------------------------------------
40     preset             .equ   $  
41              mov.l     #H'FFEFC0,ER7     ;Set stack pointers
42              mov.b     @MDCR,R0L         ;Latch mode
43              jsr       @io_int           ;Initialize IEBus, etc.
44              andc.b    #H'7F,CCR         ;Clear interrupt mask
45              jsr       @iebus_main
46     preset01
47              bra       preset01          ;Infinite loop
48 ;---------------------------------------------------------------------
49     ;-io_int                                                            --
50     ;-Initialize I/O, IEBus, and DTC                                    --
51     ;-[Input]: None                                                     --
52     ;-[Output]: None                                                    --
53     ;---------------------------------------------------------------------
54     io_int             .equ   $  
55              sub.l     ER0,ER0
56              mov.l     #ram_wroks,ER1  ;Clear work area of internal RAM to 0
57     io_int01
58              bra       preset01         ;Infinite loop
59 ;---------------------------------------------------------------------
60     ;-dtc_int                                                           --
61     ;---------------------------------------------------------------------
62 ;--DTC vector table                                                 --
63 ;---------------------------------------------------------------------
64 ;--io_int                                                            --
65 ;--Initialize I/O, IEBus, and DTC                                    --
66 ;--dtc_int                                                           --
.;-Initialize DTC related registers in internal RAM                  --
.;-[Input]: None                                                   --
.;-[Output]: None                                                 --
;---------------------------------------------------------------------
dtc_int            .equ   $
              bclr.b    #MSTPA6,@MSTPCRA  ;Cancel module stop mode for DTC
              mov.b     #B'10000000,R0L   ;Initialize DTC registers used for master transmission
              mov.b     R0L,@T_MRA
              mov.b     #B'00000000,R0L   ;Initialize DTC registers used for slave reception
              mov.b     R0L,@T_MRB
              mov.l     #IETBR,ER0        ;Set IETBR as destination address
              mov.w     R0,@T_DAR_L
              mov.w     E0,R0
              mov.b     R0L,@T_DAR_H
              mov.b     #B'00100000,R0L   ;Enable receive operation
              mov.b     R0L,@IECTR
              mov.b     #(H'A0 | H'04),R0L ;Lower 4 bits of local address: H'A
              mov.b     R0L,@IEAR1
              mov.b     #H'AA,R0L         ;Upper 8 bits of local address: H'AA
              mov.b     R0L,@IEAR2
              rts
;---------------------------------------------------------------------
iebus_int
              bclr.b    #MSTPC3,@MSTPCRC  ;Cancel module stop mode for IEBus controller
              mov.b     #B'00101000,R0L   ;Master transmission settings
              mov.b     #B'10111111,R0L   ;Set number of retransmissions in case of arbitration loss to 3
              mov.b     R0L,@IEMCR       ;Control field is H'0F
              mov.b     #(H'A0 | H'04),R0L ;Lower 4 bits of local address: H'A
              mov.b     R0L,@IEAR1       ;Operating mode: Mode 1
              mov.b     #H'AA,R0L        ;Enable receive interrupts
124             mov.b     #B'00000111,R0L    ;Receive start interrupt
125             mov.b     R0L,@IEIER         ;Receive normal end interrupt
126             ;Receive error interrupt
127                                          ;Enable transmit interrupts
128             mov.b     #B'00001111,R0L    ;Runaway interrupt
129             mov.b     R0L,@IEIET         ;Transmit start interrupt
130                                          ;Transmit normal end interrupt
131                                          ;Transmit error interrupt
132                                          ;Enable transmit interrupts
133             bset.b    #IEE,@IECTR        ;Start IEBus operation
134                                          
135             rts
136     ;--------------------------------------------------------------------------------------
137     ;-iebus_main                                                                         --
138     ;-Transmit frame to the slave 3 times and receive other frame from the master 3 times--
139     ;-
140     ;[Input]: None                                                                      --
141     ;[Output]: None                                                                     --
142     ;--------------------------------------------------------------------------------------
143     iebus_main         .equ      $
144              mov.b     @TRANS_FRAMECNT,R0L    ;Frame transmitted 3 times?
145              cmp.b     #3,R0L
146              beq       iebus_main_05
147     iebus_main_01
148              jsr       @master_trans          ;Transmit 1 frame
149     iebus_main_02
150              btst.b    #0,@IEBUS_CONDITION    ;Recognize start of master transmission
151              bne       iebus_main_03          ;
152              btst.b    #1,@IEBUS_CONDITION    ;Recognize start of slave reception
153              bne       iebus_main_04          ;
154              mov.b     @IEBUS_ERROR,R0L       ;
155              bne       iebus_main_01          ;
156              bra       iebus_main_02          ;
157     iebus_main_03
158              btst.b    #0,@IEBUS_CONDITION    ;Recognize end of master transmission
159              bne       iebus_main_03          ;
160              mov.b     @IEBUS_ERROR,R0L       ;Recognize transmit error
161              beq       iebus_main          ;
162              mov.b     #0,R0L
163              mov.b     R0L,@IEBUS_ERROR       ;
164              bra       iebus_main_01          ;Retransmission processing
165     iebus_main_04
166              btst.b    #1,@IEBUS_CONDITION    ;Recognize end of slave reception
167              bne       iebus_main_04          ;
168              mov.b     @IEBUS_ERROR,R0L       ;Recognize receive error
169              beq       iebus_main_02          ;
170              mov.b     #0,R0L
171              mov.b     R0L,@IEBUS_ERROR       ;Handle receive error within interrupt
172              bra       iebus_main_02          ;
173     iebus_main_05
174              mov.b     @RECEIV_FRAMECNT,R0L  ;Frame received 3 times?
175              cmp.b     #3,R0L
176              beq       iebus_main_08        ;
177     iebus_main_06
H8S Family
Master Transmission/Slave Reception Example Using IEBus Controller

178    btst.b  #1,@IEBUS_CONDITION ;Recognize start of slave reception
179    beq iebus_main_06
180 iebus_main_07
181    btst.b  #1,@IEBUS_CONDITION ;Recognize end of slave reception
182    bne iebus_main_07
183    mov.b  @IEBUS_ERROR,ROL ;Recognize receive error
184    beq iebus_main_05
185    mov.b  #0,ROL
186    mov.b  ROL,@IEBUS_ERROR ;Handle receive error within interrupt
187    bra iebus_main_05
188 iebus_main_08
189    rts
190
191 ;-master_trans
192 ;-Perform master transmission
193 ;-[Input]: None
194 ;-[Output]: Output @IEBUS_CMD command
195 ;---------------------------------------------------------------------
196 master_trans .equ $
197    jsr @mtrans_setup ;Setup for master transmission
198    mov.b  #H'02,ROL ;Transfer request as master
199    mov.b  ROL,@IEBUS_CMD
200    jsr @iebuscmd_input
201    rts
202 ;---------------------------------------------------------------------
203 ;-mtrans_setup
204 ;-Setup for performing master transmission
205 ;-[Input]: None
206 ;-[Output]: None
207 ;---------------------------------------------------------------------
208 mtrans_setup .equ $
209    mov.l  #transdata,ER1 ;Search which frame will be transmitted
210    mov.b  @TRANS_FRAMECNT,ROL
211    sub.b  R0H,R0H
212 mtrans_setup01
213    cmp.b  R0H,ROL
214    beq mtrans_setup02
215    add.l  #37,ER1
216    inc.b  R0H
217    bra mtrans_setup01
218 mtrans_setup02
219    mov.b  @ER1+,ROL ;Set broadcast/individual
220    beq mtrans_setup03
221    bclr.b  #SS,@IEMCR
222    bra mtrans_setup04
223 mtrans_setup03
224    bset.b  #SS,@IEMCR
225 mtrans_setup04 ;Set slave address for transmission destination
226    mov.b  @ER1+,ROL
227    mov.b  ROL,@IESA2
228    mov.b  @ER1+,ROL
229    mov.b  ROL,@IESA1
230
231    mov.b  @IEMCR,ROL ;Set control field
232       mov.b  #H'F0,R0L
233       and.b  R0L,R0H
234       mov.b  @ER1+,R0L
235       or.b   R0L,R0H
236       mov.b  R0H,@IEMCR
237       mtrans_setup05
238       sub.w  R0,R0       ;Set transmit message length and CRA of DTC
239       mov.b  @ER1+,R0L
240       mov.b  R0L,@IETBFL
241       dec.b  R0L
242       mov.w  R1,@T_SAR_L ;Set SAR of DTC
243       mtrans_setup06
244       mov.b  @ER1+,R0L
245       mov.b  R0L,@IETBR ;Write 1st byte of data to IETBR
246       bclr.b  #TxRDY,@IETSR ;Set 1st byte of transmission data
247       transfer 2nd and subsequent bytes using DTC
248       mov.w  R1,@T_SAR_L ;Set SAR of DTC
249       mov.w  E1,R0       ;Storage destination for transmission data
250       mtrans_setup07
251       rts
252       iebuscmd_input .equ    $
253       btst.b  #CMX,@IEFLG ;What for CMX flag to be cleared to 0
254       bne      iebuscmd_input
255       mov.b   @IEBUS_CMD,R0L
256       mov.b   R0L,@IECMR ;Input command
257       rts
258       txrdyi   .equ       $
259       bclr.b  #TxRDYE,@IEIET ;Halts interrupt
260       rte
261       txsti    .equ       $
262       push.l  ER0
263       push.l  ER1
H8S Family
Master Transmission/Slave Reception Example Using IEBus Controller

```
286              btst.b    #IRA,@IETSR    ;Detect interrupt source
287              bne       txsti01
288              btst.b    #TxS,@IETSR
289              bne       txsti02
290              btst.b    #TxF,@IETSR
291              bne       txsti03
292
293              jsr       @txe_sub    ;To transmit error routine
294              bra       txsti04
295     txsti01
296              jsr       @ira_sub    ;To IEBus runaway routine
297              bra       txsti04
298     txsti02
299              jsr       @txs_sub    ;To transmit start routine
300              bra       txsti04
301     txsti03
302              jsr       @txf_sub    ;To transmit end routine
303     txsti04
304     pop.l     ER1
305     pop.l     ER0
306     rte
307     ;---------------------------------------------------------------------
308     ;-ira_sub
309     ;-IEBus runaway routine
310     ;-[Input]: None
311     ;-[Output]:@IEBUS_ERROR=H'0A
312     ;-@IEBUS_CONDITION=0
313     ;---------------------------------------------------------------------
314     ira_sub           .equ       $
315              bclr.b    #IRA,@IETSR              ;Clear interrupt source
316              bclr.b    #TxRDYE,@IEIET           ;Halt transmission data setup interrupt
317              bclr.b    #DTCEG5,@DTCERG          ;Halt DTC operation
318              bclr.b    #RxRDYE,@IEIER           ;Halt ready-to-receive interrupt
319              bclr.b    #DTCEG6,@DTCERG          ;Halt DTC
320
321              bset.b   #MSTPC3,@MSTP CRC          ;Set module stop mode for IEBus (reset)
322              jsr      @iebus_int                ;Reinitialize IEBus
323              mov.b    #H'0A,R0L
324              mov.b    R0L,@IEBUS_ERROR          ;Set error code
325              mov.b    R0L,R0L                   ;End transmit/receive
326              mov.b    R0L,@IEBUS_CONDITION
327              rts
328     ;-------------------------------------------------------------------------------------
329     ;-txs_sub
330     ;-Processing of IEBus transmit start interrupt (start data transmission using DTC)
331     ;-[Input]: None
332     ;-[Output]:@IEBUS_CONDITION=H'01
333     ;-------------------------------------------------------------------------------------
334     txs_sub            .equ       $
335              bclr.b    #TxS,@IETSR
336              bset.b    #DTCEG5,@DTCERG        ;Enable DTC operation
337              bset.b    #TxRDYE,@IEIET         ;Enable TxRDYI interrupt
338              bset.b    #0,@IEBUS_CONDITION
339              rts
```
;--txf_sub
;--End IEBus transmission. Count number of frames transmitted
;--[Input]: None
;--[Output]: @TRANS_FRAMECNT++
;--  @IEBUS_CONDITION=0
;--
;---------------------------------------------------------------------

txf_sub    .equ       $

bclr.b    #TxF,@IETSR  ;Clear interrupt source
mov.b     @TRANS_FRAMECNT,R0L
inc.b     R0L
mov.b     R0L,TRANS_FRAMECNT
bclr.b    #0,@IEBUS_CONDITION  ;End transmission
rts

;---------------------------------------------------------------------

;--txe_sub
;--Transmit error processing of IEBus
;--[Input]: No
;--[Output]: @IEBUS_ERROR
;--  H'05: Arbitration error
;--  H'06: Underrun error
;--  H'07: Timing error
;--  H'08: Transfer byte over
;--  H'09: ACK error
;--  @IEBUS_CONDITION=0
;--
;---------------------------------------------------------------------
txe_sub  .equ      $

bclr.b    #TxE,@IETSR  ;Clear interrupt source
bclr.b    #TxRDYE,@IEIET  ;Halt transmit-data-ready interrupt
bclr.b    #DTCEG5,@DTCERG  ;Halt DTC
btst.b    #AL,@IETEF  ;Search error contents
bne       txe_sub01
btst.b    #UE,@IETEF
bne       txe_sub02
btst.b    #TTME,@IETEF
bne       txe_sub03
btst.b    #RO,@IETEF  ;Clear transfer byte over
mov.b     #H'08,R0L  ;Set error code
bra       txe_sub04  ;Transfer byte over caused by ACK error?
bclir.b    #AL,@IETEF  ;Clear arbitration loss error
mov.b     #H'05,R0L  ;Set error code
bra       txe_sub05
bclir.b    #UE,@IETEF  ;Clear underrun error
mov.b     #H'06,R0L  ;Set error code
bra       txe_sub05
bclir.b    #TTME,@IETEF  ;Clear timing error
mov.b     #H'07,R0L  ;Set error code
bra       txe_sub05
394     txe_sub04
395         btst.b   #ACK,@IETEF
396         beq     txe_sub05
397         bclr.b   #ACK,@IETEF ;Clear ACK error
398         mov.b    #H'09,R0L  ;Set error code
399     txe_sub05
400         bclr.b    #0,@IEBUS_CONDITION ;End transmission
401         mov.b     R0L,@IEBUS_ERROR
402         rts
403     ;---------------------------------------------------------------------
404     ;-rxrdyi interrupt                                              --
405     ;-Halt receive data setup interrupt (end data reception using DTC) --
406     ;-[Input]: None                                             --
407     ;-[Output]: None                                            --
408     ;---------------------------------------------------------------------
409     rxrdyi   .equ      $                       
410         bclr.b    #RxRDY,@IERSR  ;Clear interrupt source
411         bclr.b    #RxRDYE,@IEIER ;Halt ready-to-transmit interrupt
412         rte
413     ;---------------------------------------------------------------------
414     ;-rxsti interrupt                                            --
415     ;-IEBus reception related interrupt                          --
416     ;-[Input]: None                                             --
417     ;-[Output]: See output values for individual subroutines     --
418     ;---------------------------------------------------------------------
419     rxsti    .equ        $                      
420         push.l     ER0
421         push.l     ER1
422     btst.b    #RxS,@IERSR ;Search for interrupt request source
423         bne     rxsti01
424         btst.b    #RxF,@IERSR
425         bne     rxsti02
426         jsr     @rxe_sub ;To rxe_sub
427         bra     rxsti03
428     rxsti01
429         jsr     @rxs_sub ; To rxs_sub
430         bra     rxsti03
431     rxsti02
432         jsr     @rxf_sub ; To rxf_sub
433     rxsti03
434         pop      ER1
435         pop      ER0
436         rte
437     ;---------------------------------------------------------------------
438     ;-rxs_sub                                                      --
439     ;-Processing of IEBus receive start interrupt (start data reception using DTC) --
440     ;-[Input]: None                                             --
441     ;-[Output]:@IEBUS_CONDITION=H'02                           --
442     ;---------------------------------------------------------------------
443     rxs_sub  .equ     $                            
444         bclr.b    #RxS,@IERSR
445         jsr     @sreceiv_setup
446         bset.b    #DTCEG6,@DTCERG ;Enable DTC start
bset.b #RxRDYE, @IEIER ; Enable RxRDY interrupt

bset.b #1, @IEBUS_CONDITION ;

rxs_sub01

rts

;---------------------------------------------------------------

; -sreceiv_setup

; -Setup for performing slave reception

; -[Input]: None

; -[Output]: None

;---------------------------------------------------------------

sreceiv_setup .equ $

btst.b #RSS, @IEFLG ; Determine broadcast/individual

beq sreceiv_setup02 ; Select receive buffer

mov.l #NORMAL_BUFF, ER1 ; Use normal communication receive buffer

bra sreceiv_setup04

sreceiv_setup02

btst.b #GG, @IEFLG ; Verify general broadcast or group broadcast

beq sreceiv_setup03

mov.l #BROAD_BUFF, ER1 ; Use general broadcast communication receive buffer

bra sreceiv_setup04

sreceiv_setup03

mov.l #GROUP_BUFF, ER1 ; Use group broadcast communication receive buffer

sreceiv_setup04

mov.b @IEMA2, R0L ; Set master address in receive buffer

mov.b R0L, @ER1

inc.l #1, ER1

mov.b @IEMA1, R0L

mov.b R0L, @ER1

inc.l #1, ER1

mov.b @IERCTL, R0L ; Set control field in receive buffer

mov.b R0L, @ER1

inc.l #1, ER1

sub.w R0, R0

mov.b @IERBFL, R0L ; Set message length in receive buffer

mov.b R0L, @ER1

mov.w R0, @R_CRA ; Set CRA (transfer counter) for bytes received by DTC

inc.l #1, ER1

mov.w R1, @R_DAR_L ; Set DAR of DTC

mov.w E1, R0

mov.b R0L, @R_DAR_H

rts

;---------------------------------------------------------------

; -rxf_sub

; -IEBus receive end interrupt, count the number of frames received

; -[Input]: None

; -[Output]: @RECEIV_FRAMECNT++

; - @IEBUS_CONDITION=0

; ---------------------------------------------------------------

rxf_sub .equ $

bclr.b #RxF, @IERSR

mov.b @RECEIV_FRAMECNT, R0L
inc.b ROL
mov.b ROL,RECEIY_FRAMECNT
bclr.b #1,IEBUS_CONDITION ;End reception
rts

;---------------------------------------------------------------------
;.--rxe_sub                                                           --
;.--IEBus receive error interrupt                                     --
;.--[Input]: None                                                     --
;.--[Output]:@IEBUS_ERROR                                             --
;.--                      H'01: Timing error                          --
;.--                      H'02: Transfer byte count error             --
;.--                      H'03: Parity error                          --
;.--                      H'04: Overrun error                         --
;.--        @IEBUS_CONDITION=0                                        --
;---------------------------------------------------------------------
rxe_sub .equ       $
bcir.b #RxE,IERSR ;Clear RxE
bcir.b #RxRDY,RIIER ;Halt receive ready interrupt
bcir.b #DTCEG6,DTCEG ;Halt DTC
btst.b #RTME,IEREF ;Search error contents
bne rxe_sub01
btst.b #DLE,IEREF ;Transfer byte over
bne rxe_sub02
btst.b #RSS,IEFLG ;During normal communication reception,
bne rxe_sub06 ;discard frame being received in case of
;parity error or overrun error.
bra rxe_sub03 ;(Wait for transfer byte over error.)
ra_sub01
mov.b #H'01,ROL ;Set error code
bclr.b #RTME,IEREF ;Clear timing error
bra rxe_sub05
ra_sub02
mov.b #H'02,ROL ;Set error code
bcir.b #DLE,IEREF ;Clear transfer byte error
ra_sub03
btst.b #PE,IEREF
beq rxe_sub04
mov.b #H'03,ROL ;Set error code
bcir.b #PE,IEREF ;Clear parity error
bra rxe_sub05
ra_sub04
btst.b #OVE,IEREF
beq rxe_sub05
mov.b #H'04,ROL ;Set error code
bcir.b #OVE,IEREF ;Clear overrun error
bcir.b #RxRDY,IERSR
ra_sub05
mov.b ROL,IEBUS_ERROR
bclr.b #1,IEBUS_CONDITION ;End reception
ra_sub06
rts
;-----------------------------------------------
;Transmission data table
;The following frames are transmitted to the slave
;-----------------------------------------------
transdata .equ    $

561 tdata0   .data.b  H'00 ;Normal communication
562       .data.b  H'BB,H'B0 ;Slave address
563       .data.b  H'0F ;Control field
564       .data.b  H'20 ;Message length
565       .data.b  H'11,H'11,H'11,H'11,H'11,H'11,H'11,H'11 ;Data (32 bytes)
566 tdata1   .data.b  H'01 ;General broadcast communication
567       .data.b  H'FF,H'F0 ;Slave address
568       .data.b  H'0F ;Control field
569       .data.b  H'20 ;Message length
570       .data.b  H'22,H'22,H'22,H'22,H'22,H'22,H'22,H'22 ;Data (32 bytes)
571 tdata2   .data.b  H'01 ;Group broadcast communication
572       .data.b  H'BB,H'B0 ;Slave address
573       .data.b  H'0F ;Control field
574       .data.b  H'20 ;Message length
575       .data.b  H'33,H'33,H'33,H'33,H'33,H'33,H'33,H'33 ;Data (32 bytes)
576 ;-----------------------------------------------
;--Variable definitions (internal RAM area)
;-----------------------------------------------
NORMAL_BUFF        .res.b   36 ;Normal communication receive buffer
BROAD_BUFF         .res.b   36 ;Simultaneous broadcast communication receive buffer
GROUP_BUFF         .res.b   36 ;Group broadcast communication receive buffer
IEBUS_ERROR        .res.b   1 ;IEBus error variable
IEBUS_CMD          .res.b   1 ;IEBus command storage
TRANS_FRAMECNT     .res.b   1 ;Transmit frame counter
RECEIV_FRAMECNT    .res.b   1 ;Receive frame counter
IEBUS_CONDITION    .res.b   1 ;IEBus transmit/receive status monitor

;--DTC register information during master transmission
T_MRA              .res.b   1 ;MRA
T_SAR_H            .res.b   1 ;SAR
T_SAR_L            .res.w   1 ;
T_MRB              .res.b   1 ;MRB
T_DAR_H            .res.b   1 ;DAR
T_DAR_L            .res.w   1 ;
T_CRA              .res.w   1 ;CRA
609  T_CRB       .res.w  1       ;CRB
610 ;--DTC register information during slave reception
611  R_MRA       .res.b  1       ;MRA
612  R_SAR_H     .res.b  1       ;SAR
613  R_SAR_L     .res.w  1       ;
614  R_MRB       .res.b  1       ;MRB
615  R_DAR_H     .res.b  1       ;DAR
616  R_DAR_L     .res.w  1       ;
617  R_CRA       .res.w  1       ;CRA
618  R_CRB       .res.w  1       ;CRB
619  ram_wroke   .equ         $
### Revision Record

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Mar.09.05</td>
<td>—</td>
<td>First edition issued</td>
</tr>
</tbody>
</table>
Keep safety first in your circuit designs!

1. Renesas Technology Corp. puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage. Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corp. product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corp. or a third party.
2. Renesas Technology Corp. assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors. Renesas Technology Corp. assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corp. by various means, including the Renesas Technology Corp. Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corp. semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corp. is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corp. for further details on these materials or the products contained therein.