Cautions

Keep safety first in your circuit designs!

1. Renesas Technology Corporation 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 Corporation 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 Corporation or a third party.

2. Renesas Technology Corporation 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 Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation 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 Corporation 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 Corporation by various means, including the Renesas Technology Corporation 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 Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.

5. Renesas Technology Corporation 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 Corporation or an authorized Renesas Technology Corporation 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 Corporation 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 Corporation for further details on these materials or the products contained therein.

Rev. 1.00, 08/03, page iv of viii
Preface

Outline

The HCAN-2 control area network is a module designed to control a CAN (controller area network) for implementing real-time communications in an automobile, industrial machinery system, or the like.

This Application Note consists of communications operation examples using the HCAN-2, which incorporates the SH7047F. It is hoped that these examples will produce useful to designers of software and hardware systems.

The task examples and application examples contained in this Application Note have been tested and verified to work properly. Nevertheless, their operation should be confirmed in actual use before final implementation.

Device Used for Operation Verification

SH7047F
Contents

Section 1  Example 1 (Standard Format, 1 Byte of Data) .............................. 1
  1.1 Transmission and Reception Specifications .............................................. 1
  1.2 Transmission and Reception Specifications, Function Description ............. 1
  1.3 Transmission Flowchart ............................................................................. 3
  1.4 Software Description (Transmission)............................................................ 4
  1.5 Transmission Program Listing......................................................................... 5
  1.6 Reception Flowchart....................................................................................... 6
  1.7 Software Description (Reception)................................................................. 7
  1.8 Reception Program Listing............................................................................. 8
  1.9 Operation Waveforms (Transmission and Reception)................................. 9

Section 2  Example 2 (Standard Format, 8 Bytes of Data, Using DTC)....... 10
  2.1 Transmission and Reception Specifications ................................................. 10
  2.2 Transmission and Reception Specifications, Function Description ............ 10
  2.3 Transmission Flowchart .............................................................................. 13
  2.4 Software Description (Transmission)................................................................. 14
  2.5 Transmission Program Listing...................................................................... 16
  2.6 Reception Flowchart..................................................................................... 18
  2.7 Software Description (Reception)................................................................. 19
  2.8 Reception Program Listing.......................................................................... 21
  2.9 Operation Waveforms (Transmission and Reception)............................... 23

Section 3  Example 3 (Extended Format, 1 Byte of Data) ....................... 24
  3.1 Transmission and Reception Specifications ................................................ 24
  3.2 Transmission and Reception Specifications, Function Description ............... 24
  3.3 Transmission Flowchart .............................................................................. 26
  3.4 Software Description (Transmission)............................................................... 28
  3.5 Transmission Program Listing...................................................................... 29
  3.6 Reception Flowchart...................................................................................... 31
  3.7 Software Description (Reception)................................................................. 33
  3.8 Reception Program Listing.......................................................................... 34
  3.9 Operation Waveforms (Transmission and Reception)............................... 36

Section 4  Example 4 (Standard Format, 8 Bytes of Data, Prioritized)......... 37
  4.1 Transmission and Reception Specifications ................................................ 37
  4.2 Transmission and Reception Specifications, Function Description ............ 38
  4.3 Transmission Flowchart ............................................................................. 41
  4.4 Software Description (Transmission)............................................................... 43
  4.5 Transmission Program Listing...................................................................... 44
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.6</td>
<td>Reception Flowchart</td>
<td>49</td>
</tr>
<tr>
<td>4.7</td>
<td>Software Description (Reception)</td>
<td>50</td>
</tr>
<tr>
<td>4.8</td>
<td>Reception Program Listing</td>
<td>52</td>
</tr>
<tr>
<td>4.9</td>
<td>Operation Waveforms (Transmission and Reception)</td>
<td>54</td>
</tr>
<tr>
<td>Section 5</td>
<td>Example 5 (Remote Frame)</td>
<td>56</td>
</tr>
<tr>
<td>5.1</td>
<td>Transmission and Reception Specifications</td>
<td>56</td>
</tr>
<tr>
<td>5.2</td>
<td>Transmission and Reception Specifications, Function Description</td>
<td>57</td>
</tr>
<tr>
<td>5.3</td>
<td>Transmission Flowchart</td>
<td>59</td>
</tr>
<tr>
<td>5.4</td>
<td>Software Description (Transmission)</td>
<td>61</td>
</tr>
<tr>
<td>5.5</td>
<td>Transmission Program Listing</td>
<td>63</td>
</tr>
<tr>
<td>5.6</td>
<td>Reception Flowchart</td>
<td>66</td>
</tr>
<tr>
<td>5.7</td>
<td>Software Description (Reception)</td>
<td>67</td>
</tr>
<tr>
<td>5.8</td>
<td>Reception Program Listing</td>
<td>70</td>
</tr>
<tr>
<td>5.9</td>
<td>Operation Waveforms (Transmission and Reception)</td>
<td>72</td>
</tr>
</tbody>
</table>
Section 1  Example 1  
(Standard Format, 1 Byte of Data)

1.1 Transmission and Reception Specifications

Transmission and reception of 1 byte of data in the standard format using two SH7047F devices.

Common Transmission and Reception Specifications

- The data transfer speed is 250 kbps (during 50 MHz operation).
- The identifier is H'555.

Transmission Specifications

- Mailbox 1 is used.
- The data length is 1 byte, and the data transmitted is H'AA.
- The transmission end flag is polled while transmission is in progress.
- After confirmation that the transmission end flag has been set, the transmission end flag is cleared, completing the operation.

Reception Specifications

- Mailbox 0 is used.
- The identifier is masked and reception takes place when a match occurs.
- The received data is stored in on-chip RAM, completing the operation.

1.2 Transmission and Reception Specifications, Function Description

Table 1.1 lists the functions allocated to pins and registers.
### Table 1.1 HCAN-2 Function Allocation

<table>
<thead>
<tr>
<th>Pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTxD1</td>
<td>Used to perform message transmission using the HCAN-2 (pin 57).</td>
</tr>
<tr>
<td>HRxD1</td>
<td>Used to perform message reception using the HCAN-2 (pin 56).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBCR1</td>
<td>Port B control registers</td>
</tr>
<tr>
<td>PBCR2</td>
<td>Set to functions of pins HTxD1 and HRxD1.</td>
</tr>
<tr>
<td>MCR</td>
<td>Master control register</td>
</tr>
<tr>
<td></td>
<td>Controls operation of HCAN-2.</td>
</tr>
<tr>
<td>BCR0</td>
<td>Bit timing configuration registers</td>
</tr>
<tr>
<td>BCR1</td>
<td>Used to set baud rate prescaler and bit timing parameters of CAN.</td>
</tr>
<tr>
<td>IRR</td>
<td>Interrupt request register</td>
</tr>
<tr>
<td></td>
<td>Shows status of interrupt sources.</td>
</tr>
<tr>
<td>MBx</td>
<td>Message control register</td>
</tr>
<tr>
<td></td>
<td>Used to set identifier and whether frames are data frames or remote frames.</td>
</tr>
<tr>
<td>MC4</td>
<td>Message control register</td>
</tr>
<tr>
<td></td>
<td>Used to select transmission or reception.</td>
</tr>
<tr>
<td>MC5</td>
<td>Message control register</td>
</tr>
<tr>
<td></td>
<td>Used to set the data length.</td>
</tr>
<tr>
<td>MD7</td>
<td>Message data register</td>
</tr>
<tr>
<td></td>
<td>Stores transmitted and received CAN message data.</td>
</tr>
<tr>
<td>LAFM15</td>
<td>Local acceptance filter mask</td>
</tr>
<tr>
<td></td>
<td>Used to set filter mask for identifier of reception mailbox.</td>
</tr>
<tr>
<td>TXPR</td>
<td>Transmission wait register</td>
</tr>
<tr>
<td></td>
<td>Used to set transmission wait status after transmitted message is stored in mailbox.</td>
</tr>
<tr>
<td>TXACK</td>
<td>Transmission acknowledge register</td>
</tr>
<tr>
<td></td>
<td>Indicates that the transmitted message has been properly transmitted to the corresponding mailbox.</td>
</tr>
<tr>
<td>RXPR</td>
<td>Reception end register</td>
</tr>
<tr>
<td></td>
<td>Indicates that the data has been properly received by the corresponding mailbox.</td>
</tr>
</tbody>
</table>

Rev. 1.00, 08/03, page 2 of 74
1.3 Transmission Flowchart

Figure 1.1 Transmission Flowchart
## 1.4 Software Description (Transmission)

### Module Description

<table>
<thead>
<tr>
<th>Module</th>
<th>Label</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>t_main</td>
<td>Performs initial settings and transmission settings for HCAN-2.</td>
</tr>
</tbody>
</table>

### Description of Registers Used

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
<th>Initial Value</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>PB.PBCR1</td>
<td>Sets pins PB1 and PB0 (pins 56 and 57) to HRxD1 and HTxD1.</td>
<td>0x0000</td>
<td>Main routine</td>
</tr>
<tr>
<td>PB.PBCR2</td>
<td></td>
<td>0x000F</td>
<td></td>
</tr>
<tr>
<td>work</td>
<td>Work register used for mailbox initialization.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>HCAN_MCR</td>
<td>Clears reset request bit.</td>
<td>0x0000</td>
<td></td>
</tr>
<tr>
<td>HCAN_IRR</td>
<td>Clears reset, hold, and sleep interrupt flags. (To clear, write 1.)</td>
<td>0x0001</td>
<td></td>
</tr>
<tr>
<td>HCAN_BCR0</td>
<td>Sets bit rate to 250 kbps when $\phi$ is 50 MHz.</td>
<td>0x0009</td>
<td></td>
</tr>
<tr>
<td>HCAN_BCR1</td>
<td></td>
<td>0x4300</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC0</td>
<td>Sets data frame and standard format for mailbox 1. Also sets identifier (H'555).</td>
<td>0x5550</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC1</td>
<td>Sets mailbox 1 as for transmission.</td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC5</td>
<td>Sets mailbox 1 transmission size to a data length of 1 byte.</td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MD7</td>
<td>Sets mailbox 1 transmission data (H'AA).</td>
<td>0xAA</td>
<td></td>
</tr>
<tr>
<td>HCAN_TXPR0</td>
<td>Sets mailbox 1 to transmission wait status.</td>
<td>0x0002</td>
<td></td>
</tr>
<tr>
<td>HCAN_TXACK0</td>
<td>Checks and clears mailbox 1 transmission end flag. (To clear, write 1.)</td>
<td>0x0002</td>
<td></td>
</tr>
</tbody>
</table>
1.5 Transmission Program Listing

```c
#include <stdio.h>                /* Library function header file */
#include <machine.h>              /* Library function header file */
#include "SH7047.h"               /* Peripheral register definition header file */

void t_main(void){
    unsigned short *work;
    /* Clear bit MCR0 */
    HCAN_MCR = 0x0000;               /* Clear reset request bit */
    /* Clear bit IRR0 */
    HCAN_IRR = 0x0001;      /* Clear reset interrupt flag (to clear, write 1) */
    /* Set pins */
    PB.PBCR1 = 0x0000;               /* Set PB HCAN-2 */
    PB.PBCR2 = 0x000F;               /* Set PB HCAN-2 */
    /* Set bit rate (BCR): bit rate is 250 kbps when \( \phi = 50 \) MHz */
    HCAN_BCR0 = 0x0009;              /* BRP=9(10 system clock) */
    HCAN_BCR1 = 0x4300;              /* TSEG1=4(5tq), TSEG2=3(4tq) */
    /* Initialize mailbox */
    work = (unsigned short *)0xFFFFB100;
    do {
        *work = 0xFFFF;
        work++;
    } while(work < (unsigned short *) 0xFFFFB4F4);
    /* Set MBC */
    HCAN_MB1.MC4 = 0x01;             /* Set mailbox 1 as for transmission */
    /* Set transmission method */
    HCAN_MB1.MC0 = 0x5550;           /* Select data frame and standard format, set identifier */
    HCAN_MB1.MC5 = 0x01;             /* Set data length: 1 byte */
    /* Set transmission data */
    HCAN_MB1.MD7 = 0xAA;             /* Set transmission data: 10101010 */
    /* Set message transmission wait status */
    HCAN_TXPR0 = 0x0002;         /* Set mailbox 1 to transmission wait status */
    /* Wait for transmission end */
    while((HCAN_TXACK0 & 0x0002) != 0x0002);
    /* Clear transmission end flag */
    HCAN_TXACK0 = 0x0002; /* Clear transmission end flag (to clear, write 1) */
    while(1);
}
```

Rev. 1.00, 08/03, page 5 of 74
1.6 Reception Flowchart

Reception operation: main routine

- Initialize reception data storage area
- Clear bit MCR0
- Set local acceptance filter
- Clear bit IRR0
- Set pins
- Set bit rate
- Initialize mailbox
- Set MBC
- Set reception method

Reception complete?

- No
- Yes
  - Store received data
  - Clear reception end flag
- End of reception

Figure 1.2 Reception Flowchart
1.7 Software Description (Reception)

Module Description

<table>
<thead>
<tr>
<th>Module</th>
<th>Label</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>r_main</td>
<td>Performs initial settings and reception settings for HCAN-2.</td>
</tr>
</tbody>
</table>

Description of Registers Used

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
<th>Initial Value</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBbuff</td>
<td>Storage area for received data (address: H'FFFFFD100).</td>
<td>—</td>
<td>Main routine</td>
</tr>
<tr>
<td>work</td>
<td>Work register used for mailbox initialization.</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>PB.PBCR1</td>
<td>Sets pins PB1 and PB0 (pins 56 and 57) to HRxD1 and HTxD1.</td>
<td>0x0000</td>
<td>—</td>
</tr>
<tr>
<td>PB.PBCR2</td>
<td>Sets pins PB1 and PB0 (pins 56 and 57) to HRxD1 and HTxD1.</td>
<td>0x000F</td>
<td>—</td>
</tr>
<tr>
<td>HCAN_MCR</td>
<td>Clears reset request bit.</td>
<td>0x0000</td>
<td>—</td>
</tr>
<tr>
<td>HCAN_IRR</td>
<td>Clears reset, hold, and sleep interrupt flags. (To clear, write 1.)</td>
<td>0x0001</td>
<td>—</td>
</tr>
<tr>
<td>HCAN_BCR0</td>
<td>Sets bit rate to 250 kbps when φ is 50 MHz.</td>
<td>0x0009</td>
<td>—</td>
</tr>
<tr>
<td>HCAN_BCR1</td>
<td>Sets bit rate to 250 kbps when φ is 50 MHz.</td>
<td>0x4300</td>
<td>—</td>
</tr>
<tr>
<td>HCAN_MB0.MC0</td>
<td>Sets data frame and standard format for mailbox 0. Also sets identifier (H'555).</td>
<td>0x5550</td>
<td>—</td>
</tr>
<tr>
<td>HCAN_MB0.MC4</td>
<td>Sets mailbox 0 as for reception.</td>
<td>0x03</td>
<td>—</td>
</tr>
<tr>
<td>HCAN_MB0.MC5</td>
<td>Sets mailbox 0 reception size to a data length of 1 byte.</td>
<td>0x01</td>
<td>—</td>
</tr>
<tr>
<td>HCAN_MB0.LAFM15</td>
<td>Sets identifier filter mask for mailbox 0.</td>
<td>0x0000</td>
<td>—</td>
</tr>
<tr>
<td>HCAN_RXPR0</td>
<td>Checks and clears mailbox 0 reception end flag. (To clear, write 1.)</td>
<td>0x0001</td>
<td>—</td>
</tr>
</tbody>
</table>
1.8 Reception Program Listing

```c
#include <stdio.h>    /* Library function header file */
#include <machine.h>  /* Library function header file */
#include "SH7047.h"   /* Peripheral register definition header file */

/* Function prototype declarations */

void r_main(void);

/* Define constants */

#define MBbuff (*(unsigned char *)0xFFFFD100)  /* Storage for mailbox 0 reception data */

/* Main routine */

void r_main(void){
    unsigned short *work;
    // Initialize storage area for received data
    MBbuff = 0x00;
    // Clear bit MCR0
    HCAN_MCR = 0x0000;  /* Clear reset request bit */
    // Set local acceptance filter
    HCAN_MB0.LAFM15 = 0x0000;  /* Set identifier filter mask for mailbox 0 */
    // Clear bit IRR0
    HCAN_IRR = 0x0001;  /* Clear reset interrupt flag (to clear, write 1) */
    // Set pins
    PB.PBCR1 = 0x0000;  /* Set PB HCAN-2 */
    PB.PBCR2 = 0x000F;  /* Set PB HCAN-2 */
    // Set bit rate (BCR): bit rate is 250 kbps when \( \Phi = 50 \text{ MHz} \)
    HCAN_BCR0 = 0x0009;  /* BRP=9(10 system clock) */
    HCAN_BCR1 = 0x4300;  /* TSEG1=4(5tq),TSEG2=3(4tq) */
    // Initialize mailbox
    work = (unsigned short *)0xFFFFFB100;
    do {
        *work = 0xFFFF;
        work++;
    } while(work < (unsigned short *)0xFFFFFB4F4);
    // Set MBC
    HCAN_MB0.MC4 = 0x03;  /* Set mailbox 0 as for reception */
    // Set reception method
    HCAN_MB0.MC0 = 0x5550;  /* Select data frame and standard format, set identifier */
    HCAN_MB0.MC5 = 0x01;  /* Set data length: 1 byte */
    // Wait for reception end
    while((HCAN_RXPR0 & 0x0001) != 0x0001);  /* Wait for reception end */
}
```

Rev. 1.00, 08/03, page 8 of 74
1.9 Operation Waveforms (Transmission and Reception)

Figure 1.3 shows the waveforms during the operation of this application.
Section 2  Example 2  
(Standard Format, 8 Bytes of Data, Using DTC)

2.1  Transmission and Reception Specifications

Transmission and reception of 8 bytes of data in the standard format using two SH7047F devices and storage of received data using DTC.

Common Transmission and Reception Specifications

- The data transfer speed is 250 kbps (during 50 MHz operation).
- The identifier is H'555.

Transmission Specifications

- Mailbox 1 is used.
- The data length is 8 bytes, and the data transmitted is H'55, H'66, H'77, H'88, H'99, H'AA, H'BB, H'FF.
- The mailbox empty interrupt (IRR8) is used. After message transmission wait status has been set, the transmission end flag is cleared and mailbox empty interrupts are prohibited within the mailbox empty interrupt routine, completing the operation.

Reception Specifications

- Mailbox 0 is used.
- The identifier is masked and reception takes place when a match occurs.
- The message reception interrupt (IRR1) is used.
  (a) When the data is received DTC is triggered by the message reception interrupt, and the received data is stored in on-chip RAM.
  (b) Block transfer mode is used for the DTC transfer, and the 8 bytes of data are transferred as a single block.
  (c) After the DTC transfer is completed, the reception end flag is cleared and message reception interrupts are prohibited within the message reception interrupt routine, completing the operation.

2.2  Transmission and Reception Specifications, Function Description

Tables 2.1 and 2.2 list the functions allocated to registers. (See example 1 for information on pins and port registers.)
<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Common transmission and reception registers</strong></td>
<td></td>
</tr>
</tbody>
</table>
| MCR | Master control register  
Controls operation of HCAN-2. |
| BCR0 | Bit timing configuration registers |
| BCR1 | Used to set baud rate prescaler and bit timing parameters of CAN. |
| IRR | Interrupt request register  
Shows status of interrupt sources. |
| MBx MC0 | Message control register  
Used to set identifier and whether frames are data frames or remote frames. |
| MBx MC4 | Message control register  
Used to select transmission or reception. |
| MBx MC5 | Message control register  
Used to set the data length. |
| MD7 to MD14 | Message data registers  
Store transmitted and received CAN message data. |
| LAFM15 | Local acceptance filter mask  
Used to set filter mask for identifier of reception mailbox. |
| **Transmission registers** | TXPR | Transmission wait register  
Used to set transmission wait status after transmitted message is stored in mailbox. |
| TXACK | Transmission acknowledgment register  
Indicates that the transmitted message has been properly transmitted to the corresponding mailbox. |
| **Reception register** | RXPR | Reception end register  
Indicates that the data has been properly received by the corresponding mailbox. |
| **Interrupt registers** | MBIMR | Mailbox interrupt mask register  
Used to enable interrupt requests for mailboxes. |
| IMR | Interrupt mask register  
Used to enable interrupt requests using IRR interrupt flag. |
| IPRK | Interrupt priority register  
Used to set the priority of HCAN-2 interrupt requests. |
<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTMR</td>
<td>DTC mode register</td>
</tr>
<tr>
<td></td>
<td>Controls the DTC operation mode.</td>
</tr>
<tr>
<td>DTSAR</td>
<td>DTC source address register</td>
</tr>
<tr>
<td></td>
<td>Specifies the source address for data to be transferred using DTC.</td>
</tr>
<tr>
<td>DTDAR</td>
<td>DTC destination address register</td>
</tr>
<tr>
<td></td>
<td>Specifies the destination address for data to be transferred using DTC.</td>
</tr>
<tr>
<td>DTCRA</td>
<td>DTC transfer count register A</td>
</tr>
<tr>
<td></td>
<td>Specifies the number of DTC transfers.</td>
</tr>
<tr>
<td>DTCRB</td>
<td>DTC transfer count register B</td>
</tr>
<tr>
<td></td>
<td>In the block transfer mode, specifies the block length.</td>
</tr>
<tr>
<td>DTBR</td>
<td>DTC database register</td>
</tr>
<tr>
<td></td>
<td>Specifies the top 16 bits of the memory address at which data transferred</td>
</tr>
<tr>
<td></td>
<td>using DTC is to be stored.</td>
</tr>
<tr>
<td>DTEF</td>
<td>DTC enable register</td>
</tr>
<tr>
<td></td>
<td>Selects the interrupt source (RM1 in HCAN-2) for starting DTC.</td>
</tr>
</tbody>
</table>
2.3 Transmission Flowchart

Transmission Flowchart (1)

- Transmission operation: main routine
  - Clear bit MCR0
  - Clear bit IRR0
  - Set pins
  - Set bit rate
  - Initialize mailbox
  - Set MBC
  - Set mailbox empty interrupt
  - Set transmission method
  - Set transmission data
  - Set to transmission wait status
2.4 Software Description (Transmission)

Module Description

<table>
<thead>
<tr>
<th>Module</th>
<th>Label</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>t_main</td>
<td>Performs initial settings and transmission settings for HCAN-2.</td>
</tr>
<tr>
<td>Mailbox empty interrupt</td>
<td>SLE1_IRR8</td>
<td>Clears transmission end flag and sets interrupt prohibition.</td>
</tr>
</tbody>
</table>

Figure 2.2 Transmission Flowchart (2)
### Description of Registers Used (See example 1 for information on pins and port registers.)

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
<th>Initial Value</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>work</td>
<td>Work register used for mailbox initialization.</td>
<td>—</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_MCR</td>
<td>Clears reset request bit.</td>
<td>0x0000</td>
<td></td>
</tr>
<tr>
<td>HCAN_IRR</td>
<td>Clears reset, hold, and sleep interrupt flags. (To clear, write 1.)</td>
<td>0x0001</td>
<td></td>
</tr>
<tr>
<td>HCAN_BCR0</td>
<td>Sets bit rate to 250 kbps when ( \phi ) is 50 MHz.</td>
<td>0x0009</td>
<td></td>
</tr>
<tr>
<td>HCAN_BCR1</td>
<td></td>
<td>0x4300</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC0</td>
<td>Sets data frame and standard format for mailbox 1. Also sets identifier (H'555).</td>
<td>0x5550</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC4</td>
<td>Sets mailbox 1 as for transmission.</td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC5</td>
<td>Sets mailbox 1 transmission size to a data length of 8 bytes.</td>
<td>0x08</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MD7</td>
<td>Sets mailbox 1 byte 1 transmission data (H'55).</td>
<td>0x55</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MD8</td>
<td>Sets mailbox 1 byte 2 transmission data (H'66).</td>
<td>0x66</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MD9</td>
<td>Sets mailbox 1 byte 3 transmission data (H'77).</td>
<td>0x77</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MD10</td>
<td>Sets mailbox 1 byte 4 transmission data (H'88).</td>
<td>0x88</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MD11</td>
<td>Sets mailbox 1 byte 5 transmission data (H'99).</td>
<td>0x99</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MD12</td>
<td>Sets mailbox 1 byte 6 transmission data (H'AA).</td>
<td>0xAA</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MD13</td>
<td>Sets mailbox 1 byte 7 transmission data (H'BB).</td>
<td>0xBB</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MD14</td>
<td>Sets mailbox 1 byte 8 transmission data (H'FF).</td>
<td>0xFF</td>
<td></td>
</tr>
<tr>
<td>HCAN_IMR</td>
<td>Enables mailbox empty interrupts.</td>
<td>0xFEFF</td>
<td></td>
</tr>
<tr>
<td>HCAN_MBIMR0</td>
<td>Enables mailbox 1 interrupt requests.</td>
<td>0xFFFDF</td>
<td></td>
</tr>
<tr>
<td>INTC.IPRK</td>
<td>Sets the priority of HCAN-2 interrupt requests.</td>
<td>0x00F0</td>
<td></td>
</tr>
<tr>
<td>HCAN_TXPR0</td>
<td>Sets mailbox 1 to transmission wait status.</td>
<td>0x0002</td>
<td></td>
</tr>
<tr>
<td>HCAN_TXACK0</td>
<td>Clears mailbox 1 transmission end flag. (To clear, write 1.)</td>
<td>0x0002</td>
<td>Mailbox empty interrupt routine</td>
</tr>
<tr>
<td>HCAN_IMR</td>
<td>Prohibits mailbox empty interrupts.</td>
<td>0xFFFF</td>
<td></td>
</tr>
</tbody>
</table>
2.5 Transmission Program Listing

```c
#include <stdio.h>                /* Library function header file */
#include <machine.h>              /* Library function header file */
#include "SH7047.h"               /* Peripheral register definition header file */

/* Function prototype declarations */

void t_main(void);
void SLE1_IRR8(void);

/* Main routine */

void t_main(void){
    unsigned short *work;
    /* Clear bit MCR0 */
    HCAN_MCR = 0x0000;                         /* Clear reset request bit */
    /* Clear bit IRR0 */
    HCAN_IRR = 0x0001;      /* Clear reset interrupt flag (to clear, write 1) */
    /* Set pins */
    PB.PBCR1 = 0x0000;                         /* Set PB HCAN-2 */
    PB.PBCR2 = 0x000F;                         /* Set PB HCAN-2 */
    /* Set bit rate (BCR): bit rate is 250 kbps when ω = 50 MHz */
    HCAN_BCR0 = 0x0009;                        /* BRP=9(10 system clock) */
    HCAN_BCR1 = 0x4300;                        /* TSEG1=4(5tq), TSEG2=3(4tq) */
    /* Initialize mailbox */
    work = (unsigned short *)0xFFFFB100;
    do {
        *work = 0xFFFF;
        work++;
    } while(work < (unsigned short *)0xFFFFB4F4);
    /* Set MBC */
    HCAN_MB1.MC4 = 0x01;             /* Set mailbox 1 as for transmission */
    /* Set interrupts */
    HCAN.IMR = 0xFFEF;                   /* Enable interrupt IRR8 */
    HCAN_MBMIMRO = 0xFFFFD;             /* Enable mailbox 1 interrupt requests */
    INTC.IPRK = 0x00F0;                /* Set SLE1 priority */
    /* Set transmission method */
    HCAN_MB1.MC0 = 0x5550;             /* Select data frame and standard format, set identifier */
    HCAN_MB1.MC5 = 0x08;              /* Set data length: 8 bytes */
    /* Set transmission data */
    HCAN_MB1.MD7 = 0x55;               /* Set transmission data: 01010101 */
    HCAN_MB1.MD8 = 0x66;               /* Set transmission data: 01100110 */
    HCAN_MB1.MD9 = 0x77;               /* Set transmission data: 01110111 */
    HCAN_MB1.MD10 = 0x88;              /* Set transmission data: 10001000 */
}
```

Rev. 1.00, 08/03, page 16 of 74

RENESAS
HCAN_MB1.MD11 = 0x99;  /* Set transmission data: 10011001 */
HCAN_MB1.MD12 = 0xAA;  /* Set transmission data: 10101010 */
HCAN_MB1.MD13 = 0xBB;  /* Set transmission data: 10111011 */
HCAN_MB1.MD14 = 0xFF;  /* Set transmission data: 11111111 */
/* Set message transmission wait status */
HCAN_TXPR0 = 0x0002;  /* Set mailbox 1 to transmission wait status */
set_imask(0);
while(1);
}

 /***************************************************************************
 /* Mailbox empty interrupt routine                                        */
 /***************************************************************************/
#pragma interrupt(SLE1_IRR8)
void SLE1_IRR8(void){
/* Clear transmission end flag */
  HCAN_TXACK0 = 0x0002;  /* Clear transmission end flag (to clear, write 1) */
/* Prohibit mailbox empty interrupts */
  HCAN_IMR = 0xFFFF;  /* Prohibit interrupt IRR8 */
}

 REKOMBO RESCN
2.6 Reception Flowchart

**Figure 2.3** Reception Flowchart (1)
2.7 Software Description (Reception)

Module Description

<table>
<thead>
<tr>
<th>Module</th>
<th>Label</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>r_main</td>
<td>Performs initial settings and reception settings for HCAN-2.</td>
</tr>
<tr>
<td>Reception interrupt routine</td>
<td>RM1_IRR1</td>
<td>Clears reception end flag and sets interrupt prohibition.</td>
</tr>
</tbody>
</table>
Description of Registers Used (See example 1 for information on pins and port registers.)

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
<th>Initial Value</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBbuff</td>
<td>Storage area for received data (address: H'FFFFD100–H'FFFFD107).</td>
<td>—</td>
<td>Main routine</td>
</tr>
<tr>
<td>work</td>
<td>Work register used for mailbox initialization.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>DTMR</td>
<td>Increments both DTSAR and DTDAR after transfer completes, sets block transfer mode and byte transfer.</td>
<td>0xA890</td>
<td></td>
</tr>
<tr>
<td>DTSAR</td>
<td>Sets transfer source address as mailbox 0 message data area.</td>
<td>0xFFFFB108</td>
<td></td>
</tr>
<tr>
<td>DTDAR</td>
<td>Sets transfer destination address as received data storage area.</td>
<td>0xFFFFD100</td>
<td></td>
</tr>
<tr>
<td>DTCRA</td>
<td>Sets number of block transfers (1).</td>
<td>0x0001</td>
<td></td>
</tr>
<tr>
<td>DTCRB</td>
<td>Sets block length (8 bytes).</td>
<td>0x0008</td>
<td></td>
</tr>
<tr>
<td>DTC.DTBR</td>
<td>Sets top 16 bits of memory address at which data transferred using DTC is stored.</td>
<td>0xFFFF</td>
<td></td>
</tr>
<tr>
<td>DTC.DTEF</td>
<td>Selects the interrupt source (RM1 in HCAN-2) for starting DTC.</td>
<td>0x04</td>
<td></td>
</tr>
<tr>
<td>HCAN_MCR</td>
<td>Clears reset request bit.</td>
<td>0x0000</td>
<td></td>
</tr>
<tr>
<td>HCAN_I RR</td>
<td>Clears reset, hold, and sleep interrupt flags. (To clear, write 1.)</td>
<td>0x0001</td>
<td></td>
</tr>
<tr>
<td>HCAN_BCR0</td>
<td>Sets bit rate to 250 kbps when φ is 50 MHz.</td>
<td>0x0009</td>
<td></td>
</tr>
<tr>
<td>HCAN_BCR1</td>
<td></td>
<td>0x4300</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB0.MC0</td>
<td>Sets data frame and standard format for mailbox 0. Also sets identifier (H'555).</td>
<td>0x5550</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB0.MC4</td>
<td>Sets mailbox 0 as for reception.</td>
<td>0x03</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB0.MC5</td>
<td>Sets mailbox 0 reception size to a data length of 8 bytes.</td>
<td>0x08</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB0.LAFM15</td>
<td>Sets identifier filter mask for mailbox 0.</td>
<td>0x0000</td>
<td></td>
</tr>
<tr>
<td>HCAN_IMR</td>
<td>Enables message reception interrupts.</td>
<td>0xFFFFD</td>
<td></td>
</tr>
<tr>
<td>HCAN_MBIMR0</td>
<td>Enables mailbox 0 interrupt requests.</td>
<td>0xFFFFE</td>
<td></td>
</tr>
<tr>
<td>INTC.IPRK</td>
<td>Sets priority of HCAN-2 interrupt requests.</td>
<td>0x00F0</td>
<td></td>
</tr>
<tr>
<td>HCAN_RXPR0</td>
<td>Clears mailbox 0 reception end flag. (To clear, write 1.)</td>
<td>0x0001</td>
<td>Reception interrupt routine</td>
</tr>
<tr>
<td>HCAN.IMR</td>
<td>Prohibits message reception interrupts.</td>
<td>0xFFFF</td>
<td></td>
</tr>
</tbody>
</table>
2.8 Reception Program Listing

```c
#include <stdio.h>
#include <machine.h>
#include "SH7047.h"

#define DTMR (*(unsigned short *)0xFFFFD080) /* DTC register data */
#define DTCRA (*(unsigned short *)0xFFFFD082) /* DTC register data */
#define DTCRB (*(unsigned short *)0xFFFFD086) /* DTC register data */
#define DTSAR (*(unsigned long *)0xFFFFD088) /* DTC register data */
#define DTDAR (*(unsigned long *)0xFFFFD08C) /* DTC register data */
#define MBbuff (*(unsigned char *)0xFFFFD100) /* Storage area for received data */

void r_main(void);
void RM1_IRR1(void);

void r_main(void){
  unsigned short *work;

  /* Initialize storage area for received data */
  work = (unsigned short *)0xFFFFD100;
  do {
    *work = 0x0000;
    work++;
  } while(work < (unsigned short *)0xFFFFD108);

  /* DTC settings */
  DTMR = 0xA890;                         /* Increment both DTSAR and DTDAR after transfer completes, set block transfer mode and byte transfer */
  DTSAR = (unsigned long)&HCAN_MB0.MD7; /* Set transfer source address */
  DTDAR = (unsigned long)&MBbuff;       /* Set transfer destination address */
  DTCRA = 0x0001;                       /* Block transfer: 1 block */
  DTCRB = 0x0008;                       /* Block length: 8 bytes */
  DTC.DTBR = 0xFFFF;                    /* Register data base address */
  DTC.DTEF |= 0x04;                     /* HCAN-2 interrupt (RM1) */

  /* Clear bit MCR0 */
  HCAN_MCR = 0x0000;                    /* Clear reset request bit */

  /* Set local acceptance filter */
  HCAN_MB0.LAFM15 = 0x0000;            /* Set identifier filter mask for mailbox 0 */

  /* Set pins */
}
```

Rev. 1.00, 08/03, page 21 of 74
PB.PBCR1 = 0x0000;  /* Set PB HCAN-2 */
PBCR2 = 0x000F;  /* Set PB HCAN-2 */
/* Set bit rate (BCR): bit rate is 250 kbps when φ = 50 MHz */
HCAN_BCR0 = 0x0009;  /* BRP=9(10 system clock) */
HCAN_BCR1 = 0x4300;  /* TSEG1=4(5tq),TSEG2=3(4tq) */
/* Initialize mailbox */
work = (unsigned short *)0xFFFFB100;
do {
    *work = 0xFFFF;
    work++;
} while(work < (unsigned short *)0xFFFFB4F4);
/* Set MBC */
HCAN_MB0.MC4 = 0x03;  /* Set mailbox 0 as for reception */
/* Set interrupts */
HCAN_IMR = 0xFFF0;  /* Enable message reception interrupts */
HCAN_MBIMR0 = 0xFFFE;  /* Enable mailbox 0 reception interrupt requests */
INTC.IPRK = 0x00F0;  /* Set RM1 priority */
/* Set reception method */
HCAN_MB0.MC0 = 0x5550;  /* Select data frame and standard format, set identifier */
HCAN_MB0.MC5 = 0x08;  /* Set data length: 8 bytes */
set_imask(0);
while(1);

/************************************************************
 /* Reception interrupt routine */
 ************************************************************/
#pragma interrupt(RM1_IRR1)
void RM1_IRR1(void){
    /* Clear reception end register */
    HCAN_RXPR0 = 0x0001;  /* Clear reception end flag (to clear, write 1) */
    /* Prohibit reception interrupts */
    HCAN_IMR = 0xFFFF;  /* Prohibit message reception interrupts */
}

Rev. 1.00, 08/03, page 22 of 74
2.9 Operation Waveforms (Transmission and Reception)

Figure 2.5 shows the waveforms during the operation of this application.
Section 3  Example 3  
(Extended Format, 1 Byte of Data)

3.1 Transmission and Reception Specifications

Transmission and reception of 1 byte of data in the extended format using two SH7047F devices.

Common Transmission and Reception Specifications

- The data transfer speed is 250 kbps (during 50 MHz operation).
- The standard identifier is H'555, and the extended identifier is H'2AAAA.

Transmission Specifications

- Mailbox 1 is used.
- The data length is 1 byte, and the data transmitted is H'AA.
- The mailbox empty interrupt (IRR8) is used. After message transmission wait status has been set, the transmission end flag is cleared and mailbox empty interrupts are prohibited within the mailbox empty interrupt routine, completing the operation.

Reception Specifications

- Mailbox 0 is used.
- The identifier is masked and reception takes place when a match occurs.
- The message reception interrupt (IRR1) is used. The received data is stored in on-chip RAM, the reception end flag is cleared, and message reception interrupts are prohibited within the message reception interrupt routine, completing the operation.

3.2 Transmission and Reception Specifications, Function Description

Table 3.1 lists the functions allocated to registers. (See example 1 for information on pins and port registers.)
### Table 3.1 HCAN-2 Function Allocation

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Common transmission and reception registers</strong></td>
<td></td>
</tr>
<tr>
<td>MCR</td>
<td>Master control register</td>
</tr>
<tr>
<td></td>
<td>Controls operation of HCAN-2.</td>
</tr>
<tr>
<td>BCR0</td>
<td>Bit timing configuration registers</td>
</tr>
<tr>
<td>BCR1</td>
<td>Used to set baud rate prescaler and bit timing parameters of CAN.</td>
</tr>
<tr>
<td>IRR</td>
<td>Interrupt request register</td>
</tr>
<tr>
<td></td>
<td>Shows status of interrupt sources.</td>
</tr>
<tr>
<td>MBx</td>
<td>Message control register</td>
</tr>
<tr>
<td>MC0 to MC2</td>
<td>Used to set identifier and whether frames are data frames or remote frames.</td>
</tr>
<tr>
<td>MC4</td>
<td>Message control register</td>
</tr>
<tr>
<td></td>
<td>Used to select transmission or reception.</td>
</tr>
<tr>
<td>MC5</td>
<td>Message control register</td>
</tr>
<tr>
<td></td>
<td>Used to set the data length.</td>
</tr>
<tr>
<td>MD7</td>
<td>Message data registers</td>
</tr>
<tr>
<td></td>
<td>Store transmitted and received CAN message data.</td>
</tr>
<tr>
<td>LAFM15</td>
<td>Local acceptance filter mask</td>
</tr>
<tr>
<td>LAFM17</td>
<td>Used to set filter mask for identifier of reception mailbox.</td>
</tr>
<tr>
<td><strong>Transmission registers</strong></td>
<td></td>
</tr>
<tr>
<td>TXPR</td>
<td>Transmission wait register</td>
</tr>
<tr>
<td></td>
<td>Used to set transmission wait status after transmitted message is stored in mailbox.</td>
</tr>
<tr>
<td>TXACK</td>
<td>Transmission acknowledge register</td>
</tr>
<tr>
<td></td>
<td>Indicates that the transmitted message has been properly transmitted to the corresponding mailbox.</td>
</tr>
<tr>
<td><strong>Reception register</strong></td>
<td></td>
</tr>
<tr>
<td>RXPR</td>
<td>Reception end register</td>
</tr>
<tr>
<td></td>
<td>Indicates that the data has been properly received by the corresponding mailbox.</td>
</tr>
<tr>
<td><strong>Interrupt registers</strong></td>
<td></td>
</tr>
<tr>
<td>MBIMR</td>
<td>Mailbox interrupt mask register</td>
</tr>
<tr>
<td></td>
<td>Used to enable interrupt requests for mailboxes.</td>
</tr>
<tr>
<td>IMR</td>
<td>Interrupt mask register</td>
</tr>
<tr>
<td></td>
<td>Used to enable interrupt requests using IRR interrupt flag.</td>
</tr>
<tr>
<td>IPRK</td>
<td>Interrupt priority register</td>
</tr>
<tr>
<td></td>
<td>Used to set the priority of HCAN-2 interrupt requests.</td>
</tr>
</tbody>
</table>
3.3 Transmission Flowchart

Transmission operation: main routine

- Clear bit MCR0
- Clear bit IRR0
- Set pins
- Set bit rate
- Initialize mailbox
- Set MBC
- Set mailbox empty interrupt
- Set transmission method
- Set transmission data
- Set to transmission wait status

Figure 3.1 Transmission Flowchart (1)
Figure 3.1 Transmission Flowchart (2)
### 3.4 Software Description (Transmission)

#### Module Description

<table>
<thead>
<tr>
<th>Module</th>
<th>Label</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>t_main</td>
<td>Performs initial settings and transmission settings for HCAN-2.</td>
</tr>
<tr>
<td>Mailbox empty interrupt</td>
<td>SLE1_IRR8</td>
<td>Clears transmission end flag and sets interrupt prohibition.</td>
</tr>
</tbody>
</table>

#### Description of Registers Used (See example 1 for information on pins and port registers.)

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
<th>Initial Value</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>work</td>
<td>Work register used for mailbox initialization.</td>
<td>—</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_MCR</td>
<td>Clears reset request bit.</td>
<td>0x0000</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_IRR</td>
<td>Clears reset, hold, and sleep interrupt flags. (To clear, write 1.)</td>
<td>0x0001</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_BCR0</td>
<td>Sets bit rate to 250 kbps when φ is 50 MHz.</td>
<td>0x0009</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_BCR1</td>
<td></td>
<td>0x4300</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_MB1.MC0</td>
<td>Sets data frame and extended format for mailbox 1. Also sets standard identifier (H'555) and extended identifier (H'2AAAA).</td>
<td>0x5556</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_MB1.MC2</td>
<td></td>
<td>0xAAAA</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_MB1.MC4</td>
<td>Sets mailbox 1 as for transmission.</td>
<td>0x01</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_MB1.MC5</td>
<td>Sets mailbox 1 transmission size to a data length of 1 byte.</td>
<td>0x01</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_MB1.MD7</td>
<td>Sets mailbox 1 transmission data (H'AA).</td>
<td>0xAA</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_IMR</td>
<td>Enables mailbox empty interrupts.</td>
<td>0xFEFF</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_MBIMR0</td>
<td>Enables mailbox 1 interrupt requests.</td>
<td>0xFFFF</td>
<td>Main routine</td>
</tr>
<tr>
<td>INTC.IPRK</td>
<td>Sets the priority of HCAN-2 interrupt requests.</td>
<td>0x00F0</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_TXPR0</td>
<td>Sets mailbox 1 to transmission wait status.</td>
<td>0x0002</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_TXACK0</td>
<td>Clears mailbox 1 transmission end flag. (To clear, write 1.)</td>
<td>0x0002</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_IMR</td>
<td>Prohibits mailbox empty interrupts.</td>
<td>0xFFFF</td>
<td>Main routine</td>
</tr>
</tbody>
</table>
3.5 Transmission Program Listing

```c
#include <stdio.h>                /* Library function header file */
#include <machine.h>              /* Library function header file */
#include "SH7047.h"               /* Peripheral register definition header file */

void SLE1_IRR8(void);

void t_main(void);

unsigned short *work;

HCAN_MCR = 0x0000;                      /* Clear reset request bit */
HCAN_IRR = 0x0001;      /* Clear reset interrupt flag (to clear, write 1) */
PB.PBCR1 = 0x0000;                      /* Set PB HCAN-2 */
PB.PBCR2 = 0x000F;                      /* Set PB HCAN-2 */
HCAN_BCR0 = 0x0009;                     /* BRP=9(10 system clock) */
HCAN_BCR1 = 0x4300;                     /* TSEG1=4(5tq),TSEG2=3(4tq) */
work = (unsigned short *)0xFFFFB100;
do {
    *work = 0xFFFF;
    work++;
} while(work < (unsigned short *)0xFFFFB4F4);
HCAN_MB1.MC4 = 0x01;               /* Set mailbox 1 as for transmission */
HCAN_IMR = 0xFEFF;                 /* Enable interrupt IRR8 */
HCAN_MB1IMR = 0xFFFD;              /* Enable mailbox 1 interrupt requests */
INTC.IPRK = 0x00F0;                /* Set SLE1 priority */
HCAN_MB1.MC0 = 0x5556;             /* Select data frame and extended format, set identifier */
HCAN_MB1.MC2 = 0xA0AA;             /* Set (extended) identifier */
HCAN_MB1.MC5 = 0x01;               /* Set data length: 1 byte */
HCAN_MB1.MD7 = 0x0AA;              /* Set transmission data: 10101010 */
HCAN_TXPR0 = 0x0002;              /* Set mailbox 1 to transmission wait status */
set_imask(0);
```

Rev. 1.00, 08/03, page 29 of 74
while(1);
}

/* Mailbox empty interrupt routine */
#pragma interrupt(SLE1_IRR8)
void SLE1_IRR8(void){
/* Clear transmission end flag */
    HCAN_TXACK0 = 0x0002; /* Clear transmission end flag (to clear, write 1) */
/* Prohibit mailbox empty interrupts */
    HCAN_IMR = 0xFFFF; /* Prohibit interrupt IRR8 */
}
3.6 Reception Flowchart

Figure 3.3 Reception Flowchart (1)
Reception interrupt routine
- Clear reception end flag
- Store received data
- Prohibit reception interrupts
- RTE

Figure 3.4  Reception Flowchart (2)
### 3.7 Software Description (Reception)

<table>
<thead>
<tr>
<th>Module</th>
<th>Label</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>r_main</td>
<td>Performs initial settings and reception settings for HCAN-2.</td>
</tr>
<tr>
<td>Reception interrupt</td>
<td>RM1_IRR1</td>
<td>Clears reception end flag and sets interrupt prohibition.</td>
</tr>
</tbody>
</table>

#### Description of Registers Used (See example 1 for information on pins and port registers.)

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
<th>Initial Value</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBbuff</td>
<td>Storage area for received data (address: H'FFFFD100).</td>
<td>—</td>
<td>Main routine</td>
</tr>
<tr>
<td>work</td>
<td>Work register used for mailbox initialization.</td>
<td>—</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_MCR</td>
<td>Clears reset request bit.</td>
<td>0x0000</td>
<td>HCAN_MCR</td>
</tr>
<tr>
<td>HCAN_IRR</td>
<td>Clears reset, hold, and sleep interrupt flags. (To clear, write 1.)</td>
<td>0x0001</td>
<td>HCAN_IRR</td>
</tr>
<tr>
<td>HCAN_BCR0</td>
<td>Sets bit rate to 250 kbps when $\phi$ is 50 MHz.</td>
<td>0x0009</td>
<td>HCAN_BCR0</td>
</tr>
<tr>
<td>HCAN_BCR1</td>
<td></td>
<td>0x4300</td>
<td>HCAN_BCR1</td>
</tr>
<tr>
<td>HCAN_MB0.MC0</td>
<td>Sets data frame and extended format for mailbox 1. Also sets standard identifier (H’555) and extended identifier (H’2AAAA).</td>
<td>0x5556</td>
<td>HCAN_MB0.MC0</td>
</tr>
<tr>
<td>HCAN_MB0.MC2</td>
<td></td>
<td>0xAAAA</td>
<td>HCAN_MB0.MC2</td>
</tr>
<tr>
<td>HCAN_MB0.MC4</td>
<td>Sets mailbox 0 as for reception.</td>
<td>0x03</td>
<td>HCAN_MB0.MC4</td>
</tr>
<tr>
<td>HCAN_MB0.MC5</td>
<td>Sets mailbox 0 reception size to a data length of 1 byte.</td>
<td>0x01</td>
<td>HCAN_MB0.MC5</td>
</tr>
<tr>
<td>HCAN_MB0.LAFM15</td>
<td>Sets identifier filter mask for mailbox 0.</td>
<td>0x0000</td>
<td>HCAN_MB0.LAFM15</td>
</tr>
<tr>
<td>HCAN_MB0.LAFM17</td>
<td>Sets identifier filter mask for mailbox 0.</td>
<td>0x0000</td>
<td>HCAN_MB0.LAFM17</td>
</tr>
<tr>
<td>HCAN_IMR</td>
<td>Enables message reception interrupts.</td>
<td>0xFFFD</td>
<td>HCAN_IMR</td>
</tr>
<tr>
<td>HCAN_MBIMR0</td>
<td>Enables mailbox 0 interrupt requests.</td>
<td>0xFFF0</td>
<td>HCAN_MBIMR0</td>
</tr>
<tr>
<td>INTC.IPRK</td>
<td>Sets priority of HCAN-2 interrupt requests.</td>
<td>0xFFF0</td>
<td>INTC.IPRK</td>
</tr>
<tr>
<td>HCAN_RXPR0</td>
<td>Clears mailbox 0 reception end flag. (To clear, write 1.)</td>
<td>0x0001</td>
<td>HCAN_RXPR0</td>
</tr>
<tr>
<td>HCAN IMR</td>
<td>Prohibits message reception interrupts.</td>
<td>0xFFFF</td>
<td>HCAN IMR</td>
</tr>
</tbody>
</table>
3.8 Reception Program Listing

```c
#include <stdio.h>                /* Library function header file */
#include <machine.h>              /* Library function header file */
#include "SH7047.h"               /* Peripheral register definition header file */

/* Function prototype declarations */
void r_main(void);
void RM1_IRR1(void);

/* Define constants */
#define MBbuff (*(unsigned char *)0xFFFFD100) /* Storage area for received data */

/* Main routine */
void r_main(void){
  unsigned short *work;
  /* Initialize storage area for received data */
  MBbuff = 0x00;
  /* Clear bit MCR0 */
  HCAN_MCR = 0x0000; /* Clear reset request bit */
  /* Set local acceptance filter */
  HCAN_MB0.LAFM15 = 0x0000; /* Set identifier filter mask for mailbox 0 */
  HCAN_MB0.LAFM17 = 0x0000; /* Set identifier filter mask for mailbox 0 */
  /* Clear bit IRR0 */
  HCAN_IRR = 0x0001; /* Clear reset interrupt flag (to clear, write 1) */
  /* Set pins */
  PB.PBCR1 = 0x0000; /* Set PB HCAN-2 */
  PB.PBCR2 = 0x000F; /* Set PB HCAN-2 */
  /* Set bit rate (BCR): bit rate is 250 kbps when φ = 50 MHz */
  HCAN_BCR0 = 0x0009; /* BRP=9(10 system clock) */
  HCAN_BCR1 = 0x4300; /* TSEG1=4(5tq),TSEG2=3(4tq) */
  /* Initialize mailbox */
  work = (unsigned short *)0xFFFFB100;
  do {
    *work = 0xFFFF;
    work++;
  } while(work < (unsigned short *)0xFFFFB4F4);
  /* Set MBC */
  HCAN_MB0.MC4 = 0x03; /* Set mailbox 0 as for reception */
  /* Set interrupts */
  HCAN_IMR = 0xFFFFD; /* Enable message reception interrupts */
  HCAN_MBIMR0 = 0xFFFE; /* Enable mailbox 0 reception interrupt requests */
  INTC.IPRK = 0x00F0; /* Set RM1 priority */
}
```

Rev. 1.00, 08/03, page 34 of 74
/ * Set reception method */
    HCAN_MB0.MC0 = 0x5556; /* Select data frame and extended format,
set identifier */
    HCAN_MB0.MC2 = 0xAAAA; /* Set (extended) identifier */
    HCAN_MB0.MC5 = 0x01;   /* Set data length: 1 byte */

    set_imask(0);
    while(1);
}

/********************************************************************************
/*     Reception interrupt routine                                              */
********************************************************************************/
#pragma interrupt(RM1_IRR1)
void RM1_IRR1(void){
    /* Clear reception end register */
    HCAN_RXPR0 = 0x0001;   /* Clear reception end flag (to clear, write 1) */
    /* Store received data */
    MBbuff = HCAN_MB0.MD7;
    /* Prohibit reception interrupts */
    HCAN_IMR = 0xFFFF;     /* Prohibit interrupt IRR1 */
}

3.9 Operation Waveforms (Transmission and Reception)

Figure 3.5 shows the waveforms during the operation of this application.

Figure 3.5 Operation Waveforms
Section 4  Example 4  
(Standard Format, 8 Bytes of Data, Prioritized)

4.1 Transmission and Reception Specifications

Using two SH7047F devices, transmission of 8 bytes of data in the standard format from multiple mailboxes using identifier priority settings and reception by mailbox 0 only.

Common Transmission and Reception Specifications

- The data transfer speed is 250 kbps (during 50 MHz operation).

Transmission Specifications

- Mailboxes 1 to 15 are used.
- Messages are transmitted based on identifier priority settings.
- The data length for each mailbox is 8 bytes. The identifier and data for each mailbox are listed in table 4.1.

Table 4.1  Mailbox Settings

<table>
<thead>
<tr>
<th>MBx</th>
<th>Identifier (11 Bits)</th>
<th>Data (8 Bytes)</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0x666</td>
<td>0x1111 1111 1111 1111</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>0x2AA</td>
<td>0x2222 2222 2222 2222</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0x777</td>
<td>0x3333 3333 3333 3333</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>0x333</td>
<td>0x4444 4444 4444 4444</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>0x088</td>
<td>0x5555 5555 5555 5555</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0x4CC</td>
<td>0x6666 6666 6666 6666</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>0x199</td>
<td>0x7777 7777 7777 7777</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>0x7FF</td>
<td>0x8888 8888 8888 8888</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>0x111</td>
<td>0x9999 9999 9999 9999</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>0x444</td>
<td>0xA AAAA AAAAA AAAA</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>0x555</td>
<td>0xB BBBB BBBB BBBB</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>0x6EE</td>
<td>0xC CCC CCC CCC CCC</td>
<td>13</td>
</tr>
<tr>
<td>13</td>
<td>0x3BB</td>
<td>0xDDDD DDDD DDDD DDDD</td>
<td>7</td>
</tr>
<tr>
<td>14</td>
<td>0x222</td>
<td>0xEEEE EEEE EEEE EEEE</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>0x5DD</td>
<td>0xFFFF FFFF FFFF FFFF</td>
<td>11</td>
</tr>
</tbody>
</table>
• The data from mailboxes 1 to 15 is transmitted in a single batch.
• The transmission end flag is polled while transmission is in progress.
• After confirmation that the transmission end flag has been set, the transmission end flag is cleared, completing the operation.

Reception Specifications

• Mailbox 0 is used.
• Identifiers are not masked and all transmissions are received.
• The message reception interrupt (IRR1) is used.
  (a) When data is received DTC is triggered by the message reception interrupt, and the received data is stored in on-chip RAM.
  (b) Block transfer mode is used for DTC transfers. Fifteen blocks are transferred, with each block containing 8 bytes of data.
  (c) After the DTC transfers are completed, the reception end flag is cleared and message reception interrupts are prohibited within the message reception interrupt routine, completing the operation.

4.2 Transmission and Reception Specifications, Function Description

Tables 4.2 and 4.3 list the functions allocated to registers. (See example 1 for information on pins and port registers.)
Table 4.2  HCAN-2 Function Allocation

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common transmission and reception</td>
<td></td>
</tr>
<tr>
<td>registers</td>
<td></td>
</tr>
<tr>
<td>MCR</td>
<td>Master control register</td>
</tr>
<tr>
<td></td>
<td>Controls operation of HCAN-2.</td>
</tr>
<tr>
<td>BCR0</td>
<td>Bit timing configuration registers</td>
</tr>
<tr>
<td>BCR1</td>
<td>Used to set baud rate prescaler and bit timing parameters of CAN.</td>
</tr>
<tr>
<td>IRR</td>
<td>Interrupt request register</td>
</tr>
<tr>
<td></td>
<td>Shows status of interrupt sources.</td>
</tr>
<tr>
<td>MBx</td>
<td>Message control register</td>
</tr>
<tr>
<td></td>
<td>Used to set identifier and whether frames are data frames or remote frames.</td>
</tr>
<tr>
<td>MC0</td>
<td>Message control register</td>
</tr>
<tr>
<td></td>
<td>Used to select transmission or reception.</td>
</tr>
<tr>
<td>MC4</td>
<td>Message control register</td>
</tr>
<tr>
<td></td>
<td>Used to set the data length.</td>
</tr>
<tr>
<td>MD7 to MD14</td>
<td>Message data registers</td>
</tr>
<tr>
<td></td>
<td>Store transmitted and received CAN message data.</td>
</tr>
<tr>
<td>LAFM15</td>
<td>Local acceptance filter mask</td>
</tr>
<tr>
<td></td>
<td>Used to set filter mask for identifier of reception mailbox.</td>
</tr>
<tr>
<td>Transmission registers</td>
<td></td>
</tr>
<tr>
<td>TXPR</td>
<td>Transmission wait register</td>
</tr>
<tr>
<td></td>
<td>Used to set transmission wait status after transmitted message is stored in mailbox.</td>
</tr>
<tr>
<td>TXACK</td>
<td>Transmission acknowledge register</td>
</tr>
<tr>
<td></td>
<td>Indicates that the transmitted message has been properly transmitted to the corresponding mailbox.</td>
</tr>
<tr>
<td>Reception register</td>
<td></td>
</tr>
<tr>
<td>RXPR</td>
<td>Reception end register</td>
</tr>
<tr>
<td></td>
<td>Indicates that the data has been properly received by the corresponding mailbox.</td>
</tr>
<tr>
<td>Interrupt registers</td>
<td></td>
</tr>
<tr>
<td>MBIMR</td>
<td>Mailbox interrupt mask register</td>
</tr>
<tr>
<td></td>
<td>Used to enable interrupt requests for mailboxes.</td>
</tr>
<tr>
<td>IMR</td>
<td>Interrupt mask register</td>
</tr>
<tr>
<td></td>
<td>Used to enable interrupt requests using IRR interrupt flag.</td>
</tr>
<tr>
<td>IPRK</td>
<td>Interrupt priority register</td>
</tr>
<tr>
<td></td>
<td>Used to set the priority of HCAN-2 interrupt requests.</td>
</tr>
<tr>
<td>Register</td>
<td>Function</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
</tr>
</tbody>
</table>
| DTMR     | DTC mode register  
           | Controls the DTC operation mode. |
| DTSAR    | DTC source address register  
           | Specifies the source address for data to be transferred using DTC. |
| DTDAR    | DTC destination address register  
           | Specifies the destination address for data to be transferred using DTC. |
| DTCRA    | DTC transfer count register A  
           | Specifies the number of DTC transfers. |
| DTCRB    | DTC transfer count register B  
           | In the block transfer mode, specifies the block length. |
| DTBR     | DTC database register  
           | Specifies the top 16 bits of the memory address at which data transferred using DTC is to be stored. |
| DTEF     | DTC enable register  
           | Selects the interrupt source (RM1 in HCAN-2) for starting DTC. |
4.3 Transmission Flowchart

Transmission operation: main routine

- Clear bit MCR0
- Clear bit IRR0
- Set pins
- Set bit rate
- Initialize mailboxes

- Mailbox 1
  - Set MBC
  - Set transmission method
  - Set transmission data

- Mailbox 2
  - Set MBC
  - Set transmission method
  - Set transmission data

- Mailbox 14
  - Set MBC
  - Set transmission method
  - Set transmission data

- Mailbox 15
  - Set MBC
  - Set transmission method
  - Set transmission data

Figure 4.1 Transmission Flowchart (1)
Figure 4.2 Transmission Flowchart (2)
### 4.4 Software Description (Transmission)

#### Module Description

<table>
<thead>
<tr>
<th>Module</th>
<th>Label</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>t_main</td>
<td>Performs initial settings and transmission settings for HCAN-2.</td>
</tr>
</tbody>
</table>

#### Description of Registers Used (See example 1 for information on pins and port registers.)

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
<th>Initial Value</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>work</td>
<td>Work register used for mailbox initialization.</td>
<td>—</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_MCR</td>
<td>Clears reset request bit. Also sets transmissions based on priority of mailbox identifier.</td>
<td>0x0000</td>
<td></td>
</tr>
<tr>
<td>HCAN_IRR</td>
<td>Clears reset, hold, and sleep interrupt flags. (To clear, write 1.)</td>
<td>0x0001</td>
<td></td>
</tr>
<tr>
<td>HCAN_BCR0</td>
<td>Sets bit rate to 250 kbps when $\phi$ is 50 MHz.</td>
<td>0x0009</td>
<td></td>
</tr>
<tr>
<td>HCAN_BCR1</td>
<td>Sets bit rate to 250 kbps when $\phi$ is 50 MHz.</td>
<td>0x4300</td>
<td></td>
</tr>
<tr>
<td>HCAN_MBx.MC0</td>
<td>Sets data frame and standard format for mailboxes 1 to 15. Also sets identifiers (see table 4.1 for setting values).</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>HCAN_MBx.MC4</td>
<td>Sets mailboxes 1 to 15 as for transmission.</td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td>HCAN_MBx.MC5</td>
<td>Sets transmission size for mailboxes 1 to 15 to a data length of 8 bytes.</td>
<td>0x08</td>
<td></td>
</tr>
<tr>
<td>HCAN_MBx.MD7 to 14</td>
<td>Sets transmission data for mailboxes 1 to 15. Also sets identifiers (see table 4.1 for setting values).</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>HCAN_TXPR0</td>
<td>Sets mailboxes 1 to 15 to transmission wait status.</td>
<td>0xFFFE</td>
<td></td>
</tr>
<tr>
<td>HCAN_TXACK0</td>
<td>Checks and clears transmission end flags for mailboxes 1 to 15. (To clear, write 1.)</td>
<td>0xFFFE</td>
<td></td>
</tr>
</tbody>
</table>
4.5 Transmission Program Listing

/***************************************************************/
/* HCAN-2 Transmission Program (Example 4) */
/***************************************************************/
#include <stdio.h> /* Library function header file */
#include <machine.h> /* Library function header file */
#include "SH7047.h" /* Peripheral register definition header file */
/**************************************************************/
/* Function prototype declaration */
/**************************************************************/
void t_main(void);
/**************************************************************/
/* Main routine */
/**************************************************************/
void t_main(void)
{
    unsigned short *work;

    /* Clear bit MCR0 */
    HCAN_MCR = 0x0000; /* Clear reset request bit, identifier priority-based transmission */
    /* Clear bit IRR0 */
    HCAN_IRR = 0x0001; /* Clear reset interrupt flag (to clear, write 1) */
    /* Set pins */
    PB.PBCR1 = 0x0000; /* Set PB HCAN-2 */
    PB.PBCR2 = 0x000F; /* Set PB HCAN-2 */
    /* Set bit rate (BCR): bit rate is 250 kbps when φ = 50 MHz */
    HCAN_BCR0 = 0x0009; /* BRP=9(10 system clock) */
    HCAN_BCR1 = 0x4300; /* TSEG1=4(5tq),TSEG2=3(4tq) */
    /* Initialize mailboxes */
    work = (unsigned short *)0xFFFFB100;
    do {
        *work = 0xFFFF;
        work++;
    } while(work < (unsigned short *)0xFFFFB4F4);

    /* Set transmission data */
    /** MB1 **/
    HCAN_MB1.MC4 = 0x01; /* Set mailbox 1 as for transmission */
    HCAN_MB1.MC0 = 0x6660; /* Select data frame and standard format, set identifier */
    HCAN_MB1.MC5 = 0x08; /* Set data length: 8 bytes */
    HCAN_MB1.MD7 = 0x11; /* Set transmission data: 00010001 */
    HCAN_MB1.MD8 = 0x11; /* Set transmission data: 00010001 */
    HCAN_MB1.MD9 = 0x11; /* Set transmission data: 00010001 */
    HCAN_MB1.MD10 = 0x11; /* Set transmission data: 00010001 */
    HCAN_MB1.MD11 = 0x11; /* Set transmission data: 00010001 */
    HCAN_MB1.MD12 = 0x11; /* Set transmission data: 00010001 */
    HCAN_MB1.MD13 = 0x11; /* Set transmission data: 00010001 */
    HCAN_MB1.MD14 = 0x11; /* Set transmission data: 00010001 */
}
/** MB2 **/  
HCAN_MB2.MC4 = 0x01; /* Set mailbox 2 as for transmission */
HCAN_MB2.MC0 = 0x2AA0; /* Select data frame and standard format, set identifier */
HCAN_MB2.MC5 = 0x08; /* Set data length: 8 bytes */
HCAN_MB2.MD7 = 0x22; /* Set transmission data: 00100010 */
HCAN_MB2.MD8 = 0x22; /* Set transmission data: 00100010 */
HCAN_MB2.MD9 = 0x22; /* Set transmission data: 00100010 */
HCAN_MB2.MD10 = 0x22; /* Set transmission data: 00100010 */
HCAN_MB2.MD11 = 0x22; /* Set transmission data: 00100010 */
HCAN_MB2.MD12 = 0x22; /* Set transmission data: 00100010 */
HCAN_MB2.MD13 = 0x22; /* Set transmission data: 00100010 */
HCAN_MB2.MD14 = 0x22; /* Set transmission data: 00100010 */

/*** MB3 **/  
HCAN_MB3.MC4 = 0x01; /* Set mailbox 3 as for transmission */
HCAN_MB3.MC0 = 0x7770; /* Select data frame and standard format, set identifier */
HCAN_MB3.MC5 = 0x08; /* Set data length: 8 bytes */
HCAN_MB3.MD7 = 0x33; /* Set transmission data: 00110011 */
HCAN_MB3.MD8 = 0x33; /* Set transmission data: 00110011 */
HCAN_MB3.MD9 = 0x33; /* Set transmission data: 00110011 */
HCAN_MB3.MD10 = 0x33; /* Set transmission data: 00110011 */
HCAN_MB3.MD11 = 0x33; /* Set transmission data: 00110011 */
HCAN_MB3.MD12 = 0x33; /* Set transmission data: 00110011 */
HCAN_MB3.MD13 = 0x33; /* Set transmission data: 00110011 */
HCAN_MB3.MD14 = 0x33; /* Set transmission data: 00110011 */

/*** MB4 **/  
HCAN_MB4.MC4 = 0x01; /* Set mailbox 4 as for transmission */
HCAN_MB4.MC0 = 0x3330; /* Select data frame and standard format, set identifier */
HCAN_MB4.MC5 = 0x08; /* Set data length: 8 bytes */
HCAN_MB4.MD7 = 0x44; /* Set transmission data: 01000100 */
HCAN_MB4.MD8 = 0x44; /* Set transmission data: 01000100 */
HCAN_MB4.MD9 = 0x44; /* Set transmission data: 01000100 */
HCAN_MB4.MD10 = 0x44; /* Set transmission data: 01000100 */
HCAN_MB4.MD11 = 0x44; /* Set transmission data: 01000100 */
HCAN_MB4.MD12 = 0x44; /* Set transmission data: 01000100 */
HCAN_MB4.MD13 = 0x44; /* Set transmission data: 01000100 */
HCAN_MB4.MD14 = 0x44; /* Set transmission data: 01000100 */

/*** MB5 **/  
HCAN_MB5.MC4 = 0x01; /* Set mailbox 5 as for transmission */
HCAN_MB5.MC0 = 0x0880; /* Select data frame and standard format, set identifier */
HCAN_MB5.MC5 = 0x08; /* Set data length: 8 bytes */
HCAN_MB5.MD7 = 0x55; /* Set transmission data: 01010101 */
HCAN_MB5.MD8 = 0x55; /* Set transmission data: 01010101 */
HCAN_MB5.MD9 = 0x55; /* Set transmission data: 01010101 */
HCAN_MB5.MD10 = 0x55; /* Set transmission data: 01010101 */
HCAN_MB5.MD11 = 0x55;  /* Set transmission data: 01010101 */
HCAN_MB5.MD12 = 0x55;  /* Set transmission data: 01010101 */
HCAN_MB5.MD13 = 0x55;  /* Set transmission data: 01010101 */
HCAN_MB5.MD14 = 0x55;  /* Set transmission data: 01010101 */

/*** MB6 ***/
HCAN_MB6.MC4 = 0x01;   /* Set mailbox 6 as for transmission */
HCAN_MB6.MC0 = 0x4CC0; /* Select data frame and standard format, set identifier */
HCAN_MB6.MC5 = 0x08;   /* Set data length: 8 bytes */
HCAN_MB6.MD7 = 0x66;   /* Set transmission data: 01100110 */
HCAN_MB6.MD8 = 0x66;   /* Set transmission data: 01100110 */
HCAN_MB6.MD9 = 0x66;   /* Set transmission data: 01100110 */
HCAN_MB6.MD10 = 0x66;  /* Set transmission data: 01100110 */
HCAN_MB6.MD11 = 0x66;  /* Set transmission data: 01100110 */
HCAN_MB6.MD12 = 0x66;  /* Set transmission data: 01100110 */
HCAN_MB6.MD13 = 0x66;  /* Set transmission data: 01100110 */
HCAN_MB6.MD14 = 0x66;  /* Set transmission data: 01100110 */

/*** MB7 ***/
HCAN_MB7.MC4 = 0x01;   /* Set mailbox 7 as for transmission */
HCAN_MB7.MC0 = 0x1990; /* Select data frame and standard format, set identifier */
HCAN_MB7.MC5 = 0x08;   /* Set data length: 8 bytes */
HCAN_MB7.MD7 = 0x77;   /* Set transmission data: 01110111 */
HCAN_MB7.MD8 = 0x77;   /* Set transmission data: 01110111 */
HCAN_MB7.MD9 = 0x77;   /* Set transmission data: 01110111 */
HCAN_MB7.MD10 = 0x77;  /* Set transmission data: 01110111 */
HCAN_MB7.MD11 = 0x77;  /* Set transmission data: 01110111 */
HCAN_MB7.MD12 = 0x77;  /* Set transmission data: 01110111 */
HCAN_MB7.MD13 = 0x77;  /* Set transmission data: 01110111 */
HCAN_MB7.MD14 = 0x77;  /* Set transmission data: 01110111 */

/*** MB8 ***/
HCAN_MB8.MC4 = 0x01;   /* Set mailbox 8 as for transmission */
HCAN_MB8.MC0 = 0x7FF0; /* Select data frame and standard format, set identifier */
HCAN_MB8.MC5 = 0x08;   /* Set data length: 8 bytes */
HCAN_MB8.MD7 = 0x88;   /* Set transmission data: 10001000 */
HCAN_MB8.MD8 = 0x88;   /* Set transmission data: 10001000 */
HCAN_MB8.MD9 = 0x88;   /* Set transmission data: 10001000 */
HCAN_MB8.MD10 = 0x88;  /* Set transmission data: 10001000 */
HCAN_MB8.MD11 = 0x88;  /* Set transmission data: 10001000 */
HCAN_MB8.MD12 = 0x88;  /* Set transmission data: 10001000 */
HCAN_MB8.MD13 = 0x88;  /* Set transmission data: 10001000 */
HCAN_MB8.MD14 = 0x88;  /* Set transmission data: 10001000 */
HCAN_MB9.MC4 = 0x01; /* Set mailbox 9 as for transmission */
HCAN_MB9.MC0 = 0x1110; /* Select data frame and standard format, set identifier */
HCAN_MB9.MC5 = 0x08; /* Set data length: 8 bytes */
HCAN_MB9.MD7 = 0x99; /* Set transmission data: 10011001 */
HCAN_MB9.MD8 = 0x99; /* Set transmission data: 10011001 */
HCAN_MB9.MD9 = 0x99; /* Set transmission data: 10011001 */
HCAN_MB9.MD10 = 0x99; /* Set transmission data: 10011001 */
HCAN_MB9.MD11 = 0x99; /* Set transmission data: 10011001 */
HCAN_MB9.MD12 = 0x99; /* Set transmission data: 10011001 */
HCAN_MB9.MD13 = 0x99; /* Set transmission data: 10011001 */
HCAN_MB9.MD14 = 0x99; /* Set transmission data: 10011001 */

HCAN_MB10.MC4 = 0x01; /* Set mailbox 10 as for transmission */
HCAN_MB10.MC0 = 0x4440; /* Select data frame and standard format, set identifier */
HCAN_MB10.MC5 = 0x08; /* Set data length: 8 bytes */
HCAN_MB10.MD7 = 0xAA; /* Set transmission data: 10101010 */
HCAN_MB10.MD8 = 0xAA; /* Set transmission data: 10101010 */
HCAN_MB10.MD9 = 0xAA; /* Set transmission data: 10101010 */
HCAN_MB10.MD10 = 0xAA; /* Set transmission data: 10101010 */
HCAN_MB10.MD11 = 0xAA; /* Set transmission data: 10101010 */
HCAN_MB10.MD12 = 0xAA; /* Set transmission data: 10101010 */
HCAN_MB10.MD13 = 0xAA; /* Set transmission data: 10101010 */
HCAN_MB10.MD14 = 0xAA; /* Set transmission data: 10101010 */

HCAN_MB11.MC4 = 0x01; /* Set mailbox 11 as for transmission */
HCAN_MB11.MC0 = 0x5550; /* Select data frame and standard format, set identifier */
HCAN_MB11.MC5 = 0x08; /* Set data length: 8 bytes */
HCAN_MB11.MD7 = 0xBB; /* Set transmission data: 10111011 */
HCAN_MB11.MD8 = 0xBB; /* Set transmission data: 10111011 */
HCAN_MB11.MD9 = 0xBB; /* Set transmission data: 10111011 */
HCAN_MB11.MD10 = 0xBB; /* Set transmission data: 10111011 */
HCAN_MB11.MD11 = 0xBB; /* Set transmission data: 10111011 */
HCAN_MB11.MD12 = 0xBB; /* Set transmission data: 10111011 */
HCAN_MB11.MD13 = 0xBB; /* Set transmission data: 10111011 */
HCAN_MB11.MD14 = 0xBB; /* Set transmission data: 10111011 */

HCAN_MB12.MC4 = 0x01; /* Set mailbox 12 as for transmission */
HCAN_MB12.MC0 = 0x6660; /* Select data frame and standard format, set identifier */
HCAN_MB12.MC5 = 0x08; /* Set data length: 8 bytes */
HCAN_MB12.MD7 = 0xCC; /* Set transmission data: 11001100 */
HCAN_MB12.MD8 = 0xCC; /* Set transmission data: 11001100 */
HCAN_MB12.MD9 = 0xCC; /* Set transmission data: 11001100 */
HCAN_MB12.MD10 = 0xCC; /* Set transmission data: 11001100 */
HCAN_MB12.MD11 = 0xCC;              /* Set transmission data: 11001100 */
HCAN_MB12.MD12 = 0xCC;              /* Set transmission data: 11001100 */
HCAN_MB12.MD13 = 0xCC;              /* Set transmission data: 11001100 */
HCAN_MB12.MD14 = 0xCC;              /* Set transmission data: 11001100 */

/*** MB13 ***/
HCAN_MB13.MC4 = 0x01;                /* Set mailbox 13 as for transmission */
HCAN_MB13.MC0 = 0x3BB0;              /* Select data frame and standard format, 
set identifier */
HCAN_MB13.MC5 = 0x08;                /* Set data length: 8 bytes */
HCAN_MB13.MD7 = 0xDD;                /* Set transmission data: 11011101 */
HCAN_MB13.MD8 = 0xDD;                /* Set transmission data: 11011101 */
HCAN_MB13.MD9 = 0xDD;                /* Set transmission data: 11011101 */
HCAN_MB13.MD10 = 0xDD;               /* Set transmission data: 11011101 */
HCAN_MB13.MD11 = 0xDD;               /* Set transmission data: 11011101 */
HCAN_MB13.MD12 = 0xDD;               /* Set transmission data: 11011101 */
HCAN_MB13.MD13 = 0xDD;               /* Set transmission data: 11011101 */
HCAN_MB13.MD14 = 0xDD;               /* Set transmission data: 11011101 */

/*** MB14 ***/
HCAN_MB14.MC4 = 0x01;                /* Set mailbox 14 as for transmission */
HCAN_MB14.MC0 = 0x2220;              /* Select data frame and standard format, 
set identifier */
HCAN_MB14.MC5 = 0x08;                /* Set data length: 8 bytes */
HCAN_MB14.MD7 = 0xEE;                /* Set transmission data: 11101110 */
HCAN_MB14.MD8 = 0xEE;                /* Set transmission data: 11101110 */
HCAN_MB14.MD9 = 0xEE;                /* Set transmission data: 11101110 */
HCAN_MB14.MD10 = 0xEE;               /* Set transmission data: 11101110 */
HCAN_MB14.MD11 = 0xEE;               /* Set transmission data: 11101110 */
HCAN_MB14.MD12 = 0xEE;               /* Set transmission data: 11101110 */
HCAN_MB14.MD13 = 0xEE;               /* Set transmission data: 11101110 */
HCAN_MB14.MD14 = 0xEE;               /* Set transmission data: 11101110 */

/*** MB15 ***/
HCAN_MB15.MC4 = 0x01;                /* Set mailbox 15 as for transmission */
HCAN_MB15.MC0 = 0x5DD0;              /* Select data frame and standard format, 
set identifier */
HCAN_MB15.MC5 = 0x08;                /* Set data length: 8 bytes */
HCAN_MB15.MD7 = 0xFF;                /* Set transmission data: 11111111 */
HCAN_MB15.MD8 = 0xFF;                /* Set transmission data: 11111111 */
HCAN_MB15.MD9 = 0xFF;                /* Set transmission data: 11111111 */
HCAN_MB15.MD10 = 0xFF;               /* Set transmission data: 11111111 */
HCAN_MB15.MD11 = 0xFF;               /* Set transmission data: 11111111 */
HCAN_MB15.MD12 = 0xFF;               /* Set transmission data: 11111111 */
HCAN_MB15.MD13 = 0xFF;               /* Set transmission data: 11111111 */
HCAN_MB15.MD14 = 0xFF;               /* Set transmission data: 11111111 */

/* Set message transmission wait status */
HCAN_TXPR0 = 0xFFFE; /* Set mailboxes 1 to 15 to transmission wait status */
/* Wait for transmission end */
while((HCAN_TXACK0 & 0xFFFE) != 0xFFFE);

Rev. 1.00, 08/03, page 48 of 74
/* Clear transmission end flag */
HCAN_TXACK0 = 0xFFFE;  /* Clear transmission end flag (to clear, write 1) */
while(1);

4.6 Reception Flowchart

![Reception Flowchart](image)

**Figure 4.3** Reception Flowchart (1)
4.7 Software Description (Reception)

Module Description

<table>
<thead>
<tr>
<th>Module</th>
<th>Label</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>r_main</td>
<td>Performs initial settings and reception settings for HCAN-2.</td>
</tr>
<tr>
<td>Reception interrupt routine</td>
<td>RM1_IRR1</td>
<td>Clears reception end flag and sets interrupt prohibition.</td>
</tr>
</tbody>
</table>
### Description of Registers Used (See example 1 for information on pins and port registers.)

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
<th>Initial Value</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBbuff</td>
<td>Storage area for received data base address (H'FFFFD100).</td>
<td>---</td>
<td>Main routine</td>
</tr>
<tr>
<td>work</td>
<td>Work register used for mailbox initialization.</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>DTMR</td>
<td>Increments both DTSAR and DTDAR after transfer completes, sets block transfer mode and byte transfer.</td>
<td>0xA890</td>
<td></td>
</tr>
<tr>
<td>DTSAR</td>
<td>Sets transfer source address as mailbox 0 message data area.</td>
<td>0xFFFBB108</td>
<td></td>
</tr>
<tr>
<td>DTDAR</td>
<td>Sets transfer destination address as received data storage area.</td>
<td>0xFFFBD100</td>
<td></td>
</tr>
<tr>
<td>DTCRA</td>
<td>Sets number of block transfers (15).</td>
<td>0x000F</td>
<td></td>
</tr>
<tr>
<td>DTCRB</td>
<td>Sets block length (8 bytes).</td>
<td>0x0008</td>
<td></td>
</tr>
<tr>
<td>DTC.DTBR</td>
<td>Sets top 16 bits of memory address at which data transferred using DTC is stored.</td>
<td>0xFFFF</td>
<td></td>
</tr>
<tr>
<td>DTC.DTEF</td>
<td>Selects the interrupt source (RM1 in HCAN-2) for starting DTC.</td>
<td>0x04</td>
<td></td>
</tr>
<tr>
<td>HCAN_MCR</td>
<td>Clears reset request bit.</td>
<td>0x0000</td>
<td></td>
</tr>
<tr>
<td>HCAN_IRR</td>
<td>Clears reset, hold, and sleep interrupt flags. (To clear, write 1.)</td>
<td>0x0001</td>
<td></td>
</tr>
<tr>
<td>HCAN_BCR0</td>
<td>Sets bit rate to 250 kbps when (\phi) is 50 MHz.</td>
<td>0x0009</td>
<td></td>
</tr>
<tr>
<td>HCAN_BCR1</td>
<td></td>
<td>0x4300</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB0.MC0</td>
<td>Sets data frame and standard format for mailbox 0.</td>
<td>0x0000</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB0.MC4</td>
<td>Sets mailbox 0 as for reception.</td>
<td>0x03</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB0.MC5</td>
<td>Sets mailbox 0 reception size to a data length of 8 bytes.</td>
<td>0x08</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB0.LAFM15</td>
<td>Sets identifier filter mask for mailbox 0.</td>
<td>0x7FF0</td>
<td></td>
</tr>
<tr>
<td>HCAN.IMR</td>
<td>Enables message reception interrupts.</td>
<td>0xFFFFD</td>
<td></td>
</tr>
<tr>
<td>HCAN_MBMIR0</td>
<td>Enables mailbox 0 interrupt requests.</td>
<td>0xFFFFE</td>
<td></td>
</tr>
<tr>
<td>INTC.IPRK</td>
<td>Sets priority of HCAN-2 interrupt requests.</td>
<td>0x00F0</td>
<td></td>
</tr>
<tr>
<td>HCAN_RXPR0</td>
<td>Clears mailbox 0 reception end flag. (To clear, write 1.)</td>
<td>0x0001</td>
<td>Reception interrupt routine</td>
</tr>
<tr>
<td>HCAN.IMR</td>
<td>Prohibits message reception interrupts.</td>
<td>0xFFFF</td>
<td></td>
</tr>
</tbody>
</table>
4.8 Reception Program Listing

#include <stdio.h>                /* Library function header file */
#include <machine.h>              /* Library function header file */
#include "SH7047.h"               /* Peripheral register definition header file */

---

/* Function prototype declarations */

void r_main(void);
void RM1_IRR1(void);

---

/* Define constants */

#define DTMR  (*(unsigned short *)0xFFFFD080) /* DTC register data */
#define DTCRA (*(unsigned short *)0xFFFFD082) /* DTC register data */
#define DTCRB (*(unsigned short *)0xFFFFD086) /* DTC register data */
#define DTSAR (*(unsigned long *)0xFFFFD088)  /* DTC register data */
#define DTDAR (*(unsigned long *)0xFFFFD08C)  /* DTC register data */
#define MBbuff (*(unsigned char *)0xFFFFD100) /* Storage area for received data */

---

Main routine

void r_main(void){
    unsigned short *work;
    /* Initialize storage area for received data */
    work = (unsigned short *)0xFFFFD100;
    do {
        *work = 0x0000;
        work++;
    } while(work < (unsigned short *)0xFFFFD180);
    /* DTC settings */
    DTMR = 0xA890;                        /* Increment both DTSAR and DTDAR after transfer completes, set block transfer mode and byte transfer */
    DTSAR = (unsigned long)&HCAN_MB0.MD7; /* Set transfer source address */
    DTDAR = (unsigned long)&MBbuff;       /* Set transfer destination address */
    DTCRA = 0x000F;                       /* Block transfer: 15 blocks */
    DTCRB = 0x0008;                       /* Block length: 8 bytes */
    DTC.DTBR = 0xFFFF;                    /* Register data base address */
    DTC.DTEF |= 0x04;                     /* HCAN-2 interrupt (RM1) */
    /* Clear bit MCR0 */
    HCAN_MCR = 0x0000;                    /* Clear reset request bit */
    /* Set local acceptance filter */
    HCAN_MB0.LAFM15 = 0x7FF0;            /* Set identifier filter mask for mailbox 0 */
    /* Clear bit IRR0 */
    HCAN_IIR = 0x0001;                   /* Clear reset interrupt flag (to clear, write 1) */
    /* Set pins */

Rev. 1.00, 08/03, page 52 of 74
PB.PBCR1 = 0x0000;                     /* Set PB HCAN-2 */
PB.PBCR2 = 0x0000;                     /* Set PB HCAN-2 */
/* Set bit rate (BCR): bit rate is 250 kbps when φ = 50 MHz */
HCAN_BCR0 = 0x0009;                    /* BRP=9(10 system clock) */
HCAN_BCR1 = 0x4300;                    /* TSEG1=4(5tq),TSEG2=3(4tq) */
/* Initialize mailbox */
  work = (unsigned short *)0xFFFFB100;
  do {
    *work = 0xFFFF;
    work++;
  } while(work < (unsigned short *)0xFFFFB4F4);
/* Set MBC */
  HCAN_MB0.MC4 = 0x03;                   /* Set mailbox 0 as for reception */
/* Set interrupts */
  HCAN_IMR = 0xFFFFD;                     /* Enable message reception interrupts */
    HCAN_MBIMR0 = 0xFFFFE;    /* Enable mailbox 0 reception interrupt requests */
  /* Set reception method */
    HCAN_MB0.MC0 = 0x0000;             /* Select data frame and standard format, set identifier */
    HCAN_MB0.MC5 = 0x08;               /* Set data length: 8 bytes */
    set_imask(0);
    while(1);
}

/* *********************************************************************** */
/*     Reception interrupt routine                                        */
/* *********************************************************************** */
#pragma interrupt(RM1_IRR1)
void RM1_IRR1(void){
  /* Clear reception end register */
    HCAN_RXPR0 = 0x0001; /* Clear reception end flag (to clear, write 1) */
  /* Prohibit reception interrupts */
    HCAN_IMR = 0xFFFF; /* Prohibit message reception interrupts */
}
4.9 Operation Waveforms (Transmission and Reception)

Figures 4.5 and 4.6 show the waveforms at the start and end of the operation of this application.

![Operation Waveforms Diagram]

**Figure 4.5 Operation Waveforms (Start of Operation)**
Figure 4.6  Operation Waveforms (End of Operation)
Section 5  Example 5  
(Remote Frame)

5.1 Transmission and Reception Specifications

Remote frame transmission and reception in the standard format using two SH7047F devices.

Common Transmission and Reception Specifications

- The data transfer speed is 250 kbps (during 50 MHz operation).
- The identifier is H'555.
- The identifier is masked and reception takes place when a match occurs.

Remote Frame Transmission Specifications

- Mailbox 1 is used for remote frame transmission and data frame reception.
- Eight bytes of data are requested.
- The message reception interrupt (IRR1) is used.
  (a) After remote frame transmission the system enters message reception interrupt wait status.
  (b) The reception end flag is cleared and message reception interrupts are prohibited within the message reception interrupt routine.
  (c) When the data is received DTC settings are made and the software is triggered within the message reception interrupt routine, and the received data is stored in on-chip RAM.
  (d) Block transfer mode is used for the DTC transfer, and the 8 bytes of data are transferred as a single block.
- The DTC transfer end interrupt is used.
  Triggering of the DTC software is prohibited within the DTC transfer end interrupt routine, completing the operation.

Remote Frame Reception Specifications

- Mailbox 1 is used for remote frame reception and data frame transmission.
- The data frame is prepared in advance, and the automatic transmission function (ATX) is used.
- The data transmitted is H'55, H'66, H'77, H'88, H'99, H'AA, H'BB, H'FF.
- The remote frame request interrupt (IRR2) is used.
  (a) The remote frame reception end flag is cleared and reception interrupts are prohibited within the remote frame request interrupt routine, and the data frame transmission end flag is polled.
(b) Once it has been confirmed that the transmission end flag has been set, the transmission end flag is cleared, completing the operation.

5.2 Transmission and Reception Specifications, Function Description

Tables 5.1 and 5.2 list the functions allocated to registers. (See example 1 for information on pins and port registers.)

Table 5.1 HCAN-2 Function Allocation

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCR</td>
<td>Master control register</td>
</tr>
<tr>
<td></td>
<td>Controls operation of HCAN-2.</td>
</tr>
<tr>
<td>BCR0</td>
<td>Bit timing configuration registers</td>
</tr>
<tr>
<td>BCR1</td>
<td>Used to set baud rate prescaler and bit timing parameters of CAN.</td>
</tr>
<tr>
<td>IRR</td>
<td>Interrupt request register</td>
</tr>
<tr>
<td></td>
<td>Shows status of interrupt sources.</td>
</tr>
<tr>
<td>MBx MC0</td>
<td>Message control register</td>
</tr>
<tr>
<td></td>
<td>Used to set identifier and whether frames are data frames or remote frames.</td>
</tr>
<tr>
<td>MC4</td>
<td>Message control register</td>
</tr>
<tr>
<td></td>
<td>Used to select transmission or reception.</td>
</tr>
<tr>
<td>MC5</td>
<td>Message control register</td>
</tr>
<tr>
<td></td>
<td>Used to set the data length.</td>
</tr>
<tr>
<td>MD7 to MD14</td>
<td>Message data registers</td>
</tr>
<tr>
<td></td>
<td>Store transmitted and received CAN message data.</td>
</tr>
<tr>
<td>LAFM15</td>
<td>Local acceptance filter mask</td>
</tr>
<tr>
<td></td>
<td>Used to set filter mask for identifier of reception mailbox.</td>
</tr>
<tr>
<td>TXPR</td>
<td>Transmission wait register</td>
</tr>
<tr>
<td></td>
<td>Used to set transmission wait status after transmitted message is stored in mailbox.</td>
</tr>
<tr>
<td>TXACK</td>
<td>Transmission acknowledge register</td>
</tr>
<tr>
<td></td>
<td>Indicates that the transmitted message has been properly transmitted to the corresponding mailbox.</td>
</tr>
</tbody>
</table>
## Register Function

**Reception**

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXPR</td>
<td>Reception end register</td>
</tr>
<tr>
<td></td>
<td>Indicates that the data has been properly received by the corresponding mailbox.</td>
</tr>
<tr>
<td>RFPR</td>
<td>Remote request register</td>
</tr>
<tr>
<td></td>
<td>Indicates reception of remote frames by the corresponding mailbox.</td>
</tr>
</tbody>
</table>

**Interrupt registers**

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBIMR</td>
<td>Mailbox interrupt mask register</td>
</tr>
<tr>
<td></td>
<td>Used to enable interrupt requests for mailboxes.</td>
</tr>
<tr>
<td>IMR</td>
<td>Interrupt mask register</td>
</tr>
<tr>
<td></td>
<td>Used to enable interrupt requests using IRR interrupt flag.</td>
</tr>
<tr>
<td>IPRK</td>
<td>Interrupt priority register</td>
</tr>
<tr>
<td></td>
<td>Used to set the priority of HCAN-2 interrupt requests.</td>
</tr>
</tbody>
</table>

### Table 2.2  DTC Function Allocation

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTMR</td>
<td>DTC mode register</td>
</tr>
<tr>
<td></td>
<td>Controls the DTC operation mode.</td>
</tr>
<tr>
<td>DTSAR</td>
<td>DTC source address register</td>
</tr>
<tr>
<td></td>
<td>Specifies the source address for data to be transferred using DTC.</td>
</tr>
<tr>
<td>DTDAR</td>
<td>DTC destination address register</td>
</tr>
<tr>
<td></td>
<td>Specifies the destination address for data to be transferred using DTC.</td>
</tr>
<tr>
<td>DTCRA</td>
<td>DTC transfer count register A</td>
</tr>
<tr>
<td></td>
<td>Specifies the number of DTC transfers.</td>
</tr>
<tr>
<td>DTCRB</td>
<td>DTC transfer count register B</td>
</tr>
<tr>
<td></td>
<td>In the block transfer mode, specifies the block length.</td>
</tr>
<tr>
<td>DTBR</td>
<td>DTC database register</td>
</tr>
<tr>
<td></td>
<td>Specifies the top 16 bits of the memory address at which data transferred using DTC is to be stored.</td>
</tr>
<tr>
<td>DTCSR</td>
<td>DTC control/status register</td>
</tr>
<tr>
<td></td>
<td>Used to enable or prohibit software triggering of DTC and to set the vector address for software triggering of DTC.</td>
</tr>
<tr>
<td>IPRG</td>
<td>Interrupt priority register</td>
</tr>
<tr>
<td></td>
<td>Used to set the priority of DTC interrupt requests.</td>
</tr>
</tbody>
</table>
5.3 Transmission Flowchart

Transmission operation: main routine (remote frame transmission)

- Initialize reception data storage area
- Clear bit MCR0
- Set local acceptance filter
- Clear bit IRR0
- Set pins
- Set bit rate
- Initialize mailbox
- Set MBC
- Set reception interrupt
- Set remote frame transmission
- Set to transmission wait status
- Transmission complete?
  - No
  - Yes
  - Clear transmission end flag

Figure 5.1 Transmission Flowchart (1)
Reception interrupt routine
   Clear reception end flag
   Prohibit reception interrupts
   Set and trigger DTC
   RTE

Figure 5.2 Transmission Flowchart (2)

DTC transfer end interrupt routine
   Prohibit DTC software triggering
   RTE

Figure 5.3 Transmission Flowchart (3)
## 5.4 Software Description (Transmission)

### Module Description

<table>
<thead>
<tr>
<th>Module</th>
<th>Label</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>t_main</td>
<td>Performs initial settings and transmission settings for HCAN-2.</td>
</tr>
<tr>
<td>Reception interrupt routine</td>
<td>RM1_IRR1</td>
<td>Clears reception end flag, prohibits reception interrupts, and stored received data by triggering DTC.</td>
</tr>
<tr>
<td>DTC transfer end interrupt routine</td>
<td>DTC_SWDTEND</td>
<td>Prohibits DTC triggering by software.</td>
</tr>
</tbody>
</table>

### Description of Registers Used (See example 1 for information on pins and port registers.)

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
<th>Initial Value</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBbuff</td>
<td>Storage area for received data (address: H'FFFFD100–H'FFFFD108).</td>
<td>—</td>
<td>Main routine</td>
</tr>
<tr>
<td>work</td>
<td>Work register used for mailbox initialization.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>HCAN_MCR</td>
<td>Clears reset request bit.</td>
<td>0x0000</td>
<td></td>
</tr>
<tr>
<td>HCAN_IRR</td>
<td>Clears reset, hold, and sleep interrupt flags. (To clear, write 1.)</td>
<td>0x0001</td>
<td></td>
</tr>
<tr>
<td>HCAN_BCR0</td>
<td>Sets bit rate to 250 kbps when φ is 50 MHz.</td>
<td>0x0009</td>
<td></td>
</tr>
<tr>
<td>HCAN_BCR1</td>
<td>—</td>
<td>0x4300</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC0</td>
<td>Sets remote frame and standard format for mailbox 1. Also sets identifier (H'555).</td>
<td>0x5558</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC4</td>
<td>Sets mailbox 1 to allow remote frame transmission and data frame reception.</td>
<td>0x05</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC5</td>
<td>Sets mailbox 1 transmission size to a data length of 8 bytes.</td>
<td>0x08</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.LAFM15</td>
<td>Sets identifier filter mask for mailbox 1.</td>
<td>0x0000</td>
<td></td>
</tr>
<tr>
<td>HCAN_IMR</td>
<td>Enables message reception interrupts.</td>
<td>0xFFF0</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1MR0</td>
<td>Enables mailbox 1 interrupt requests.</td>
<td>0xFFF0</td>
<td></td>
</tr>
<tr>
<td>INTC.IPRK</td>
<td>Sets priority of HCAN-2 interrupt requests.</td>
<td>0x00F0</td>
<td></td>
</tr>
<tr>
<td>INTC.IPRG</td>
<td>Sets priority of DTC interrupt requests.</td>
<td>0x0E00</td>
<td></td>
</tr>
<tr>
<td>HCAN_TXPR0</td>
<td>Sets mailbox 1 to transmission wait status.</td>
<td>0x0002</td>
<td></td>
</tr>
<tr>
<td>HCAN_TXACK0</td>
<td>Checks and clears mailbox 1 transmission end flag. (To clear, write 1.)</td>
<td>0x0002</td>
<td></td>
</tr>
<tr>
<td>Register</td>
<td>Function</td>
<td>Initial Value</td>
<td>Module</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>HCAN_RXPR0</td>
<td>Checks and clears mailbox 1 reception end flag. (To clear, write 1.)</td>
<td>0x0002</td>
<td>Reception interrupt routine</td>
</tr>
<tr>
<td>HCAN_IMR</td>
<td>Prohibits message reception interrupts.</td>
<td>0xFFFF</td>
<td></td>
</tr>
<tr>
<td>DTMR</td>
<td>Increments both DTSAR and DTDAR after transfer completes, sets block</td>
<td>0xA890</td>
<td></td>
</tr>
<tr>
<td></td>
<td>transfer mode and byte transfer.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DTSAR</td>
<td>Sets transfer source address as mailbox 1 message data area.</td>
<td>H'FFFFB128</td>
<td></td>
</tr>
<tr>
<td>DTDAR</td>
<td>Sets transfer destination address as received data storage area.</td>
<td>H'FFFFD100</td>
<td></td>
</tr>
<tr>
<td>DTCRA</td>
<td>Sets number of block transfers (1).</td>
<td>0x0001</td>
<td></td>
</tr>
<tr>
<td>DTCRB</td>
<td>Sets block length (8 bytes).</td>
<td>0x0008</td>
<td></td>
</tr>
<tr>
<td>DTC.DTBR</td>
<td>Sets top 16 bits of memory address at which data transferred using DTC is</td>
<td>0xFFFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>stored.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DTC.DTCSR</td>
<td>Sets DTC software trigger and DTC vector address (H'0000045A).</td>
<td>0x015A</td>
<td></td>
</tr>
<tr>
<td>DTC.DTCSR</td>
<td>Prohibits DTC software triggering.</td>
<td>0x06FF</td>
<td>DTC transfer end interrupt routine</td>
</tr>
</tbody>
</table>
5.5 Transmission Program Listing

```c
#include <stdio.h>                 /* Library function header file */
#include <machine.h>               /* Library function header file */
#include "SH7047.h"                /* Peripheral register definition header file */

void t_main(void);
void RM1_IRR1(void);
void DTC_SWDTEND(void);

main:
    unsigned short *work;
    /* Initialize storage area for received data */
    work = (unsigned short *)0xFFFFD100;
    do {
        *work = 0x0000;
        work++;
    } while(work < (unsigned short *)0xFFFFD108);
    /* Clear bit MCR0 */
    HCAN_MCR = 0x0000;             /* Clear reset request bit */
    /* Set local acceptance filter */
    HCAN_MB1.LAFM15 = 0x0000;     /* Set identifier filter mask for mailbox 1 */
    /* Clear bit IRR0 */
    HCAN_IRR = 0x0001;            /* Clear reset interrupt flag (to clear, write 1) */
    /* Set pins */
    PB.PBCR1 = 0x0000;               /* Set PB HCAN-2 */
    PB.PBCR2 = 0x0000;               /* Set PB HCAN-2 */
    /* Set bit rate (BCR): bit rate is 250 kbps when \( \phi = 50 \text{ MHz} \) */
    HCAN_BCR0 = 0x0009;            /* BRP=9(10 system clock) */
    HCAN_BCR1 = 0x4300;            /* TSEG1=4(5tq),TSEG2=3(4tq) */
    /* Initialize mailbox */
    work = (unsigned short *)0xFFFFD100;
    do {
```
```
*work = 0xFFFF;
work++;
} while(work < (unsigned short *)0xFFFFB4F4);

/* Set MBC */
HCAN_MB1.MC4 = 0x05;               /* Set mailbox 1 to enable remote frame transmission and data frame reception */
/* Set reception interrupt */
HCAN_IMR = 0xFFFF;               /* Enable message reception interrupts */
HCAN_MBMIR0 = 0xFFFF;               /* Enable mailbox 1 interrupt requests */
INTC.IPRK = 0x00F0;                /* Set RM1 priority */
INTC.IPRG = 0x0E00;                /* Set DTC interrupt priority */
/* Set remote frame transmission */
HCAN_MB1.MC0 = 0x5558;               /* Select remote frame and standard format, set identifier */
HCAN_MB1.MC5 = 0x08;               /* Set data length: 8 bytes */
/* Set message transmission wait status */
HCAN_TXPR0 = 0x0002;               /* Set mailbox 1 to transmission wait status */
/* Wait for transmission end */
while((HCAN_TXACK0 & 0x0002) != 0x0002);
/* Clear transmission end flag */
HCAN_TXACK0 = 0x0002;           /* Clear transmission end flag (to clear, write 1) */
set_imask(0);
while(1);

/********************************************************************************
/*     Reception interrupt routine                                              */
********************************************************************************/
#pragma interrupt(RM1_IRR1)
void RM1_IRR1(void){
/* Clear reception end flag */
HCAN_RXPR0 = 0x0002;               /* Clear reception end flag (to clear, write 1) */
/* Prohibit reception interrupts */
HCAN_IMR = 0xFFFF;                 /* Prohibit message reception interrupts */
/* Set and trigger DTC */
DTRM = 0xA890;                     /* Increment both DTSAR and DTDAR after transfer completes, set block transfer mode and byte transfer */
DTSAR = (unsigned long)&HCAN_MB1.MD7; /* Set transfer source address */
DTDAR = (unsigned long)&MBbuff;     /* Set transfer destination address */
DTCRA = 0x00001;                  /* Block transfer: 1 block */
DTCRB = 0x00008;                  /* Block length: 8 bytes */
DTC.DTBR = 0xFFFFF;                /* Register data base address */
while((DTC.DTCSR & 0x0100) != 0x0000); /* Confirm DTC software triggering is prohibited */
DTC.DTCSR = 0x015A;                /* Enable DTC software triggering */
while((DTC.DTCSR & 0x00FF) != 0x005A); /* Confirm DTC trigger vector address */
}

/*@*********************************************************************************/
/* DTC transfer end interrupt routine */
/*@*********************************************************************************/
#pragma interrupt(DTC_SWDTEND)
void DTC_SWDTEND(void)
{
    DTC.DTCSR &= 0x06FF;  /* Prohibit DTC software triggering */
}
5.6 Reception Flowchart

Figure 5.4 Reception Flowchart (1)
5.7 Software Description (Reception)

Module Description

<table>
<thead>
<tr>
<th>Module</th>
<th>Label</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>r_main</td>
<td>Performs initial settings and reception settings for HCAN-2.</td>
</tr>
<tr>
<td>Remote request</td>
<td>RM1_IRR2</td>
<td>Clears remote frame reception end flag, sets interrupt prohibition, and checks and clears transmission end flag.</td>
</tr>
<tr>
<td>interrupt routine</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Description of Registers Used (See example 1 for information on pins and port registers.)

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
<th>Initial Value</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>work</td>
<td>Work register used for mailbox initialization.</td>
<td>—</td>
<td>Main routine</td>
</tr>
<tr>
<td>HCAN_MCR</td>
<td>Clears reset request bit.</td>
<td>0x0000</td>
<td></td>
</tr>
<tr>
<td>HCAN_IRR</td>
<td>Clears reset, hold, and sleep interrupt flags. (To clear, write 1.)</td>
<td>0x0001</td>
<td></td>
</tr>
<tr>
<td>HCAN_BCR0</td>
<td>Sets bit rate to 250 kbps when $\phi$ is 50 MHz.</td>
<td>0x0009</td>
<td></td>
</tr>
<tr>
<td>HCAN_BCR1</td>
<td></td>
<td>0x4300</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC0</td>
<td>Sets data frame and standard format for mailbox 1. Also sets identifier (H'555).</td>
<td>0x5550</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC4</td>
<td>Enables remote frame reception and data frame transmission for mailbox 1. Also sets ATX (automatic data frame transmission).</td>
<td>0x11</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC5</td>
<td>Sets mailbox 1 transmission size to a data length of 8 bytes.</td>
<td>0x08</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC7</td>
<td>Sets mailbox 1 byte 1 transmission data (H'55).</td>
<td>0x55</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC8</td>
<td>Sets mailbox 1 byte 2 transmission data (H'66).</td>
<td>0x66</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC9</td>
<td>Sets mailbox 1 byte 3 transmission data (H'77).</td>
<td>0x77</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC10</td>
<td>Sets mailbox 1 byte 4 transmission data (H'88).</td>
<td>0x88</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC11</td>
<td>Sets mailbox 1 byte 5 transmission data (H'99).</td>
<td>0x99</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC12</td>
<td>Sets mailbox 1 byte 6 transmission data (H'AA).</td>
<td>0xAA</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC13</td>
<td>Sets mailbox 1 byte 7 transmission data (H'BB).</td>
<td>0xBB</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.MC14</td>
<td>Sets mailbox 1 byte 8 transmission data (H'FF).</td>
<td>0xFF</td>
<td></td>
</tr>
<tr>
<td>HCAN_MB1.LAFM15</td>
<td>Sets identifier filter mask for mailbox 1.</td>
<td>0x0000</td>
<td></td>
</tr>
<tr>
<td>HCAN_IMR</td>
<td>Enables remote request interrupts.</td>
<td>0xFFFFB</td>
<td></td>
</tr>
<tr>
<td>HCAN_MBIMR0</td>
<td>Enables mailbox 1 interrupt requests.</td>
<td>0xFFFFD</td>
<td></td>
</tr>
<tr>
<td>INTC.IPRK</td>
<td>Sets the priority of HCAN-2 interrupt requests.</td>
<td>0x00F0</td>
<td></td>
</tr>
<tr>
<td>Register</td>
<td>Function</td>
<td>Initial Value</td>
<td>Module</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------</td>
<td>---------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>HCAN_RFPR0</td>
<td>Clears mailbox 1 remote request transmission end flag. (To clear, write 1.)</td>
<td>0x0002</td>
<td>Remote request interrupt routine</td>
</tr>
<tr>
<td>HCAN_IMR</td>
<td>Prohibits remote request interrupts.</td>
<td>0xFFFF</td>
<td></td>
</tr>
<tr>
<td>HCAN_TXACK0</td>
<td>Checks and clears mailbox 1 transmission end flag. (To clear, write 1.)</td>
<td>0x0002</td>
<td></td>
</tr>
</tbody>
</table>
5.8 Reception Program Listing

/******************************************/
/*     HCAN-2 Reception Program (Example 5)*/
/******************************************/
#include <stdio.h>                 /* Library function header file */
#include <machine.h>               /* Library function header file */
#include "SH7047.h"                /* Peripheral register definition header file*/
/******************************************/
/*     Function prototype declarations*/
/******************************************/
void r_main(void);
void RM1_IRR2(void);
(/^
/*     Main routine*/)*/
void r_main(void){
  unsigned short *work;
  /* Clear bit MCR0 */
  HCAN_MCR = 0x0000;                  /* Clear reset request bit */
  /* Set local acceptance filter */
  HCAN_MB1.LAFM15 = 0x0000;     /* Set identifier filter mask for mailbox 1 */
  /* Clear bit IRR0 */
  HCAN_IRR = 0x0001;      /* Clear reset interrupt flag (to clear, write 1) */
  /* Set pins */
  PB.PBCR1 = 0x0000;                  /* Set PB HCAN-2 */
  PB.PBCR2 = 0x000F;                  /* Set PB HCAN-2 */
  /* Set bit rate (BCR): bit rate is 250 kbps when φ = 50 MHz */
  HCAN_BCR0 = 0x0009;                 /* BRP=9(10 system clock) */
  HCAN_BCR1 = 0x4300;                 /* TSEG1=4(5tq),TSEG2=3(4tq) */
  /* Initialize mailbox */
  work = (unsigned short *)0xFFFFB100;
  do {
    *work = 0xFFFF;
    work++;
  } while(work < (unsigned short *)0xFFFFB4F4);
  /* Set MBC */
  HCAN_MB1.MC4 = 0x11;               /* Set mailbox 1 to use ATX, enable */
  remote frame reception, and enable data frame transmission */
  /* Set remote request interrupt */
  HCAN_IMR = 0xFFFFB;                /* Enable remote request interrupts */
  HCAN_MBIMR0 = 0xFFFFD;              /* Enable mailbox 1 interrupt requests */
  INTC.IPRK = 0x00F0;                /* Set RMI priority */
  /* Set transmission data */
  HCAN_MB1.MC0 = 0x5550;             /* Select data frame and standard format, */
  set identifier */
  HCAN_MB1.MC5 = 0x08;                /* Set data length: 8 bytes */
  HCAN_MB1.MD7 = 0x55;                /* Set transmission data: 01010101 */
  HCAN_MB1.MD8 = 0x66;                /* Set transmission data: 01100110 */
HCAN_MB1.MD9 = 0x77;               /* Set transmission data: 01110111 */
HCAN_MB1.MD10 = 0x88;              /* Set transmission data: 10001000 */
HCAN_MB1.MD11 = 0x99;              /* Set transmission data: 10011001 */
HCAN_MB1.MD12 = 0xAA;              /* Set transmission data: 10101010 */
HCAN_MB1.MD13 = 0xBB;              /* Set transmission data: 10111011 */
HCAN_MB1.MD14 = 0xFF;              /* Set transmission data: 11111111 */

set_imask(0);
while(1);
}

/********************************************************************************
/*     Remote request interrupt routine                                         */
********************************************************************************/
#pragma interrupt(RM1_IRR2)
void RM1_IRR2(void){

    /* Clear remote request register */
    HCAN_RPPR0 = 0x0002;     /* Clear remote frame reception end flag (to clear, write 1) */
    /* Prohibit reception interrupts */
    HCAN_IMR = 0xFFFF;          /* Prohibit remote frame reception interrupts */
    /* Wait for transmission end */
    while((HCAN_TXACK0 & 0x0002) != 0x0002);
    /* Clear transmission end flag */
    HCAN_TXACK0 = 0x0002;  /* Clear transmission end flag (to clear, write 1) */
}

Rev. 1.00, 08/03, page 71 of 74
5.9  Operation Waveforms (Transmission and Reception)

Figure 5.6 shows the waveforms during the operation of this application.

![Diagram of waveforms]

Figure 5.6  Operation Waveforms
Reference Documents

(1) SH7047 Series Hardware Manual, Version 1.0 (Renesas Technology Corp.)