To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

Application Note

32

SH7047F HCAN2

Application Note

Renesas 32-Bit RISC Microcomputer
SuperH™ RISC engine Family/
SH7047 Series

Renesas Electronics
www.renesas.com

Rev.1.00 2003.8

Renesas 32-Bit RISC Microcomputer
SuperH™ RISC engine Family/
SH7047 Series

SH7047F HCAN2

Application Note

RENESAS

REJ05B0144-0100O

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.

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 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

Pin			Function
Pin	HTxD1		Used to perform message transmission using the HCAN-2 (pin 57).
	HRxD1		Used to perform message reception using the HCAN-2 (pin 56).
Register			**Function**
Common transmission and reception registers	PBCR1		Port B control registers
	PBCR2		Set to functions of pins HTxD1 and HRxD1.
	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.
		MC4	Message control register
			Used to select transmission or reception.
		MC5	Message control register
			Used to set the data length.
		MD7	Message data register
			Stores 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 acknowledge 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.

1.3 Transmission Flowchart

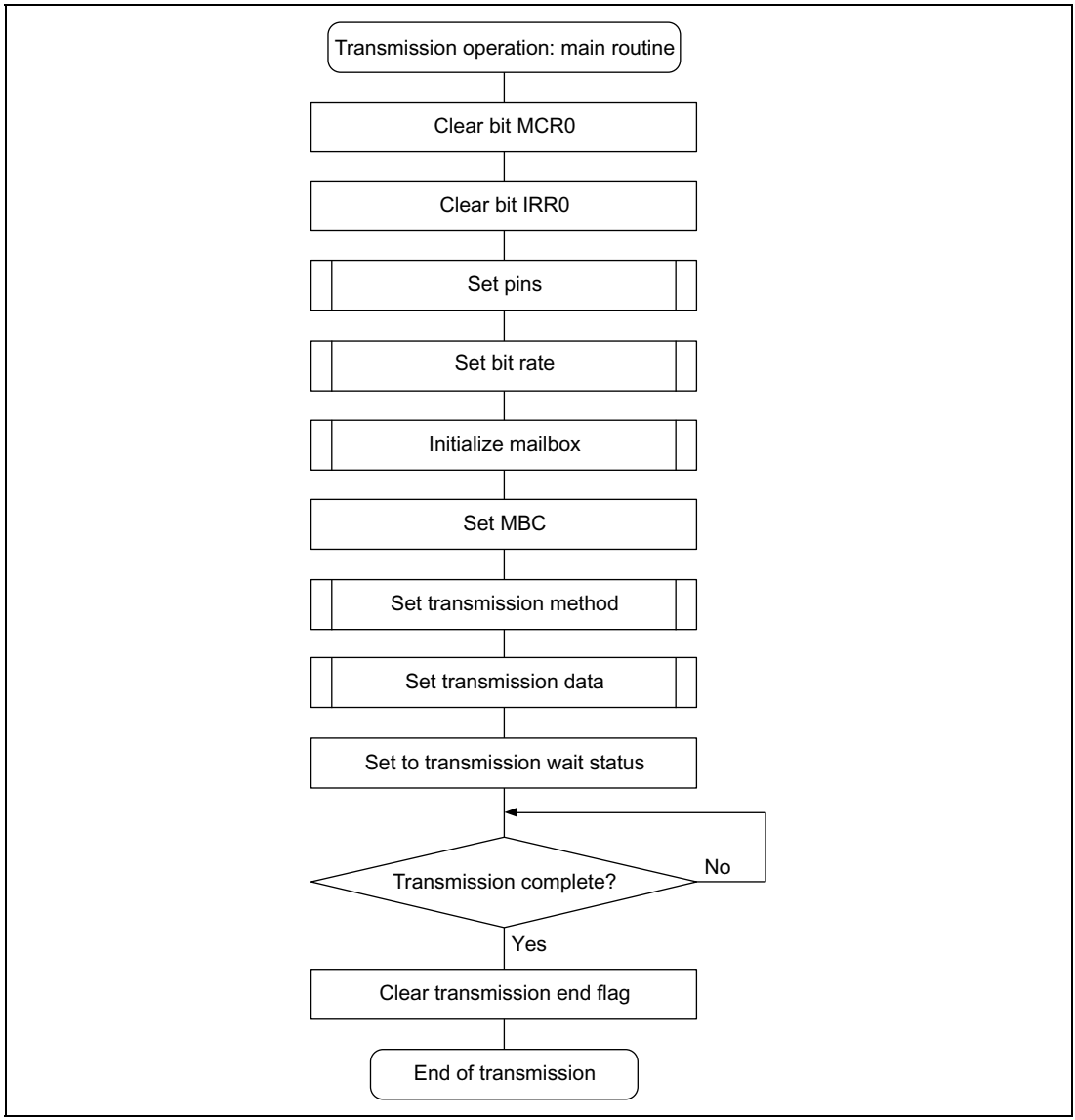

Figure 1.1 Transmission Flowchart

1.4 Software Description (Transmission)

Module Description

Module	Label	Function
Main routine	t_main	Performs initial settings and transmission settings for HCAN-2.

Description of Registers Used

Register	Function	Initial Value	Module
PB.PBCR1	Sets pins PB1 and PB0 (pins 56 and 57) to HRxD1 and HTxD1.	0x0000	Main routine
PB.PBCR2		0x000F	
work	Work register used for mailbox initialization.	—	
HCAN_MCR	Clears reset request bit.	0x0000	
HCAN_IRR	Clears reset, hold, and sleep interrupt flags. (To clear, write 1.)	0x0001	
HCAN_BCR0	Sets bit rate to 250 kbps when ϕ is 50 MHz.	0x0009	
HCAN_BCR1		0x4300	
HCAN_MB1.MC0	Sets data frame and standard format for mailbox 1. Also sets identifier (H'555).	0x5550	
HCAN_MB1.MC4	Sets mailbox 1 as for transmission.	0x01	
HCAN_MB1.MC5	Sets mailbox 1 transmission size to a data length of 1 byte.	0x01	
HCAN_MB1.MD7	Sets mailbox 1 transmission data (H'AA).	0xAA	
HCAN_TXPR0	Sets mailbox 1 to transmission wait status.	0x0002	
HCAN_TXACK0	Checks and clears mailbox 1 transmission end flag. (To clear, write 1.)	0x0002	

1.5 Transmission Program Listing

```
/*****************************************************************************/
/*     HCAN-2 Transmission Program (Example 1)                               */
/*****************************************************************************/
#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 );
/*****************************************************************************/
/*     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 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 = 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);
}
```

1.6 Reception Flowchart

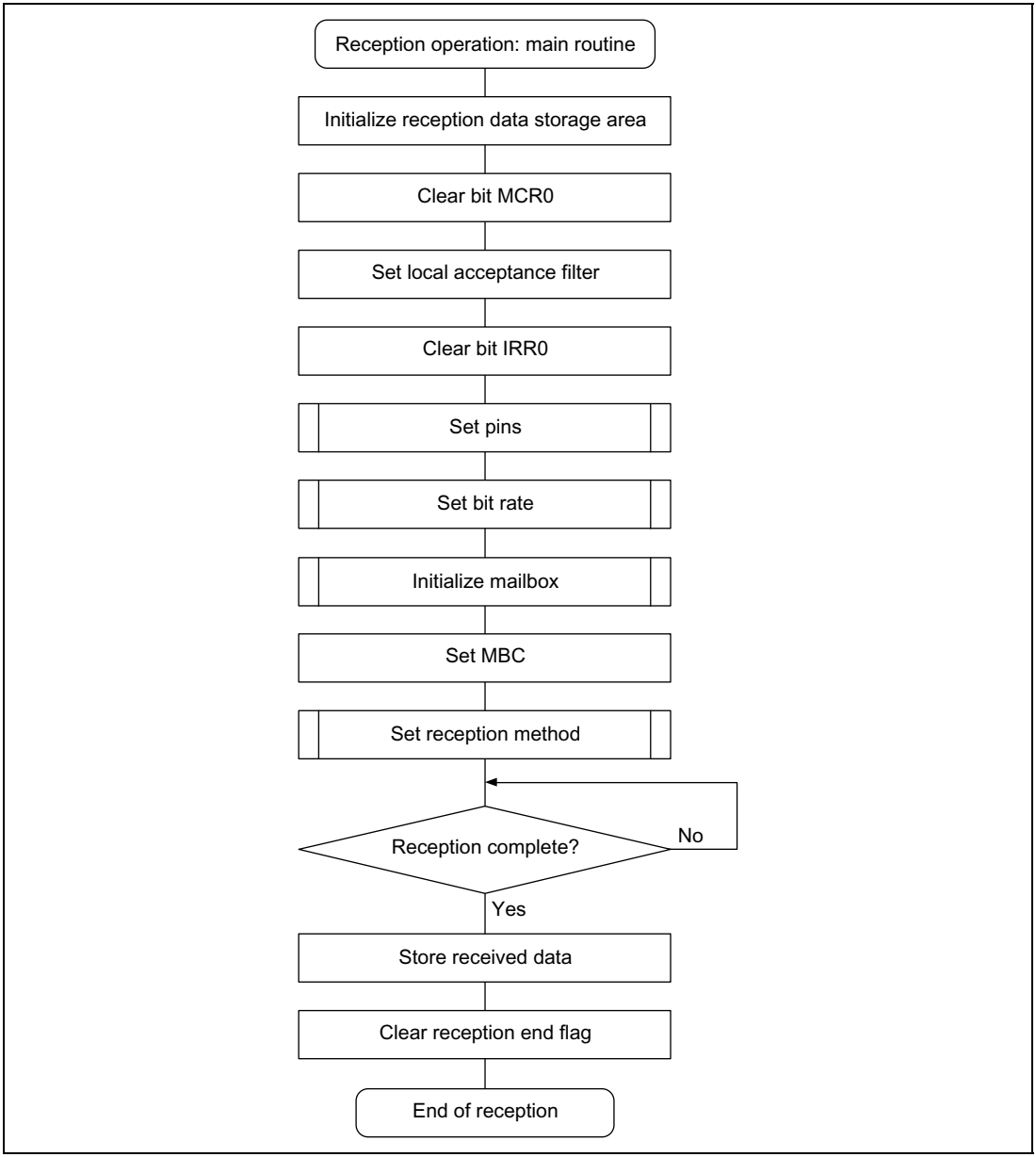

Figure 1.2 Reception Flowchart

1.7 Software Description (Reception)

Module Description

Module	Label	Function
Main routine	r_main	Performs initial settings and reception settings for HCAN-2.

Description of Registers Used

Register	Function	Initial Value	Module
MBbuff	Storage area for received data (address: H'FFFFD100).	—	Main routine
work	Work register used for mailbox initialization.	—	
PB.PBCR1	Sets pins PB1 and PB0 (pins 56 and 57) to HRxD1 and HTxD1.	0x0000	
PB.PBCR2		0x000F	
HCAN_MCR	Clears reset request bit.	0x0000	
HCAN_IRR	Clears reset, hold, and sleep interrupt flags. (To clear, write 1.)	0x0001	
HCAN_BCR0	Sets bit rate to 250 kbps when ϕ is 50 MHz.	0x0009	
HCAN_BCR1		0x4300	
HCAN_MB0.MC0	Sets data frame and standard format for mailbox 0. Also sets identifier (H'555).	0x5550	
HCAN_MB0.MC4	Sets mailbox 0 as for reception.	0x03	
HCAN_MB0.MC5	Sets mailbox 0 reception size to a data length of 1 byte.	0x01	
HCAN_MB0.LAFM15	Sets identifier filter mask for mailbox 0.	0x0000	
HCAN_RXPR0	Checks and clears mailbox 0 reception end flag. (To clear, write 1.)	0x0001	

1.8 Reception Program Listing

```
/*****************************************************************************/
/*     HCAN-2 Reception Program (Example 1)                                  */
/*****************************************************************************/
#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 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 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 */
```

```
/* Store received data */
     MBbuff = HCAN_MB0.MD7;              /* Store received data in on-chip RAM */
/* Clear reception end flag */
     HCAN_RXPR0 = 0x0001;      /* Clear reception end flag (to clear, write 1) */
     while(1);
}
```

1.9 Operation Waveforms (Transmission and Reception)

Figure 1.3 shows the waveforms during the operation of this application.

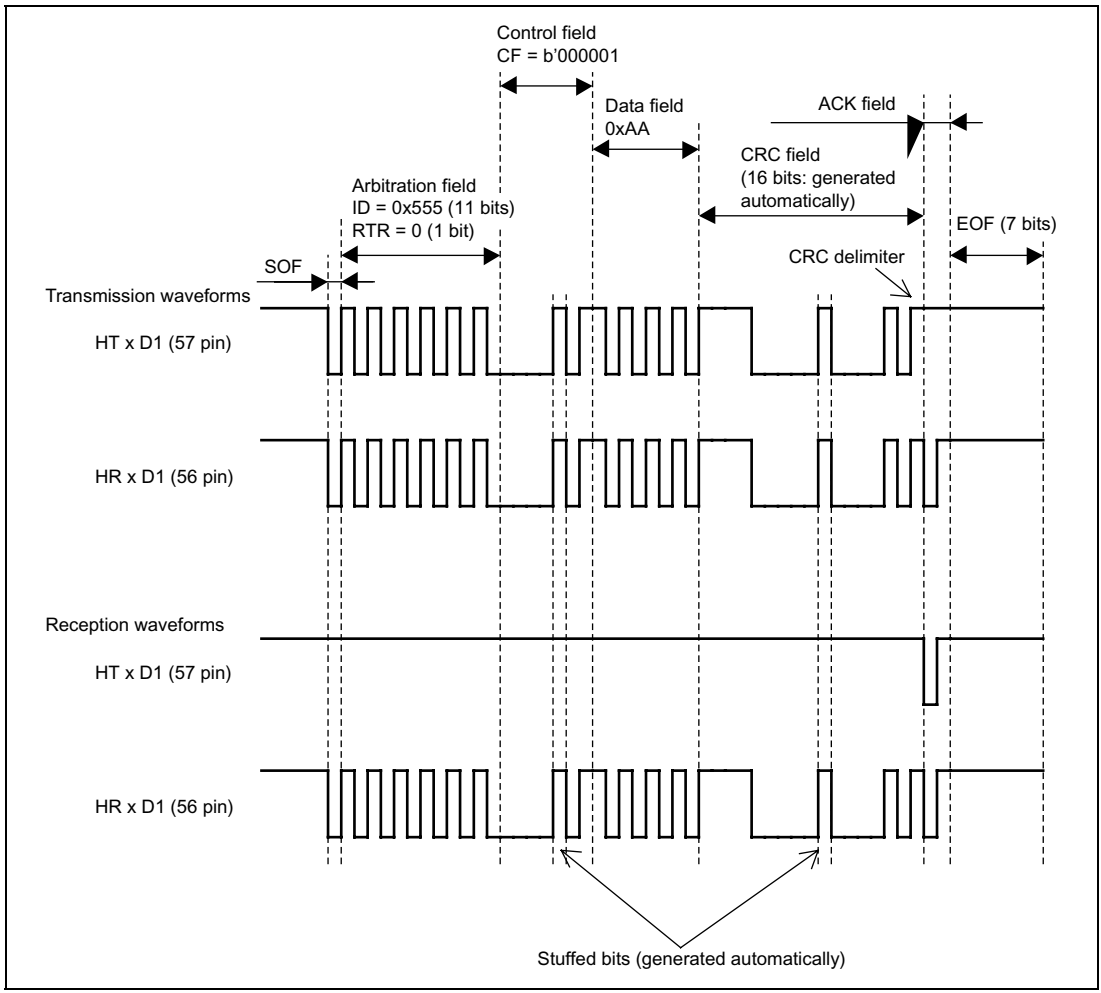

Figure 1.3 Operation Waveforms

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 2.1 HCAN-2 Function Allocation

Register			Function
Common transmission and reception registers	MCR		Master control register Controls operation of HCAN-2.
	BCR0		Bit timing configuration registers Used to set baud rate prescaler and bit timing parameters of CAN.
	BCR1		
	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.
		MC4	Message control register Used to select transmission or reception.
		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 acknowledge 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 2.2 DTC Function Allocation

Register	Function
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.

2.3 Transmission Flowchart

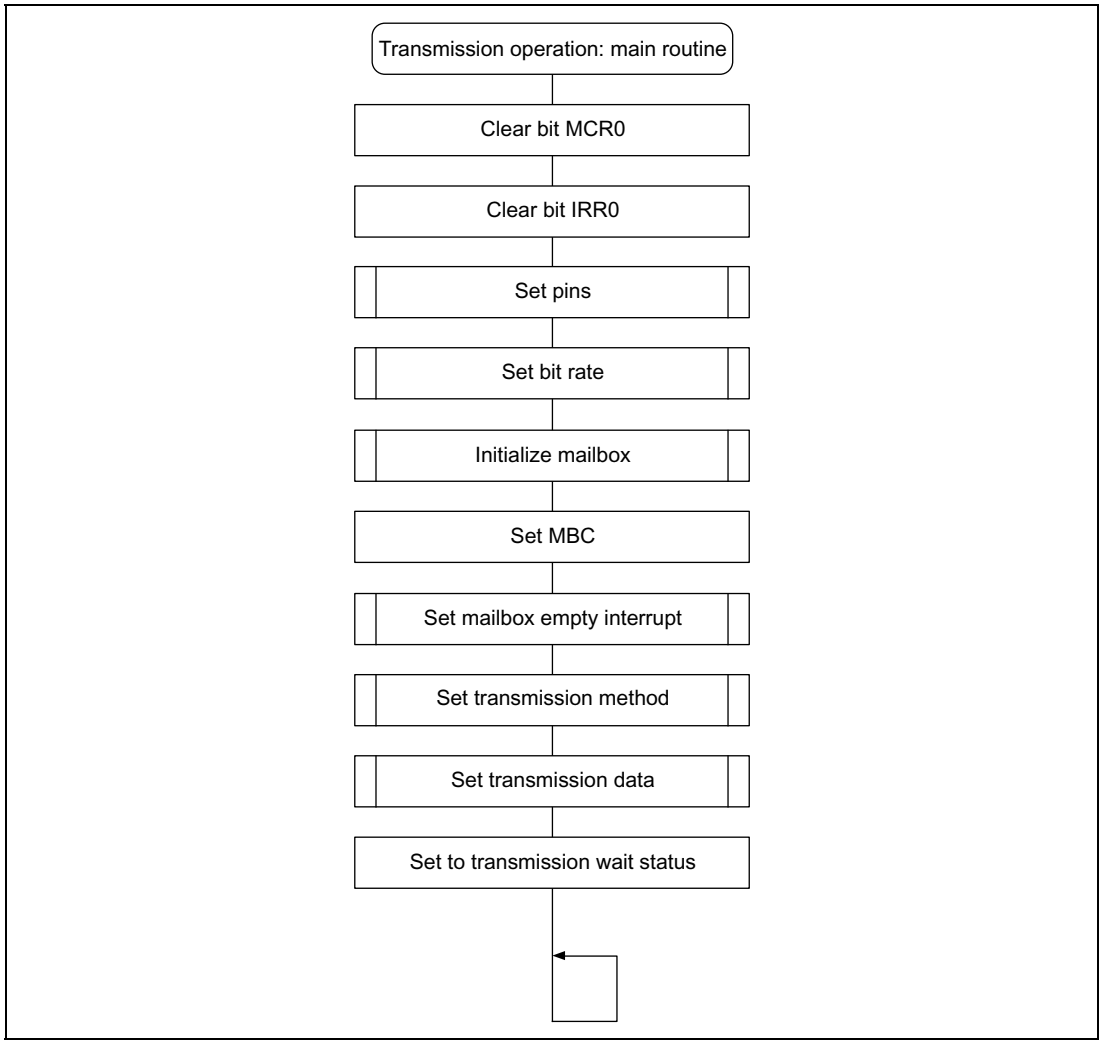

Figure 2.1 Transmission Flowchart (1)

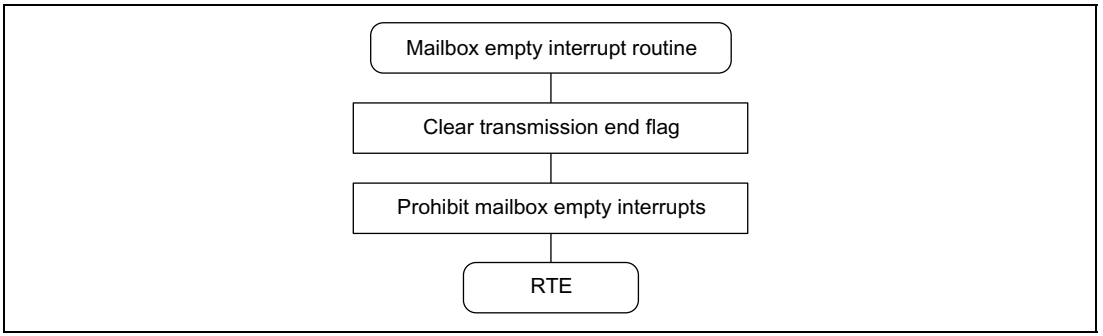

Figure 2.2 Transmission Flowchart (2)

2.4 Software Description (Transmission)

Module Description

Module	Label	Function
Main routine	t_main	Performs initial settings and transmission settings for HCAN-2.
Mailbox empty interrupt routine	SLE1_IRR8	Clears transmission end flag and sets interrupt prohibition.

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

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

2.5 Transmission Program Listing

```
/*****************************************************************************/
/*     HCAN-2 Transmission Program (Example 2)                               */
/*****************************************************************************/
#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 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 = 0x01;              /* Set mailbox 1 as for transmission */
/* Set interrupts */
     HCAN_IMR = 0xFEFF;                /* Enable interrupt IRR8 */
     HCAN_MBIMR0 = 0xFFFD;             /* 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 */
```

```
      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 */
}
```

2.6 Reception Flowchart

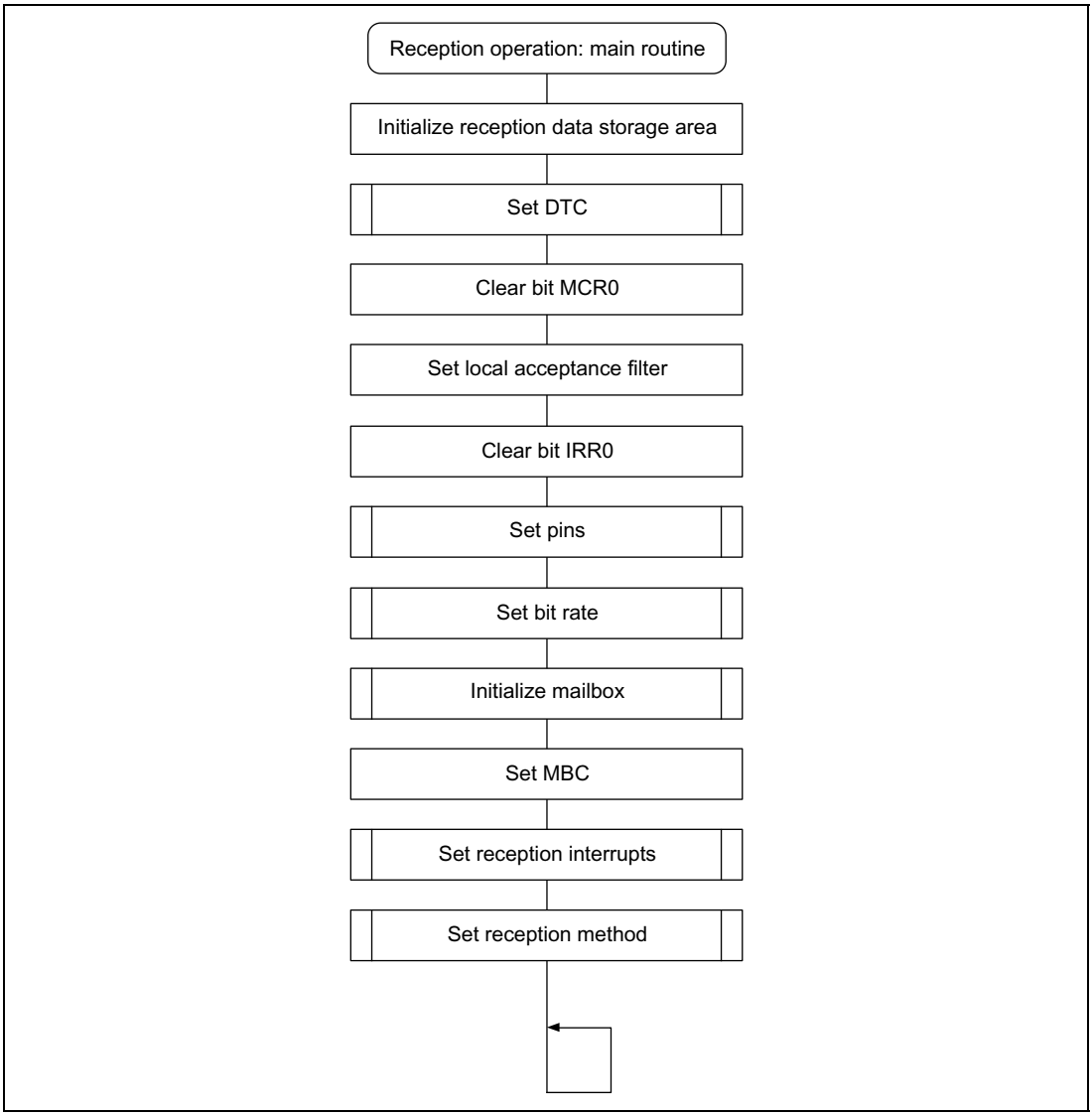

Figure 2.3 Reception Flowchart (1)

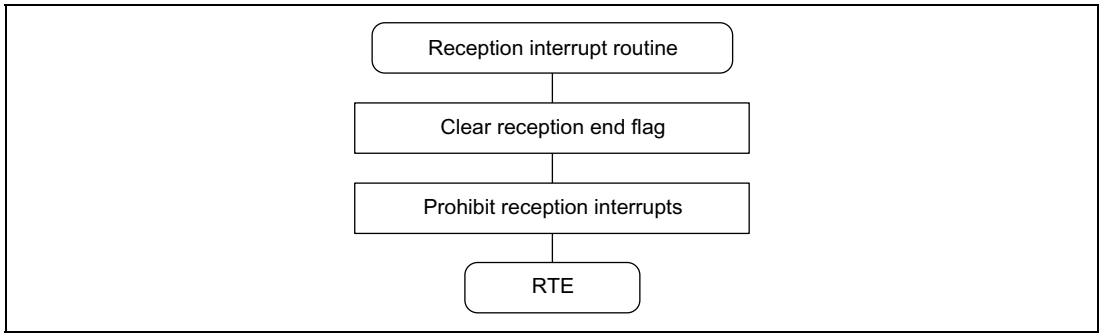

Figure 2.4 Reception Flowchart (2)

2.7 Software Description (Reception)

Module Description

Module	Label	Function
Main routine	r_main	Performs initial settings and reception settings for HCAN-2.
Reception interrupt routine	RM1_IRR1	Clears reception end flag and sets interrupt prohibition.

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

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

2.8 Reception Program Listing

```
/*****************************************************************************/
/*     HCAN-2 Reception Program (Example 2)                                  */
/*****************************************************************************/
#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 *)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 */
/* 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 = 0xFFFD;                /* 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 */

}
```

2.9 Operation Waveforms (Transmission and Reception)

Figure 2.5 shows the waveforms during the operation of this application.

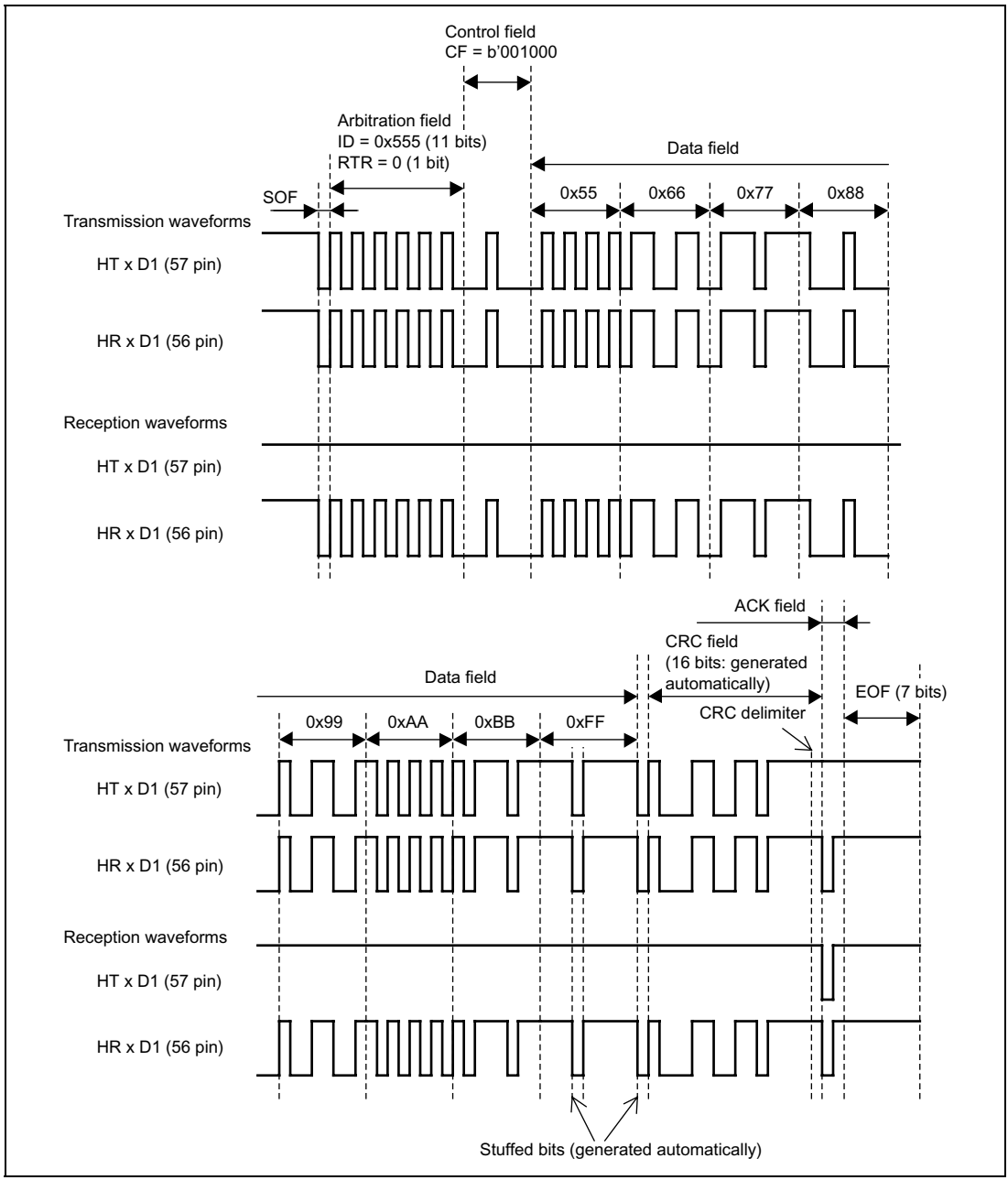

Figure 2.5 Operation Waveforms

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

Register			Function
Common transmission and reception registers	MCR		Master control register Controls operation of HCAN-2.
	BCR0		Bit timing configuration registers Used to set baud rate prescaler and bit timing parameters of CAN.
	BCR1		
	IRR		Interrupt request register Shows status of interrupt sources.
	MBx	MC0 to MC2	Message control register Used to set identifier and whether frames are data frames or remote frames.
		MC4	Message control register Used to select transmission or reception.
		MC5	Message control register Used to set the data length.
		MD7	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.
		LAFM17	
Transmission registers	TXPR		Transmission wait register Used to set transmission wait status after transmitted message is stored in mailbox.
	TXACK		Transmission acknowledge 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.

3.3 Transmission Flowchart

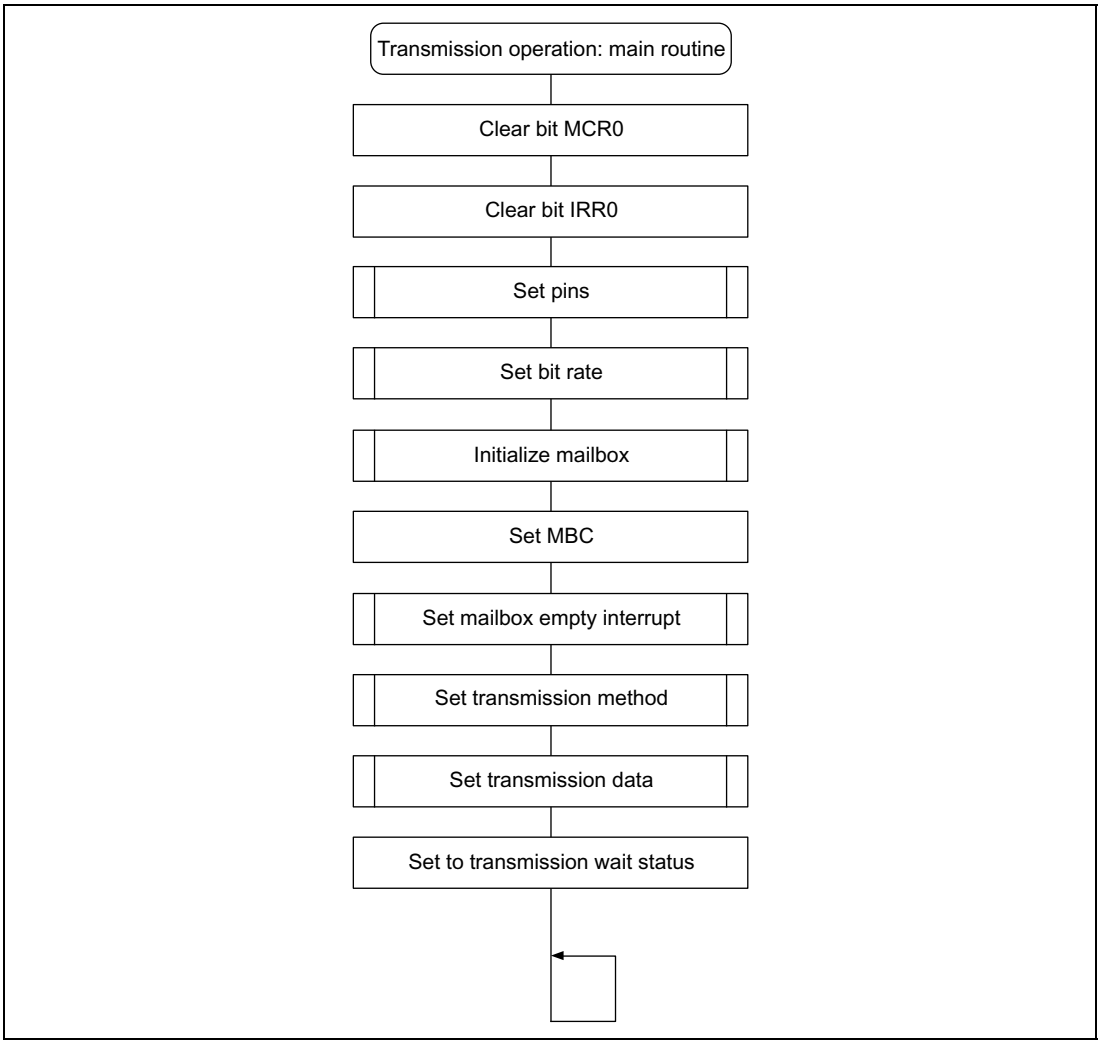

Figure 3.1 Transmission Flowchart (1)

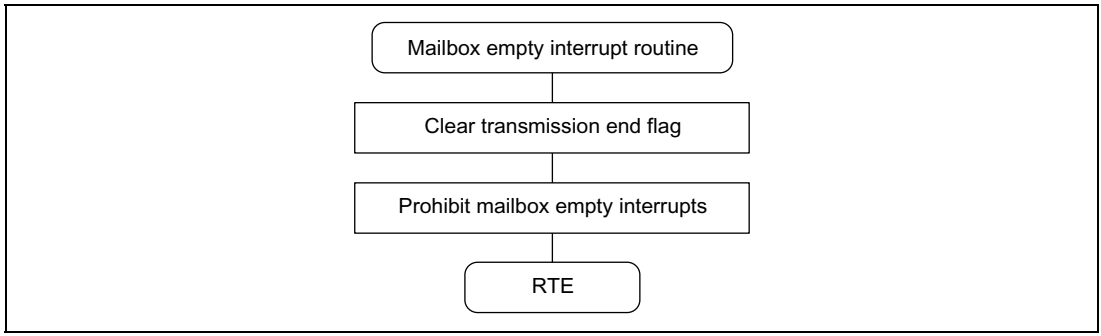

Figure 3.1 Transmission Flowchart (2)

3.4 Software Description (Transmission)

Module Description

Module	Label	Function
Main routine	t_main	Performs initial settings and transmission settings for HCAN-2.
Mailbox empty interrupt routine	SLE1_IRR8	Clears transmission end flag and sets interrupt prohibition.

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

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

3.5 Transmission Program Listing

```
/*****************************************************************************/
/*     HCAN-2 Transmission Program (Example 3)                               */
/*****************************************************************************/
#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 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 = 0x01;              /* Set mailbox 1 as for transmission */
/* Set interrupts */
     HCAN_IMR = 0xFEFF;                /* Enable interrupt IRR8 */
     HCAN_MBIMR0 = 0xFFFD;             /* Enable mailbox 1 interrupt requests */
     INTC.IPRK = 0x00F0;               /* Set SLE1 priority */
/* Set transmission method */
     HCAN_MB1.MC0 = 0x5556;            /* Select data frame and extended format,
set identifier */
     HCAN_MB1.MC2 = 0xAAAA;            /* Set (extended) 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 */
     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 */
}
```

3.6 Reception Flowchart

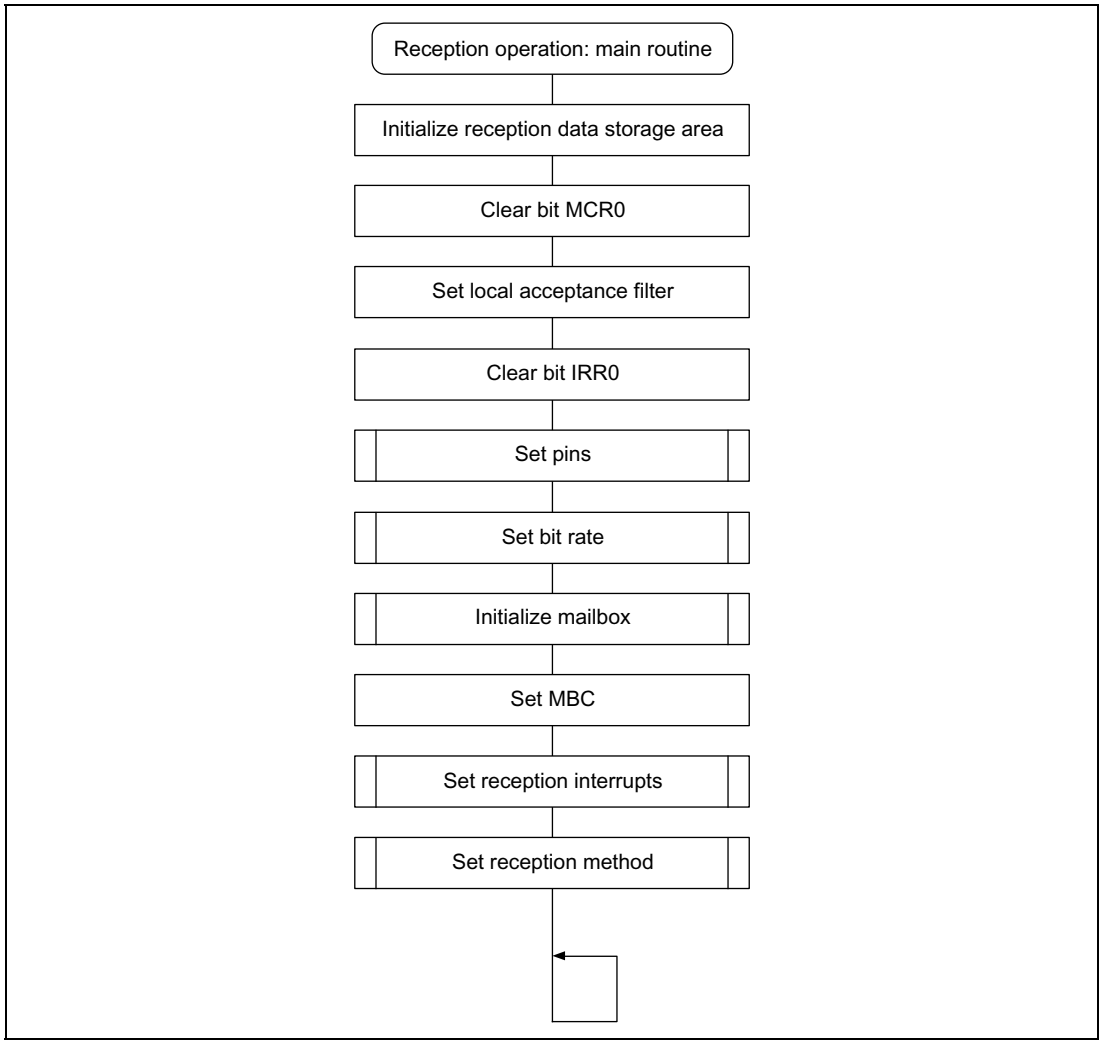

Figure 3.3 Reception Flowchart (1)

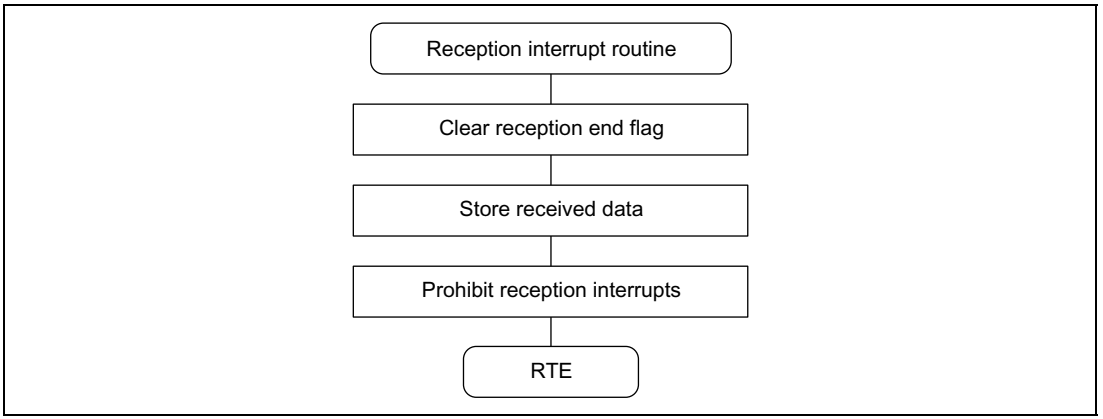

Figure 3.4 Reception Flowchart (2)

3.7 Software Description (Reception)

Module	Label	Function
Main routine	r_main	Performs initial settings and reception settings for HCAN-2.
Reception interrupt routine	RM1_IRR1	Clears reception end flag and sets interrupt prohibition.

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

Register	Function	Initial Value	Module
MBbuff	Storage area for received data (address: H'FFFFD100).	—	Main routine
work	Work register used for mailbox initialization.	—	
HCAN_MCR	Clears reset request bit.	0x0000	
HCAN_IRR	Clears reset, hold, and sleep interrupt flags. (To clear, write 1.)	0x0001	
HCAN_BCR0	Sets bit rate to 250 kbps when ϕ is 50 MHz.	0x0009	
HCAN_BCR1		0x4300	
HCAN_MB0.MC0	Sets data frame and extended format for mailbox 1. Also sets standard identifier (H'555) and extended identifier (H'2AAAA).	0x5556	
HCAN_MB0.MC2		0xAAAA	
HCAN_MB0.MC4	Sets mailbox 0 as for reception.	0x03	
HCAN_MB0.MC5	Sets mailbox 0 reception size to a data length of 1 byte.	0x01	
HCAN_MB0.LAFM15	Sets identifier filter mask for mailbox 0.	0x0000	
HCAN_MB0.LAFM17	Sets identifier filter mask for mailbox 0.	0x0000	
HCAN_IMR	Enables message reception interrupts.	0xFFFD	
HCAN_MBIMR0	Enables mailbox 0 interrupt requests.	0xFFFE	
INTC.IPRK	Sets priority of HCAN-2 interrupt requests.	0x00F0	
HCAN_RXPR0	Clears mailbox 0 reception end flag. (To clear, write 1.)	0x0001	Reception interrupt routine
HCAN_IMR	Prohibits message reception interrupts.	0xFFFF	

3.8 Reception Program Listing

```
/*****************************************************************************/
/*     HCAN-2 Reception Program (Example 3)                                  */
/*****************************************************************************/
#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 = 0xFFFD;       /* 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 = 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.

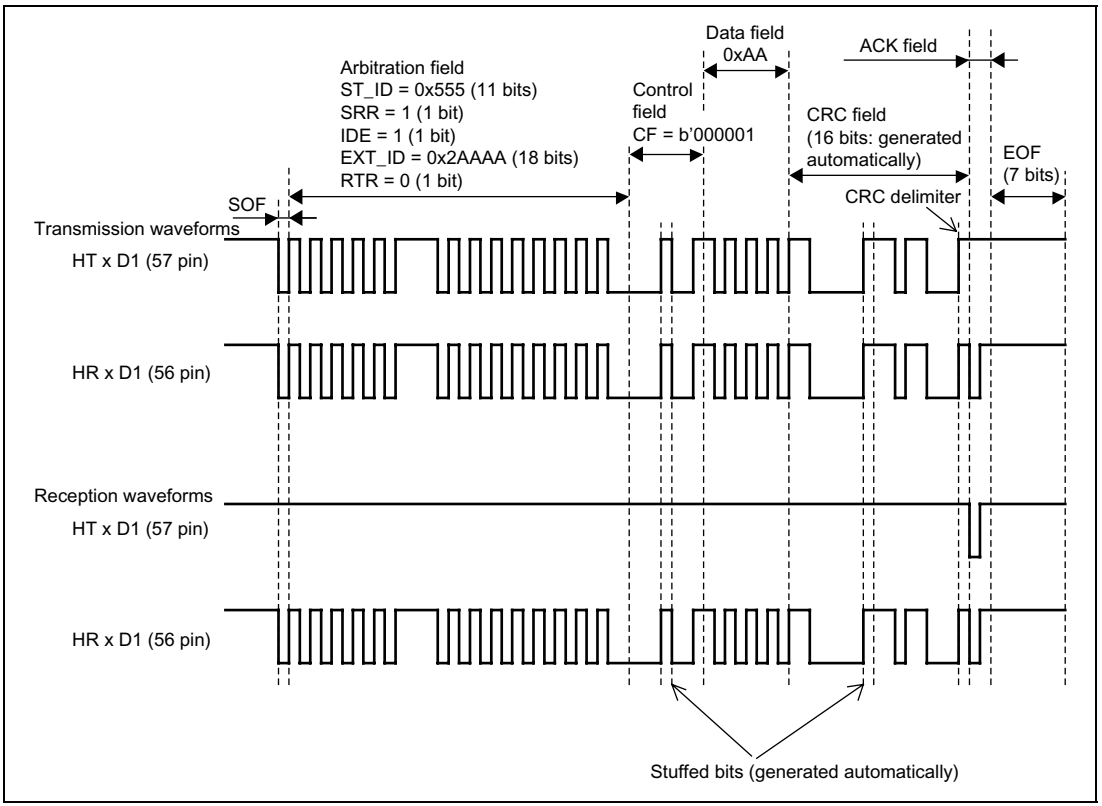

Figure 3.5 Operation Waveforms

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

MBx	Identifier (11 Bits)	Data (8 Bytes)	Priority
1	0x666	0x1111 1111 1111 1111	12
2	0x2AA	0x2222 2222 2222 2222	5
3	0x777	0x3333 3333 3333 3333	14
4	0x333	0x4444 4444 4444 4444	6
5	0x088	0x5555 5555 5555 5555	1
6	0x4CC	0x6666 6666 6666 6666	9
7	0x199	0x7777 7777 7777 7777	3
8	0x7FF	0x8888 8888 8888 8888	15
9	0x111	0x9999 9999 9999 9999	2
10	0x444	0xAAAA AAAA AAAA AAAA	8
11	0x555	0xBBBB BBBB BBBB BBBB	10
12	0x6EE	0xCCCC CCCC CCCC CCCC	13
13	0x3BB	0xDDDD DDDD DDDD DDDD	7
14	0x222	0xEEEE EEEE EEEE EEEE	4
15	0x5DD	0xFFFF FFFF FFFF FFFF	11

- 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

Register			Function
Common transmission and reception registers	MCR		Master control register Controls operation of HCAN-2.
	BCR0		Bit timing configuration registers Used to set baud rate prescaler and bit timing parameters of CAN.
	BCR1		
	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.
		MC4	Message control register Used to select transmission or reception.
		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 acknowledge 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 4.3 DTC Function Allocation

Register	Function
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

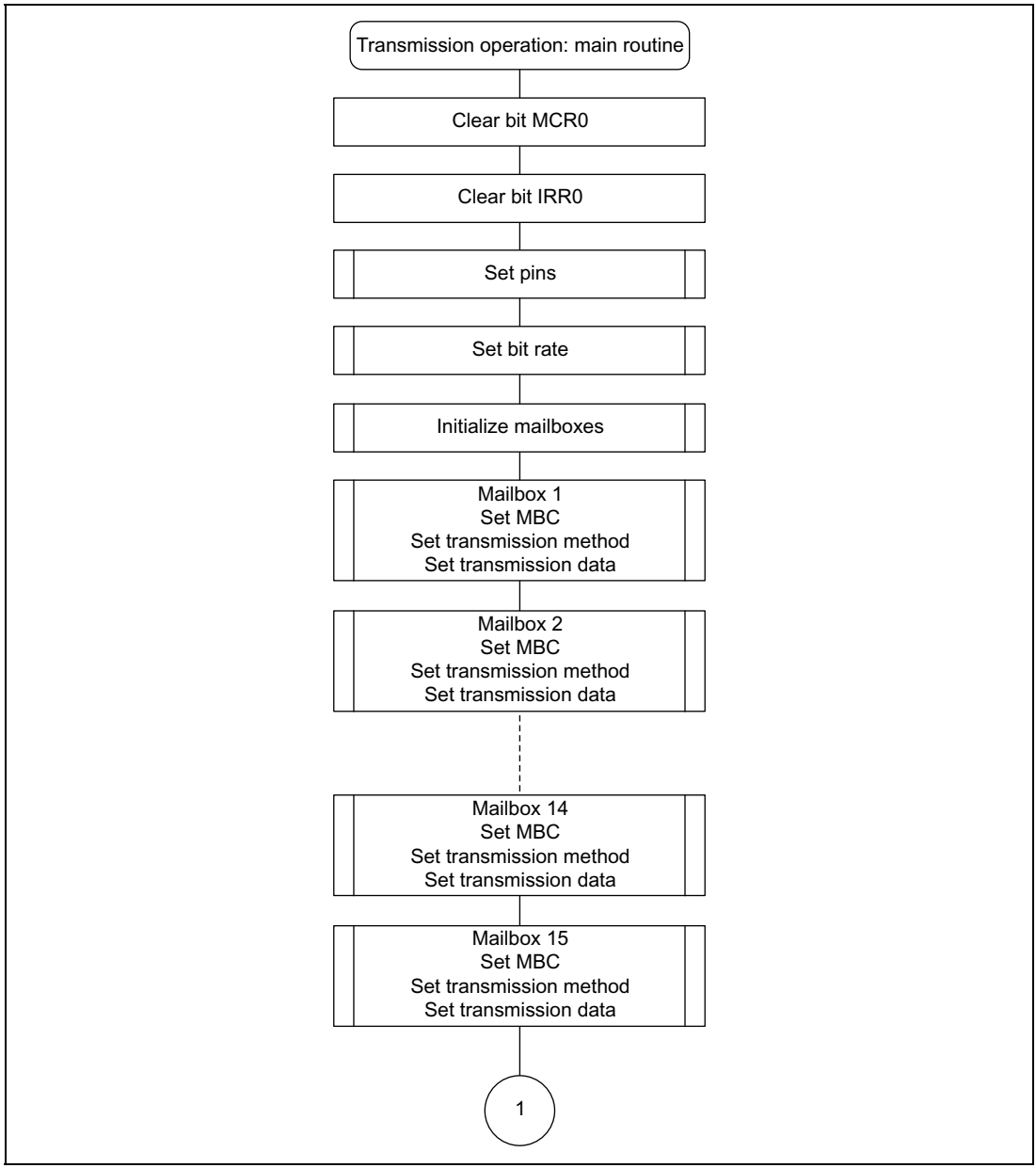

Figure 4.1 Transmission Flowchart (1)

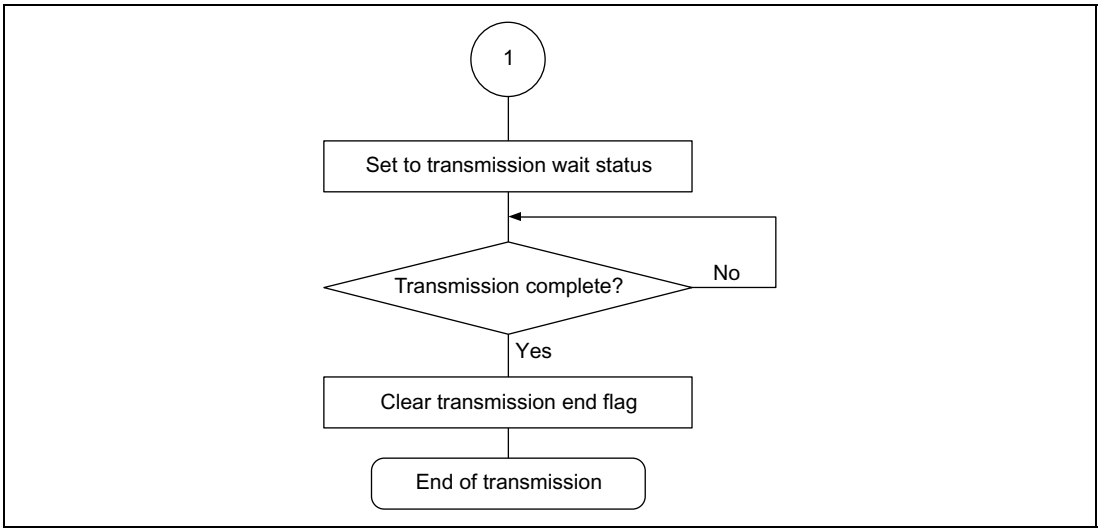

Figure 4.2 Transmission Flowchart (2)

4.4 Software Description (Transmission)

Module Description

Module	Label	Function
Main routine	t_main	Performs initial settings and transmission settings for HCAN-2.

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

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

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 */
```

```
/*** MB9 ***/
      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 */

/*** MB10 ***/
      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 */

/*** MB11 ***/
      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 */

/*** MB12 ***/
      HCAN_MB12.MC4 = 0x01;             /* Set mailbox 12 as for transmission */
      HCAN_MB12.MC0 = 0x6EE0;           /* 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);
```

```
/* Clear transmission end flag */
      HCAN_TXACK0 = 0xFFFE;  /* Clear transmission end flag (to clear, write 1) */
      while(1);

}
```

4.6 Reception Flowchart

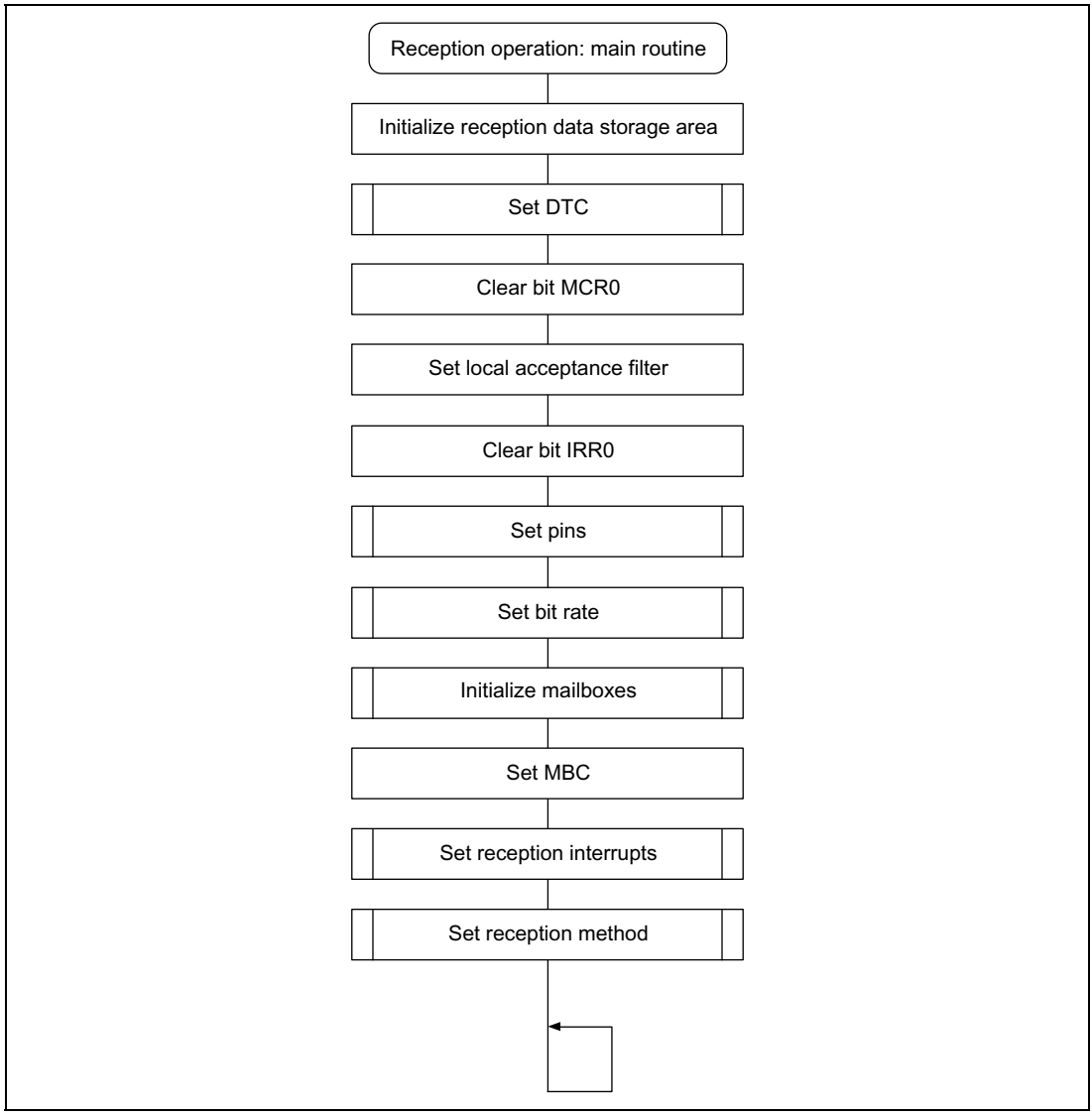

Figure 4.3 Reception Flowchart (1)

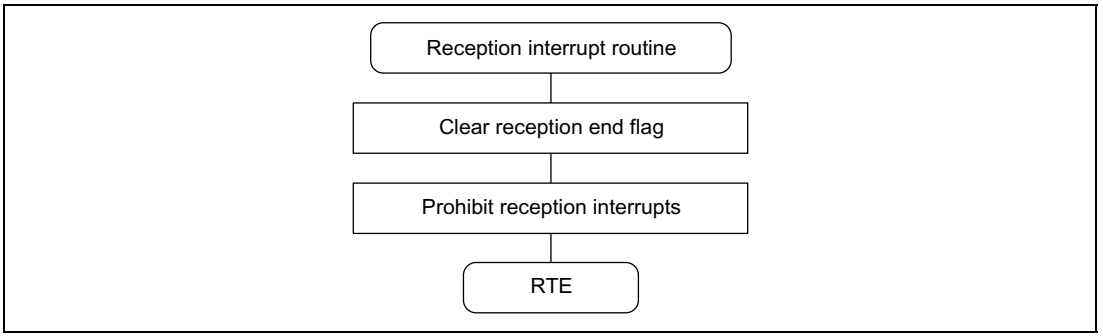

Figure 4.4 Reception Flowchart (2)

4.7 Software Description (Reception)

Module Description

Module	Label	Function
Main routine	r_main	Performs initial settings and reception settings for HCAN-2.
Reception interrupt routine	RM1_IRR1	Clears reception end flag and sets interrupt prohibition.

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

Register	Function	Initial Value	Module
MBbuff	Storage area for received data base address (H'FFFFD100).	—	Main routine
work	Work register used for mailbox initialization.	—	
DTMR	Increments both DTSAR and DTDAR after transfer completes, sets block transfer mode and byte transfer.	0xA890	
DTSAR	Sets transfer source address as mailbox 0 message data area.	0xFFFFB108	
DTDAR	Sets transfer destination address as received data storage area.	0xFFFFD100	
DTCRA	Sets number of block transfers (15).	0x000F	
DTCRB	Sets block length (8 bytes).	0x0008	
DTC.DTBR	Sets top 16 bits of memory address at which data transferred using DTC is stored.	0xFFFF	
DTC.DTEF	Selects the interrupt source (RM1 in HCAN-2) for starting DTC.	0x04	
HCAN_MCR	Clears reset request bit.	0x0000	
HCAN_IRR	Clears reset, hold, and sleep interrupt flags. (To clear, write 1.)	0x0001	
HCAN_BCR0	Sets bit rate to 250 kbps when ϕ is 50 MHz.	0x0009	
HCAN_BCR1		0x4300	
HCAN_MB0.MC0	Sets data frame and standard format for mailbox 0.	0x0000	
HCAN_MB0.MC4	Sets mailbox 0 as for reception.	0x03	
HCAN_MB0.MC5	Sets mailbox 0 reception size to a data length of 8 bytes.	0x08	
HCAN_MB0.LAFM15	Sets identifier filter mask for mailbox 0.	0x7FF0	
HCAN_IMR	Enables message reception interrupts.	0xFFFD	
HCAN_MBIMR0	Enables mailbox 0 interrupt requests.	0xFFFE	
INTC.IPRK	Sets priority of HCAN-2 interrupt requests.	0x00F0	
HCAN_RXPR0	Clears mailbox 0 reception end flag. (To clear, write 1.)	0x0001	Reception interrupt routine
HCAN_IMR	Prohibits message reception interrupts.	0xFFFF	

4.8 Reception Program Listing

```
/*****************************************************************************/
/*     HCAN-2 Reception 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 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_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 = 0xFFFD;                    /* 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 = 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.

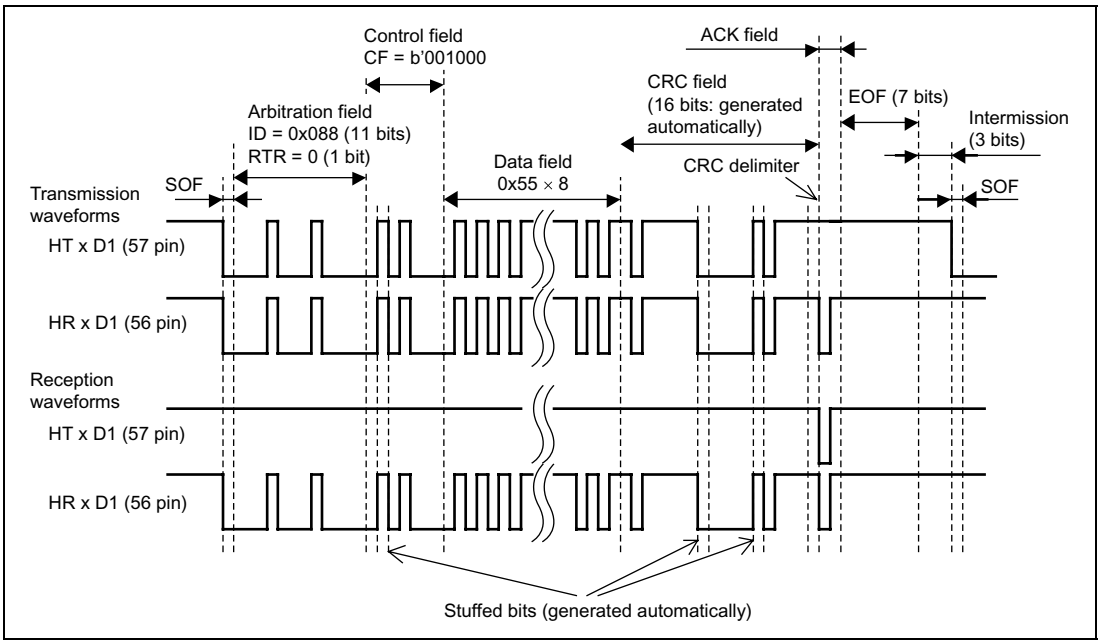

Figure 4.5 Operation Waveforms (Start of Operation)

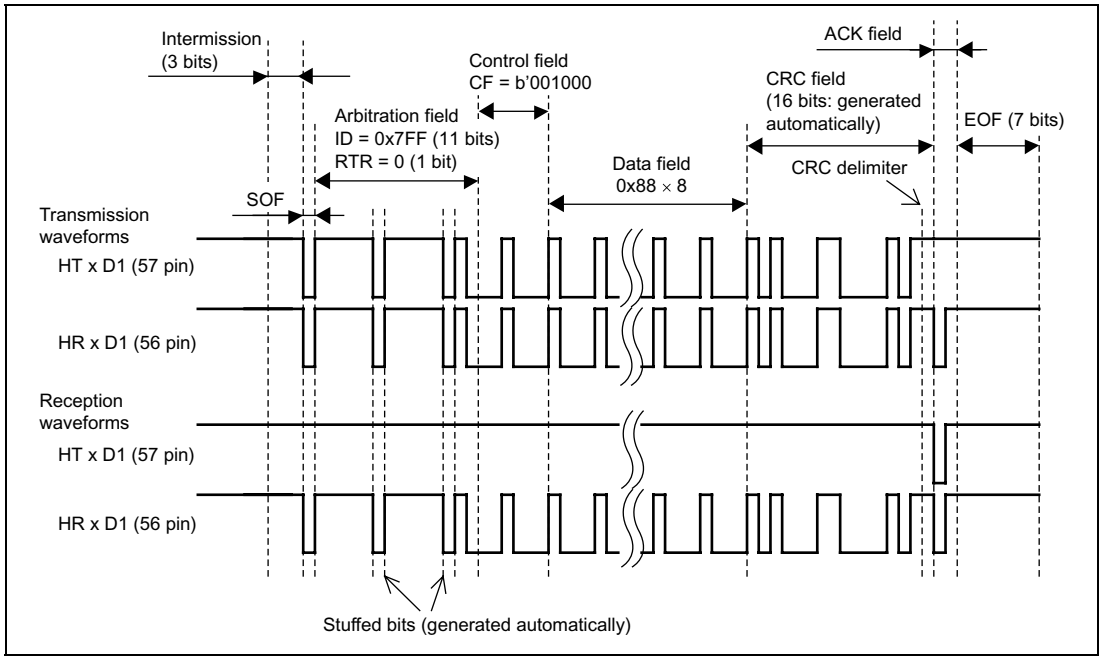

Figure 4.6 Operation Waveforms (End of Operation)

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

Register			Function
Common transmission and reception registers	MCR		Master control register Controls operation of HCAN-2.
	BCR0 BCR1		Bit timing configuration registers 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.
		MC4	Message control register Used to select transmission or reception.
		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 acknowledge register Indicates that the transmitted message has been properly transmitted to the corresponding mailbox.

Register		Function
Reception register	RXPR	Reception end register Indicates that the data has been properly received by the corresponding mailbox.
	RFPR	Remote request register Indicates reception of remote frames 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 2.2 DTC Function Allocation

Register	Function
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.
DTCSR	DTC control/status register Used to enable or prohibit software triggering of DTC and to set the vector address for software triggering of DTC.
IPRG	Interrupt priority register Used to set the priority of DTC interrupt requests.

5.3 Transmission Flowchart

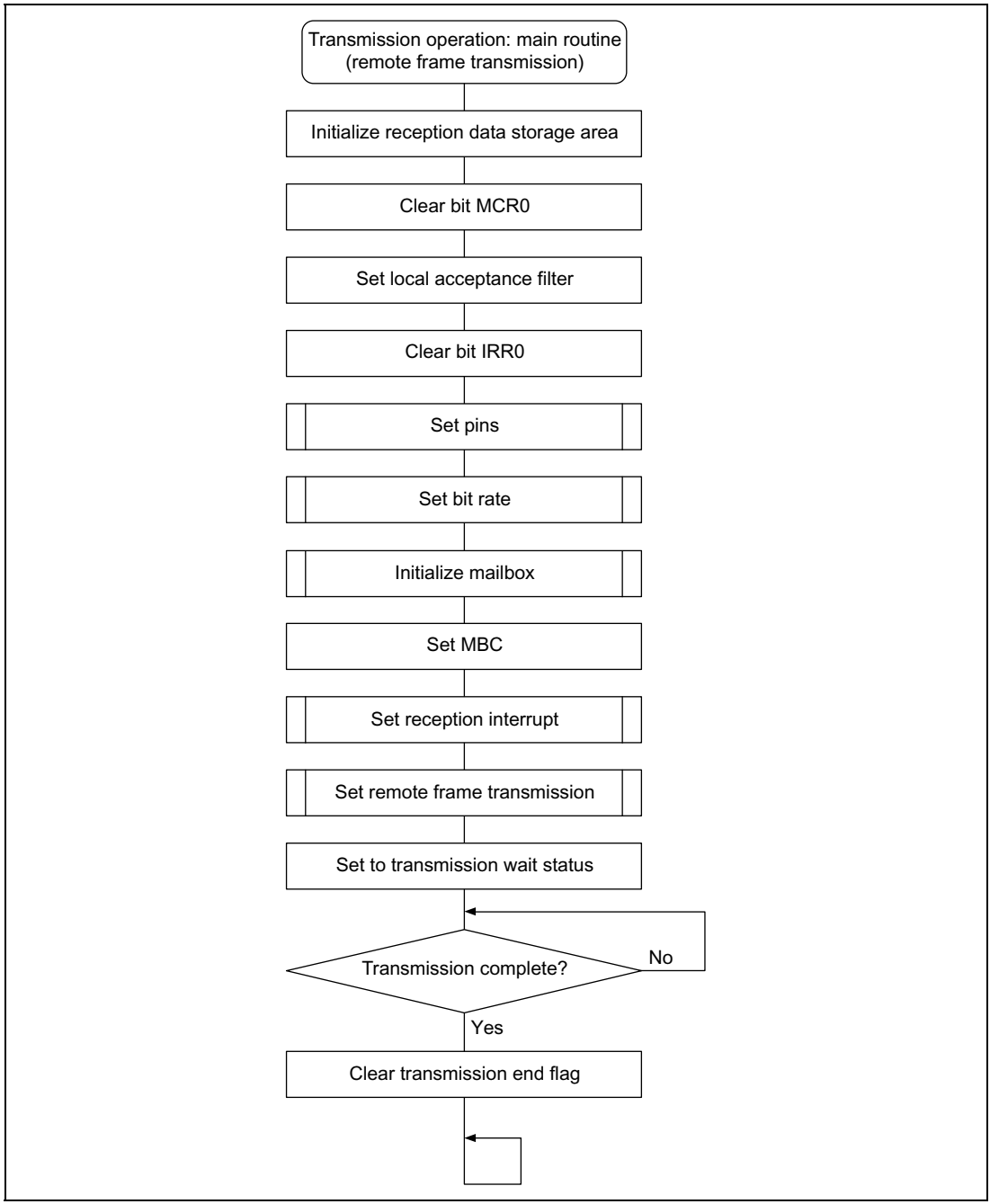

Figure 5.1 Transmission Flowchart (1)

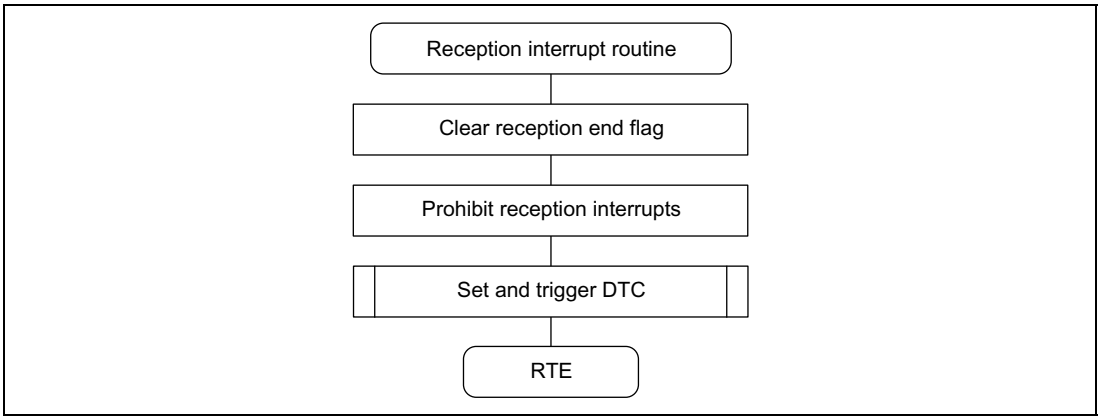

Figure 5.2 Transmission Flowchart (2)

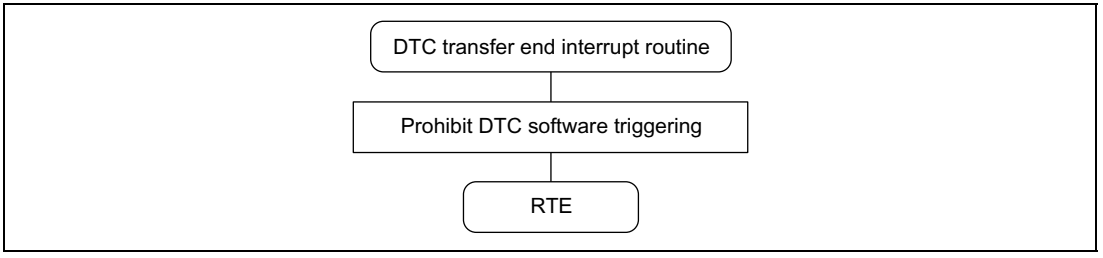

Figure 5.3 Transmission Flowchart (3)

5.4 Software Description (Transmission)

Module Description

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

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

Register	Function	Initial Value	Module
MBbuff	Storage area for received data (address: H'FFFFD100–H'FFFFD108).	—	Main routine
work	Work register used for mailbox initialization.	—	
HCAN_MCR	Clears reset request bit.	0x0000	
HCAN_IRR	Clears reset, hold, and sleep interrupt flags. (To clear, write 1.)	0x0001	
HCAN_BCR0	Sets bit rate to 250 kbps when ϕ is 50 MHz.	0x0009	
HCAN_BCR1		0x4300	
HCAN_MB1.MC0	Sets remote frame and standard format for mailbox 1. Also sets identifier (H'555).	0x5558	
HCAN_MB1.MC4	Sets mailbox 1 to allow remote frame transmission and data frame reception.	0x05	
HCAN_MB1.MC5	Sets mailbox 1 transmission size to a data length of 8 bytes.	0x08	
HCAN_MB1.LAFM15	Sets identifier filter mask for mailbox 1.	0x0000	
HCAN_IMR	Enables message reception interrupts.	0xFFFD	
HCAN_MBIMR0	Enables mailbox 1 interrupt requests.	0xFFFD	
INTC.IPRK	Sets priority of HCAN-2 interrupt requests.	0x00F0	
INTC.IPRG	Sets priority of DTC interrupt requests.	0x0E00	
HCAN_TXPR0	Sets mailbox 1 to transmission wait status.	0x0002	
HCAN_TXACK0	Checks and clears mailbox 1 transmission end flag. (To clear, write 1.)	0x0002	

Register	Function	Initial Value	Module
HCAN_RXPR0	Checks and clears mailbox 1 reception end flag. (To clear, write 1.)	0x0002	Reception interrupt routine
HCAN_IMR	Prohibits message reception interrupts.	0xFFFF	
DTMR	Increments both DTSAR and DTDAR after transfer completes, sets block transfer mode and byte transfer.	0xA890	
DTSAR	Sets transfer source address as mailbox 1 message data area.	H'FFFFB128	
DTDAR	Sets transfer destination address as received data storage area.	H'FFFFD100	
DTCRA	Sets number of block transfers (1).	0x0001	
DTCRB	Sets block length (8 bytes).	0x0008	
DTC.DTBR	Sets top 16 bits of memory address at which data transferred using DTC is stored.	0xFFFF	
DTC.DTCSR	Sets DTC software trigger and DTC vector address (H'0000045A).	0x015A	
DTC.DTCSR	Prohibits DTC software triggering.	0x06FF	DTC transfer end interrupt routine

5.5 Transmission Program Listing

```
/*****************************************************************************/
/*     HCAN-2 Transmission 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 t_main(void );
void RM1_IRR1(void);
void DTC_SWDTEND(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 t_main(void){
     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 = 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 = 0x05;               /* Set mailbox 1 to enable remote frame
transmission and data frame reception */
/* Set reception interrupt */
      HCAN_IMR = 0xFFFD;                 /* Enable message reception interrupts */
      HCAN_MBIMR0 = 0xFFFD;              /* 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 */
      DTMR = 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 = 0x0001;                       /* Block transfer: 1 block */
      DTCRB = 0x0008;                       /* Block length: 8 bytes */
      DTC.DTBR = 0xFFFF;                    /* 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

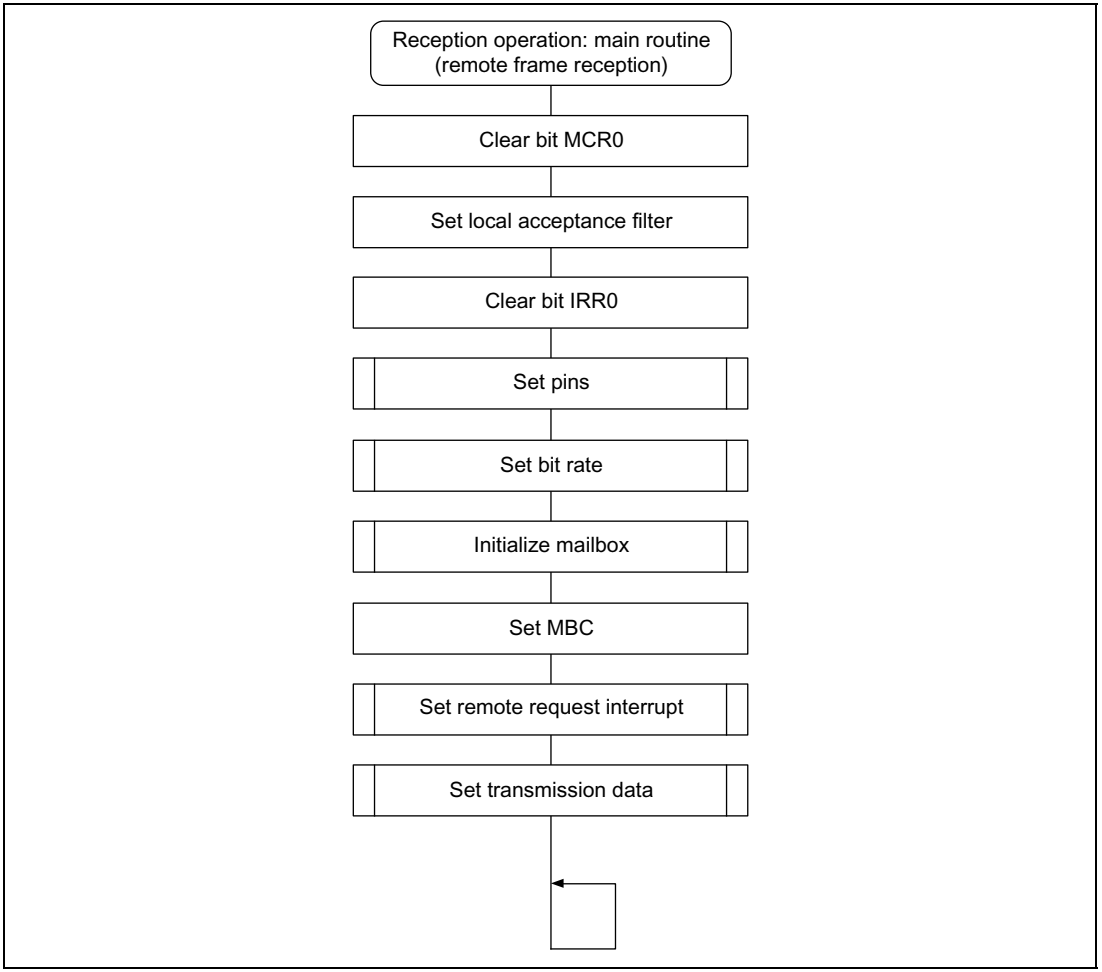

Figure 5.4 Reception Flowchart (1)

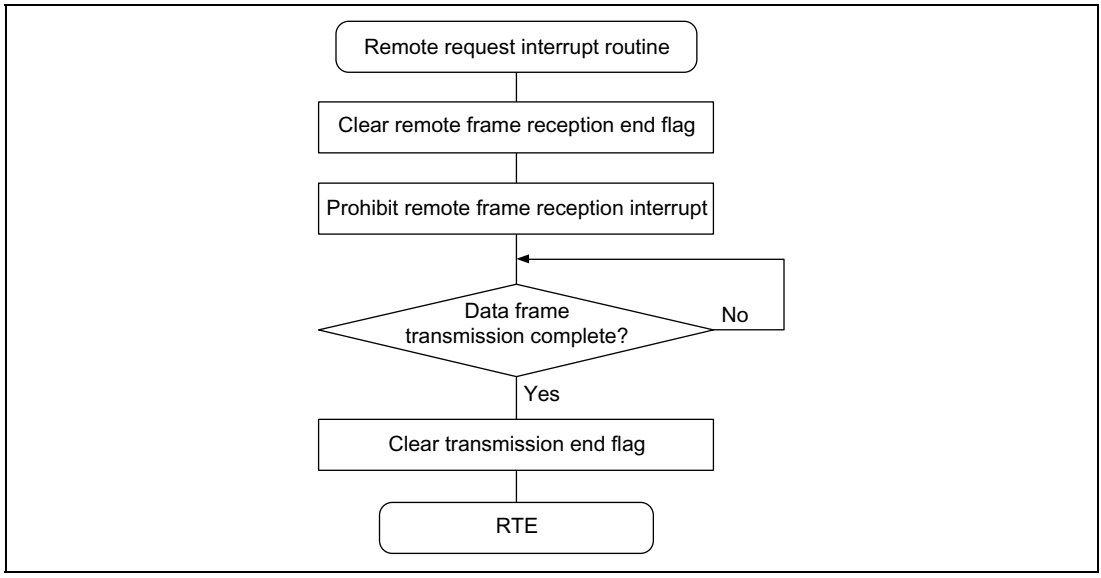

Figure 5.5 Reception Flowchart (2)

5.7 Software Description (Reception)

Module Description

Module	Label	Function
Main routine	r_main	Performs initial settings and reception settings for HCAN-2.
Remote request interrupt routine	RM1_IRR2	Clears remote frame reception end flag, sets interrupt prohibition, and checks and clears transmission end flag.

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

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

Register	Function	Initial Value	Module
HCAN_RFPR0	Clears mailbox 1 remote request transmission end flag. (To clear, write 1.)	0x0002	Remote request interrupt routine
HCAN_IMR	Prohibits remote request interrupts.	0xFFFF	
HCAN_TXACK0	Checks and clears mailbox 1 transmission end flag. (To clear, write 1.)	0x0002	

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 = 0xFFFB;                 /* Enable remote request interrupts */
      HCAN_MBIMR0 = 0xFFFD;              /* Enable mailbox 1 interrupt requests */
      INTC.IPRK = 0x00F0;                /* Set RM1 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_RFPR0 = 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) */

}
```

5.9 Operation Waveforms (Transmission and Reception)

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

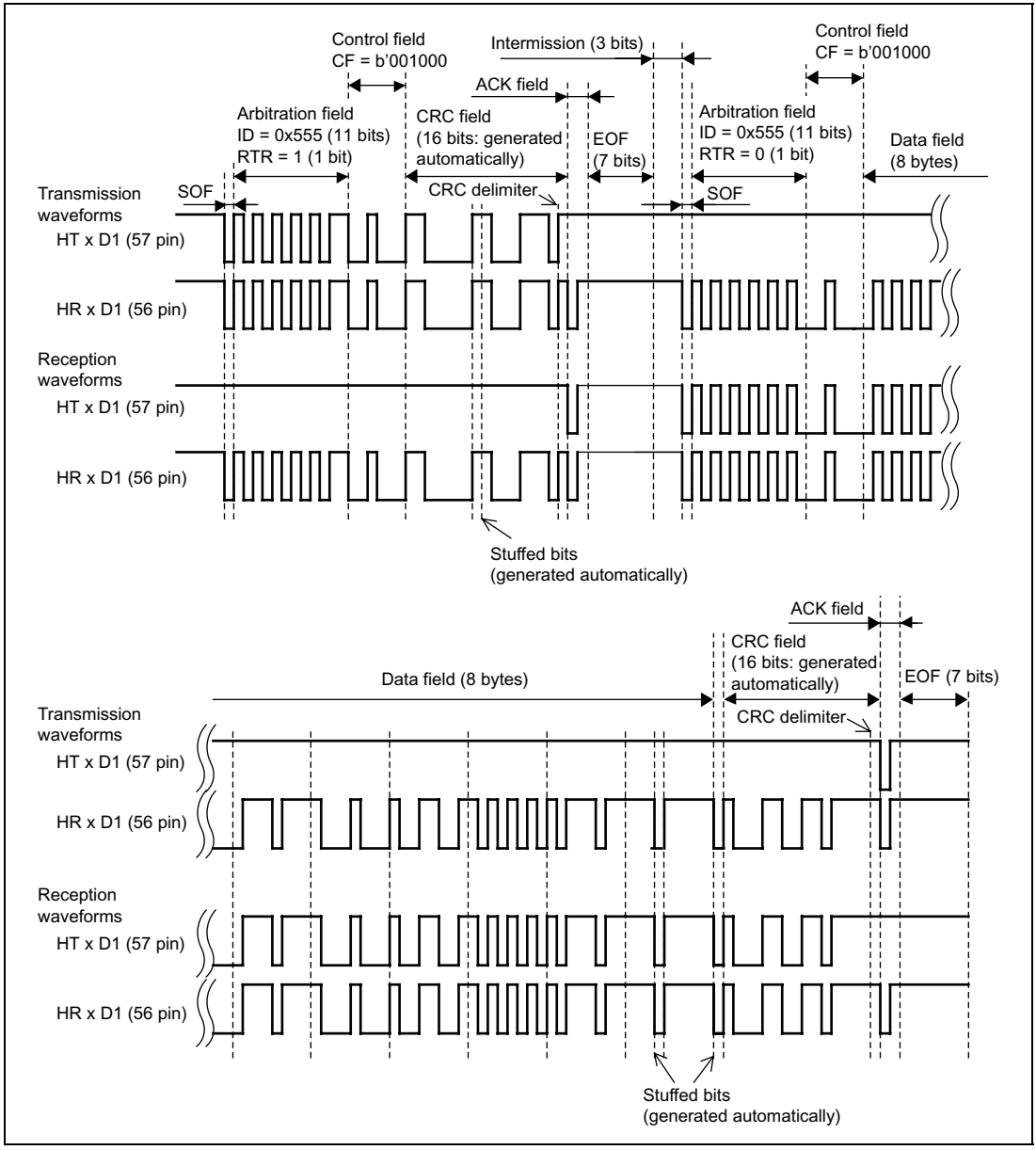

Figure 5.6 Operation Waveforms

Reference Documents

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

SH7047F HCAN2 Application Note

Publication Date: 1st Edition, August 18, 2003
Published by: Sales Strategic Planning Div.
Renesas Technology Corp.
Edited by: Technical Documentation & Information Department
Renesas Kodaira Semiconductor Co., Ltd.

Renesas Technology Corp. Sales Strategic Planning Div. Nippon Bldg., 2-6-2, Ohte-machi, Chiyoda-ku, Tokyo 100-0004, Japan

http://www.renesas.com

SH7047F HCAN2
Application Note

Renesas Electronics Corporation
1753, Shimonumabe, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8668 Japan

REJ05B0144-0100O