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.

Controller Area Network (RCAN-ET)

Application Note

Summary

This application note consists of examples of communications operation using RCAN-ET. It was compiled as a reference for users performing software and hardware design.

Although the operation of each sample task and sample application contained in this application note has been verified, make sure to conduct operation checks before actually using them.

Device on which Operation Has Been Confirmed

H8SX/1544

Table of Contents

1. Overview of RCAN-ET

1.1 Features of RCAN-ET

1.1.1 Features

The main features of RCAN-ET are as follows.

- Supports CAN specification 2.0B and ISO-11898.
- 16 mailboxes (15 transmit and receive, 1 receive only)
- Supports receive filter mask (LAFM) for all mailboxes.
- Built-in test function
- Register configuration analogous to HCAN2

1.1.2 Additional Functions

The following functions have been newly added to RCAN-ET.

- ID reordering function (Order of ID, RTR, and IDE can be set to provide compatibility with HCAN2.)
- Bus-off halt (Enables immediate transition to halt mode after transition to bus-off status.)
- IDE bit filter mask bit added.
- Mailboxes comprise RAM and registers.

1.2 I/O Pins

The RCAN-ET pin configuration of the H8SX/1544 is shown in table 1.2.1. The H8SX/1544 has two-channel RCAN-ET functionality built in.

Table 1.2.1 Pin Configuration (H8SX/1544)

Channel	Name	Pin Name	I/O	Function
0	Transmit data pin	CTx_0	Output	CAN bus transmit pin
	Receive data pin	CRx_0	Input	CAN bus receive pin
1	Transmit data pin	CTx_1	Output	CAN bus transmit pin
	Receive data pin	CRx_1	Input	CAN bus receive pin

1.3 Register Functions

Table 1.3.1 lists the register functions of RCAN-ET.

Table 1.3.1 Register Functions

Register Name	Abbreviation	Function
Message control 0	MB[x].CONTROL0	Selects standard format/extended format, selects data frame/remote frame, and sets standard ID and extended ID.
Local acceptance filter	MB[x].LAFM	Sets filter mask for IDE bit, standard ID, and extended ID.
Message data	MB[x].MSG_DATA[n] (n = 0 to 7)	Sets transmit and receive data.
Message control 1	MB[x].CONTROL1	Sets new message control (NMC), automatic data frame transmit (ATX), disable automatic retransmit (DART), mailbox configuration (MBC), and data length (DLC).
Master control register	MCR	Controls RCAN-ET operation.
General status register	GSR	Indicates RCAN-ET status.
Bit timing configuration register	BCR1/BCR0	Sets CAN bit timing parameters and baud rate prescaler.
Interrupt request register	IRR	Status flags of interrupt sources.
Interrupt mask register	IMR	Masks interrupt requests corresponding to IRR bits.
Transmit error counter Receive error counter	TEC/REC	Counter indicating the number of transmit/receive message errors.
Transmit wait register	TXPR	Sets a message stored in a mailbox to transmit wait status.
Transmit cancel register	TXCR	Cancels a transmit request for a message in transmit-wait status.
Transmit acknowledge register	TXACK	Status register indicating that a message in transmit status in a mailbox has been transmitted properly.
Abort acknowledge register	ABACK	Status register indicating that a message in transmit status in a mailbox has been cancelled properly.
Data frame receive end register	RXPR	Status register indicating that a mailbox has properly received a data frame.
Remote frame receive end register	RFPR	Status register indicating that a mailbox has properly received a remote frame.
Mailbox interrupt mask register	MBIMR	Enables interrupt requests for mailboxes.
Unread message status register	UMSR	Status register indicating that an overwrite/overrun has occurred because a mailbox containing an unread message has received a new message.
RCAN-ET monitor register	RCANMON*[2]	Enables/disables RCAN-ET transmit and receive pins, performs transmit-stop control of transmit pins, and monitors status of transmit and receive pins.

Notes: 1. x: Mailbox number (0 to 15)
2. Some devices do not have an RCAN-ET monitor register (RCANMON).
3. The RCAN-ET registers have access size limitations. See the hardware manual for information on the access sizes of individual registers.

1.4 Interrupt Sources

Table 1.4.1 lists the RCAN-ET interrupt sources. With the exception of the reset interrupt (IRR0) triggered by a power-on reset, these sources are maskable. The mailbox interrupt mask register (MBIMR) and interrupt mask register (IMR) are used to mask interrupt sources. See section 5, Interrupt Controller, in the *H8SX/1544 Hardware Manual* for information on the interrupt vectors of the interrupt requests.

Table 1.4.1 RCAN-ET Interrupt Sources

Name	Description	Interrupt Flag	DMAC Startup
ERS_n	Error passive (TEC ≥ 128 or REC ≥ 128)	IRR5	No
	Bus-off (TEC ≥ 256)/recovery from bus-off	IRR6	
	Error warning (TEC ≥ 96)	IRR3	
	Error warning (REC ≥ 96)	IRR4	
OVR_n	Message error detect (enabled in test mode only)	IRR13	
	Reset/halt/CAN sleep transition	IRR0	
	Overload frame transmit	IRR7	
	Unread message overwrite/overrun	IRR9	
	CAN bus operation detect in CAN sleep status	IRR12	
SLE_n	Message transmit/receive cancel (mailbox empty)	IRR8	
RM1_n	Mailbox 1 to 15 remote frame receive	IRR2	
	Mailbox 1 to 15 data frame receive	IRR1	
RM0_n	Mailbox 0 remote frame receive	IRR2	Yes
	Mailbox 0 data frame receive	IRR1	

Note: n: RCAN channel number (0 or 1)

1.5 Bit Rate Setting

The bit timing configuration registers (BCR0 and BCR1) are used to set the RCAN-ET bit rate. The following formula is used to calculate the bit rate.

Bit rate = fclk / 2 × (BRP + 1) × (TSEG1 + TSEG2 + 1)

Notes: BRP: Bit rate prescaler
TSEG1 and TSEG2: Time segments 1 and 2
fclk: Peripheral clock (Pϕ)

This section describes how to set the BCR registers by using a BCR listing. If a BCR listing is used, it is possible to make BCR register settings that fall within the CAN specifications without the need to calculate values by means of the above formula.

BCR Table Usage Method

1. Select the bit rate from among 1 Mbps, 500 kbps, and 250 kbps.*
2. Select the peripheral clock (Pϕ) setting value from among 16 MHz, 18 MHz, or 20 MHz.*
3. Select setting values for SJW (resynchronize jump width) and BSP (bit sample point). See table 1.5.1 for the setting method.
4. Set the corresponding BCR register values according to the BCR listing (tables 1.5.2 to 1.5.4). Also, confirm that the SJW setting value is less than or equal to TSEG2.
   1 Mbps bit rate: See table 1.5.2.
   500 kbps bit rate: See table 1.5.3.
   250 kbps bit rate: See table 1.5.4.

Note: * When using a bit rate other than 1 Mbps, 500 kbps, or 250 kbps, or a Pϕ frequency of other than 16 MHz, 18 MHz, or 20 MHz, calculate the BCR setting values by using the above formula. Refer to the hardware manual for details of the setting method.

Usage example: BCR register settings when the bit rate is 500 kbps, Pϕ = 16 MHz, and the sample point is 75%

Pϕ (MHz)	SP*[1] (%)	1-bit TQ count	SYNC_SEG	TSEG1	TSEG2	BCR1*[2]	BCR0
16	68.8	16	1	10	5	H'94xx	H'0000
	75.0	8	1	5	2	H'41xx	H'0001
		16	1	11	4	H'A3xx	H'0000

Notes: 1. SP: Sample point = (1 + TSEG1) TQ count / 1-bit TQ count
(TQ: Time quantum)
2. The 8 lower-order bits of BCR1 set the SJW (resynchronize jump width) and BSP (bit sample point).

According to the above table, the BCR register settings are as follows when SJW = 0 and BSP = 0.

BCR1 = H'4100, BCR0 = H'0001 or BCR1 = H'A300, BCR0 = H'0000

Table 5.1.1 shows the setting method for SJW and BSP, and tables 5.1.2 to 5.1.4 list BCR register settings for bit rates of 1 Mbps, 500 kbps, and 250 kbps.

Table 1.5.1 SJW and BSP Setting Method

Bit Name	Bits	Function	Setting Value	Setting Description
SJW	Bits 5 and 4 in BCR1	Resynchronize jump width	0	Jump width = 1 time quantum
			1	Jump width = 2 time quanta
			2	Jump width = 3 time quanta
			3	Jump width = 4 time quanta
BSP	Bit 0 in BCR1	Bit sample point	0	Bit sampling at 1 location (end of TSEG1)
			1	Bit sampling at 3 locations (rising edges of last 3 clock cycles of TSEG1)

Notes: 1. Bits 7, 6, 3, 2, and 1 in BCR1 are reserved. Always write 0 to them.
2. Under the CAN specification SJW must satisfy the following conditions: 1 ≤ SJW ≤ 4TQ (time quanta), TSEG1 > TSEG2 ≥ SJW.

Table 1.5.2 BCR Listing (1 Mbps)

Pϕ (MHz)	SP*1 (%)	1-bit TQ count	SYNC_SEG	TSEG1	TSEG2	BCR1*2	BCR0
16	62.5	8	1	4	3	H'32xx	H'0000
	75.0	8	1	5	2	H'41xx	H'0000
18	66.7	9	1	5	3	H'42xx	H'0000
	77.8	9	1	6	2	H'51xx	H'0000
20	60.0	10	1	5	4	H'43xx	H'0000
	70.0	10	1	6	3	H'52xx	H'0000
	80.0	10	1	7	2	H'61xx	H'0000

Notes: 1. SP: Sample point = (1 + TSEG1) TQ count / 1-bit TQ count
(TQ: Time quantum)
2. The 8 lower-order bits of BCR1 set the SJW (resynchronize jump width) and BSP (bit sample point). See table 1.5.1 for the setting method.

Table 1.5.3 BCR Listing (500 kbps)

Pϕ (MHz)	SP*1 (%)	1-bit TQ count	SYNC_SEG	TSEG1	TSEG2	BCR1*2	BCR0
16	56.3	16	1	8	7	H'76xx	H'0000
	62.5	8	1	4	3	H'32xx	H'0001
		16	1	9	6	H'85xx	H'0000
	68.8	16	1	10	5	H'94xx	H'0000
	75.0	8	1	5	2	H'41xx	H'0001
		16	1	11	4	H'A3xx	H'0000
	81.3	16	1	12	3	H'B2xx	H'0000
	87.5	16	1	13	2	H'C1xx	H'0000
18	55.6	18	1	9	8	H'87xx	H'0000
	61.1	18	1	10	7	H'96xx	H'0000
	66.7	9	1	5	3	H'42xx	H'0001
		18	1	11	6	H'A5xx	H'0000
	72.2	18	1	12	5	H'B4xx	H'0000
	77.8	9	1	6	2	H'51xx	H'0001
		18	1	13	4	H'C3xx	H'0000
	83.3	18	1	14	3	H'D2xx	H'0000
	88.9	18	1	15	2	H'E1xx	H'0000
20	60.0	10	1	5	4	H'43xx	H'0001
		20	1	11	8	H'A7xx	H'0000
	65.0	20	1	12	7	H'B6xx	H'0000
	70.0	10	1	6	3	H'52xx	H'0001
		20	1	13	6	H'C5xx	H'0000
	75.0	20	1	14	5	H'D4xx	H'0000
	80.0	10	1	7	2	H'61xx	H'0001
		20	1	15	4	H'E3xx	H'0000
	85.0	20	1	16	3	H'F2xx	H'0000

Notes: 1. SP: Sample point = (1 + TSEG1) TQ count / 1-bit TQ count
(TQ: Time quantum)
2. The 8 lower-order bits of BCR1 set the SJW (resynchronize jump width) and BSP (bit sample point). See table 1.5.1 for the setting method.

Table 1.5.4 BCR Listing (250 kbps)

Pϕ (MHz)	SP*1 (%)	1-bit TQ count	SYNC_SEG	TSEG1	TSEG2	BCR1*2	BCR0
16	56.3	16	1	8	7	H'76xx	H'0001
	62.5	8	1	4	3	H'32xx	H'0003
		16	1	9	6	H'85xx	H'0001
	68.8	16	1	10	5	H'94xx	H'0001
	75.0	8	1	5	2	H'41xx	H'0003
		16	1	11	4	H'A3xx	H'0001
	81.3	16	1	12	3	H'B2xx	H'0001
	87.5	16	1	13	2	H'C1xx	H'0001
18	55.6	18	1	9	8	H'87xx	H'0001
	61.1	18	1	10	7	H'96xx	H'0001
	66.7	9	1	5	3	H'42xx	H'0003
		18	1	11	6	H'A5xx	H'0001
	72.2	18	1	12	5	H'B4xx	H'0001
	77.8	9	1	6	2	H'51xx	H'0003
		18	1	13	4	H'C3xx	H'0001
	83.3	18	1	14	3	H'D2xx	H'0001
	88.9	18	1	15	2	H'E1xx	H'0001
20	60.0	10	1	5	4	H'43xx	H'0003
		20	1	11	8	H'A7xx	H'0001
	65.0	20	1	12	7	H'B6xx	H'0001
	70.0	10	1	6	3	H'52xx	H'0003
		20	1	13	6	H'C5xx	H'0001
	75.0	20	1	14	5	H'D4xx	H'0001
	80.0	10	1	7	2	H'61xx	H'0003
		20	1	15	4	H'E3xx	H'0001
	85.0	20	1	16	3	H'F2xx	H'0001

Notes: 1. SP: Sample point = (1 + TSEG1) TQ count / 1-bit TQ count
(TQ: Time quantum)
2. The 8 lower-order bits of BCR1 set the SJW (resynchronize jump width) and BSP (bit sample point). See table 1.5.1 for the setting method.

2. RCAN-ET Application Examples

2.1 Initial Settings

This section describes the initial settings of RCAN-ET.

2.1.1 Specification

Initial RCAN-ET settings are performed to transmit a CAN message with standard ID H'555, as shown in figure 2.1.1.

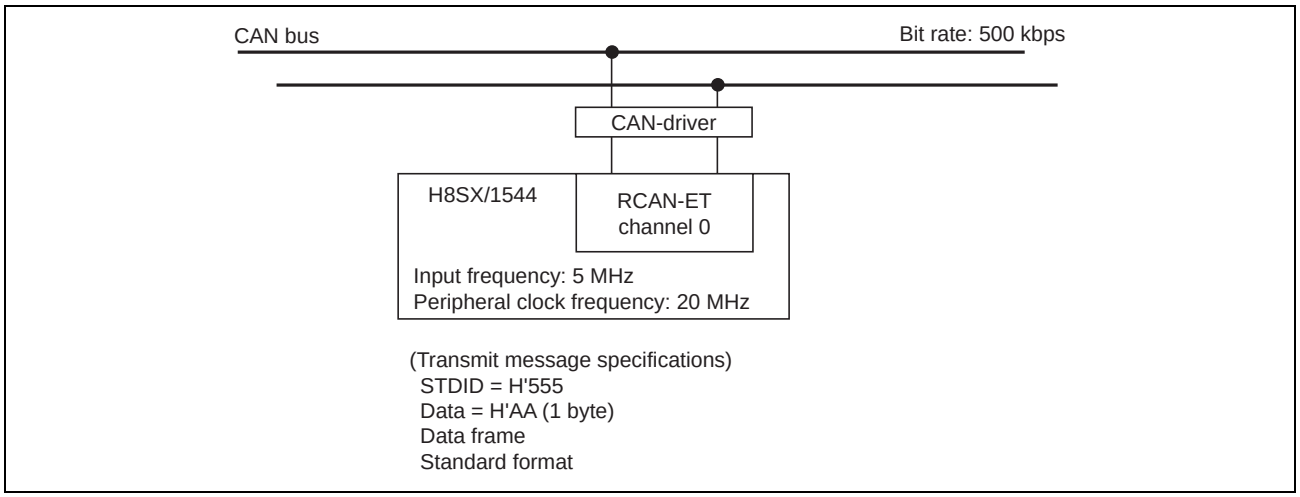


Figure 2.1.1 Initial Setting Specifications

2.1.2 Software Description

(1) Module Description

Table 2.1.1 lists the common initial setting modules used in this application note.

Table 2.1.1 Initial Setting Modules

Module	Label	Function	Settings
H8SX/1544 initial settings	set_1544_init	Performs clock settings and transmit and receive pin settings for the H8SX/1544. Cancels RCAN-ET module-stop mode.	Device-specific
H8SX/1544 interrupt settings	set_1544_INTC_init	Performs RCAN-ET interrupt priority level setting for the H8SX/1544.	
RCAN initial settings	set_RCAN0_init	Sets RCAN-ET reset request, clears IRR0.	RCAN-ET common
RCAN mailbox initial settings	set_RCAN0MB_init	Set mailbox to be used for transmit or receive.	
RCAN start	set_RCAN0_start	Cancels RCAN-ET reset.	

(2) Description of Registers Used

Tables 2.1.2 and 2.1.3 list the register settings of the initial setting modules. The settings of registers indicated in the tables as changeable may be changed according to the system specifications.

Table 2.1.2 Description of Registers Used (Device-Specific)

Module	Register	Setting Value	Function	Setting Value Changeable*[1]
H8SX/1544 initial settings	SCKCR.BIT.ICK	0	Sets the system clock frequency to 40 MHz (input clock: 5 MHz × 8).	Yes
	SCKCR.BIT.BCK	1	Sets the peripheral module clock frequency to 20 MHz (input clock: 5 MHz × 4).	
	SCKCR.BIT.PCK	1	Sets the external bus clock frequency to 20 MHz (input clock: 5 MHz × 4).	
	SBYCR.BIT.SSBY	1	Enables transition to software standby mode after execution of Sleep instruction.	
	MSTP.CRC.BIT._RCAN01	0	Cancels RCAN-ET module-stop mode.	No*[2]
	RCANET0.RCANMON.BYTE	0x20	Enables RCAN-ET transmit and receive pins.	
	P6.ICR.BIT.B4	1	Sets P64 (pin 67) as RCAN-ET channel 0 receive pin.	
H8SX/1544 interrupt settings	INTC.INTCR.BIT.INTM	2	Sets interrupt controller to interrupt control mode 2.	Yes
	INTC.IPRQ.BIT._RCAN01	7	Set the RCAN-ET interrupt priority level.	

Notes: 1. For registers whose settings may be changed, do not alter the sequence in which register settings are made.
2. The setting method for the RCAN-ET transmit and receive pins differs depending on the device. When using a device other than the H8SX/1544, make sure to check the setting method and make any necessary changes.

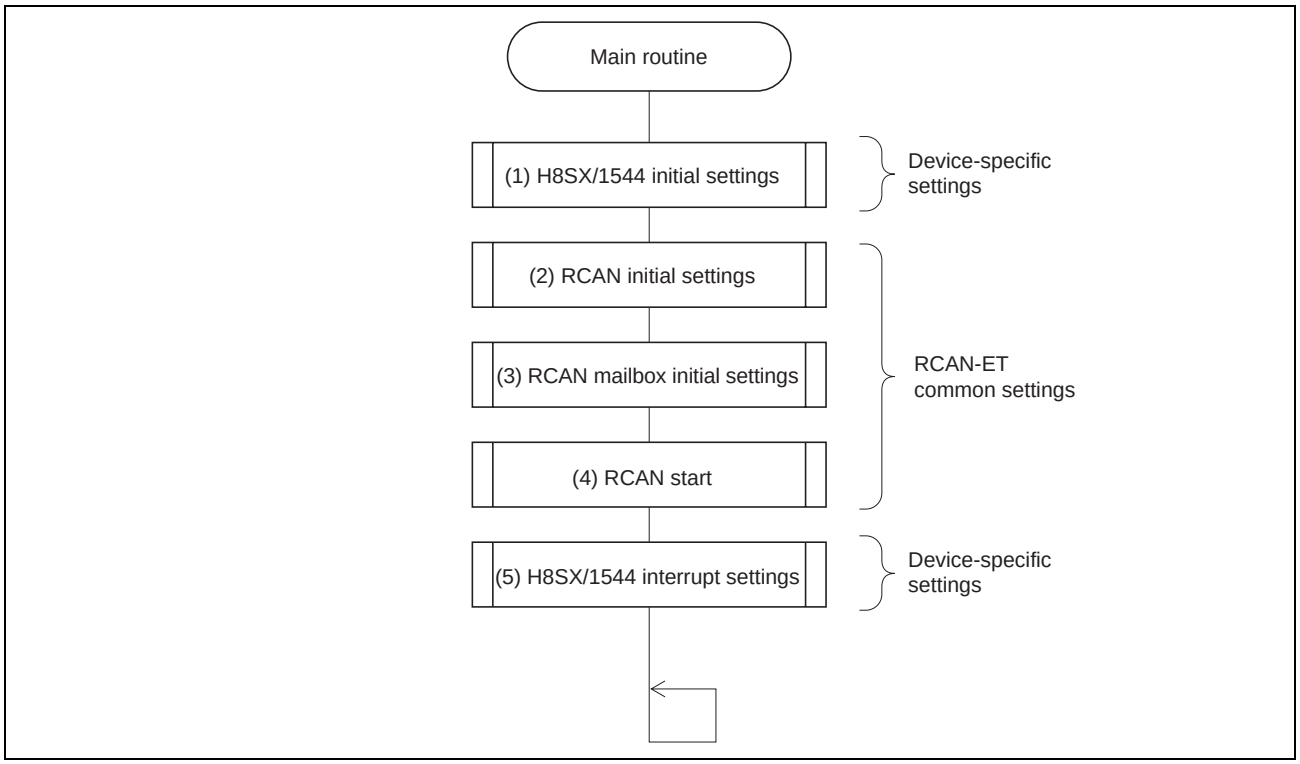
Table 2.1.3 Description of Registers Used (RCAN-ET Common)

Module	Register	Setting Value	Function	Setting Value Changeable*1
RCAN initial settings	RCANET0.MCR.WORD	0x0001	Sets the reset request bit (MCR0 = 1). Note: Set automatically by a hardware reset only.	No
	RCANET0.GSR.WORD	—	Confirms that RCAN-ET is in reset status. (Confirms that GSR3 = 1.)	
	RCANET0.IRR.WORD	—	Confirms that RCAN-ET has transitioned to reset mode. (Confirms that IRR0 = 1.)	
	RCANET0.IRR.WORD	0x0001	Clears reset/halt/sleep interrupt flag (IRR0). (Clearing condition: write 1)	
	RCANET0.MCR.WORD	0x8000	Sets ID order to other than HCAN2 (MCR15 = 1).	
	RCANET0.MB[m].CTRL0.WORD.H RCANET0.MB[m].CTRL0.WORD.L RCANET0.MB[m].LAFM.WORD.H RCANET0.MB[m].LAFM.WORD.L RCANET0.MB[m].MSG_DATA[n] (m = 0 to 15, n = 0 to 7)	0	Initializes mailbox (RAM area).	
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFEFF	Enables mailbox empty interrupt (IMR8 = 0).	Yes
	RCANET0.MBIMR0.WORD	0xFFFD	Enables interrupts for mailbox 1.	
	RCANET0.MB[1].CTRL0.WORD.H	0x1554	Sets mailbox 1 to standard format, data frame. Also sets ID (H'555).	
	RCANET0.MB[1].CTRL1.BYTE.H	0x00	Sets mailbox 1 to transmit.	
	RCANET0.MB[1].CTRL1.BYTE.L	0x01	Sets the data length (1 byte).	
	RCANET0.MB[1].MSG_DATA[0]	0xAA	Sets the transmit data.	
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)	
	RCANET0.BCR0.WORD	0x0001		
RCAN start	RCANET0.MCR.WORD	0xFFFE	Clears reset request bit (MCR0 = 0).	No
	RCANET0.GSR.WORD	—	Confirms that RCAN-ET reset has been cancelled. (Confirms that GSR3 = 0.)	

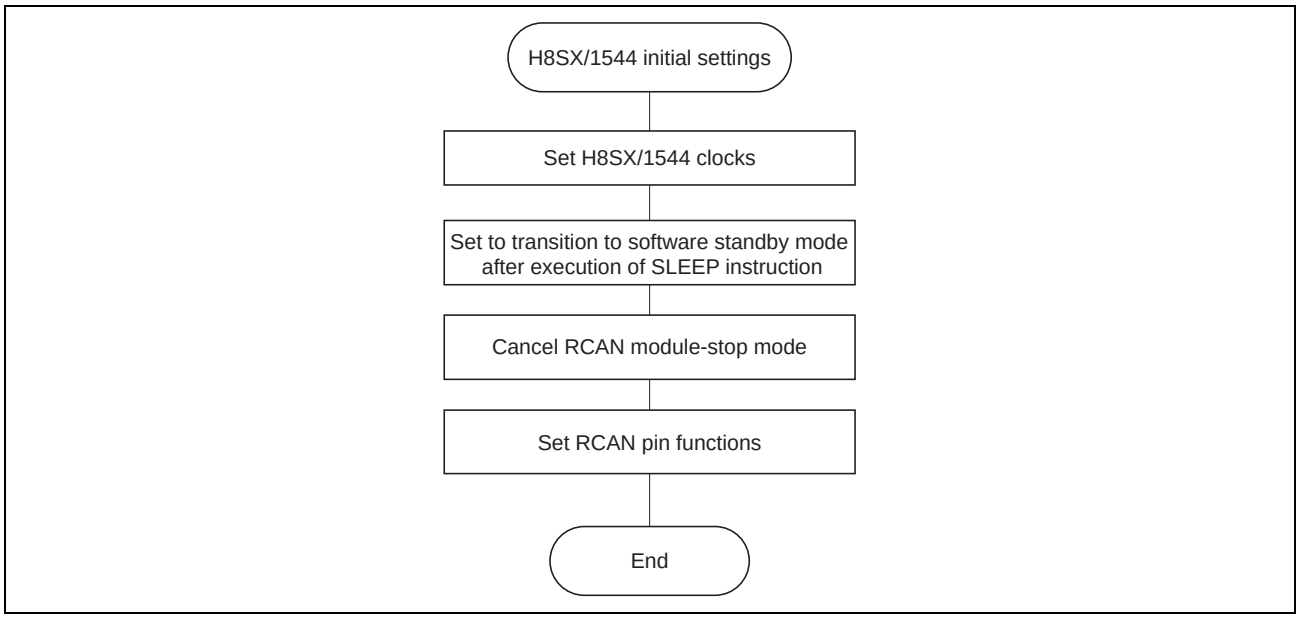
Notes: 1. For registers whose settings may be changed, do not alter the sequence in which register settings are made.
2. The bit names listed in the table correspond to the bit names used in the hardware manual.

2.1.3 Flowcharts

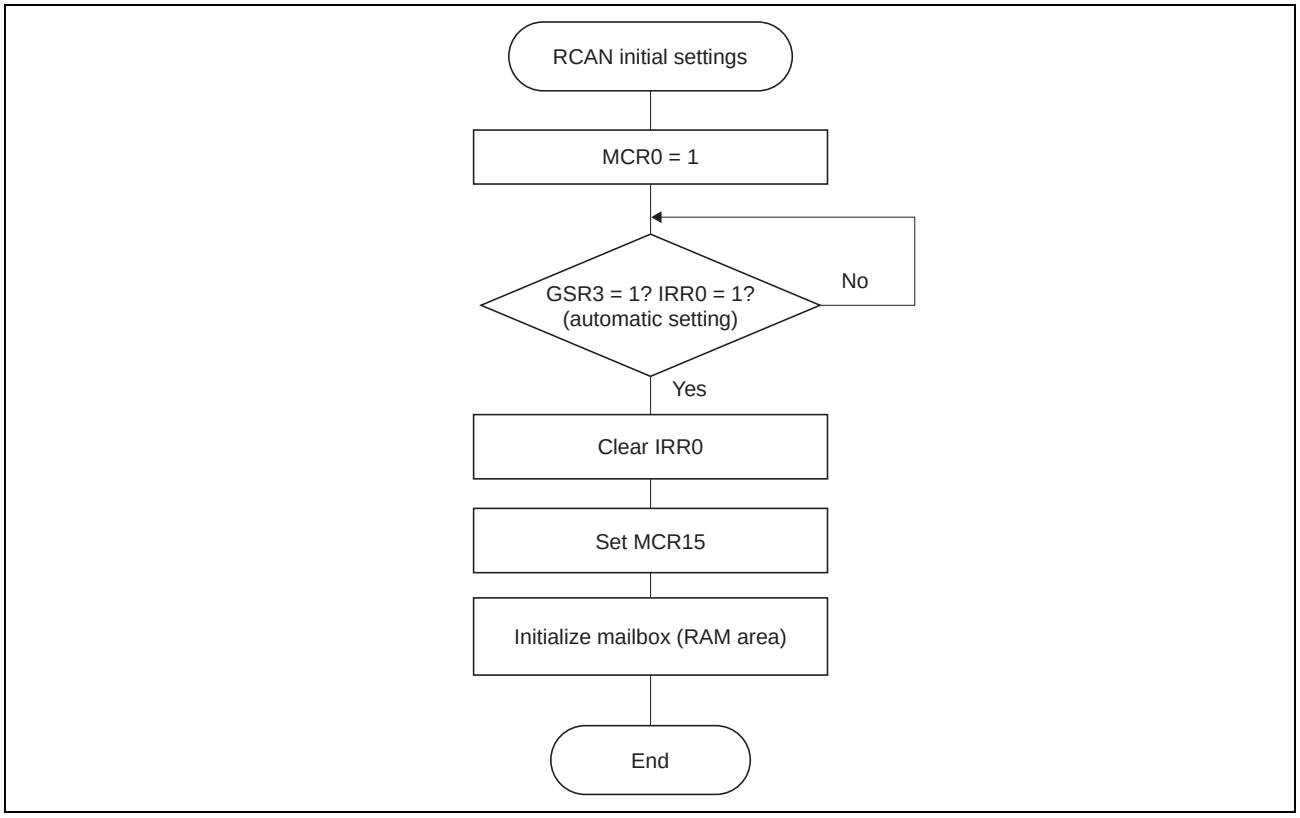
Main Routine



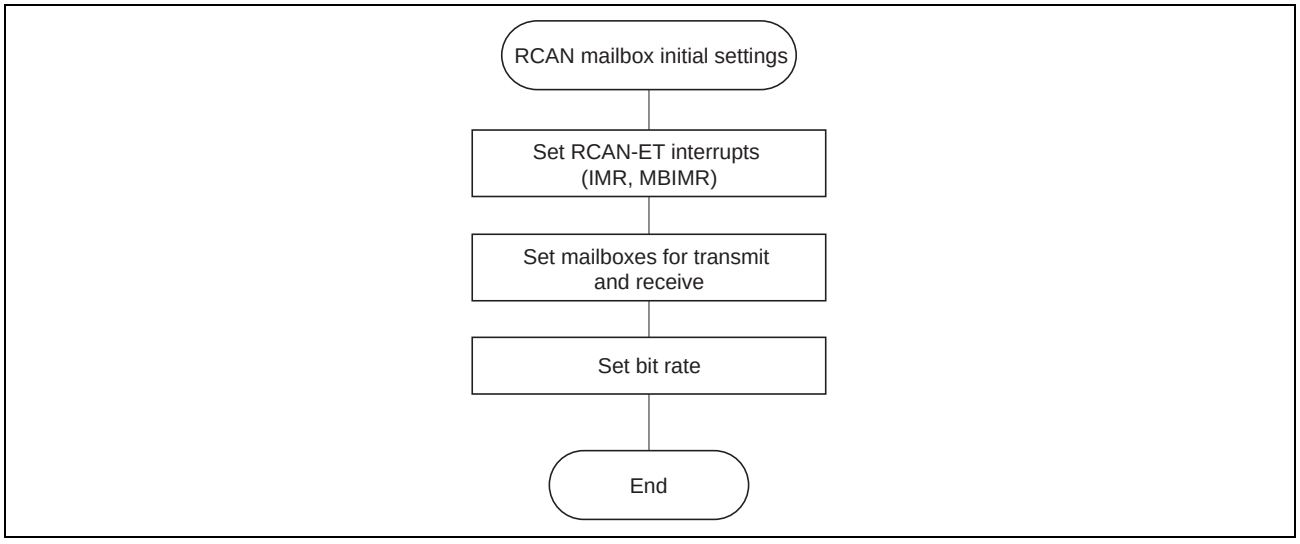
(1) H8SX/1544 Initialize Routine



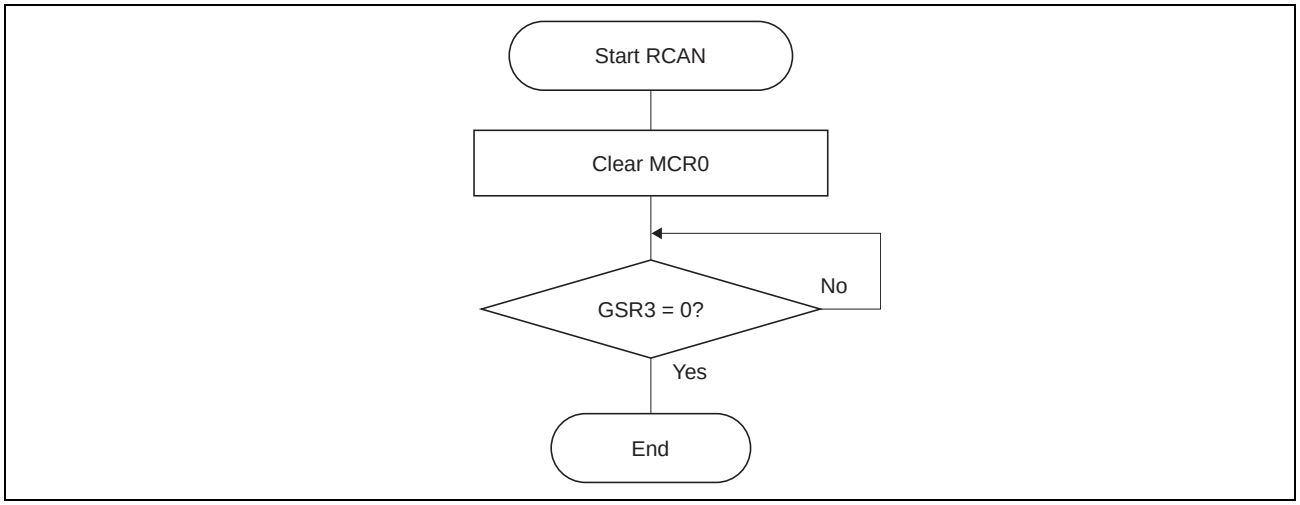
(2) RCAN Initial Settings Routine



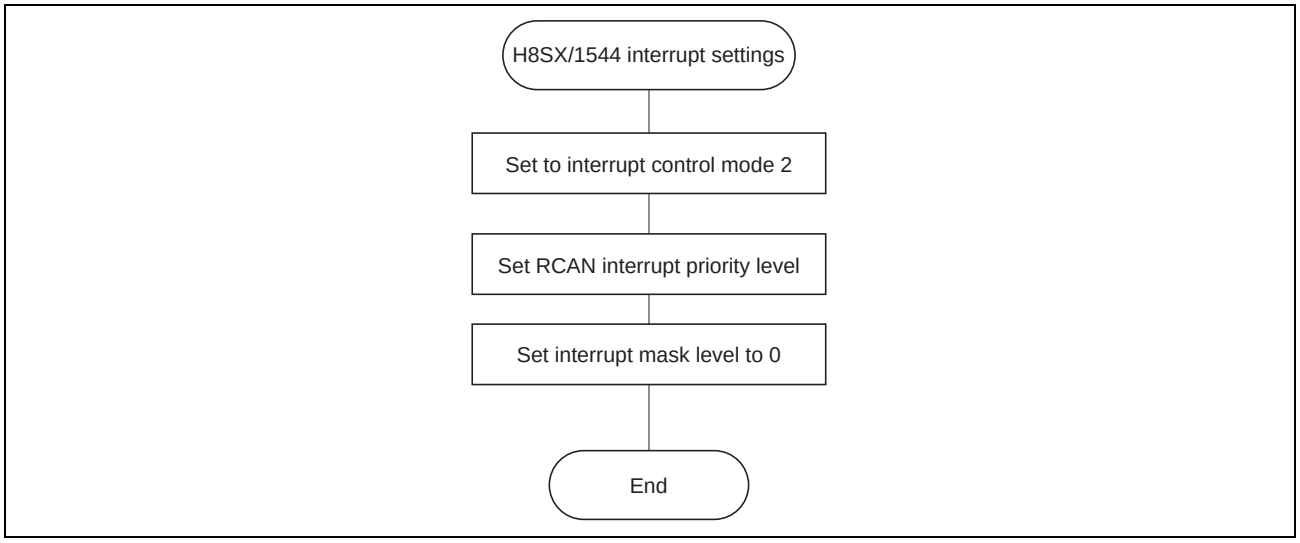
(3) RCAN Mailbox Initial Settings Routine



(4) RCAN Start Routine



(5) H8SX/1544 Interrupt Settings Routine



2.1.4 Program Listing

(1) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{

    set_1544_init();                        /*  H8SX/1544 initialization    */
    set_RCAN0_init();                       /*  RCAN initialization         */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();                   /*  H8SX/1544 interrupt settings*/

    while(1);

}


/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)       */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz)   */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz) */
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode       */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN*/

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins    */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin)*/

}
```

```
/*****************************************************************************/
/*      H8SX/1544 INTC Initialize routine                                    */
/*****************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;                /* Interrupt control mode 2       */
    INTC.IPRQ.BIT._RCAN01 = 7;              /* Set interrupt priority level: RCAN */
    set_imask_exr(0);                       /* Interrupt mask level 0         */


}
/*****************************************************************************/
```

(2) rcan.c

```
/*****************************************************************************/
/*  Filename    :   rcan.c                                                   */
/*  Written     :   '06/06/01   REV.1.00                                     */
/*  Purpose     :   for H8SX/1544   RCAN-ET                                  */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;           /* Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/* GSR3 = 1? (RCAN-ET reset status)    */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/* IRR0 = 1? (reset/halt/sleep interrupt) */
    RCANET0.IRR.WORD = 0x0001;            /* Clear IRR0 (clearing condition: write 1)    */
    RCANET0.MCR.WORD |= 0x8000;           /* Reorder ID: Set to MCR15 = 1 (initial setting) */

    for(i = 0;i < 16;i++){                /*   Initialize mailbox (RAM area)             */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }


}

/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    RCANET0.IMR.WORD &= 0xFEFF;           /*   Enable mailbox empty interrupt          */
    RCANET0.MBIMR0.WORD &= 0xFFFD;        /*   Enable MB1 interrupt                    */

    RCANET0.MB[1].CTRL0.WORD.H = 0x1554;  /*   ID: H'555, standard format, data frame  */
    RCANET0.MB[1].CTRL1.BYTE.H = 0x00;    /*   Set mailbox 1 to transmit               */
    RCANET0.MB[1].CTRL1.BYTE.L = 0x01;    /*   Data length: 1 byte                     */
    RCANET0.MB[1].MSG_DATA[0] = 0xAA;     /*   Transmit data: 0xAA                     */
```

```
    /*  Bit rate = 500 kbps */
    RCANET0.BCR1.WORD = 0x4300;    /*  TSEG1 = 4 (5tq), TSEG2 = 3 (4tq), SJW = 0, BSP = 0, (pϕ = 20 MHz) */
    RCANET0.BCR0.WORD = 0x0001;                              /*    BRP = 1                          */


}

/*****************************************************************************/
/*      RCAN start routine                                                   */
/*****************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                          /*   Clear MCR0                        */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000); /*   GSR3 = 0?                         */


}
/*****************************************************************************/
```

2.2 Data Frame Transmit and Receive (Standard ID)

2.2.1 Specification

A data frame with standard ID H'555 is transmitted by node A and received by node B (no interrupts used), as shown in figure 2.2.1.

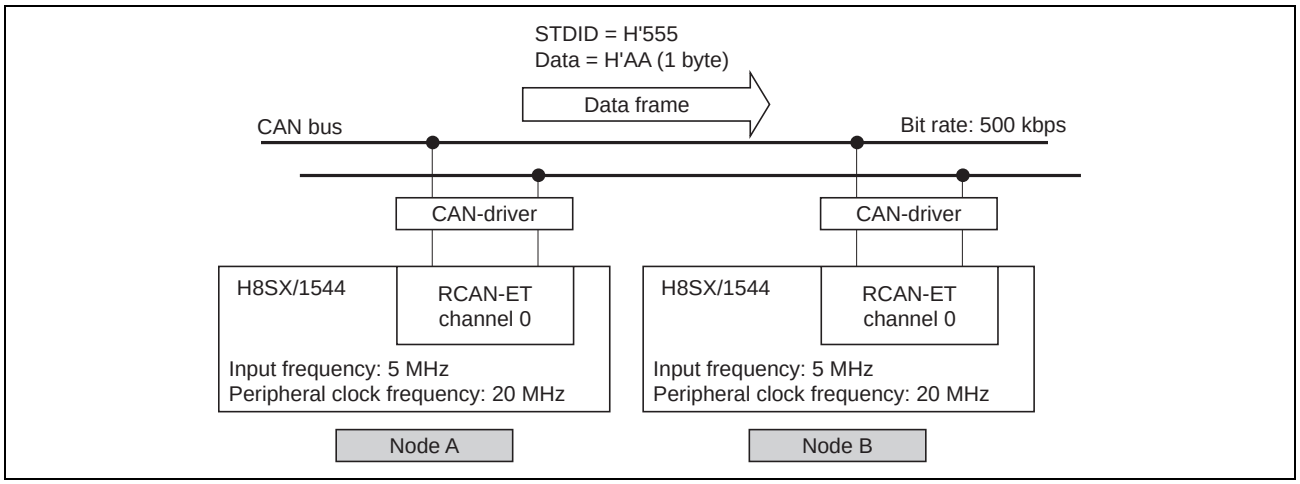


Figure 2.2.1 Communication Specification

2.2.2 Software Description

(1) Module Description

Table 2.2.1 Module Description

Module	Label	Function	Settings
H8SX/1544 initial settings	set_1544_init	See 2.1, Initial Settings.	Device-specific
RCAN initial settings	set_RCAN0_init		RCAN-ET common
RCAN mailbox initial settings	set_RCAN0MB_init		
RCAN start	set_RCAN0_start		
CAN message transmit	RCAN0_Tx	Transmits CAN message, clears transmit-end flag.	
CAN message receive	RCAN0_Rx	Receives CAN message, clears receive-end flag	
Receive buffer initialization	Clear_MBbuff	Initializes on-chip RAM for storing CAN message.	—

(2) Description of Registers Used

(a) Transmitting Side

Table 2.2.2 Description of Registers Used (Transmitting Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.MB[1].CTRL0.WORD.H	0x1554	Sets mailbox 1 to standard format, data frame. Also sets ID (H'555).
	RCANET0.MB[1].CTRL1.BYTE.H	0x00	Sets mailbox 1 to transmit.
	RCANET0.MB[1].CTRL1.BYTE.L	0x01	Sets the data length (1 byte).
	RCANET0.MB[1].MSG_DATA[0]	0xAA	Sets the transmit data.
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
CAN message transmit	RCANET0.TXPR0.LONG	0x00000002	Sets mailbox 1 to transmit-wait status.
	RCANET0.TXACK0.WORD	—	Mailbox 1 waits for transmit-end. (Flag is polled until bit 1 in TXACK0 is set to 1.)
	RCANET0.TXACK0.WORD	0x0002	Clears mailbox 1 transmit-end flag.

Note: The bit names listed in the table correspond to the bit names used in the hardware manual.

(b) Receiving Side

Table 2.2.3 Description of Registers Used (Receiving Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.MB[0].CTRL0.WORD.H	0x1554	Sets mailbox 0 to standard format, data frame. Also sets ID (H'555).
	RCANET0.MB[0].LAFM.WORD.H	0x0000	Sets filter mask for standard ID and IDE bit of mailbox 0.
	RCANET0.MB[0].CTRL1.BYTE.H	0x02	Sets mailbox 0 to receive.
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
CAN message transmit	RCANET0.RXPR0.WORD	—	Mailbox 0 waits for transmit-end. (Flag is polled until bit 0 in RXPR0 is set to 1.)
	RCANET0.RXPR0.WORD	0x0001	Clears mailbox 0 receive-end flag. (Clearing condition: write 1)

Note: The bit names listed in the table correspond to the bit names used in the hardware manual.

(3) Description of RAM Used

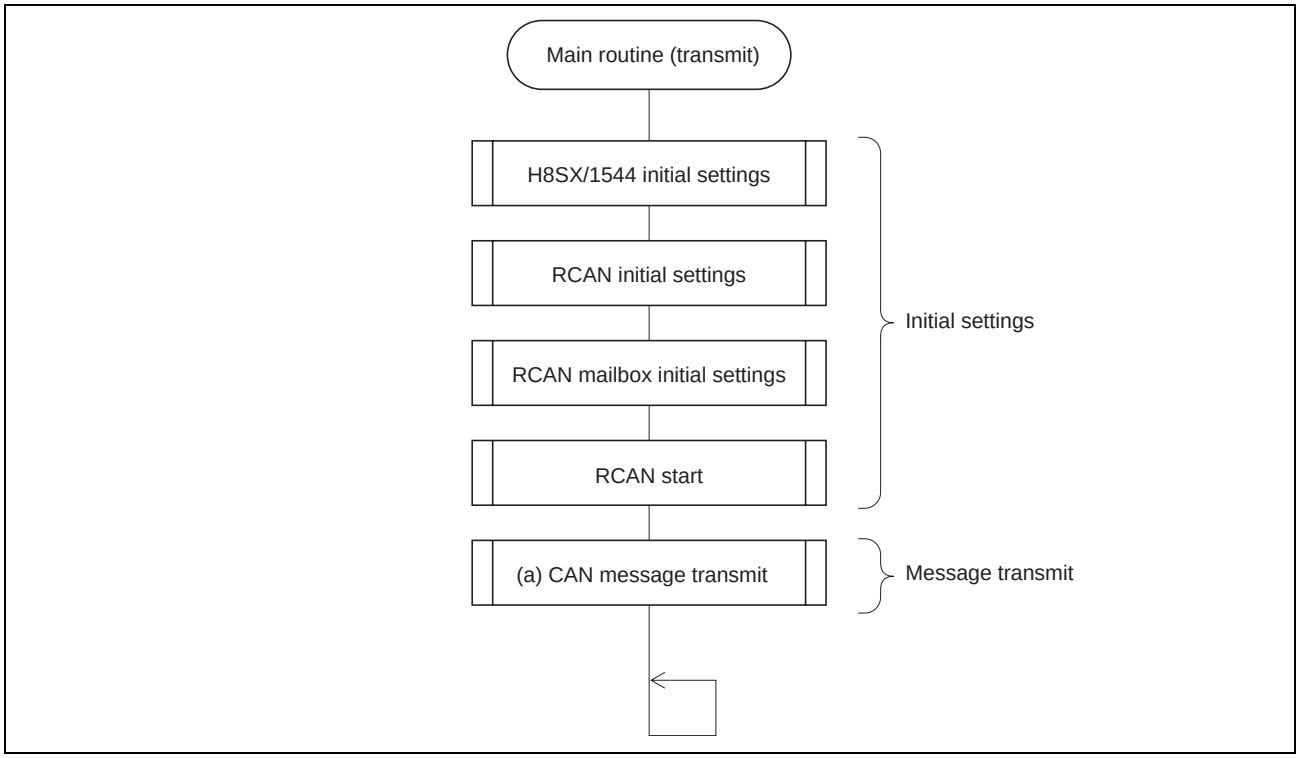
Table 2.2.4 Description of RAM Used

Module	Label	Function
Receive buffer initialization	MBbuff.ID.LONG MBbuff.ID.WORD.H MBbuff.ID.WORD.L	Stores receive ID.
CAN message transmit	MBbuff.DATA.LONG[0] to [1] MBbuff.DATA.WORD[0] to [3] MBbuff.DATA.BYTE[0] to [7]	Stores receive data.

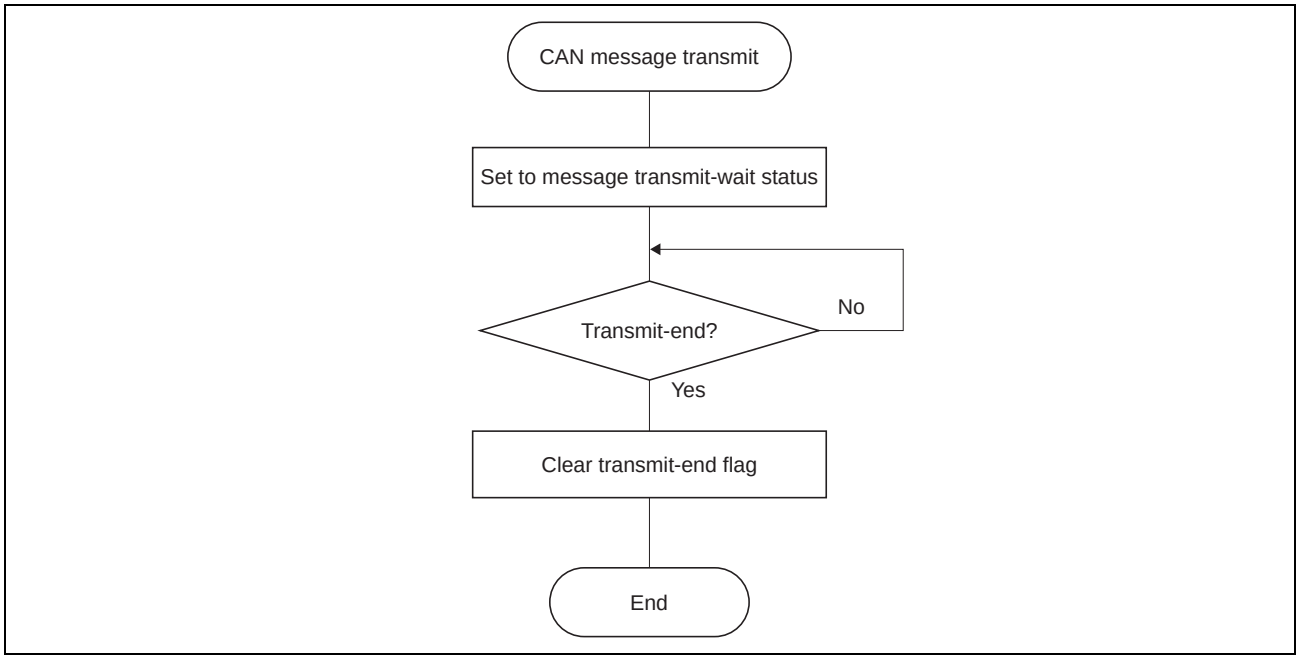
2.2.3 Flowcharts

(1) Transmitting Side Flowcharts

Main Routine

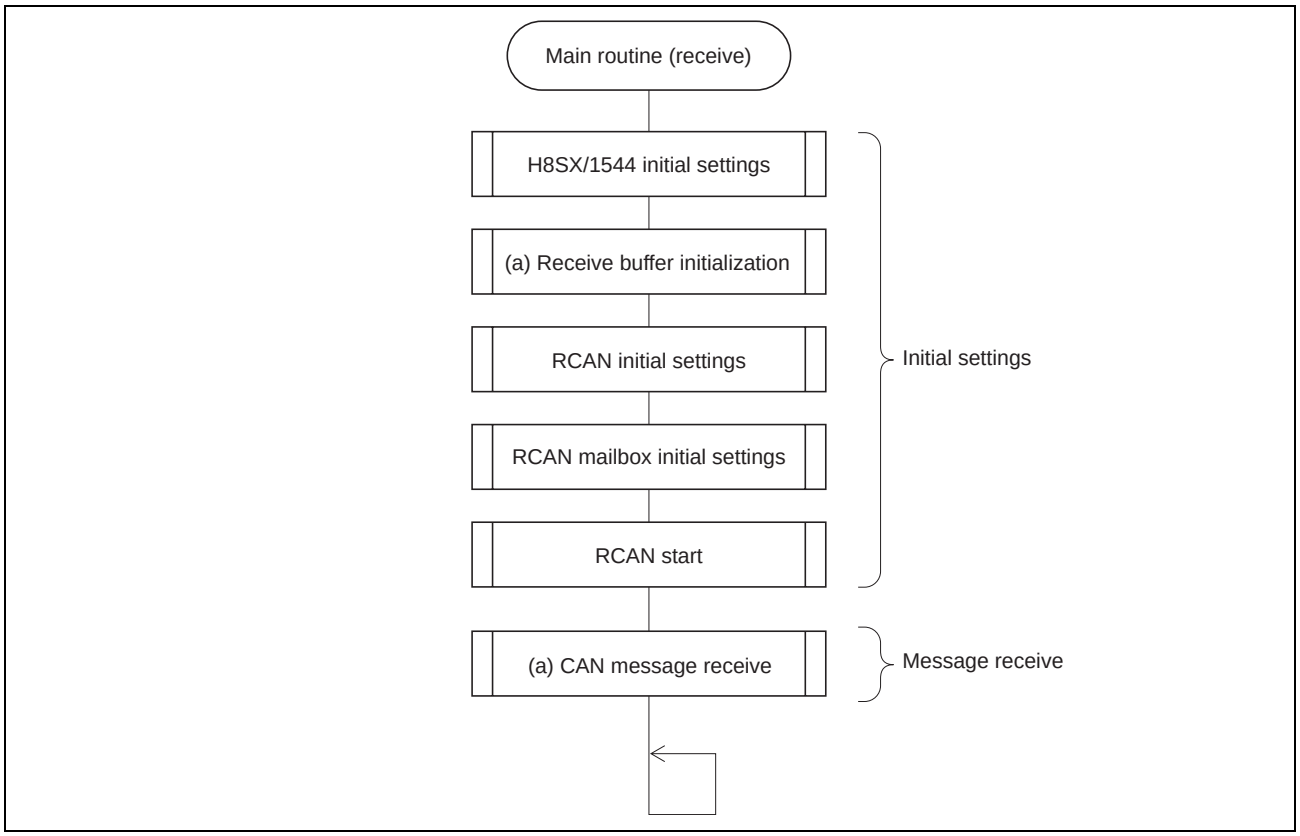


(a) CAN Message Transmit Routine (RCAN-ET Common Settings)

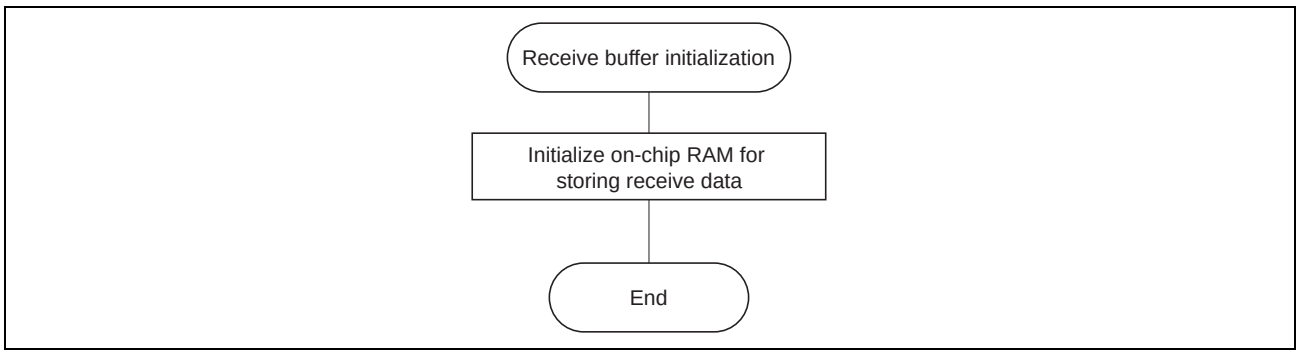


(2) Receiving Side Flowcharts

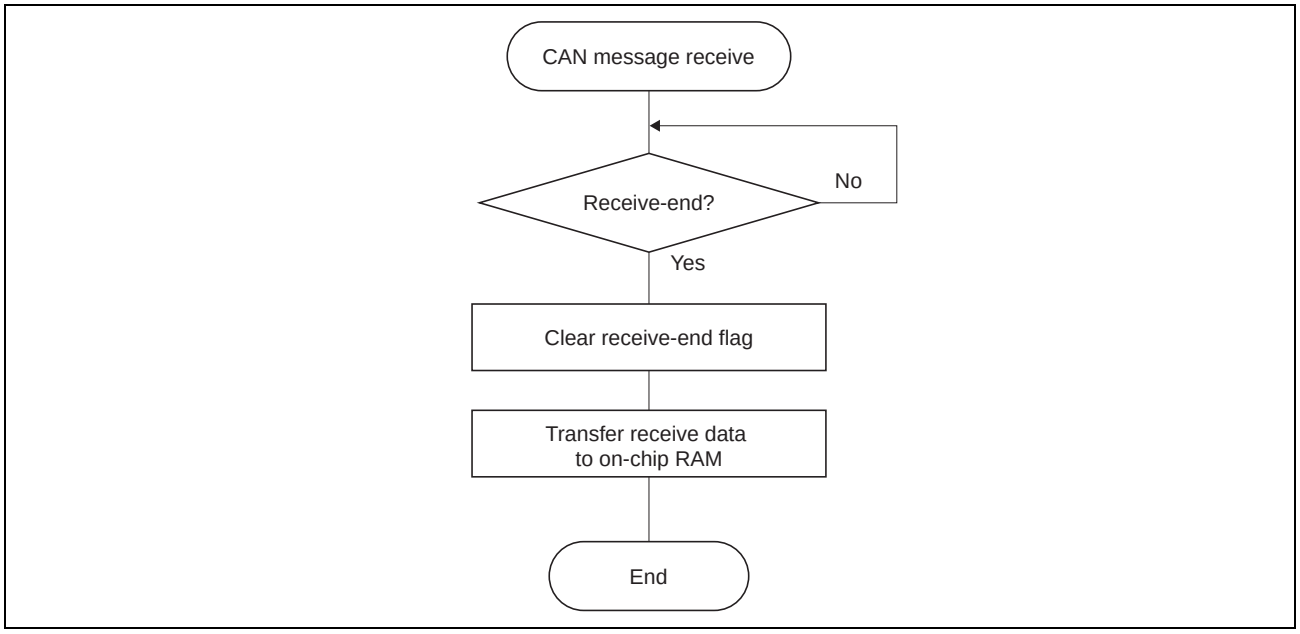
Main Routine



(a) Receive Buffer Initialization Routine (Device-Specific Settings)



(b) CAN Message Receive Routine (RCAN-ET Common Settings)



2.2.4 Program Listing

(1) Transmitting Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void RCAN0_Tx(void);
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{

    set_1544_init();                        /*  H8SX/1544 initialization    */
    set_RCAN0_init();                       /*  RCAN initialization         */
    set_RCAN0MB_init();
    set_RCAN0_start();
    RCAN0_Tx();                             /*  CAN message transmit        */
    while(1);

}

/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){
    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)      */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz)  */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz)*/
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode       */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin)*/


}
/*****************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename    :  rcan.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void RCAN0_Tx(void);
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*  Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)    */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)  */
    RCANET0.IRR.WORD = 0x0001;           /*  Clear IRR0 (clearing condition: write 1)    */
    RCANET0.MCR.WORD |= 0x8000;          /*  Reorder ID: Set to MCR15 = 1 (initial setting)  */

    for(i = 0;i < 16;i++){               /*  Initialize mailbox (RAM area)               */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }

}
/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    RCANET0.MB[1].CTRL0.WORD.H = 0x1554;    /*  ID: H'555, standard format, data frame  */
    RCANET0.MB[1].CTRL1.BYTE.H = 0x00;      /*  Set mailbox 1 to transmit               */
    RCANET0.MB[1].CTRL1.BYTE.L = 0x01;      /*  Data length: 1 byte                     */
    RCANET0.MB[1].MSG_DATA[0] = 0xAA;       /*  Transmit data: 0xAA                     */

    /*  Bit rate = 500 kbps  */
    RCANET0.BCR1.WORD = 0x4300; /*  TSEG1 = 4 (5tq), TSEG2 = 3 (4tq), SJW = 0, BSP = 0, (pϕ = 20 MHz)*/
    RCANET0.BCR0.WORD = 0x0001;             /*  BRP = 1                                 */

}
```

```
/*****************************************************************************/
/*      RCAN start routine                                                   */
/*****************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                     /*   Clear MCR0          */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000); /*   GSR3 = 0?           */


}



/*****************************************************************************/
/*      RCAN send message routine                                            */
/*****************************************************************************/
void RCAN0_Tx(void){

    RCANET0.TXPR0.LONG = 0x00000002;          /*   Set MB1 to transmit-wait status   */
    while ((RCANET0.TXACK0.WORD & 0x0002) != 0x0002);/*   Wait for transmit-end     */
    RCANET0.TXACK0.WORD = RCANET0.TXACK0.WORD;
                                        /*   Clear transmit-end flag (clearing condition: write 1)   */


}
/*****************************************************************************/
```

(2) Receiving Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose      : for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void Clear_MBbuff(void);
extern void RCAN0_Rx(void);
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{
    set_1544_init();                        /*  H8SX/1544 initialization                */
    Clear_MBbuff();                         /*  Initialize RAM area for storing receive data */
    set_RCAN0_init();                       /*  RCAN initialization                     */
    set_RCAN0MB_init();
    set_RCAN0_start();
    RCAN0_Rx();                             /*  CAN message receive                     */
    while(1);

}

/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)   */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz) */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz) */
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode                   */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN           */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin)            */

}
/*****************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename    :  rcan.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void RCAN0_Rx(void);
void Clear_MBbuff(void);
/*---------------------------------------------------------------------------*/
struct {
    union {
        unsigned long LONG;
        struct{
            unsigned short H;
            unsigned short L;
        }WORD;
    }ID;
    union {
        unsigned char BYTE[8];
        unsigned short WORD[4];
        unsigned long LONG[2];
    }DATA;
} MBbuff;
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*   Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)    */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)  */
    RCANET0.IRR.WORD = 0x0001;           /*   Clear IRR0 (clearing condition: write 1)           */
    RCANET0.MCR.WORD |= 0x8000;          /*   Reorder ID: Set to MCR15 = 1 (initial setting)     */

    for(i = 0;i < 16;i++){               /*   Initialize mailbox (RAM area)                      */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }

}
```

```
/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    RCANET0.MB[0].CTRL0.WORD.H = 0x1554;    /*  ID: H'555, standard format, data frame      */
    RCANET0.MB[0].LAFM.WORD.H = 0x0000;     /*  STD_LAFM and IDE_LAFM settings              */
    RCANET0.MB[0].CTRL1.BYTE.H = 0x02;      /*  Set mailbox 0 to receive                    */

    /*  Bit rate = 500 kbps  */
    RCANET0.BCR1.WORD = 0x4300;/*   TSEG1 = 4 (5tq), TSEG2 = 3 (4tq), SJW = 0, BSP = 0, (pϕ = 20 MHz)*/
    RCANET0.BCR0.WORD = 0x0001;             /*  BRP = 1                                     */


}

/*****************************************************************************/
/*      RCAN start routine                                                   */
/*****************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                     /*  Clear MCR0                  */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000); /*  GSR3 = 0?                   */


}


/*****************************************************************************/
/*      RCAN receive message routine                                         */
/*****************************************************************************/
void RCAN0_Rx(void){

    while ((RCANET0.RXPR0.WORD & 0x0001) != 0x0001);/*  Wait for receive-end        */
    RCANET0.RXPR0.WORD = RCANET0.RXPR0.WORD;
                                        /*  Clear receive-end flag (clearing condition: write 1)  */
    MBbuff.DATA.BYTE[0] = RCANET0.MB[0].MSG_DATA[0];/*  Store receive data in RAM   */


}


/*****************************************************************************/
/*      RAM area Initialize routine                                          */
/*****************************************************************************/
void Clear_MBbuff(void){

    MBbuff.ID.LONG = 0;                         /*  Initialize RAM area for storing receive data*/
    MBbuff.DATA.LONG[0] = 0;
    MBbuff.DATA.LONG[1] = 0;

}
/*****************************************************************************/
```

2.3 Data Frame Transmit and Receive (Extended ID)

2.3.1 Specification

A data frame with standard ID H'555 and extended ID H'2AAAA is transmitted by node A and received by node B (interrupts used), as shown in figure 2.3.1.

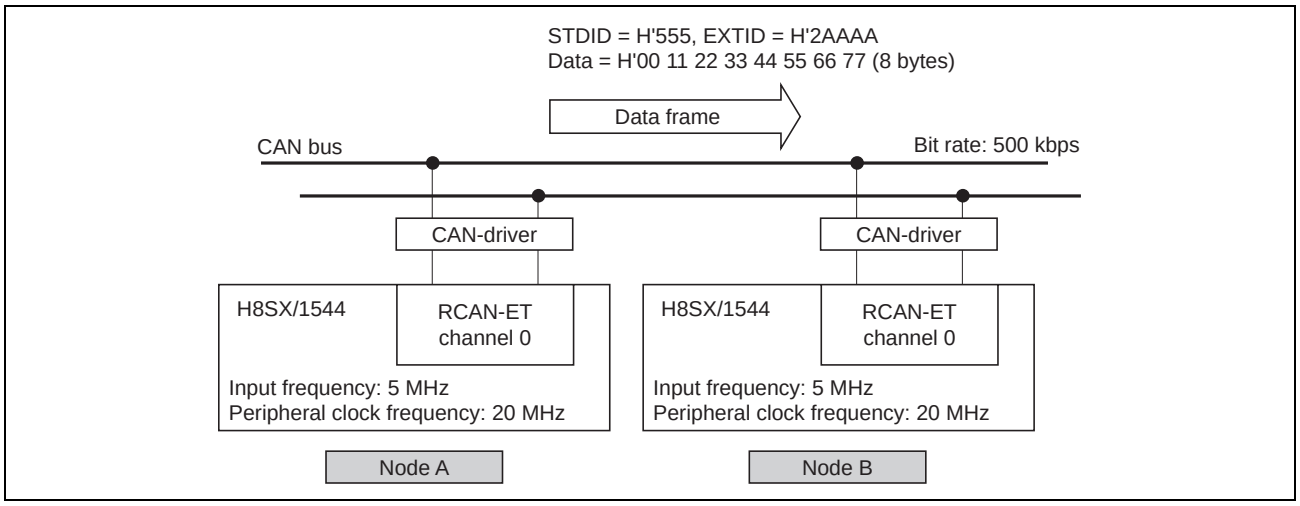


Figure 2.3.1 Communication Specification

2.3.2 Software Description

(1) Module Description

Table 2.3.1 Module Description

Module	Label	Function	Settings
H8SX/1544 initial settings	set_1544_init	See 2.1, Initial Settings.	Device-specific
H8SX/1544 interrupt settings	set_1544_INTC_init		
RCAN initial settings	set_RCAN0_init		RCAN-ET common
RCAN mailbox initial settings	set_RCAN0MB_init		
RCAN start	set_RCAN0_start		
CAN message transmit	RCAN0_Tx	Transmits CAN message.	
Mailbox empty interrupt	SLE0_0	Clears CAN message transmit-end flag.	
Data frame receive interrupt	RM0_0	Clears CAN message receive-end flag.	
Receive buffer initialization	Clear_MBbuff	Initializes on-chip RAM for storing CAN message.	—

(2) Description of Registers Used

(a) Transmitting Side

Table 2.3.2 Description of Registers Used (Transmitting Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFEFF	Enables mailbox empty interrupt (IMR8 = 0).
	RCANET0.MBIMR0.WORD	0xFFFD	Enables interrupts for mailbox 1.
	RCANET0.MB[1].CTRL0.WORD.H	0x9556	Sets mailbox 1 to extended format, data frame. Also sets standard ID (H'555) and extended ID (H'2AAAA).
	RCANET0.MB[1].CTRL0.WORD.L	0xAAAA	
	RCANET0.MB[1].CTRL1.BYTE.H	0x00	Sets mailbox 1 to transmit.
	RCANET0.MB[1].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[1].MSG_DATA[0]	0x00	Sets the transmit data.
	RCANET0.MB[1].MSG_DATA[1]	0x11	
	RCANET0.MB[1].MSG_DATA[2]	0x22	
	RCANET0.MB[1].MSG_DATA[3]	0x33	
	RCANET0.MB[1].MSG_DATA[4]	0x44	
	RCANET0.MB[1].MSG_DATA[5]	0x55	
	RCANET0.MB[1].MSG_DATA[6]	0x66	
	RCANET0.MB[1].MSG_DATA[7]	0x77	
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
H8SX/1544 interrupt settings			
CAN message transmit	RCANET0.TXPR0.LONG	0x00000002	Mailbox 1 to transmit-wait status.
Mailbox empty interrupt	RCANET0.TXACK0.WORD	0x0002	Clears mailbox 1 transmit-end flag. (Clearing condition: write 1)

Note: The bit names listed in the table correspond to the bit names used in the hardware manual.

(b) Receiving Side

Table 2.3.3 Description of Registers Used (Receiving Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFFFD	Enables data frame receive interrupt (IRR1) (IMR1 = 0).
	RCANET0.MBIMR0.WORD	0xFFFE	Enables interrupts for mailbox 0.
	RCANET0.MB[0].CTRL0.WORD.H	0x9556	Sets mailbox 0 to extended format, data frame. Also sets standard ID (H'555) and extended ID (H'2AAAA).
	RCANET0.MB[0].CTRL0.WORD.L	0xAAAA	
	RCANET0.MB[0].LAFM.WORD.H	0x0000	Sets filter mask for standard ID, extended ID, and IDE bit of mailbox 0.
	RCANET0.MB[0].LAFM.WORD.L	0x0000	
	RCANET0.MB[0].CTRL1.BYTE.H	0x02	Sets mailbox 0 to receive.
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
H8SX/1544 interrupt settings			
Data frame receive interrupt	RCANET0.RXPR0.WORD	0x0001	Clears mailbox 0 receive-end flag. (Clearing condition: write 1)

Note: The bit names listed in the table correspond to the bit names used in the hardware manual.

(3) Description of RAM Used

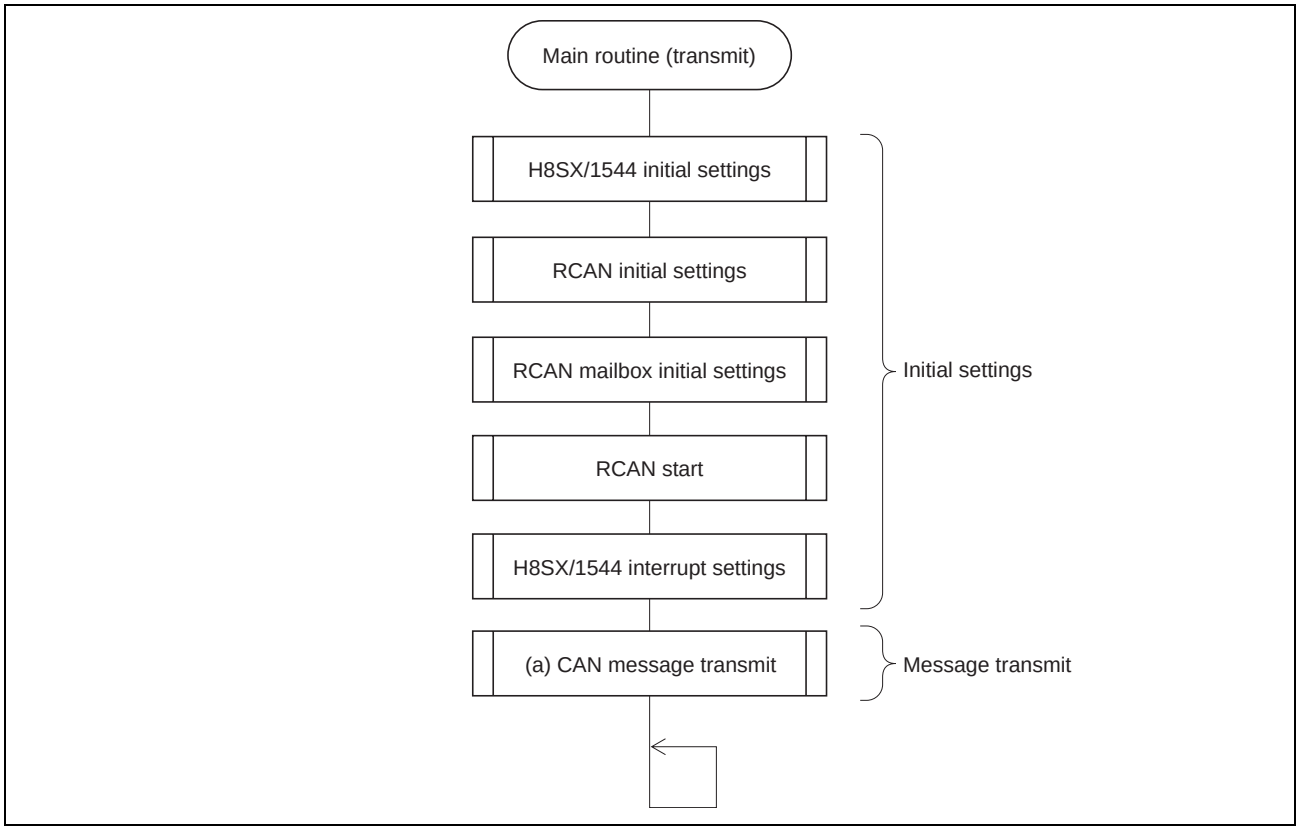
Table 2.3.4 Description of RAM Used

Module	Label	Function
Receive buffer initialization	MBbuff.ID.LONG MBbuff.ID.WORD.H MBbuff.ID.WORD.L	Stores receive ID.
Data frame receive interrupt	MBbuff.DATA.LONG[0] to [1] MBbuff.DATA.WORD[0] to [3] MBbuff.DATA.BYTE[0] to [7]	Stores receive data.

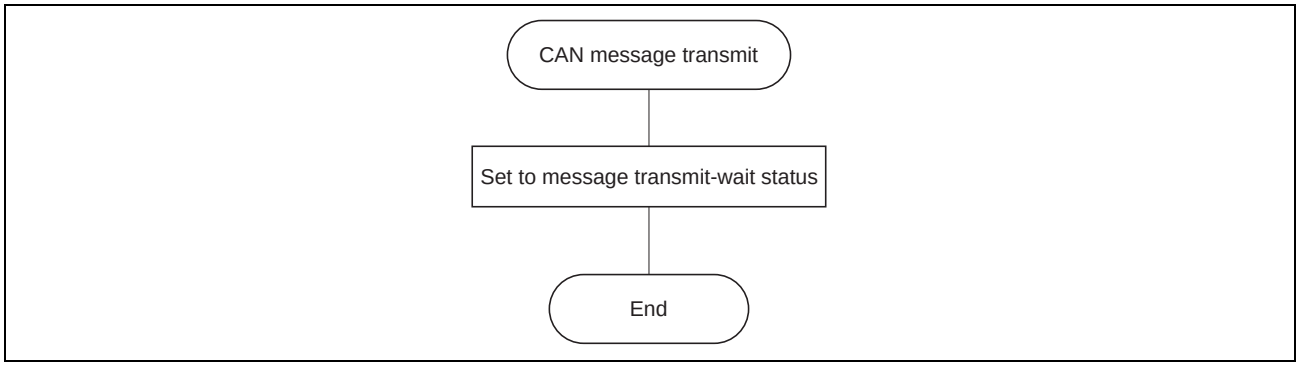
2.3.3 Flowcharts

(1) Transmitting Side Flowcharts

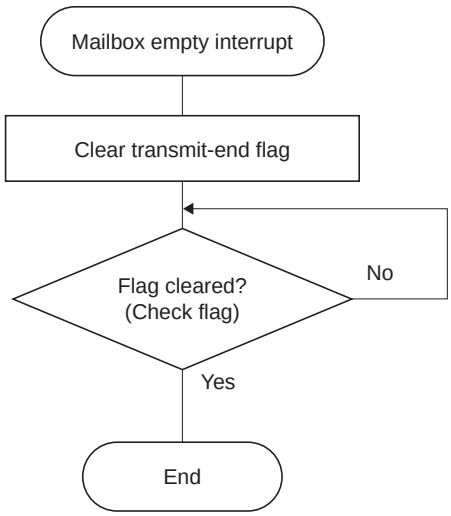
Main Routine



(a) CAN Message Transmit Routine (RCAN-ET Common Settings)



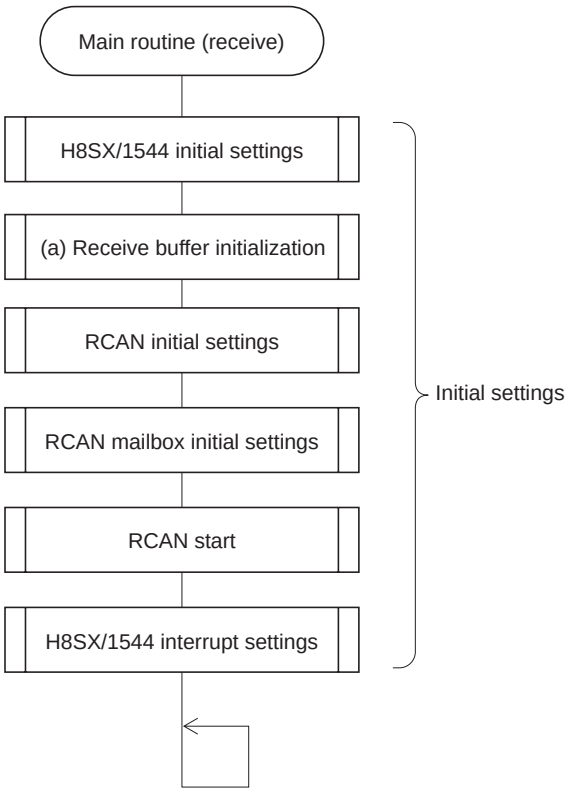
(b) Mailbox Empty Interrupt Routine (RCAN-ET Common Settings)



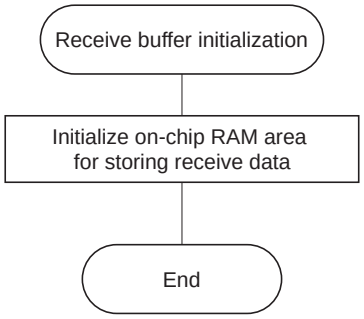
Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

(2) Receiving Side Flowcharts

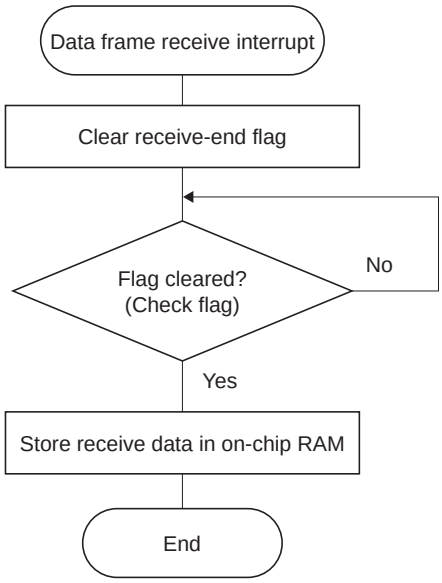
Main Routine



(a) Receive Buffer Initialization Routine (Device-Specific Settings)



(b) Data Frame Receive Interrupt Routine (RCAN-ET Common Settings)



Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

2.3.4 Program Listing

(1) Transmitting Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void RCAN0_Tx(void);
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{

    set_1544_init();                        /*  H8SX/1544 initialization     */
    set_RCAN0_init();                       /*  RCAN initialization          */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();                   /*  H8SX/1544 interrupt settings */
    RCAN0_Tx();                             /*  CAN message receive          */
    while(1);

}

/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)       */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz)   */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz) */
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode        */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin) */

}
```

```
/*****************************************************************************/
/*      H8SX/1544 INTC Initialize routine                                    */
/*****************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;                /*   Interrupt control mode 2          */
    INTC.IPRQ.BIT._RCAN01 = 7;              /*   Set interrupt priority level: RCAN */
    set_imask_exr(0);                       /*   Interrupt mask level 0            */


}
/*****************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename    :  rcan.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void RCAN0_Tx(void);
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*  Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)    */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)  */
    RCANET0.IRR.WORD = 0x0001;           /*  Clear IRR0 (clearing condition: write 1)    */
    RCANET0.MCR.WORD |= 0x8000;          /*  Reorder ID: Set to MCR15 = 1 (initial setting)  */

    for(i = 0;i < 16;i++){               /*  Initialize mailbox (RAM area)               */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }


}

/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    RCANET0.IMR.WORD &= 0xFEFF;              /*  Enable mailbox empty interrupt          */
    RCANET0.MBIMR0.WORD &= 0xFFFD;           /*  Enable MB1 interrupt                    */
```

```
    RCANET0.MB[1].CTRL0.WORD.H = 0x9556;      /*   STDID: H'555, extended format, data frame  */
    RCANET0.MB[1].CTRL0.WORD.L = 0xAAAA;      /*   EXTID: H'2AAAA                             */
    RCANET0.MB[1].CTRL1.BYTE.H = 0x00;        /*   Set mailbox 1 to transmit                  */
    RCANET0.MB[1].CTRL1.BYTE.L = 0x08;        /*   Data length: 8 bytes                       */
    RCANET0.MB[1].MSG_DATA[0] = 0x00;         /*   Transmit data: 0x00                        */
    RCANET0.MB[1].MSG_DATA[1] = 0x11;         /*   Transmit data: 0x11                        */
    RCANET0.MB[1].MSG_DATA[2] = 0x22;         /*   Transmit data: 0x22                        */
    RCANET0.MB[1].MSG_DATA[3] = 0x33;         /*   Transmit data: 0x33                        */
    RCANET0.MB[1].MSG_DATA[4] = 0x44;         /*   Transmit data: 0x44                        */
    RCANET0.MB[1].MSG_DATA[5] = 0x55;         /*   Transmit data: 0x55                        */
    RCANET0.MB[1].MSG_DATA[6] = 0x66;         /*   Transmit data: 0x66                        */
    RCANET0.MB[1].MSG_DATA[7] = 0x77;         /*   Transmit data: 0x77                        */

    /*  Bit rate = 500 kbps  */
    RCANET0.BCR1.WORD = 0x4300;  /*  TSEG1 = 4 (5tq), TSEG2 = 3 (4tq), SJW = 0, BSP = 0, (pϕ = 20 MHz)*/
    RCANET0.BCR0.WORD = 0x0001;               /*   BRP = 1                                    */

}

/***********************************************************************************/
/*      RCAN start routine                                                         */
/***********************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                       /*   Clear MCR0                 */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000); /*   GSR3 = 0?                  */


}
/***********************************************************************************/
/*      RCAN send message routine                                                  */
/***********************************************************************************/
void RCAN0_Tx(void){

    RCANET0.TXPR0.LONG = 0x00000002;          /*   Set MB1 to transmit-wait status            */

}

/***********************************************************************************/
/*      Mailbox Empty Interrupt routine                                            */
/***********************************************************************************/
#pragma interrupt(SLE0_0)
void SLE0_0(void)
{

    RCANET0.TXACK0.WORD = 0x0002;          /*   Clear transmit-end flag (clearing condition: write 1) */
    while(RCANET0.TXACK0.WORD & 0x0002); /*   Check flag                                 */

}
/***********************************************************************************/
```

(2) Receiving Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void Clear_MBbuff(void);
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{

    set_1544_init();                        /*  H8SX/1544 initialization                    */
    Clear_MBbuff();                         /*  Initialize RAM area for storing receive data */
    set_RCAN0_init();                       /*  RCAN initialization                         */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();                   /*  H8SX/1544 interrupt settings                */
    while(1);

}
/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)       */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz)   */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz) */
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode                       */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN               */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins    */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin)                */

}
```

```
/*****************************************************************************/
/*      H8SX/1544 INTC Initialize routine                                    */
/*****************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;                /*   Interrupt control mode 2          */
    INTC.IPRQ.BIT._RCAN01 = 7;              /*   Set interrupt priority level: RCAN */
    set_imask_exr(0);                       /*   Interrupt mask level 0            */


}
/*****************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename    :  rcan.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void Clear_MBbuff(void);
/*---------------------------------------------------------------------------*/
struct {
    union {
        unsigned long LONG;
        struct{
            unsigned short H;
            unsigned short L;
        }WORD;
    }ID;
    union {
        unsigned char BYTE[8];
        unsigned short WORD[4];
        unsigned long LONG[2];
    }DATA;
} MBbuff;
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*   Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008); /*  GSR3 = 1? (RCAN-ET reset status)   */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001); /*  IRR0 = 1? (reset/halt/sleep interrupt)  */
    RCANET0.IRR.WORD = 0x0001;           /*   Clear IRR0 (clearing condition: write 1)            */
    RCANET0.MCR.WORD |= 0x8000;          /*   Reorder ID: Set to MCR15 = 1 (initial setting)      */

    for(i = 0;i < 16;i++){               /*   Initialize mailbox (RAM area)                       */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }


}
```

```
/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    RCANET0.IMR.WORD &= 0xFFFD;           /*   Enable data frame receive interrupt      */
    RCANET0.MBIMR0.WORD &= 0xFFFE;        /*   Enable MB0 interrupt                     */

    RCANET0.MB[0].CTRL0.WORD.H = 0x9556; /*   STDID: H'555, extended format, data frame */
    RCANET0.MB[0].CTRL0.WORD.L = 0xAAAA; /*   EXTID: H'2AAAA                           */
    RCANET0.MB[0].LAFM.WORD.H = 0x0000;  /*   STD_LAFM and IDE_LAFM settings           */
    RCANET0.MB[0].LAFM.WORD.L = 0x0000;  /*   EXTID_LAFM settings                      */
    RCANET0.MB[0].CTRL1.BYTE.H = 0x02;   /*   Set mailbox 0 to receive                 */

    /*  Bit rate = 500 kbps  */
    RCANET0.BCR1.WORD = 0x4300; /*  TSEG1 = 4 (5tq), TSEG2 = 3 (4tq), SJW = 0, BSP = 0, (pϕ = 20 MHz) */
    RCANET0.BCR0.WORD = 0x0001;           /*   BRP = 1                                  */
}
/*****************************************************************************/
/*      RCAN start routine                                                   */
/*****************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                     /*   Clear MCR0                */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000); /*   GSR3 = 0?                 */

}
/*****************************************************************************/
/*      RAM area Initialize routine                                          */
/*****************************************************************************/
void Clear_MBbuff(void){

    MBbuff.ID.LONG = 0;                          /*   Initialize RAM area for storing receive data*/
    MBbuff.DATA.LONG[0] = 0;
    MBbuff.DATA.LONG[1] = 0;

}
/*****************************************************************************/
/*      Data Frame Received Interrupt routine(MailBox0)                      */
/*****************************************************************************/
#pragma interrupt(RM0_0)
void RM0_0(void)
{
    unsigned int i;

    RCANET0.RXPR0.WORD = 0x0001;          /*   Clear receive-end flag (clearing condition: write 1) */
    while(RCANET0.RXPR0.WORD & 0x0001);   /*   Check flag                               */
    for(i = 0;i < 8;i++){
        MBbuff.DATA.BYTE[i] = RCANET0.MB[0].MSG_DATA[i];/*   Store receive data in RAM */
    }
}
/*****************************************************************************/
```

2.4 Remote Frame Transmit and Receive (Using ATX)

2.4.1 Specification

A remote frame with standard ID H'555 is transmitted by node A, as shown in figure 2.4.1. After node B receives the remote frame, a data frame is transmitted automatically using the automatic data frame transmit function (ATX).

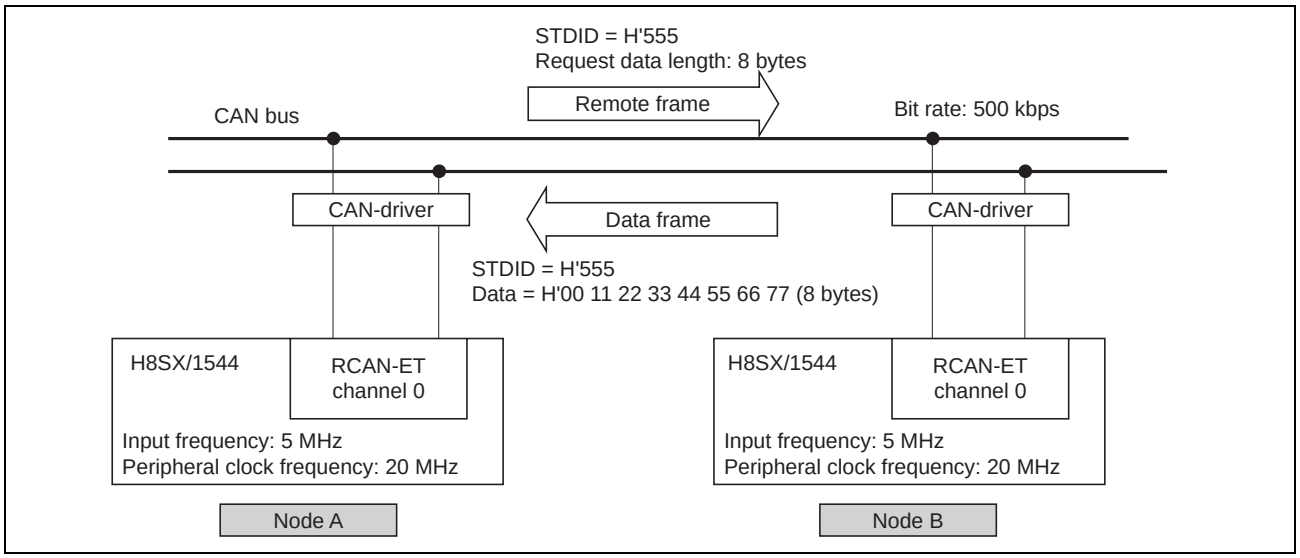


Figure 2.4.1 Communication Specification

2.4.2 Software Description

(1) Module Description

Table 2.4.1 Module Description

Module	Label	Function	Settings
H8SX/1544 initial settings	set_1544_init	See 2.1, Initial Settings.	Device-specific
H8SX/1544 interrupt settings	set_1544_INTC_init		
RCAN initial settings	set_RCAN0_init		RCAN-ET common
RCAN mailbox initial settings	set_RCAN0MB_init		
RCAN start	set_RCAN0_start		
CAN message transmit	RCAN0_Tx	Transmits a remote frame.	
Mailbox empty interrupt	SLE0_0	Clears remote frame transmit-end flag.	
Data frame receive interrupt	RM0_0	Clears mailbox 0 data frame receive-end flag. Stores receive data.	
Remote frame receive interrupt	RM1_0	Clears mailbox 1 remote frame receive-end flag, and clears data frame transmit-end flag.	
Receive buffer initialization	Clear_MBbuff	Initializes on-chip RAM area for storing CAN message.	—

Note: On the H8SX/1544, common interrupt vector addresses are used for all interrupt sources (RM1, SLE, OVR, and ERS) in the case of mailboxes other than mailbox 0. Therefore, in this operation example (which uses mailbox 1) the same interrupt function is used for the remote frame receive interrupt (IRR2) and the data frame transmit-end interrupt (IRR8).

(2) Receiving Side Program Listing

(a) Transmitting Side

Table 2.4.2 Description of Registers Used (Transmitting Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFEFD	Enables mailbox empty interrupt (IRR8) and data frame receive interrupt (IRR0) (IMR8 = 0, IMR1 = 0).
	RCANET0.MBIMR0.WORD	0xFFFC	Enables interrupts for mailboxes 0 and 1.
	RCANET0.MB[0].CTRL0.WORD.H	0x1554	Sets mailbox 0 to standard format, data frame (RTR = 0). Also sets standard ID (H'555).
	RCANET0.MB[0].LAFM.WORD.H	0x0000	Sets filter mask for standard ID and IDE bit of mailbox 0.
	RCANET0.MB[0].CTRL1.BYTE.H	0x02	Sets mailbox 0 to data frame receive.
	RCANET0.MB[1].CTRL0.WORD.H	0x5554	Sets mailbox 1 to standard format, remote frame (RTR = 1). Also sets standard ID (H'555).
	RCANET0.MB[1].CTRL1.BYTE.H	0x00	Sets mailbox 1 to remote frame transmit.
	RCANET0.MB[1].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1).
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
H8SX/1544 interrupt settings			
CAN message transmit	RCANET0.TXPR0.LONG	0x00000002	Sets mailbox 1 to transmit-wait status.
Mailbox empty interrupt	RCANET0.TXACK0.WORD	0x0002	Clears mailbox 1 transmit-end flag. (Clearing condition: write 1)
Receive buffer initialization	RCANET0.RXPR0.WORD	0x0001	Clears mailbox 0 receive-end flag. (Clearing condition: write 1)

Note: The bit names listed in the table correspond to the bit names used in the hardware manual.

(b) Receiving Side

Table 2.4.3 Description of Registers Used (Receiving Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFEFB	Enables mailbox empty interrupt (IRR8) and remote frame receive interrupt (IRR2) (IMR8 = 0, IMR2 = 0).
	RCANET0.MBIMR0.WORD	0xFFFD	Enables mailbox 1 interrupt.
	RCANET0.MB[1].CTRL0.WORD.H	0x1554	Sets mailbox 0 to standard format, data frame (RTR = 0). Also sets standard ID (H'555). Note: When using the automatic data frame transmit function (ATX), RTR is not also set after remote frame receive
	RCANET0.MB[1].LAFM.WORD.H	0x0000	Sets filter mask for standard ID and IDE bit of mailbox 1.
	RCANET0.MB[1].CTRL1.BYTE.H	0x11	Enables mailbox 1 for remote frame receive and data frame transmit. Also sets automatic data frame transmit function (ATX) (ATX = 1, MBC = b'001).
	RCANET0.MB[1].MSG_DATA[0]	0x00	Sets the transmit data.
	RCANET0.MB[1].MSG_DATA[1]	0x11	
	RCANET0.MB[1].MSG_DATA[2]	0x22	
	RCANET0.MB[1].MSG_DATA[3]	0x33	
	RCANET0.MB[1].MSG_DATA[4]	0x44	
	RCANET0.MB[1].MSG_DATA[5]	0x55	
	RCANET0.MB[1].MSG_DATA[6]	0x66	
	RCANET0.MB[1].MSG_DATA[7]	0x77	
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1).
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
H8SX/1544 interrupt settings			
Remote frame receive interrupt	RCANET0.RFPR0.WORD	0x0002	Clears mailbox 1 remote frame receive-end flag. (Clearing condition: write 1)
	RCANET0.TXACK0.WORD	0x0002	Clears mailbox 1 transmit-end flag. (Clearing condition: write 1)

Note: The bit names listed in the table correspond to the bit names used in the hardware manual.

(3) Description of RAM Used

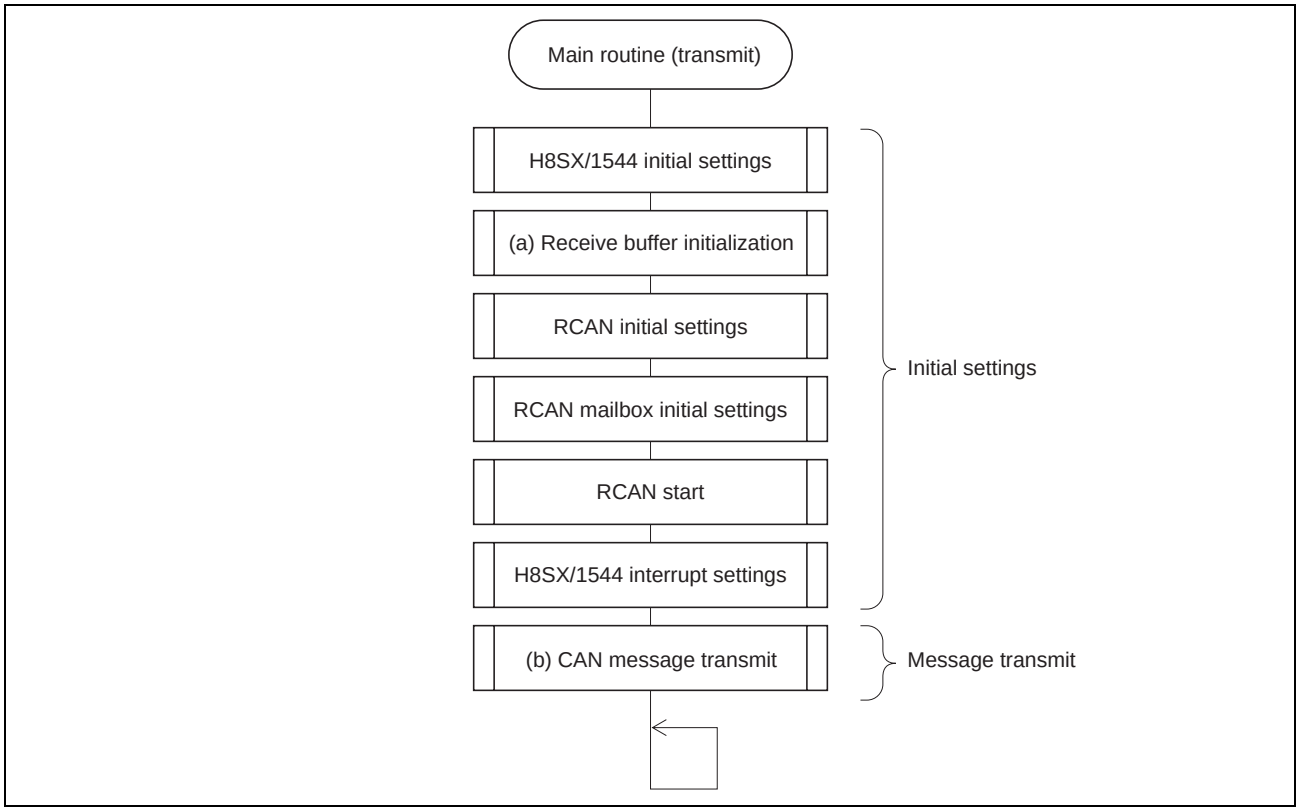
Table 2.4.4 Description of RAM Used

Module	Label	Function
Receive buffer initialization	MBbuff.ID.LONG MBbuff.ID.WORD.H MBbuff.ID.WORD.L	Stores receive ID.
Data frame receive interrupt	MBbuff.DATA.LONG[0] to [1] MBbuff.DATA.WORD[0] to [3] MBbuff.DATA.BYTE[0] to [7]	Stores receive data.

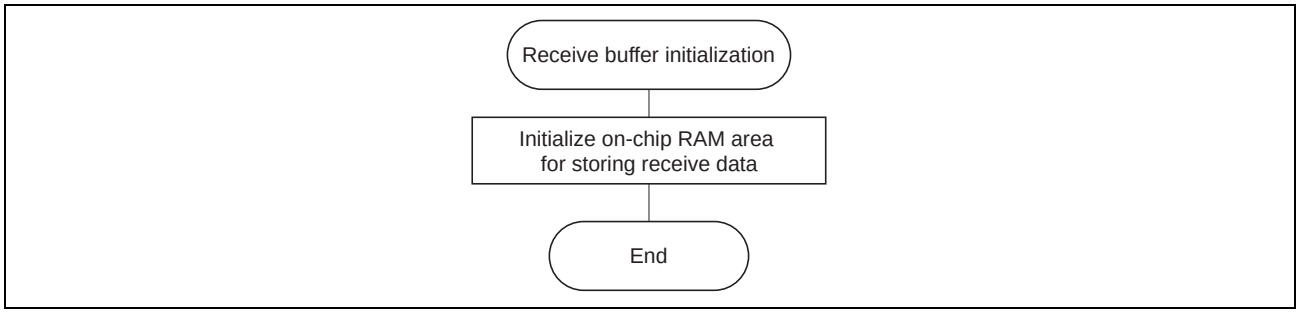
2.4.3 Flowcharts

(1) Transmitting Side Flowcharts

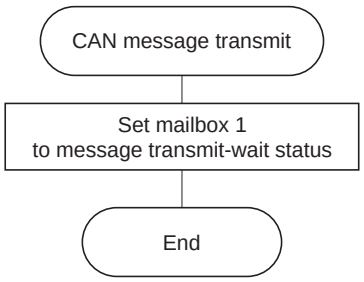
Main Routine



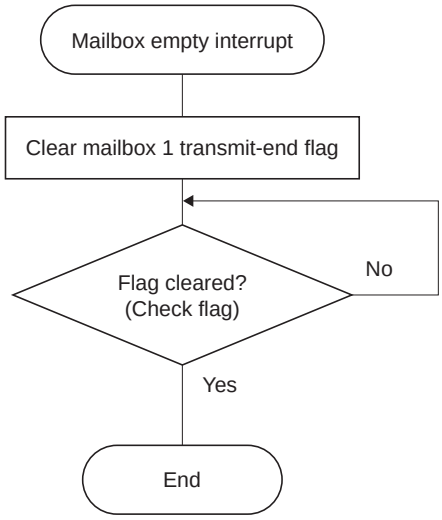
(a) Receive Buffer Initialization Routine (Device-Specific Settings)



(b) CAN Message Transmit Routine (RCAN-ET Common Settings)

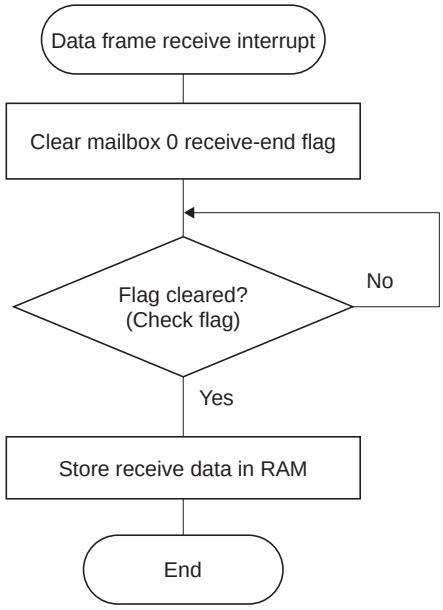


(c) Mailbox Empty Interrupt Routine (RCAN-ET Common Settings)



Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

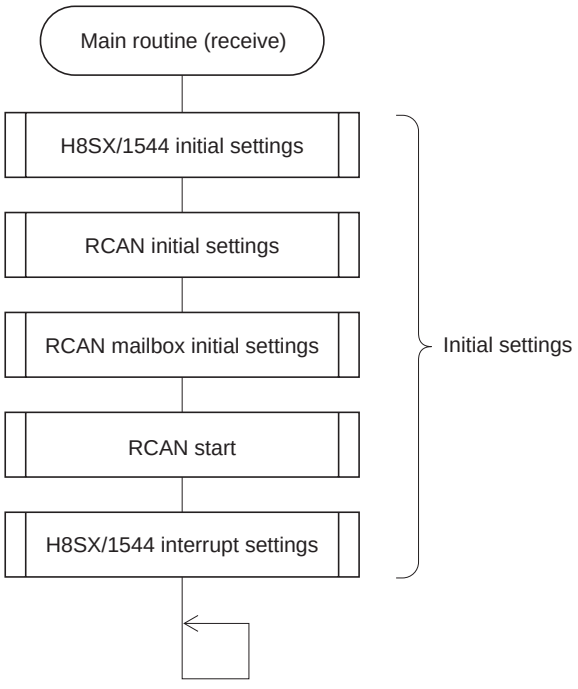
(d) Data Frame Receive Interrupt Routine (RCAN-ET Common Settings)



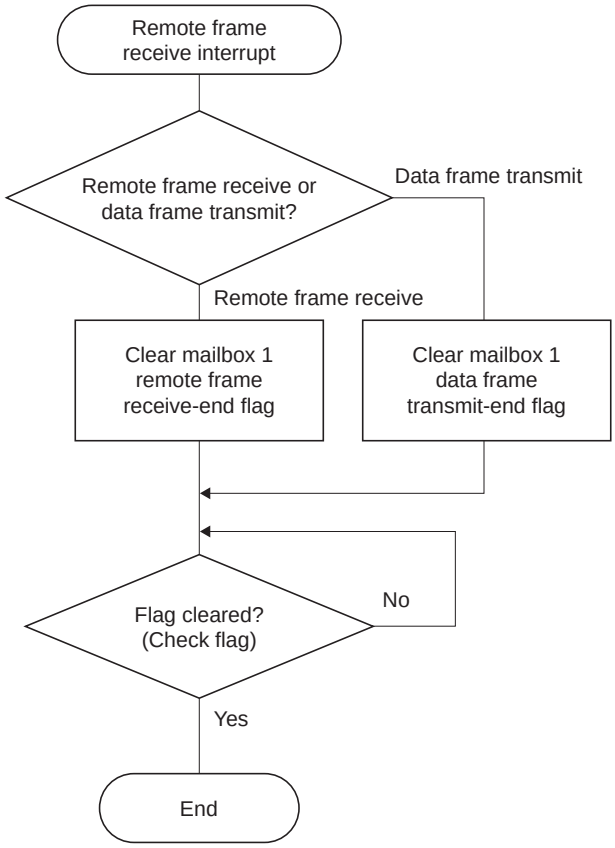
Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

(2) Receiving Side Flowcharts

Main Routine



(a) Remote Frame Receive Interrupt Routine (RCAN-ET Common Settings)



Notes: 1. On the H8SX/1544, common interrupt vector addresses are used for all interrupt sources (RM1, SLE, OVR, and ERS) in the case of mailboxes other than mailbox 0. Therefore, in this operation example (which uses mailbox 1) the same interrupt function is used for the remote frame receive interrupt (IRR2) and the data frame transmit-end interrupt (IRR8).

2. To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

2.4.4 Program Listing

(1) Transmitting Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void RCAN0_Tx(void);
extern void Clear_MBbuff(void);
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{
    set_1544_init();                        /*  H8SX/1544 initialization                    */
    Clear_MBbuff();                         /*  Initialize RAM area for storing receive data */
    set_RCAN0_init();                       /*  RCAN initialization                         */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();                   /*  H8SX/1544 interrupt settings                */
    RCAN0_Tx();                             /*  CAN message transmit                        */
    while(1);

}
/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)       */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz)   */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz) */
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode                       */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN               */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins    */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin)                */

}
```

```
/*****************************************************************************/
/*      H8SX/1544 INTC Initialize routine                                    */
/*****************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;                /*   Interrupt control mode 2          */
    INTC.IPRQ.BIT._RCAN01 = 7;              /*   Set interrupt priority level: RCAN */
    set_imask_exr(0);                       /*   Interrupt mask level 0            */


}
/*****************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename    :  rcan.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void RCAN0_Tx(void);
void Clear_MBbuff(void);
/*---------------------------------------------------------------------------*/
struct {
    union {
        unsigned long LONG;
        struct{
            unsigned short H;
            unsigned short L;
        }WORD;
    }ID;
    union {
        unsigned char BYTE[8];
        unsigned short WORD[4];
        unsigned long LONG[2];
    }DATA;
} MBbuff;
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*   Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)    */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)  */
    RCANET0.IRR.WORD = 0x0001;           /*   Clear IRR0 (clearing condition: write 1)           */
    RCANET0.MCR.WORD |= 0x8000;          /*   Reorder ID: Set to MCR15 = 1 (initial setting)     */

    for(i = 0;i < 16;i++){               /*   Initialize mailbox (RAM area)                      */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }
}
```

```
/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    RCANET0.IMR.WORD &= 0xFEFD;        /*   Enable mailbox empty and data frame receive interrupts  */
    RCANET0.MBIMR0.WORD &= 0xFFFC;     /*   Enable interrupts for MB0 and MB1                       */

    RCANET0.MB[0].CTRL0.WORD.H = 0x1554;  /*   STDID: H'555, standard format, data frame (RTR = 0)*/
    RCANET0.MB[0].LAFM.WORD.H = 0x0000; /*   STD_LAFM and IDE_LAFM settings                        */
    RCANET0.MB[0].CTRL1.BYTE.H = 0x02;  /*   Set mailbox 0 to data frame receive                   */

    RCANET0.MB[1].CTRL0.WORD.H = 0x5554;/*  STDID: H'555, standard format, remote frame (RTR = 1) */
    RCANET0.MB[1].CTRL1.BYTE.H = 0x00;  /*   Set mailbox 1 to remote frame transmit                */
    RCANET0.MB[1].CTRL1.BYTE.L = 0x08;  /*   Data length: 8 bytes                                  */

    /*  Bit rate = 500 kbps  */
    RCANET0.BCR1.WORD = 0x4300; /*  TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, (pϕ = 20 MHz)*/
    RCANET0.BCR0.WORD = 0x0001;         /*   BRP = 1                                               */

}

/*****************************************************************************/
/*      RCAN start routine                                                   */
/*****************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                          /*   Clear MCR0            */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000);       /*   GSR3 = 0?             */

}

/*****************************************************************************/
/*      RCAN send message routine                                            */
/*****************************************************************************/
void RCAN0_Tx(void){

    RCANET0.TXPR0.LONG = 0x00000002;           /*   Set MB1 to transmit-wait status          */

}

/*****************************************************************************/
/*      RAM area Initialize routine                                          */
/*****************************************************************************/
void Clear_MBbuff(void){

    MBbuff.ID.LONG = 0;                        /*   Initialize RAM area for storing receive data*/
    MBbuff.DATA.LONG[0] = 0;
    MBbuff.DATA.LONG[1] = 0;

}
```

```
/*****************************************************************************/
/*      Mailbox Empty Interrupt routine                                      */
/*****************************************************************************/
#pragma interrupt(SLE0_0)
void SLE0_0(void)
{

    RCANET0.TXACK0.WORD = 0x0002;         /*   Clear transmit-end flag (clearing condition: write 1)*/
    while(RCANET0.TXACK0.WORD & 0x0002); /*   Check flag                                          */


}


/*****************************************************************************/
/*      Data Frame Received Interrupt routine(MailBox0)                      */
/*****************************************************************************/
#pragma interrupt(RM0_0)
void RM0_0(void)
{
    unsigned int i;

    RCANET0.RXPR0.WORD = 0x0001;          /*   Clear receive-end flag (clearing condition: write 1) */
    while(RCANET0.RXPR0.WORD & 0x0001);   /*   Check flag                                          */

    for(i = 0;i < 8;i++){
        MBbuff.DATA.BYTE[i] = RCANET0.MB[0].MSG_DATA[i];/* Store receive data in RAM  */
    }


}
/*****************************************************************************/
```

(2) Receiving Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{

    set_1544_init();                        /*  H8SX/1544 initialization        */
    set_RCAN0_init();                       /*  RCAN initialization             */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();                   /*  H8SX/1544 interrupt settings    */
    while(1);

}


/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)       */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz)   */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz) */
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode           */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN   */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins    */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin)    */

}
```

```
/*****************************************************************************/
/*      H8SX/1544 INTC Initialize routine                                    */
/*****************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;                /*   Interrupt control mode 2          */
    INTC.IPRQ.BIT._RCAN01 = 7;              /*   Set interrupt priority level: RCAN */
    set_imask_exr(0);                       /*   Interrupt mask level 0            */


}
/*****************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename    :  rcan.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
/*****************************************************************************/
/*      RCAN Initialize routin                                               */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*   Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)     */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)   */
    RCANET0.IRR.WORD = 0x0001;           /*   Clear IRR0 (clearing condition: write 1)          */
    RCANET0.MCR.WORD |= 0x8000;          /*   Reorder ID: Set to MCR15 = 1 (initial setting)    */

    for(i = 0;i < 16;i++){               /*   Initialize mailbox (RAM area)                     */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }
}

/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    RCANET0.IMR.WORD &= 0xFEFB;     /*   Enable mailbox empty and remote frame receive interrupts */
    RCANET0.MBIMR0.WORD &= 0xFFFD; /*   Enable MB1 interrupt                                     */

    RCANET0.MB[1].CTRL0.WORD.H = 0x1554;/*   STDID: H'555, standard format, RTR = 0          */
    RCANET0.MB[1].LAFM.WORD.H = 0x0000; /*   STD_LAFM and IDE_LAFM settings                  */
    RCANET0.MB[1].CTRL1.BYTE.H = 0x11;
                                 /*  ATX=1, MBC = B'001 (remote frame receive, data frame transmit) */
```

```
    RCANET0.MB[1].MSG_DATA[0] = 0x00;  /*   Transmit data: 0x00                          */
    RCANET0.MB[1].MSG_DATA[1] = 0x11;  /*   Transmit data: 0x11                          */
    RCANET0.MB[1].MSG_DATA[2] = 0x22;  /*   Transmit data: 0x22                          */
    RCANET0.MB[1].MSG_DATA[3] = 0x33;  /*   Transmit data: 0x33                          */
    RCANET0.MB[1].MSG_DATA[4] = 0x44;  /*   Transmit data: 0x44                          */
    RCANET0.MB[1].MSG_DATA[5] = 0x55;  /*   Transmit data: 0x55                          */
    RCANET0.MB[1].MSG_DATA[6] = 0x66;  /*   Transmit data: 0x66                          */
    RCANET0.MB[1].MSG_DATA[7] = 0x77;  /*   Transmit data: 0x77                          */

    /*  Bit rate = 500 kbps  */
    RCANET0.BCR1.WORD = 0x4300;/*  TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, (pϕ = 20 MHz) */
    RCANET0.BCR0.WORD = 0x0001;        /*   BRP = 1                                     */


}

/*****************************************************************************/
/*      RCAN start routine                                                   */
/*****************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                     /*   Clear MCR0                 */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000); /*   GSR3 = 0?                   */


}


/*****************************************************************************/
/*      Remote Frame Received Interrupt routine(MailBox1 to 15)              */
/*****************************************************************************/
#pragma interrupt(RM1_0)
void RM1_0(void)
{
    unsigned int i;

    if(RCANET0.RFPR0.WORD){                  /*   Set IRR2, MB1 receive              */
        RCANET0.RFPR0.WORD = 0x0002;
                                  /*   Clear remote frame receive-end flag (clearing condition: write 1) */
        while(RCANET0.RFPR0.WORD & 0x0002);/*   Check flag                          */
    }
    else if(RCANET0.TXACK0.WORD){        /*   Set IRR8, MB1 transmit-end           */
        RCANET0.TXACK0.WORD = 0x0002;    /*   Clear transmit-end flag (clearing condition: write 1)  */
        while(RCANET0.TXACK0.WORD & 0x0002);/*   Check flag                         */
    }


}
/*****************************************************************************/
```

2.5 Disable Automatic Retransmit Function (DART)

2.5.1 Specification

A data frame with standard ID H'555 is transmitted by node A to node B, as shown in figure 2.5.1. If node B is not connected to the CAN bus, there is no node to transmit an ACK, so node A automatically retransmits the message repeatedly while waiting for an ACK to be received. However, if the disable automatic retransmit function (DART) is enabled for node A, the message is not retransmitted and transmission is cancelled instead.

Note: This operation example describes the operation of node A (the transmitting side).

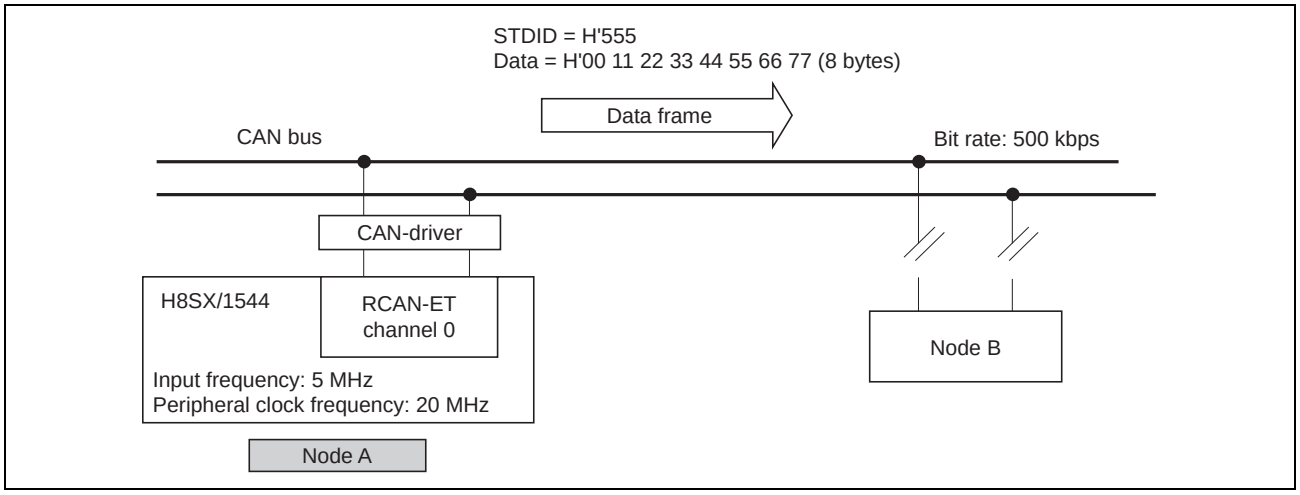


Figure 2.5.1 Communication Specification

2.5.2 Software Description

(1) Module Description

Table 2.5.1 Module Description

Module	Label	Function	Settings
H8SX/1544 initial settings	set_1544_init	See 2.1, Initial Settings.	Device-specific
H8SX/1544 interrupt settings	set_1544_INTC_init		
RCAN initial settings	set_RCAN0_init		RCAN-ET common
RCAN mailbox initial settings	set_RCAN0MB_init		
RCAN start	set_RCAN0_start		
CAN message transmit	RCAN0_Tx	Transmits CAN message.	
Mailbox empty interrupt	SLE0_0	Clears CAN message transmit-end flag.	

(2) Description of Registers Used

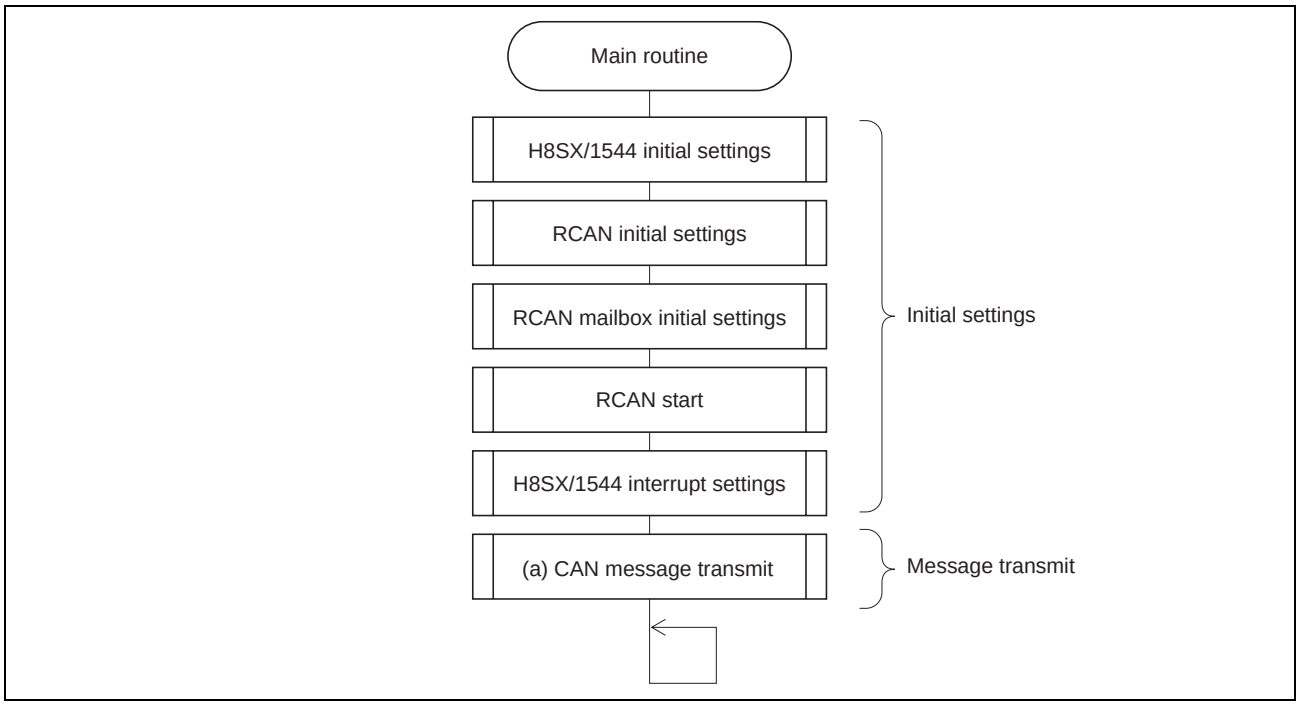
Table 2.5.2 Description of Registers Used

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFEFF	Enables mailbox empty interrupt (IRR8) (IMR8 = 0).
	RCANET0.MBIMR0.WORD	0xFFFD	Enables mailbox 1 interrupt.
	RCANET0.MB[1].CTRL0.WORD.H	0x1554	Sets mailbox 1 to standard format, data frame. Also sets standard ID (H'555).
	RCANET0.MB[1].CTRL1.BYTE.H	0x08	Sets mailbox 1 to transmit. Also sets disable automatic retransmit function (DART = 1, MBC = b'000)). When DART = 1, the corresponding TXCR bit is set automatically at the start of the transmission.
	RCANET0.MB[1].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[1].MSG_DATA[0]	0x00	Sets the transmit data.
	RCANET0.MB[1].MSG_DATA[1]	0x11	
	RCANET0.MB[1].MSG_DATA[2]	0x22	
	RCANET0.MB[1].MSG_DATA[3]	0x33	
	RCANET0.MB[1].MSG_DATA[4]	0x44	
	RCANET0.MB[1].MSG_DATA[5]	0x55	
	RCANET0.MB[1].MSG_DATA[6]	0x66	
	RCANET0.MB[1].MSG_DATA[7]	0x77	
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
H8SX/1544 interrupt settings			
CAN message transmit	RCANET0.TXPR0.LONG	0x00000002	Sets mailbox 1 to transmit-wait status.
Mailbox empty interrupt	RCANET0.ABACK0.WORD	0x0002	Clears mailbox 1 transmit-abort flag. (Clearing condition: write 1)

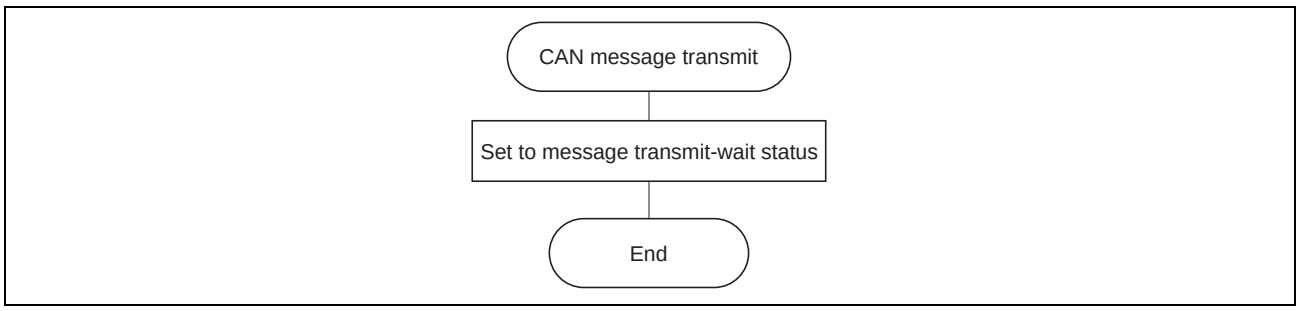
Note: The bit names listed in the table correspond to the bit names used in the hardware manual.

2.5.3 Flowcharts

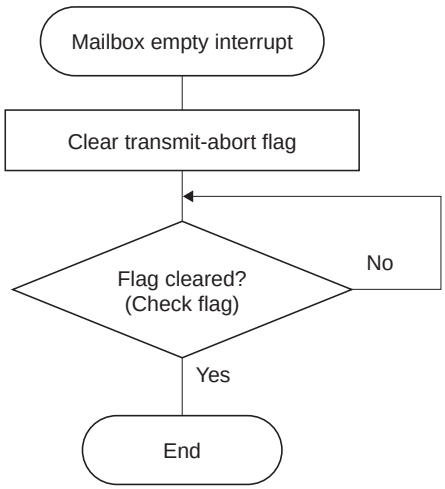
Main Routine



(a) CAN Message Transmit Routine (RCAN-ET Common Settings)



(b) Mailbox Empty Interrupt Routine (RCAN-ET Common Settings)



Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

2.5.4 Program Listing

(1) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void RCAN0_Tx(void);
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{
    set_1544_init();                        /*  H8SX/1544 initialization        */
    set_RCAN0_init();                       /*  RCAN initialization             */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();                   /*  H8SX/1544 interrupt settings    */
    RCAN0_Tx();                             /*  CAN message transmit            */
    while(1);

}


/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)       */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz)   */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz) */
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode           */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN   */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins    */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin)    */

}
```

```
/*****************************************************************************/
/*      H8SX/1544 INTC Initialize routine                                    */
/*****************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;                /*   Interrupt control mode 2          */
    INTC.IPRQ.BIT._RCAN01 = 7;              /*   Set interrupt priority level: RCAN */
    set_imask_exr(0);                       /*   Interrupt mask level 0            */


}
/*****************************************************************************/
```

(2) rcan.c

```
/*****************************************************************************/
/*  Filename    :  rcan.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void RCAN0_Tx(void);
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*   Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)     */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)  */
    RCANET0.IRR.WORD = 0x0001;           /*   Clear IRR0 (clearing condition: write 1)     */
    RCANET0.MCR.WORD |= 0x8000;          /*   Reorder ID: Set to MCR15 = 1 (initial setting)  */

    for(i = 0;i < 16;i++){               /*   Initialize mailbox (RAM area)                */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }


}

/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    RCANET0.IMR.WORD &= 0xFEFF;                 /*   Enable mailbox empty interrupt     */
    RCANET0.MBIMR0.WORD &= 0xFFFD;              /*   Enable MB1 interrupt               */

    RCANET0.MB[1].CTRL0.WORD.H = 0x1554;/*  ID: H'555, standard format, data frame      */
    RCANET0.MB[1].CTRL1.BYTE.H = 0x08;/*   DART = 1 (disable automatic retransmit), MBC = B'000*/
    RCANET0.MB[1].CTRL1.BYTE.L = 0x08;          /*   Data length: 8 bytes               */
```

```
    RCANET0.MB[1].MSG_DATA[0] = 0x00;          /*   Transmit data: 0x00                */
    RCANET0.MB[1].MSG_DATA[1] = 0x11;          /*   Transmit data: 0x11                */
    RCANET0.MB[1].MSG_DATA[2] = 0x22;          /*   Transmit data: 0x22                */
    RCANET0.MB[1].MSG_DATA[3] = 0x33;          /*   Transmit data: 0x33                */
    RCANET0.MB[1].MSG_DATA[4] = 0x44;          /*   Transmit data: 0x44                */
    RCANET0.MB[1].MSG_DATA[5] = 0x55;          /*   Transmit data: 0x55                */
    RCANET0.MB[1].MSG_DATA[6] = 0x66;          /*   Transmit data: 0x66                */
    RCANET0.MB[1].MSG_DATA[7] = 0x77;          /*   Transmit data: 0x77                */

    /*  Bit rate = 500 kbps  */
    RCANET0.BCR1.WORD = 0x4300;/*  TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, (pϕ = 20 MHz) */
    RCANET0.BCR0.WORD = 0x0001;                /*   BRP = 1                            */


}

/***********************************************************************/
/*      RCAN start routine                                             */
/***********************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                    /*   Clear MCR0                   */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000); /*   GSR3 = 0?                    */


}

/***********************************************************************/
/*      RCAN send message routine                                      */
/***********************************************************************/
void RCAN0_Tx(void){


    RCANET0.TXPR0.LONG = 0x00000002;           /*   Set MB1 to transmit-wait status    */

}

/***********************************************************************/
/*      Mailbox Empty Interrupt routine                                */
/***********************************************************************/
#pragma interrupt(SLE0_0)
void SLE0_0(void)
{

    RCANET0.ABACK0.WORD = 0x0002;         /*   Clear transmit-abort flag (clearing condition: write 1)*/
    while(RCANET0.ABACK0.WORD & 0x0002);  /*   Check flag                           */

}
/***********************************************************************/
```

2.6 New Message Control (NMC)

2.6.1 Specification

Messages 1 and 2 are transmitted by node A and received by node B, as shown in figure 2.6.1.

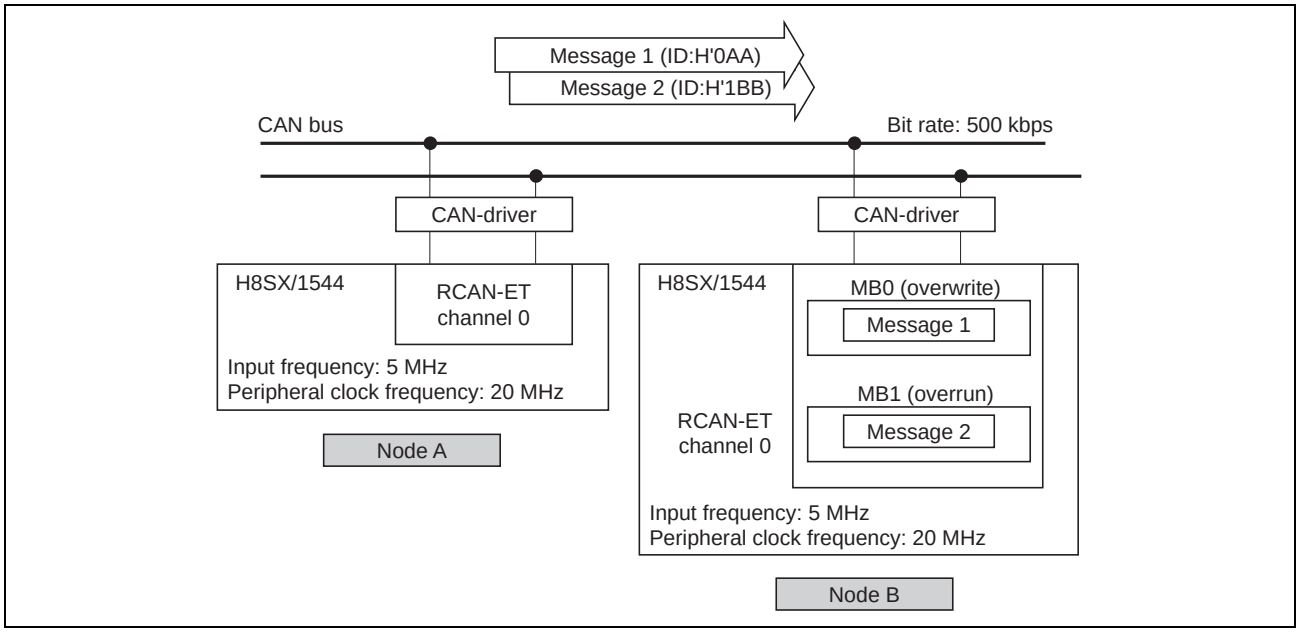


Figure 2.6.1 Communication Specification (1)

If messages 3 and 4 are transmitted by node A before the node B receive-end flag (RXPR) has been cleared, mailbox overwrite and overrun occur on node B.

- Mailbox 0 (MB0): Set to overwrite mode (NMC = 1), so message 3 overwrites the preceding message.
- Mailbox 1 (MB1): Set to overrun mode (NMC = 0), so message 4 does not overwrite the preceding message and message 2 is retained.

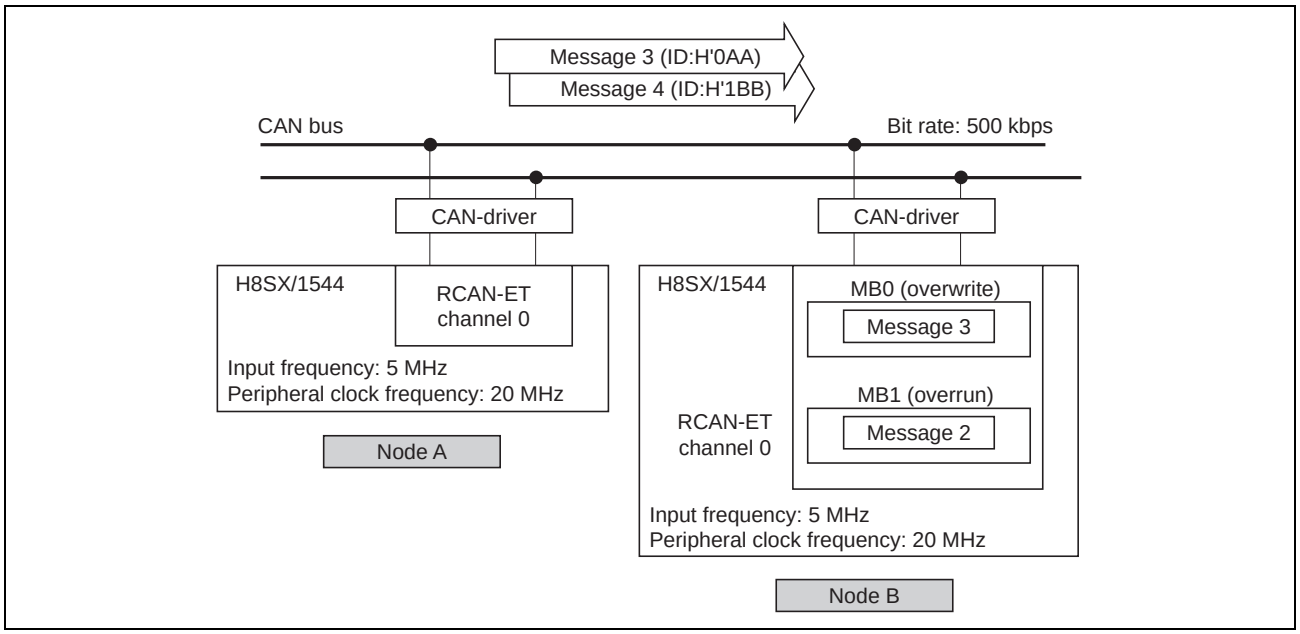


Figure 2.6.2 Communication Specification (2)

2.6.2 Software Description

(1) Module Description

Table 2.6.1 Module Description

Module	Label	Function	Settings
H8SX/1544 initial settings	set_1544_init	See 2.1, Initial Settings.	Device-specific
H8SX/1544 interrupt settings	set_1544_INTC_init		
RCAN initial settings	set_RCAN0_init		RCAN-ET common
RCAN mailbox initial settings	set_RCAN0MB_init		
RCAN start	set_RCAN0_start		
CAN message transmit	RCAN0_Tx	Transmits CAN message.	
Mailbox empty interrupt	SLE0_0	Clears CAN message transmit-end flag. Transmits next message.	
Message overwrite/overrun interrupt	OVR0_0	Clears unsent message status flag. Stores unset message in RAM.	
Receive buffer initialization	Clear_MBbuff	Initializes on-chip RAM area for storing CAN message.	—

(2) Description of Constants Used

Table 2.6.2 Description of Constants Used

Label	Setting Value	Function
ID_1	0x0AA	CAN message ID
ID_2	0x1BB	CAN message ID

(3) Description of Registers Used

(a) Transmitting Side

Table 2.6.3 Description of Registers Used (Transmitting Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFEFF	Enables mailbox empty interrupt (IRR8) (IMR8 = 0).
	RCANET0.MBIMR0.WORD	0xFFF9	Enables interrupts for mailboxes 1 and 2.
	RCANET0.MB[1].CTRL0.WORD.H	(ID_1<<2)	Sets mailbox 1 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[1].CTRL1.BYTE.H	0x00	Sets mailbox 1 to transmit.
	RCANET0.MB[1].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[1].MSG_DATA[0]	0x11	Sets the transmit data.
	RCANET0.MB[1].MSG_DATA[1]	0x11	
	RCANET0.MB[1].MSG_DATA[2]	0x11	
	RCANET0.MB[1].MSG_DATA[3]	0x11	
	RCANET0.MB[1].MSG_DATA[4]	0x11	
	RCANET0.MB[1].MSG_DATA[5]	0x11	
	RCANET0.MB[1].MSG_DATA[6]	0x11	
	RCANET0.MB[1].MSG_DATA[7]	0x11	
	RCANET0.MB[2].CTRL0.WORD.H	(ID_2<<2)	Sets mailbox 2 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[2].CTRL1.BYTE.H	0x00	Sets mailbox 2 to transmit.
	RCANET0.MB[2].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[2].MSG_DATA[0]	0x22	Sets the transmit data.
	RCANET0.MB[2].MSG_DATA[1]	0x22	
	RCANET0.MB[2].MSG_DATA[2]	0x22	
	RCANET0.MB[2].MSG_DATA[3]	0x22	
	RCANET0.MB[2].MSG_DATA[4]	0x22	
	RCANET0.MB[2].MSG_DATA[5]	0x22	
	RCANET0.MB[2].MSG_DATA[6]	0x22	
	RCANET0.MB[2].MSG_DATA[7]	0x22	
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
H8SX/1544 interrupt settings			
CAN message transmit	RCANET0.TXPR0.LONG	0x00000006	Sets mailboxes 1 and 2 to transmit-wait status.

Module	Register	Setting Value	Function
Mailbox empty interrupt	RCANET0.TXACK0.WORD	0x0002	Clears mailbox 1 transmit-end flag. (Clearing condition: write 1)
	RCANET0.TXACK0.WORD	0x0004	Clears mailbox 2 transmit-end flag. (Clearing condition: write 1)
	RCANET0.MB[1].MSG_DATA[0]	0x33	Sets the transmit data.
	RCANET0.MB[1].MSG_DATA[1]	0x33	
	RCANET0.MB[1].MSG_DATA[2]	0x33	
	RCANET0.MB[1].MSG_DATA[3]	0x33	
	RCANET0.MB[1].MSG_DATA[4]	0x33	
	RCANET0.MB[1].MSG_DATA[5]	0x33	
	RCANET0.MB[1].MSG_DATA[6]	0x33	
	RCANET0.MB[1].MSG_DATA[7]	0x33	
	RCANET0.MB[2].MSG_DATA[0]	0x44	
	RCANET0.MB[2].MSG_DATA[1]	0x44	
	RCANET0.MB[2].MSG_DATA[2]	0x44	
	RCANET0.MB[2].MSG_DATA[3]	0x44	
	RCANET0.MB[2].MSG_DATA[4]	0x44	
	RCANET0.MB[2].MSG_DATA[5]	0x44	
	RCANET0.MB[2].MSG_DATA[6]	0x44	
	RCANET0.MB[2].MSG_DATA[7]	0x44	
	RCANET0.TXPR0.LONG	0x00000006	Sets mailboxes 1 and 2 to transmit-wait status.

Notes: 1. The bit names listed in the table correspond to the bit names used in the hardware manual.
2. ID_1 and ID_2 indicate labels listed in (2), “Description of Constants Used,” in 2.6.2, “Software Description.”

(b) Receiving Side

Table 2.6.4 Description of Registers Used (Receiving Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFDFF	Enables message overwrite/overrun interrupt (IRR9) (IMR9 = 0).
	RCANET0.MBIMR0.WORD	0xFFFC	Enables interrupts for mailboxes 0 and 1.
	RCANET0.MB[0].CTRL0.WORD.H	(ID_1<<2)	Sets mailbox 0 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[0].LAFM.WORD.H	0x0000	Sets filter mask for standard ID and IDE bit of mailbox 0.
	RCANET0.MB[0].CTRL1.BYTE.H	0x22	Sets mailbox 0 to receive. Also sets overwrite mode. (NMC = 1, MBC = b'010)
	RCANET0.MB[1].CTRL0.WORD.H	(ID_2<<2)	Sets mailbox 1 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[1].LAFM.WORD.H	0x0000	Sets filter mask for standard ID and IDE bit of mailbox 1.
	RCANET0.MB[1].CTRL1.BYTE.H	0x02	Sets mailbox 0 to receive. Also sets overwrite mode. (NMC = 0, MBC = b'010)
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
H8SX/1544 interrupt settings			
Message overwrite/overrun interrupt	RCANET0.UMSR0.WORD	0x0001	Clears mailbox 0 unread message status flag. (Clearing condition: write 1)
	RCANET0.UMSR0.WORD	0x0002	Clears mailbox 1 unread message status flag. (Clearing condition: write 1)

Notes: 1. The bit names listed in the table correspond to the bit names used in the hardware manual.
2. ID_1 and ID_2 indicate labels listed in (2), “Description of Constants Used,” in 2.6.2, “Software Description.”

(4) Description of RAM Used

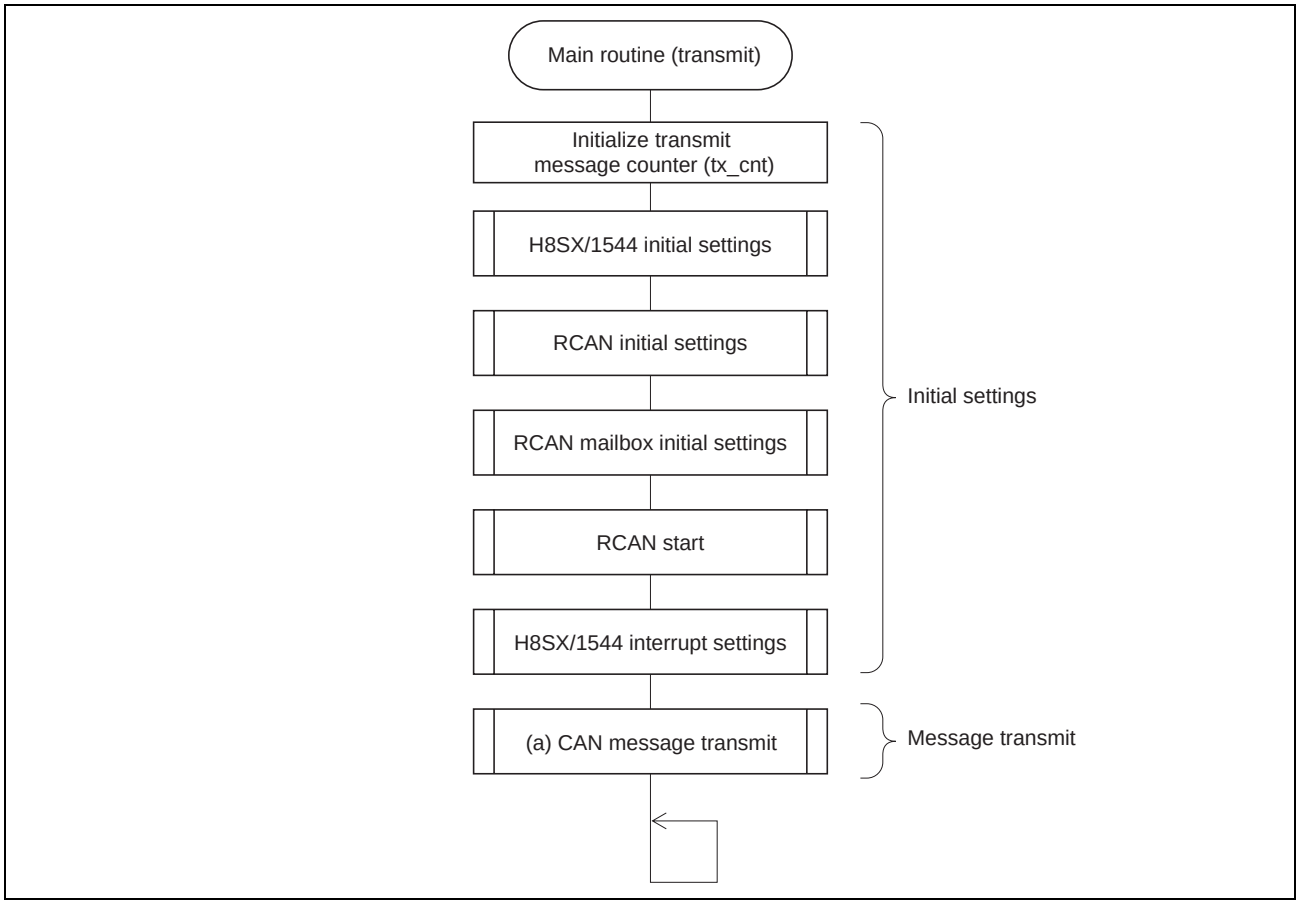
Table 2.6.5 Description of RAM Used

Module	Label	Function
Receive buffer initialization	Mbbuff[2].ID.LONG MBbuff[2].ID.WORD.H MBbuff[2].ID.WORD.L	Stores receive ID.
Message overwrite/overrun interrupt	MBbuff[2].DATA.LONG[0] to [1] MBbuff[2].DATA.WORD[0] to [3] MBbuff[2].DATA.BYTE[0] to [7]	Stores receive data.
Main routine Mailbox empty interrupt	tx_cnt	Indicates number of transmit-end messages.

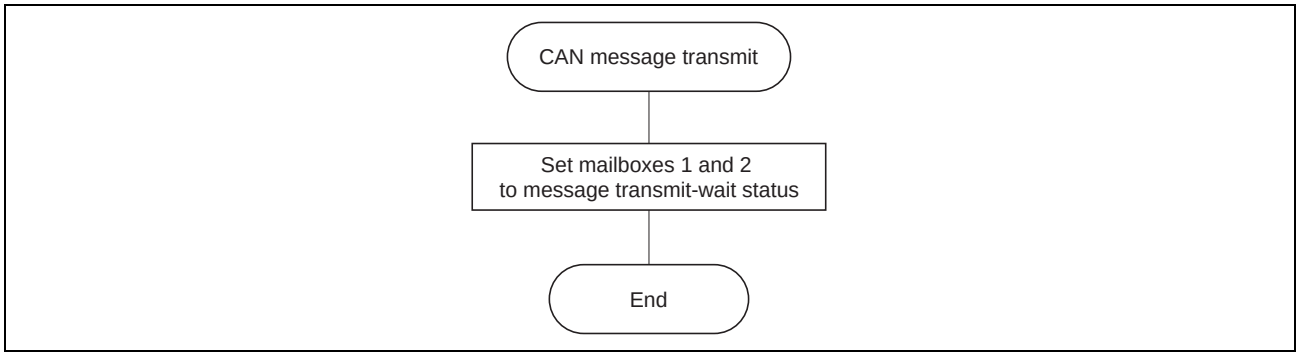
2.6.3 Flowcharts

(1) Transmitting Side Flowcharts

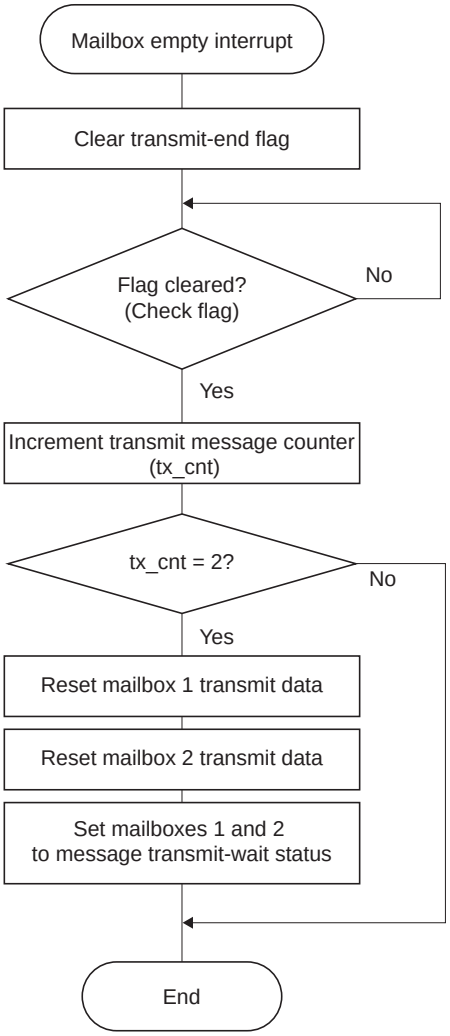
Main Routine



(a) CAN Message Transmit Routine (RCAN-ET Common Settings)



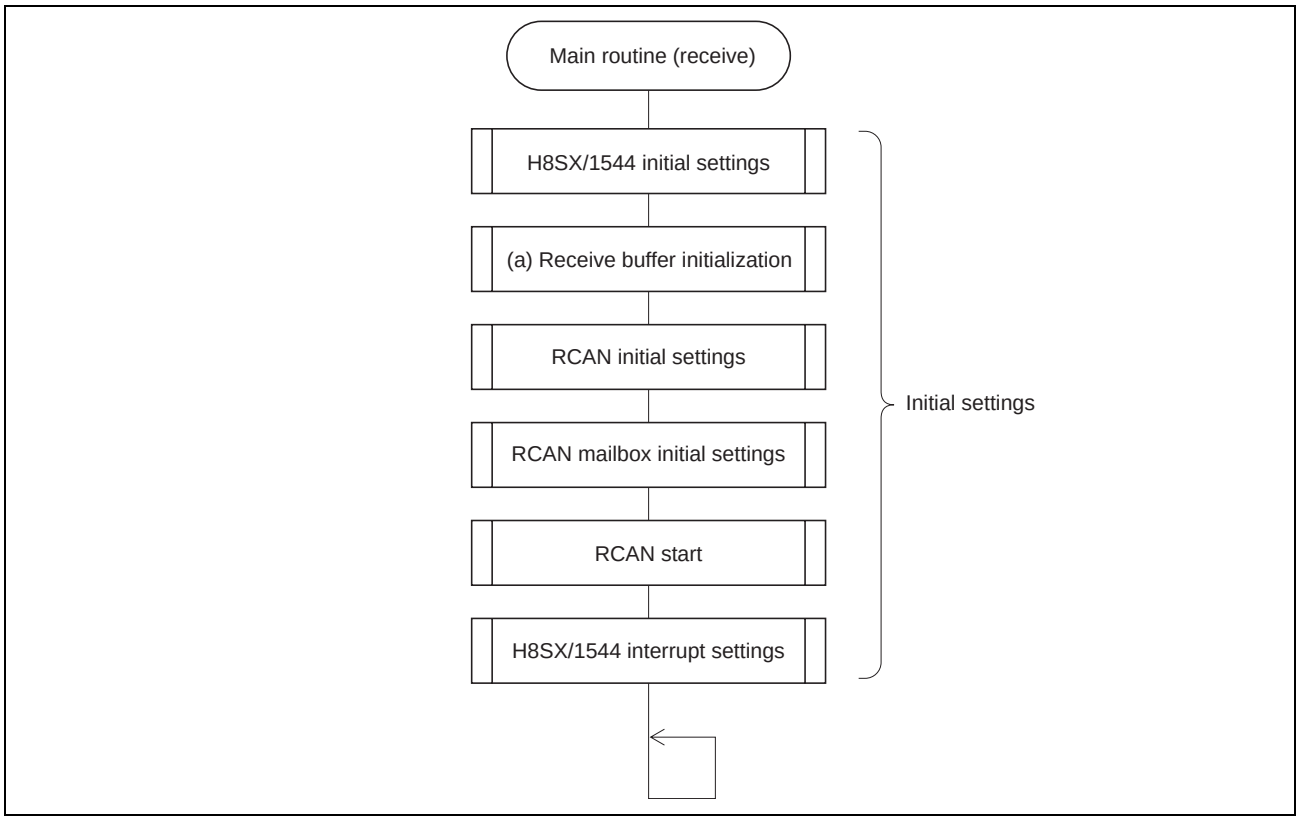
(b) Mailbox Empty Interrupt Routine (RCAN-ET Common Settings)



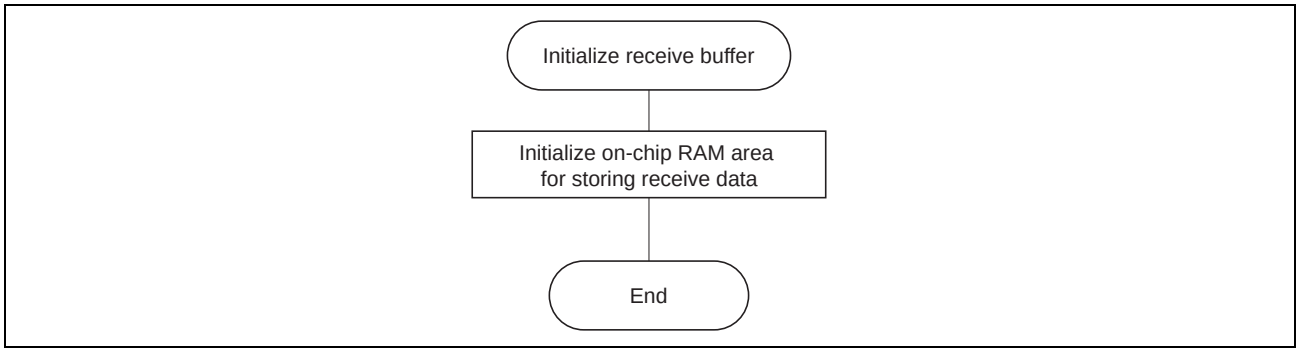
Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

(2) Receiving Side Flowcharts

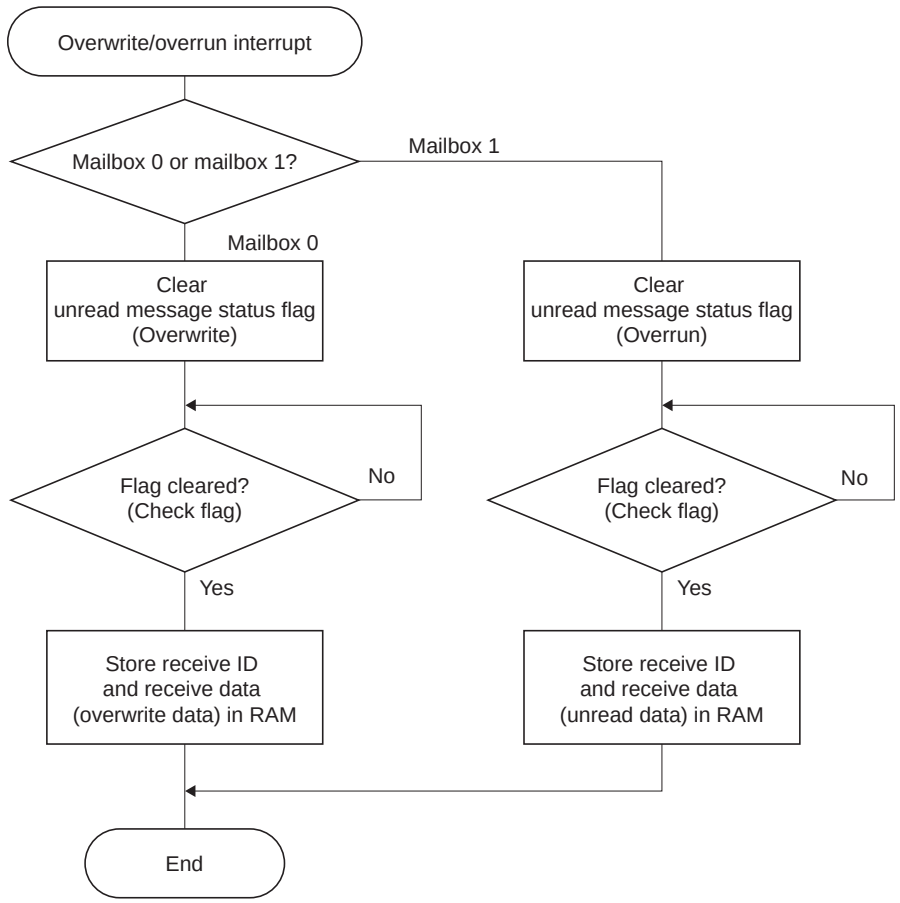
Main Routine



(a) Receive Buffer Initialization Routine (Device-Specific Settings)



(b) Message Overrun/Overwrite Interrupt Routine (RCAN-ET Common Settings)



Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

2.6.4 Program Listing

(1) Transmitting Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void RCAN0_Tx(void);
/*---------------------------------------------------------------------------*/
unsigned char tx_cnt;
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{
    tx_cnt = 0;
    set_1544_init();                        /*  H8SX/1544 initialization     */
    set_RCAN0_init();                       /*  RCAN initialization          */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();                   /*  H8SX/1544 interrupt settings */
    RCAN0_Tx();                             /*  CAN message transmit         */
    while(1);
}

/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)       */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz)   */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz) */
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode       */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin) */
}
```

```
/*****************************************************************************/
/*      H8SX/1544 INTC Initialize routine                                    */
/*****************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;                /*   Interrupt control mode 2          */
    INTC.IPRQ.BIT._RCAN01 = 7;              /*   Set interrupt priority level: RCAN */
    set_imask_exr(0);                       /*   Interrupt mask level 0            */


}
/*****************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename    :  rcan.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void RCAN0_Tx(void);
/*---------------------------------------------------------------------------*/
extern unsigned char tx_cnt;
/*---------------------------------------------------------------------------*/
#define ID_1        0x0AA
#define ID_2        0x1BB
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*  Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)    */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)  */
    RCANET0.IRR.WORD = 0x0001;           /*  Clear IRR0 (clearing condition: write 1)    */
    RCANET0.MCR.WORD |= 0x8000;          /*  Reorder ID: Set to MCR15 = 1 (initial setting)  */

    for(i = 0;i < 16;i++){               /*  Initialize mailbox (RAM area)               */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }
}
```

```
/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    unsigned int i;

    RCANET0.IMR.WORD &= 0xFEFF;                 /*  Enable mailbox empty interrupt      */
    RCANET0.MBIMR0.WORD = 0xFFF9;               /*  Enable MB1-2 interrupt              */

    RCANET0.MB[1].CTRL0.WORD.H  = (ID_1<<2);    /*  Set STDID, standard format, data frame  */
    RCANET0.MB[1].CTRL1.BYTE.H = 0x00;          /*  Set MB1 to transmit-wait status     */
    RCANET0.MB[1].CTRL1.BYTE.L = 0x08;          /*  Data length: 8 bytes                */
    for(i = 0;i < 8;i++){
        RCANET0.MB[1].MSG_DATA[i] = 0x11;       /*  Transmit data                       */
    }

    RCANET0.MB[2].CTRL0.WORD.H  = (ID_2<<2);    /*  Set STDID, standard format, data frame  */
    RCANET0.MB[2].CTRL1.BYTE.H = 0x00;          /*  Set mailbox 2 to transmit           */
    RCANET0.MB[2].CTRL1.BYTE.L = 0x08;          /*  Data length: 8 bytes                */
    for(i = 0;i < 8;i++){
        RCANET0.MB[2].MSG_DATA[i] = 0x22;       /*  Transmit data                       */
    }

    /*  Bit rate = 500 kbps  */
    RCANET0.BCR1.WORD = 0x4300;/*  TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, (pϕ = 20 MHz) */
    RCANET0.BCR0.WORD = 0x0001;                 /*  BRP = 1                             */


}

/*****************************************************************************/
/*      RCAN start routine                                                   */
/*****************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                     /*  Clear MCR0                  */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000); /*  GSR3 = 0?                   */

}

/*****************************************************************************/
/*      RCAN send message routine                                            */
/*****************************************************************************/
void RCAN0_Tx(void){

    RCANET0.TXPR0.LONG = 0x00000006;            /*  Set MB1-2 to transmit-wait status   */

}
```

```
/************************************************************************/
/*      Mailbox Empty Interrupt routine                                 */
/************************************************************************/
#pragma interrupt(SLE0_0)
void SLE0_0(void)
{

    unsigned int i;

    RCANET0.TXACK0.WORD = RCANET0.TXACK0.WORD;
                                        /* Clear transmit-end flag (clearing condition: write 1) */
    while(RCANET0.TXACK0.WORD);             /*   Check flag                                      */

    tx_cnt++;

    if(tx_cnt == 2){
        for(i = 0;i < 8;i++){
            RCANET0.MB[1].MSG_DATA[i] = 0x33; /*   Transmit data                               */
        }
        for(i = 0;i < 8;i++){
            RCANET0.MB[2].MSG_DATA[i] = 0x44; /*   Transmit data                               */
        }
        RCANET0.TXPR0.LONG = 0x00000006;    /*   Set MB1-2 to transmit-wait status           */

    }

}
/************************************************************************/
```

(2) Receiving Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void Clear_MBbuff(void);
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{

    set_1544_init();                        /*  H8SX/1544 initialization                    */
    Clear_MBbuff();                         /*  Initialize RAM area for storing receive data */
    set_RCAN0_init();                       /*  RCAN initialization                         */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();                   /*  H8SX/1544 interrupt settings                */
    while(1);

}


/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)       */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz)   */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz) */
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode                       */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN               */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins    */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin)                */

}
```

```
/*****************************************************************************/
/*      H8SX/1544 INTC Initialize routine                                    */
/*****************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;                /*   Interrupt control mode 2          */
    INTC.IPRQ.BIT._RCAN01 = 7;              /*   Set interrupt priority level: RCAN */
    set_imask_exr(0);                       /*   Interrupt mask level 0            */


}

/*****************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename:  rcan.c*/
/*  Written :  '06/06/01   REV.1.00 */
/*  Purpose :  for H8SX/1544   RCAN-ET */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void Clear_MBbuff(void);
/*---------------------------------------------------------------------------*/
struct {
    union {
        unsigned long LONG;
        struct{
            unsigned short H;
            unsigned short L;
        }WORD;
    }ID;
    union {
        unsigned char BYTE[8];
        unsigned short WORD[4];
        unsigned long LONG[2];
    }DATA;
} MBbuff[2];
/*---------------------------------------------------------------------------*/
#define ID_1        0x0AA
#define ID_2        0x1BB
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*  Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)    */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)  */
    RCANET0.IRR.WORD = 0x0001;           /*  Clear IRR0 (clearing condition: write 1)    */
    RCANET0.MCR.WORD |= 0x8000;          /*  Reorder ID: Set to MCR15 = 1 (initial setting)  */

    for(i = 0;i < 16;i++){               /*  Initialize mailbox (RAM area)               */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }
}
```

```
/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    RCANET0.IMR.WORD &= 0xFDFF;                 /*   Enable message overrun/overwrite interrupt  */
    RCANET0.MBIMR0.WORD &= 0xFFFC;              /*   Enable interrupts for MB0-1                 */

    RCANET0.MB[0].CTRL0.WORD.H  = (ID_1<<2); /*  Set STDID, standard format, data frame       */
    RCANET0.MB[0].LAFM.WORD.H = 0x0000;         /*   STD_LAFM and IDE_LAFM settings              */
    RCANET0.MB[0].CTRL1.BYTE.H = 0x22;  /*   NMC = 1 (overwrite mode), MBC = b'010 (receive)  */

    RCANET0.MB[1].CTRL0.WORD.H  = (ID_2<<2); /*  Set STDID, standard format, data frame       */
    RCANET0.MB[1].LAFM.WORD.H = 0x0000;         /*   STD_LAFM and IDE_LAFM settings              */
    RCANET0.MB[1].CTRL1.BYTE.H = 0x02;  /*   NMC = 0 (overrun mode), MBC = b'010 (receive)    */
    /*  Bit rate = 500 kbps  */
    RCANET0.BCR1.WORD = 0x4300;/*  TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, (pϕ = 20 MHz)*/
    RCANET0.BCR0.WORD = 0x0001;                 /*   BRP = 1                                     */


}

/*****************************************************************************/
/*      RCAN start routine                                                   */
/*****************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                     /*   Clear MCR0                 */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000); /*   GSR3 = 0?                  */


}


/*****************************************************************************/
/*      RAM area Initialize routine                                          */
/*****************************************************************************/
void Clear_MBbuff(void){

    unsigned int i;

    for(i = 0;i < 2;i++){                       /*   Initialize RAM area for storing receive data  */
        MBbuff[i].ID.LONG = 0;
        MBbuff[i].DATA.LONG[0] = 0;
        MBbuff[i].DATA.LONG[1] = 0;
    }

}
```

```
/*****************************************************************************/
/*      Message Overrun/Overwrite Interrupt routine                          */
/*****************************************************************************/
#pragma interrupt(OVR0_0)
void OVR0_0(void)
{
    unsigned int i;

    if(RCANET0.UMSR0.WORD&0x0001){              /*  Set IRR9, MB0 overwrite          */
        RCANET0.UMSR0.WORD = 0x0001;
                                    /*  Clear unread message status flag (clearing condition: write 1)  */
        while(RCANET0.UMSR0.WORD & 0x0001);     /*  Check flag                       */
        MBbuff[0].ID.WORD.H = RCANET0.MB[0].CTRL0.WORD.H;/*  Store receive ID in RAM  */
        MBbuff[0].ID.WORD.L = RCANET0.MB[0].CTRL0.WORD.L;/*  Store receive ID in RAM  */
        for(i = 0;i < 8;i++){                   /*  Store receive data in RAM        */
            MBbuff[0].DATA.BYTE[i] = RCANET0.MB[0].MSG_DATA[i];
        }
    }
    else if(RCANET0.UMSR0.WORD&0x0002){         /*  Set IRR9, MB1 overwrite          */
        RCANET0.UMSR0.WORD = 0x0002;
                                    /*  Clear unread message status flag (clearing condition: write 1)  */
        while(RCANET0.UMSR0.WORD & 0x0002);     /*  Check flag                       */
        MBbuff[1].ID.WORD.H = RCANET0.MB[1].CTRL0.WORD.H;/*  Store receive ID in RAM  */
        MBbuff[1].ID.WORD.L = RCANET0.MB[1].CTRL0.WORD.L;/*  Store receive ID in RAM  */
        for(i = 0;i < 8;i++){                   /*  Store receive data in RAM        */
            MBbuff[1].DATA.BYTE[i] = RCANET0.MB[1].MSG_DATA[i];
        }
    }


}
/*****************************************************************************/
```

2.7 Group Receive Using Message Filtering (LAFM)

2.7.1 Specification

As shown in figure 2.7.1, 15 messages are transmitted by node A and received by node B. Node B uses the receive message filtering function (LAFM) to sort the received messages into groups.

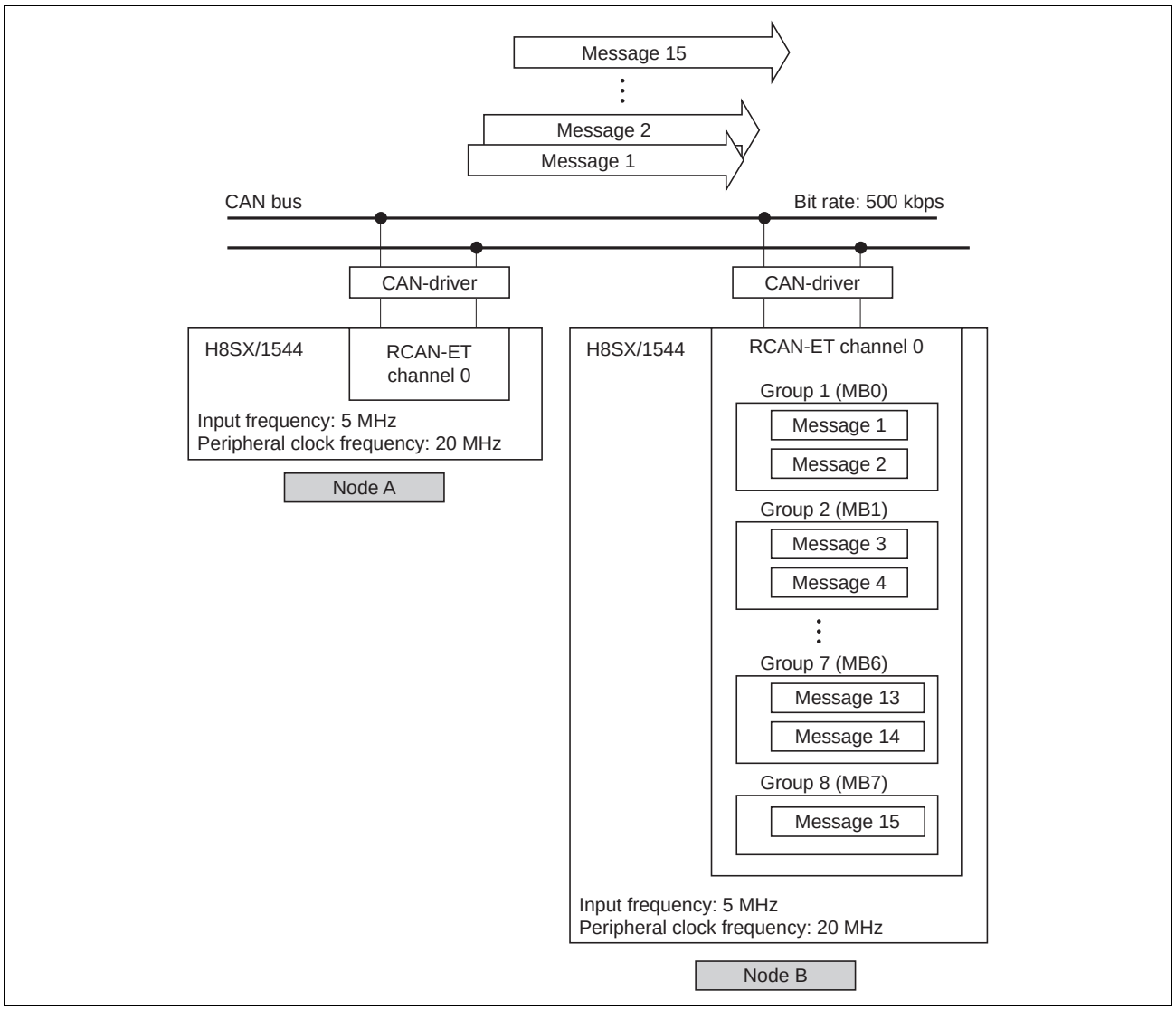


Figure 2.7.1 Communication Specification

Table 2.7.1 lists the message specifications.

Table 2.7.1 Message Specifications

Message	Standard ID	Data	Receive mailbox
Message 1	H'011	H'01 01 01 01 01 01 01 01 (8 bytes)	Mailbox 0
Message 2	H'0AA	H'02 02 02 02 02 02 02 02 (8 bytes)	
Message 3	H'122	H'03 03 03 03 03 03 03 03 (8 bytes)	Mailbox 1
Message 4	H'1BB	H'04 04 04 04 04 04 04 04 (8 bytes)	
Message 5	H'233	H'05 05 05 05 05 05 05 05 (8 bytes)	Mailbox 2
Message 6	H'2CC	H'06 06 06 06 06 06 06 06 (8 bytes)	
Message 7	H'344	H'07 07 07 07 07 07 07 07 (8 bytes)	Mailbox 3
Message 8	H'3DD	H'08 08 08 08 08 08 08 08 (8 bytes)	
Message 9	H'455	H'09 09 09 09 09 09 09 09 (8 bytes)	Mailbox 4
Message 10	H'4EE	H'0A 0A 0A 0A 0A 0A 0A 0A (8 bytes)	
Message 11	H'566	H'0B 0B 0B 0B 0B 0B 0B 0B (8 bytes)	Mailbox 5
Message 12	H'5FF	H'0C 0C 0C 0C 0C 0C 0C 0C (8 bytes)	
Message 13	H'677	H'0D 0D 0D 0D 0D 0D 0D 0D (8 bytes)	Mailbox 6
Message 14	H'6EE	H'0E 0E 0E 0E 0E 0E 0E 0E (8 bytes)	
Message 15	H'788	H'0F 0F 0F 0F 0F 0F 0F 0F (8 bytes)	Mailbox 7

Note: This operation example uses standard format. (It is not necessary to set an extended ID.)

2.7.2 Software Description

(1) Module Description

Table 2.7.2 Module Description

Module	Label	Function	Settings
H8SX/1544 initial settings	set_1544_init	See 2.1, Initial Settings.	Device-specific
H8SX/1544 interrupt settings	set_1544_INTC_init		
RCAN initial settings	set_RCAN0_init		RCAN-ET common
RCAN mailbox initial settings	set_RCAN0MB_init		
RCAN start	set_RCAN0_start		
CAN message transmit	RCAN0_Tx	Transmits CAN message.	
Mailbox empty interrupt	SLE0_0	Clears CAN message transmit-end flag.	
Data frame receive interrupt (mailbox 0)	RM0_0	Clears CAN message receive-end flag. Stores mailbox 0 receive data in RAM.	
Data frame receive interrupt (mailboxes 1 to 7)	RM1_0	Clears CAN message receive-end flag. Stores mailbox 1 to 7 receive data in RAM.	
Receive buffer initialization	Clear_MBbuff	Initializes on-chip RAM area for storing CAN message.	—

(2) Description of Constants Used

Table 2.7.3 Description of Constants Used

Label	Setting Value	Function
ID_1	0x011	CAN message ID
ID_2	0x0AA	
ID_3	0x122	
ID_4	0x1BB	
ID_5	0x233	
ID_6	0x2CC	
ID_7	0x344	
ID_8	0x3DD	
ID_9	0x455	
ID_10	0x4EE	
ID_11	0x566	
ID_12	0x5FF	
ID_13	0x677	
ID_14	0x6EE	
ID_15	0x788	
STD_ID_MB0	0x000	ID setting value for receive mailbox
STD_ID_MB1	0x100	
STD_ID_MB2	0x200	
STD_ID_MB3	0x300	
STD_ID_MB4	0x400	
STD_ID_MB5	0x500	
STD_ID_MB6	0x600	
STD_ID_MB7	0x700	
STD_LAFM	0x0FF	Filter mask setting value for standard ID
IDE_LAFM	0x0000	Filter mask setting value for IDE bit

(3) Description of Registers Used

(a) Transmitting Side

Table 2.7.4 Description of Registers Used (Transmitting Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFEFF	Enables mailbox empty interrupt (IRR8) (IMR8 = 0).
	RCANET0.MBIMR0.WORD	0x0001	Enables interrupts for mailboxes 1 to 15.
	RCANET0.MB[1].CTRL0.WORD.H	(ID_1<<2)	Sets mailbox 1 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[1].CTRL1.BYTE.H	0x00	Sets mailbox 1 to transmit.
	RCANET0.MB[1].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[1].MSG_DATA[0] to RCANET0.MB[1].MSG_DATA[7]	0x01 × 8	Sets the transmit data.
	RCANET0.MB[2].CTRL0.WORD.H	(ID_2<<2)	Sets mailbox 2 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[2].CTRL1.BYTE.H	0x00	Sets mailbox 2 to transmit.
	RCANET0.MB[2].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[2].MSG_DATA[0] to RCANET0.MB[2].MSG_DATA[7]	0x02 × 8	Sets the transmit data.
	RCANET0.MB[3].CTRL0.WORD.H	(ID_3<<2)	Sets mailbox 3 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[3].CTRL1.BYTE.H	0x00	Sets mailbox 3 to transmit.
	RCANET0.MB[3].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[3].MSG_DATA[0] to RCANET0.MB[3].MSG_DATA[7]	0x03 × 8	Sets the transmit data.
	RCANET0.MB[4].CTRL0.WORD.H	(ID_4<<2)	Sets mailbox 4 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[4].CTRL1.BYTE.H	0x00	Sets mailbox 4 to transmit.
	RCANET0.MB[4].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[4].MSG_DATA[0] to RCANET0.MB[4].MSG_DATA[7]	0x04 × 8	Sets the transmit data.
	RCANET0.MB[5].CTRL0.WORD.H	(ID_5<<2)	Sets mailbox 5 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[5].CTRL1.BYTE.H	0x00	Sets mailbox 5 to transmit.
	RCANET0.MB[5].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[5].MSG_DATA[0] to RCANET0.MB[5].MSG_DATA[7]	0x05 × 8	Sets the transmit data.
	RCANET0.MB[6].CTRL0.WORD.H	(ID_6<<2)	Sets mailbox 6 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[6].CTRL1.BYTE.H	0x00	Sets mailbox 6 to transmit.
	RCANET0.MB[6].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[6].MSG_DATA[0] to RCANET0.MB[6].MSG_DATA[7]	0x06 × 8	Sets the transmit data.
	RCANET0.MB[7].CTRL0.WORD.H	(ID_7<<2)	Sets mailbox 7 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[7].CTRL1.BYTE.H	0x00	Sets mailbox 7 to transmit.
	RCANET0.MB[7].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[7].MSG_DATA[0] to RCANET0.MB[7].MSG_DATA[7]	0x07 × 8	Sets the transmit data.

Module	Register	Setting Value	Function
RCAN mailbox initial settings	RCANET0.MB[8].CTRL0.WORD.H	(ID_8<<2)	Sets mailbox 8 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[8].CTRL1.BYTE.H	0x00	Sets mailbox 8 to transmit.
	RCANET0.MB[8].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[8].MSG_DATA[0] to RCANET0.MB[8].MSG_DATA[7]	0x08 × 8	Sets the transmit data.
	RCANET0.MB[9].CTRL0.WORD.H	(ID_9<<2)	Sets mailbox 9 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[9].CTRL1.BYTE.H	0x00	Sets mailbox 9 to transmit.
	RCANET0.MB[9].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[9].MSG_DATA[0] to RCANET0.MB[9].MSG_DATA[7]	0x09 × 8	Sets the transmit data.
	RCANET0.MB[10].CTRL0.WORD.H	(ID_10<<2)	Sets mailbox 10 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[10].CTRL1.BYTE.H	0x00	Sets mailbox 10 to transmit.
	RCANET0.MB[10].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[10].MSG_DATA[0] to RCANET0.MB[10].MSG_DATA[7]	0x0A × 8	Sets the transmit data.
	RCANET0.MB[11].CTRL0.WORD.H	(ID_11<<2)	Sets mailbox 11 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[11].CTRL1.BYTE.H	0x00	Sets mailbox 11 to transmit.
	RCANET0.MB[11].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[11].MSG_DATA[0] to RCANET0.MB[11].MSG_DATA[7]	0x0B × 8	Sets the transmit data.
	RCANET0.MB[12].CTRL0.WORD.H	(ID_12<<2)	Sets mailbox 12 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[12].CTRL1.BYTE.H	0x00	Sets mailbox 12 to transmit.
	RCANET0.MB[12].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[12].MSG_DATA[0] to RCANET0.MB[12].MSG_DATA[7]	0x0C × 8	Sets the transmit data.
	RCANET0.MB[13].CTRL0.WORD.H	(ID_13<<2)	Sets mailbox 13 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[13].CTRL1.BYTE.H	0x00	Sets mailbox 13 to transmit.
	RCANET0.MB[13].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[13].MSG_DATA[0] to RCANET0.MB[13].MSG_DATA[7]	0x0D × 8	Sets the transmit data.
	RCANET0.MB[14].CTRL0.WORD.H	(ID_14<<2)	Sets mailbox 14 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[14].CTRL1.BYTE.H	0x00	Sets mailbox 14 to transmit.
	RCANET0.MB[14].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[14].MSG_DATA[0] to RCANET0.MB[14].MSG_DATA[7]	0x0E × 8	Sets the transmit data.
	RCANET0.MB[15].CTRL0.WORD.H	(ID_15<<2)	Sets mailbox 15 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[15].CTRL1.BYTE.H	0x00	Sets mailbox 15 to transmit.
	RCANET0.MB[15].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[15].MSG_DATA[0] to RCANET0.MB[15].MSG_DATA[7]	0x0F × 8	Sets the transmit data.
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	

Module	Register	Setting Value	Function
RCAN start	See 2.1, Initial Settings.		
H8SX/1544 interrupt settings			
CAN message transmit	RCANET0.TXPR0.LONG	0x0000FFFE	Sets mailboxes 1 to 15 to transmit.
Mailbox empty interrupt	RCANET0.TXACK0.WORD	—	Clears transmit-end flag for mailboxes 1 to 15. (Clearing condition: write 1)

Notes: 1. The bit names listed in the table correspond to the bit names used in the hardware manual.
2. ID_x (x = 1 to 15) indicates labels listed in (2), “Description of Constants Used,” in 2.7.2, “Software Description.”

(b) Receiving Side

Table 2.7.5 Description of Registers Used (Receiving Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFFFD	Enables data frame receive interrupt (IRR1) (IMR1 = 0).
	RCANET0.MBIMR0.WORD	0xFF00	Enables interrupts for mailboxes 0 to 7.
	RCANET0.MB[0].CTRL0.WORD.H	(STD_ID_MB0<<2)	Sets mailbox 0 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[0].LAFM.WORD.H	0x03FC	Sets filter mask for standard ID and IDE bit of mailbox 0.
	RCANET0.MB[0].CTRL1.BYTE.H	0x02	Sets mailbox 0 to receive.
	RCANET0.MB[1].CTRL0.WORD.H	(STD_ID_MB1<<2)	Sets mailbox 1 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[1].LAFM.WORD.H	0x03FC	Sets filter mask for standard ID and IDE bit of mailbox 1.
	RCANET0.MB[1].CTRL1.BYTE.H	0x02	Sets mailbox 1 to receive.
	RCANET0.MB[2].CTRL0.WORD.H	(STD_ID_MB2<<2)	Sets mailbox 2 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[2].LAFM.WORD.H	0x03FC	Sets filter mask for standard ID and IDE bit of mailbox 2.
	RCANET0.MB[2].CTRL1.BYTE.H	0x02	Sets mailbox 2 to receive.
	RCANET0.MB[3].CTRL0.WORD.H	(STD_ID_MB3<<2)	Sets mailbox 3 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[3].LAFM.WORD.H	0x03FC	Sets filter mask for standard ID and IDE bit of mailbox 3.
	RCANET0.MB[3].CTRL1.BYTE.H	0x02	Sets mailbox 3 to receive.
	RCANET0.MB[4].CTRL0.WORD.H	(STD_ID_MB4<<2)	Sets mailbox 4 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[4].LAFM.WORD.H	0x03FC	Sets filter mask for standard ID and IDE bit of mailbox 4.
	RCANET0.MB[4].CTRL1.BYTE.H	0x02	Sets mailbox 4 to receive.

Module	Register	Setting Value	Function
RCAN mailbox initial settings	RCANET0.MB[5].CTRL0.WORD.H	(STD_ID_MB5<<2)	Sets mailbox 5 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[5].LAFM.WORD.H	0x03FC	Sets filter mask for standard ID and IDE bit of mailbox 5.
	RCANET0.MB[5].CTRL1.BYTE.H	0x02	Sets mailbox 5 to receive.
	RCANET0.MB[6].CTRL0.WORD.H	(STD_ID_MB6<<2)	Sets mailbox 6 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[6].LAFM.WORD.H	0x03FC	Sets filter mask for standard ID and IDE bit of mailbox 6.
	RCANET0.MB[6].CTRL1.BYTE.H	0x02	Sets mailbox 6 to receive.
	RCANET0.MB[7].CTRL0.WORD.H	(STD_ID_MB7<<2)	Sets mailbox 7 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[7].LAFM.WORD.H	0x03FC	Sets filter mask for standard ID and IDE bit of mailbox 7.
	RCANET0.MB[7].CTRL1.BYTE.H	0x02	Sets mailbox 7 to receive.
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
H8SX/1544 interrupt settings			
Data frame receive interrupt (mailbox 0)	RCANET0.RXPR0.WORD	0x0001	Clears mailbox 0 receive-end flag. (Clearing condition: write 1)
Data frame receive interrupt (mailboxes 1 to 7)	RCANET0.RXPR0.WORD	—	Clears receive-end flag for mailboxes 1 to 15. (Clearing condition: write 1)

Notes: 1. The bit names listed in the table correspond to the bit names used in the hardware manual.
2. ID_x (x = 1 to 7) indicates labels listed in (2), “Description of Constants Used,” in 2.7.2, “Software Description.”

(4) Description of RAM Used

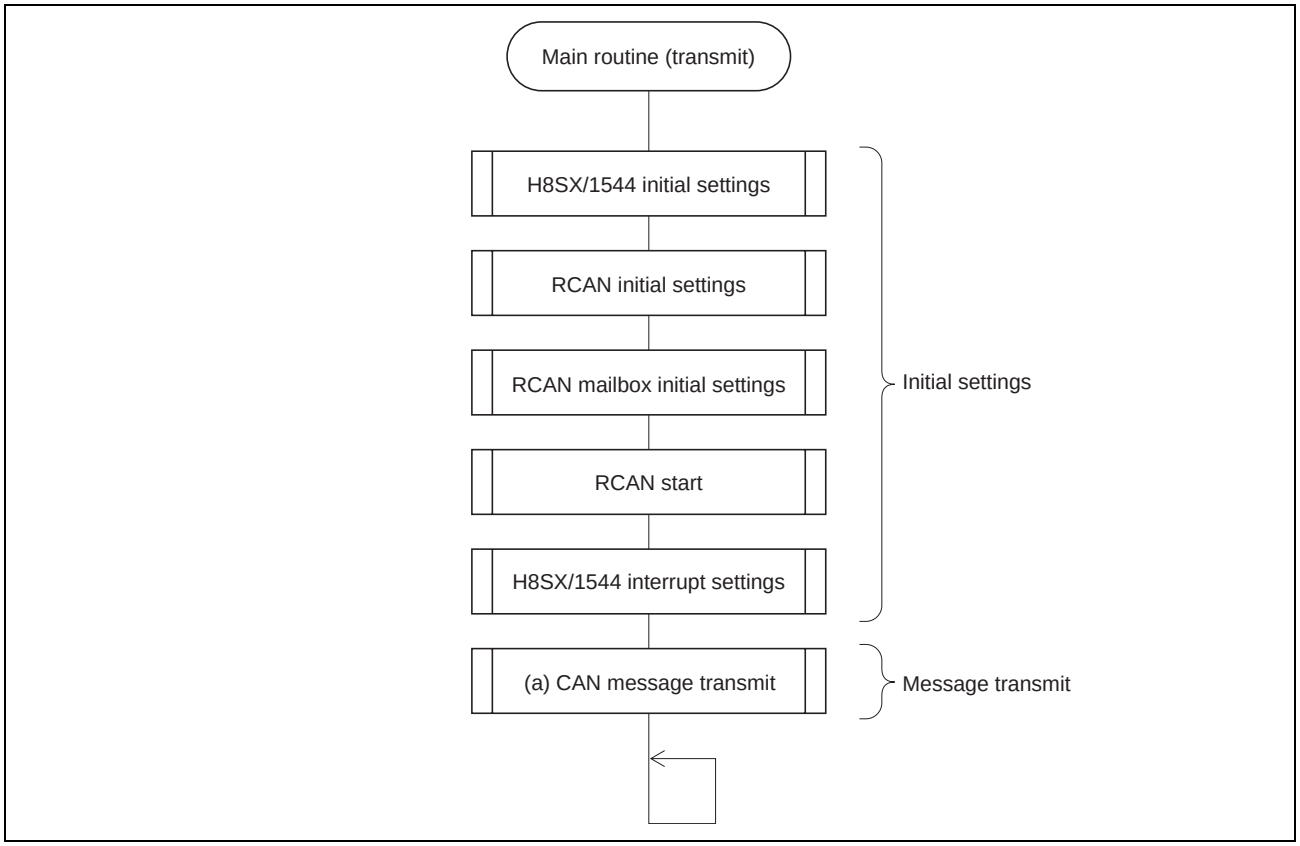
Table 2.7.6 Description of RAM Used

Module	Label	Function
Receive buffer initialization	Mbbuff[16].ID.LONG MBbuff[16].ID.WORD.H MBbuff[16].ID.WORD.L	Stores receive ID.
Data frame receive interrupt	MBbuff[16].DATA.LONG[0] to [1] MBbuff[16].DATA.WORD[0] to [3] MBbuff[16].DATA.BYTE[0] to [7]	Stores receive data.

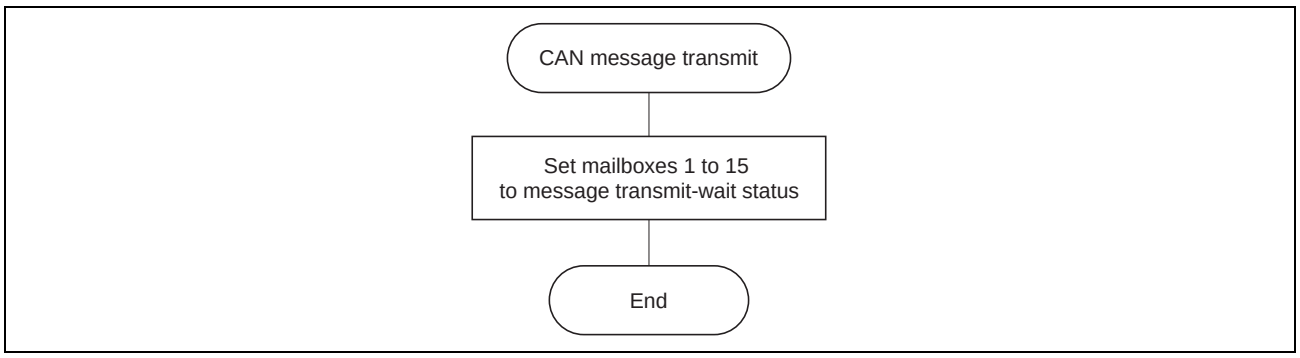
2.7.3 Flowcharts

(1) Transmitting Side Flowcharts

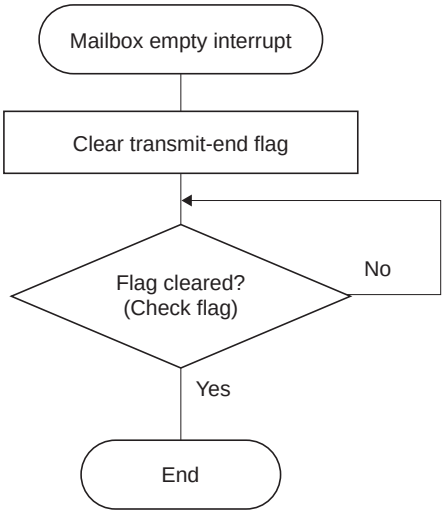
Main Routine



(a) CAN Message Transmit Routine (RCAN-ET Common Settings)



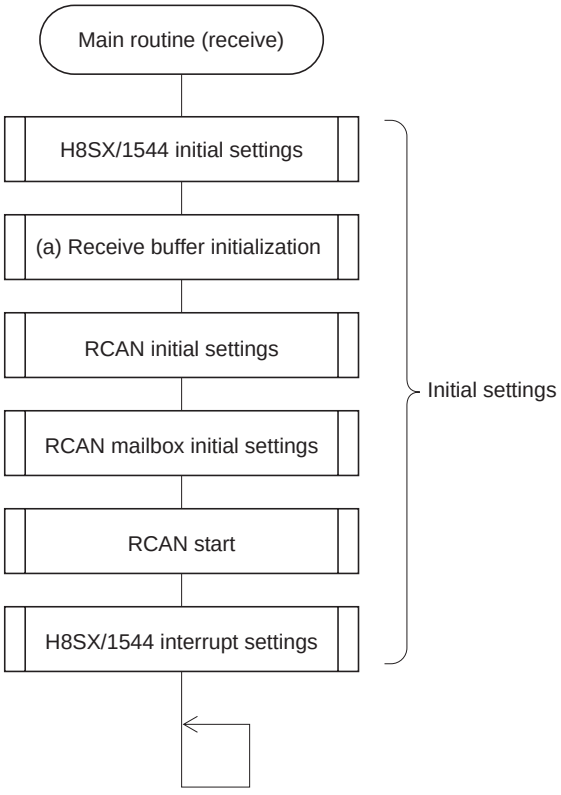
(b) Mailbox Empty Interrupt Routine (RCAN-ET Common Settings)



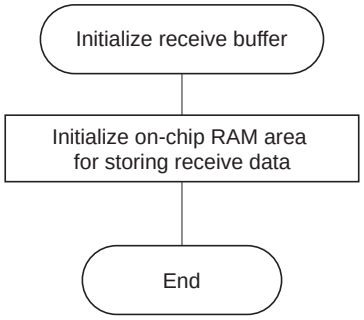
Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

(2) Receiving Side Flowcharts

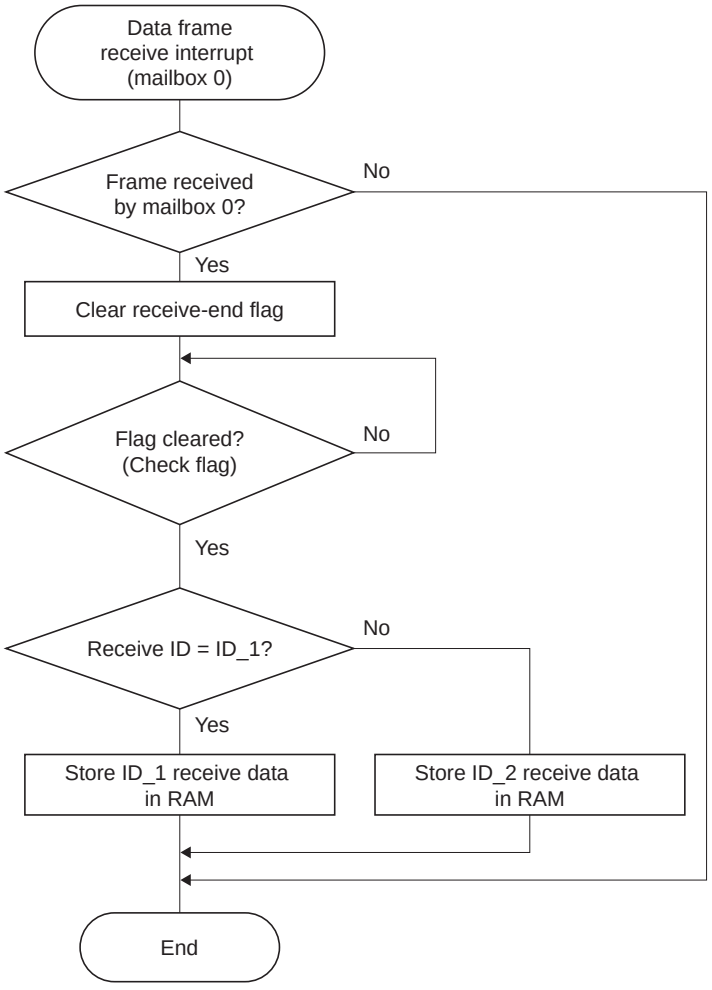
Main Routine



(a) Receive Buffer Initialization Routine (Device-Specific Settings)

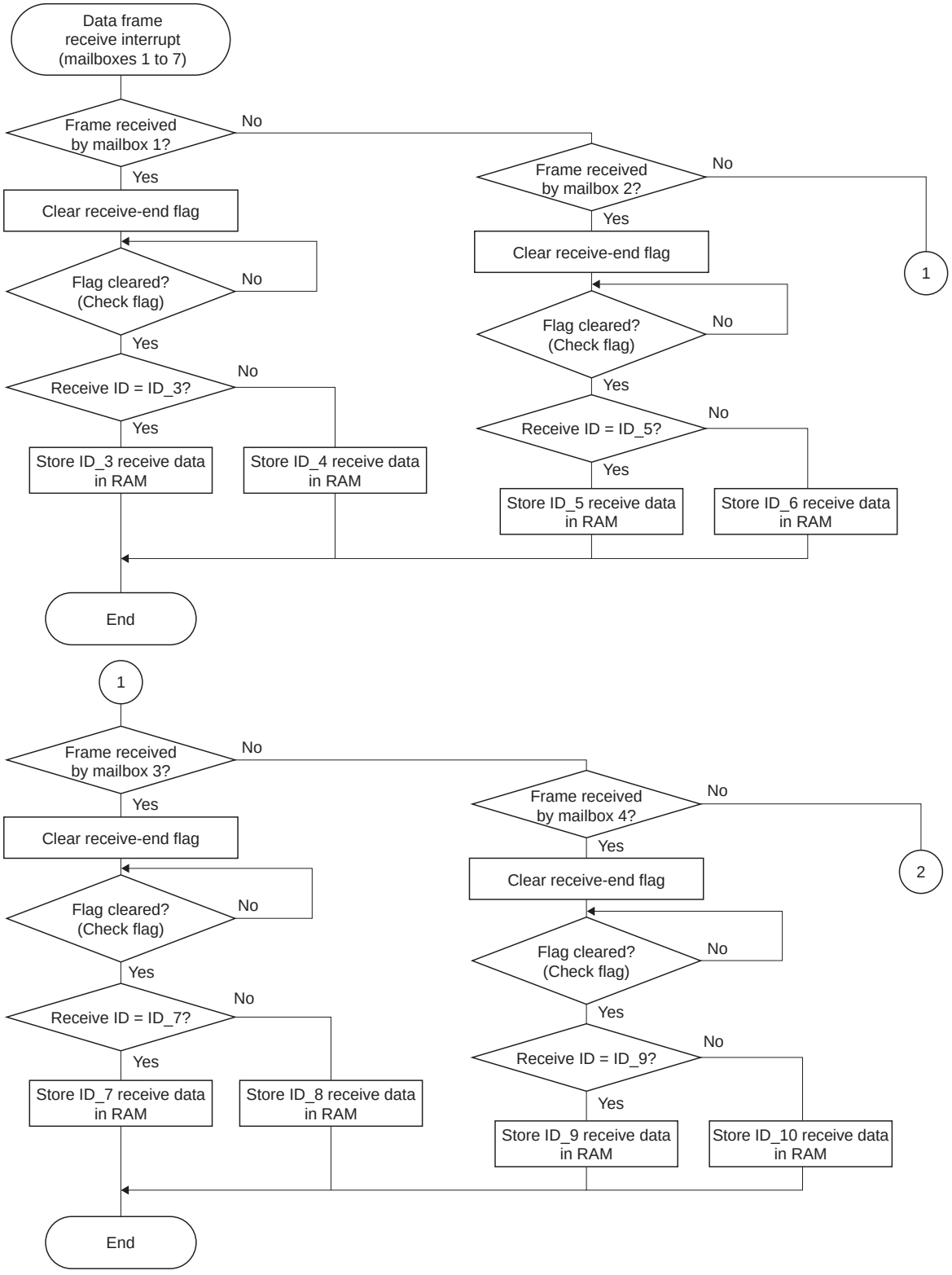


(b) Data Frame Receive Interrupt Routine (Mailbox 0, RCAN-ET Common Settings)

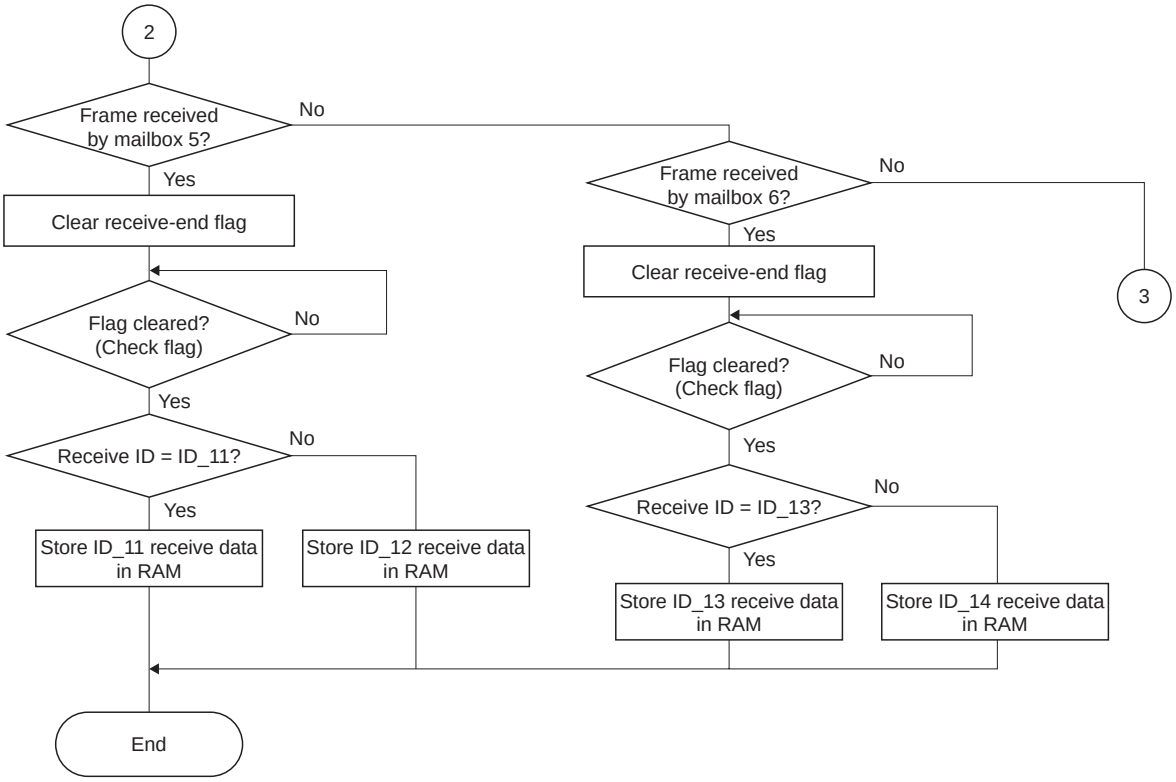


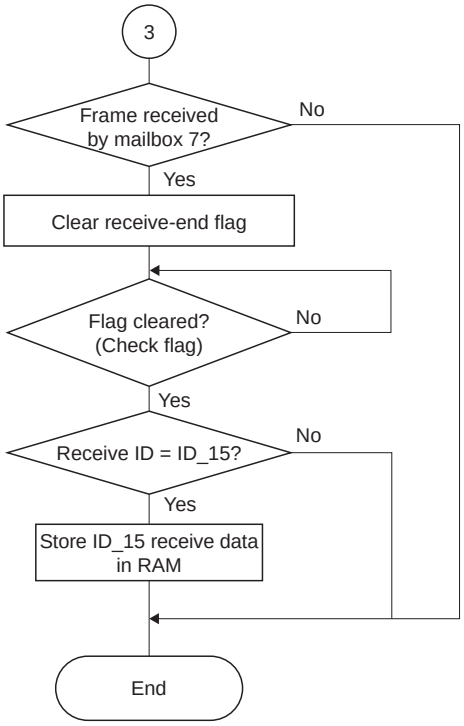
Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

(c) Data Frame Receive Interrupt Routine (Mailboxes 1 to 7, RCAN-ET Common Settings)



Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.





Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

2.7.4 Program Listing

(1) Transmitting Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void RCAN0_Tx(void);
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{
    set_1544_init();                        /*  H8SX/1544 initialization        */
    set_RCAN0_init();                       /*  RCAN initialization             */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();                   /*  H8SX/1544 interrupt settings    */
    RCAN0_Tx();                             /*  CAN message transmit            */
    while(1);

}
/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)       */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz)   */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz) */
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode           */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN   */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins    */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin)    */

}
```

```
/*****************************************************************************/
/*      H8SX/1544 INTC Initialize routine                                    */
/*****************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;                /*   Interrupt control mode 2          */
    INTC.IPRQ.BIT._RCAN01 = 7;              /*   Set interrupt priority level: RCAN */
    set_imask_exr(0);                       /*   Interrupt mask level 0            */


}
/*****************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename    :  rcan.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void RCAN0_Tx(void);
/*---------------------------------------------------------------------------*/
#define ID_1        0x011
#define ID_2        0x0AA
#define ID_3        0x122
#define ID_4        0x1BB
#define ID_5        0x233
#define ID_6        0x2CC
#define ID_7        0x344
#define ID_8        0x3DD
#define ID_9        0x455
#define ID_10       0x4EE
#define ID_11       0x566
#define ID_12       0x5FF
#define ID_13       0x677
#define ID_14       0x6EE
#define ID_15       0x788
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*  Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)    */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)  */
    RCANET0.IRR.WORD = 0x0001;           /*  Clear IRR0 (clearing condition: write 1)    */
    RCANET0.MCR.WORD |= 0x8000;          /*  Reorder ID: Set to MCR15 = 1 (initial setting)  */

    for(i = 0;i < 16;i++){               /*  Initialize mailbox (RAM area)               */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }
}
```

```
/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    unsigned int i;

    RCANET0.IMR.WORD &= 0xFEFF;                /*  Enable mailbox empty interrupt      */
    RCANET0.MBIMR0.WORD = 0x0001;              /*  Enable MB1-15 interrupt             */

    RCANET0.MB[1].CTRL0.WORD.H  = (ID_1<<2);   /*  Set STDID, standard format, data frame  */
    RCANET0.MB[1].CTRL1.BYTE.H = 0x00;         /*  Set mailbox 1 to transmit           */
    RCANET0.MB[1].CTRL1.BYTE.L = 0x08;         /*  Data length: 8 bytes                */
    for(i = 0;i < 8;i++){
        RCANET0.MB[1].MSG_DATA[i] = 0x01;      /*  Transmit data                       */
    }

    RCANET0.MB[2].CTRL0.WORD.H  = (ID_2<<2);   /*  Set STDID, standard format, data frame  */
    RCANET0.MB[2].CTRL1.BYTE.H = 0x00;         /*  Set mailbox 2 to transmit           */
    RCANET0.MB[2].CTRL1.BYTE.L = 0x08;         /*  Data length: 8 bytes                */
    for(i = 0;i < 8;i++){
        RCANET0.MB[2].MSG_DATA[i] = 0x02;      /*  Transmit data                       */
    }

    RCANET0.MB[3].CTRL0.WORD.H  = (ID_3<<2);   /*  Set STDID, standard format, data frame  */
    RCANET0.MB[3].CTRL1.BYTE.H = 0x00;         /*  Set mailbox 3 to transmit           */
    RCANET0.MB[3].CTRL1.BYTE.L = 0x08;         /*  Data length: 8 bytes                */
    for(i = 0;i < 8;i++){
        RCANET0.MB[3].MSG_DATA[i] = 0x03;      /*  Transmit data                       */
    }

    RCANET0.MB[4].CTRL0.WORD.H  = (ID_4<<2);   /*  Set STDID, standard format, data frame  */
    RCANET0.MB[4].CTRL1.BYTE.H = 0x00;         /*  Set mailbox 4 to transmit           */
    RCANET0.MB[4].CTRL1.BYTE.L = 0x08;         /*  Data length: 8 bytes                */
    for(i = 0;i < 8;i++){
        RCANET0.MB[4].MSG_DATA[i] = 0x04;      /*  Transmit data                       */
    }

    RCANET0.MB[5].CTRL0.WORD.H  = (ID_5<<2);   /*  Set STDID, standard format, data frame  */
    RCANET0.MB[5].CTRL1.BYTE.H = 0x00;         /*  Set mailbox 5 to transmit           */
    RCANET0.MB[5].CTRL1.BYTE.L = 0x08;         /*  Data length: 8 bytes                */
    for(i = 0;i < 8;i++){
        RCANET0.MB[5].MSG_DATA[i] = 0x05;      /*  Transmit data                       */
    }

    RCANET0.MB[6].CTRL0.WORD.H  = (ID_6<<2);   /*  Set STDID, standard format, data frame  */
    RCANET0.MB[6].CTRL1.BYTE.H = 0x00;         /*  Set mailbox 6 to transmit           */
    RCANET0.MB[6].CTRL1.BYTE.L = 0x08;         /*  Data length: 8 bytes                */
    for(i = 0;i < 8;i++){
        RCANET0.MB[6].MSG_DATA[i] = 0x06;      /*  Transmit data                       */
    }
```

```
RCANET0.MB[7].CTRL0.WORD.H  = (ID_7<<2);    /*  Set STDID, standard format, data frame  */
  RCANET0.MB[7].CTRL1.BYTE.H = 0x00;        /*  Set mailbox 7 to transmit               */
  RCANET0.MB[7].CTRL1.BYTE.L = 0x08;        /*  Data length: 8 bytes                    */
  for(i = 0;i < 8;i++){
      RCANET0.MB[7].MSG_DATA[i] = 0x07;     /*  Transmit data                           */
  }

  RCANET0.MB[8].CTRL0.WORD.H  = (ID_8<<2);  /*  Set STDID, standard format, data frame  */
  RCANET0.MB[8].CTRL1.BYTE.H = 0x00;        /*  Set mailbox 8 to transmit               */
  RCANET0.MB[8].CTRL1.BYTE.L = 0x08;        /*  Data length: 8 bytes                    */
  for(i = 0;i < 8;i++){
      RCANET0.MB[8].MSG_DATA[i] = 0x08;     /*  Transmit data                           */
  }

  RCANET0.MB[9].CTRL0.WORD.H  = (ID_9<<2);  /*  Set STDID, standard format, data frame  */
  RCANET0.MB[9].CTRL1.BYTE.H = 0x00;        /*  Set mailbox 9 to transmit               */
  RCANET0.MB[9].CTRL1.BYTE.L = 0x08;        /*  Data length: 8 bytes                    */
  for(i = 0;i < 8;i++){
      RCANET0.MB[9].MSG_DATA[i] = 0x09;     /*  Transmit data                           */
  }

  RCANET0.MB[10].CTRL0.WORD.H  = (ID_10<<2);/*  Set STDID, standard format, data frame  */
  RCANET0.MB[10].CTRL1.BYTE.H = 0x00;       /*  Set mailbox 10 to transmit              */
  RCANET0.MB[10].CTRL1.BYTE.L = 0x08;       /*  Data length: 8 bytes                    */
  for(i = 0;i < 8;i++){
      RCANET0.MB[10].MSG_DATA[i] = 0x0A;    /*  Transmit data                           */
  }

  RCANET0.MB[11].CTRL0.WORD.H  = (ID_11<<2);/*  Set STDID, standard format, data frame  */
  RCANET0.MB[11].CTRL1.BYTE.H = 0x00;       /*  Set mailbox 11 to transmit              */
  RCANET0.MB[11].CTRL1.BYTE.L = 0x08;       /*  Data length: 8 bytes                    */
  for(i = 0;i < 8;i++){
      RCANET0.MB[11].MSG_DATA[i] = 0x0B;    /*  Transmit data                           */
  }

  RCANET0.MB[12].CTRL0.WORD.H  = (ID_12<<2);/*  Set STDID, standard format, data frame  */
  RCANET0.MB[12].CTRL1.BYTE.H = 0x00;       /*  Set mailbox 12 to transmit              */
  RCANET0.MB[12].CTRL1.BYTE.L = 0x08;       /*  Data length: 8 bytes                    */
  for(i = 0;i < 8;i++){
      RCANET0.MB[12].MSG_DATA[i] = 0x0C;    /*  Transmit data                           */
  }

  RCANET0.MB[13].CTRL0.WORD.H  = (ID_13<<2);/*  Set STDID, standard format, data frame  */
  RCANET0.MB[13].CTRL1.BYTE.H = 0x00;       /*  Set mailbox 13 to transmit              */
  RCANET0.MB[13].CTRL1.BYTE.L = 0x08;       /*  Data length: 8 bytes                    */
  for(i = 0;i < 8;i++){
      RCANET0.MB[13].MSG_DATA[i] = 0x0D;    /*  Transmit data                           */
  }
```

```
    RCANET0.MB[14].CTRL0.WORD.H  = (ID_14<<2);/*  Set STDID, standard format, data frame  */
    RCANET0.MB[14].CTRL1.BYTE.H = 0x00;       /*  Set mailbox 14 to transmit             */
    RCANET0.MB[14].CTRL1.BYTE.L = 0x08;       /*  Data length: 8 bytes                   */
    for(i = 0;i < 8;i++){
        RCANET0.MB[14].MSG_DATA[i] = 0x0E;    /*  Transmit data                          */
    }

    RCANET0.MB[15].CTRL0.WORD.H  = (ID_15<<2);/*  Set STDID, standard format, data frame  */
    RCANET0.MB[15].CTRL1.BYTE.H = 0x00;       /*  Set mailbox 15 to transmit             */
    RCANET0.MB[15].CTRL1.BYTE.L = 0x08;       /*  Data length: 8 bytes                   */
    for(i = 0;i < 8;i++){
        RCANET0.MB[15].MSG_DATA[i] = 0x0F;    /*  Transmit data                          */
    }

    /*  Bit rate = 500 kbps  */
    RCANET0.BCR1.WORD = 0x4300;/*  TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, (pϕ = 20 MHz) */
    RCANET0.BCR0.WORD = 0x0001;               /*  BRP = 1                                */

}

/**************************************************************************/
/*      RCAN start routine                                                */
/**************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                    /*  Clear MCR0                    */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000); /*  GSR3 = 0?                     */

}

/**************************************************************************/
/*      RCAN send message routine                                         */
/**************************************************************************/
void RCAN0_Tx(void){

    RCANET0.TXPR0.LONG = 0x0000FFFE;          /*  Set MB1-15 to transmit-wait status    */

}

/**************************************************************************/
/*      Mailbox Empty Interrupt routine                                   */
/**************************************************************************/
#pragma interrupt(SLE0_0)
void SLE0_0(void)
{
    RCANET0.TXACK0.WORD = RCANET0.TXACK0.WORD;
                                          /*  Clear transmit-end flag (clearing condition: write 1)  */
    while(RCANET0.TXACK0.WORD);           /*  Check flag                            */

}
/**************************************************************************/
```

(2) Receiving Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void Clear_MBbuff(void);
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{
    set_1544_init();                        /*  H8SX/1544 initialization                    */
    Clear_MBbuff();                         /*  Initialize RAM area for storing receive data */
    set_RCAN0_init();                       /*  RCAN initialization                         */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();                   /*  H8SX/1544 interrupt settings                */
    while(1);

}

/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)       */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz)   */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz) */
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode                       */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN               */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins    */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin)                */

}
```

```
/*****************************************************************************/
/*      H8SX/1544 INTC Initialize routine                                    */
/*****************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;                /*   Interrupt control mode 2          */
    INTC.IPRQ.BIT._RCAN01 = 7;              /*   Set interrupt priority level: RCAN */
    set_imask_exr(0);                       /*   Interrupt mask level 0            */


}
/*****************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename    :   rcan.c                                                   */
/*  Written     :   '06/06/01   REV.1.00                                     */
/*  Purpose     :   for H8SX/1544   RCAN-ET                                  */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void Clear_MBbuff(void);
/*---------------------------------------------------------------------------*/
struct {
    union {
        unsigned long LONG;
        struct{
            unsigned short H;
            unsigned short L;
        }WORD;
    }ID;
    union {
        unsigned char BYTE[8];
        unsigned short WORD[4];
        unsigned long LONG[2];
    }DATA;
} MBbuff[16];
/*---------------------------------------------------------------------------*/
#define ID_1        0x011
#define ID_2        0x0AA
#define ID_3        0x122
#define ID_4        0x1BB
#define ID_5        0x233
#define ID_6        0x2CC
#define ID_7        0x344
#define ID_8        0x3DD
#define ID_9        0x455
#define ID_10       0x4EE
#define ID_11       0x566
#define ID_12       0x5FF
#define ID_13       0x677
#define ID_14       0x6EE
#define ID_15       0x788

#define STD_ID_MB0      0x000
#define STD_ID_MB1      0x100
#define STD_ID_MB2      0x200
#define STD_ID_MB3      0x300
#define STD_ID_MB4      0x400
#define STD_ID_MB5      0x500
#define STD_ID_MB6      0x600
#define STD_ID_MB7      0x700
#define STD_LAFM        0x0FF
#define IDE_LAFM        0x0000
```

```
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*   Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)    */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)  */
    RCANET0.IRR.WORD = 0x0001;           /*   Clear IRR0 (clearing condition: write 1)          */
    RCANET0.MCR.WORD |= 0x8000;          /*   Reorder ID: Set to MCR15 = 1 (initial setting)    */

    for(i = 0;i < 16;i++){               /*   Initialize mailbox (RAM area)                     */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }


}

/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    RCANET0.IMR.WORD &= 0xFFFD;                  /*   Enable data frame receive interrupt     */
    RCANET0.MBIMR0.WORD &= 0xFF00;               /*   Enable MB0-7 interrupt                  */

    RCANET0.MB[0].CTRL0.WORD.H = (STD_ID_MB0<<2);/*   Set STDID, standard format, data frame  */
    RCANET0.MB[0].LAFM.WORD.H = ((IDE_LAFM)|(STD_LAFM<<2));
                                                 /*   STD_LAFM and IDE_LAFM settings          */
    RCANET0.MB[0].CTRL1.BYTE.H = 0x02;           /*   Set mailbox 0 to receive                */

    RCANET0.MB[1].CTRL0.WORD.H = (STD_ID_MB1<<2);/*   Set STDID, standard format, data frame  */
    RCANET0.MB[1].LAFM.WORD.H = ((IDE_LAFM)|(STD_LAFM<<2));
                                                 /*   STD_LAFM and IDE_LAFM settings          */
    RCANET0.MB[1].CTRL1.BYTE.H = 0x02;           /*   Set mailbox 1 to receive                */

    RCANET0.MB[2].CTRL0.WORD.H = (STD_ID_MB2<<2);/*   Set STDID, standard format, data frame  */
    RCANET0.MB[2].LAFM.WORD.H = ((IDE_LAFM)|(STD_LAFM<<2));
                                                 /*   STD_LAFM and IDE_LAFM settings          */
    RCANET0.MB[2].CTRL1.BYTE.H = 0x02;           /*   Set mailbox 2 to receive                */

    RCANET0.MB[3].CTRL0.WORD.H = (STD_ID_MB3<<2);/*   Set STDID, standard format, data frame  */
    RCANET0.MB[3].LAFM.WORD.H = ((IDE_LAFM)|(STD_LAFM<<2));
                                                 /*   STD_LAFM and IDE_LAFM settings          */
    RCANET0.MB[3].CTRL1.BYTE.H = 0x02;           /*   Set mailbox 3 to receive                */
```

```
    RCANET0.MB[4].CTRL0.WORD.H = (STD_ID_MB4<<2);/*   Set STDID, standard format, data frame  */
    RCANET0.MB[4].LAFM.WORD.H = ((IDE_LAFM)|(STD_LAFM<<2));
                                                  /*   STD_LAFM and IDE_LAFM settings   */
    RCANET0.MB[4].CTRL1.BYTE.H = 0x02;            /*   Set mailbox 4 to receive         */

    RCANET0.MB[5].CTRL0.WORD.H = (STD_ID_MB5<<2);/*   Set STDID, standard format, data frame  */
    RCANET0.MB[5].LAFM.WORD.H = ((IDE_LAFM)|(STD_LAFM<<2));
                                                  /*   STD_LAFM and IDE_LAFM settings   */
    RCANET0.MB[5].CTRL1.BYTE.H = 0x02;            /*   Set mailbox 5 to receive         */

    RCANET0.MB[6].CTRL0.WORD.H = (STD_ID_MB6<<2);/*   Set STDID, standard format, data frame  */
    RCANET0.MB[6].LAFM.WORD.H = ((IDE_LAFM)|(STD_LAFM<<2));
                                                  /*   STD_LAFM and IDE_LAFM settings   */
    RCANET0.MB[6].CTRL1.BYTE.H = 0x02;            /*   Set mailbox 6 to receive         */

    RCANET0.MB[7].CTRL0.WORD.H = (STD_ID_MB7<<2);/*   Set STDID, standard format, data frame  */
    RCANET0.MB[7].LAFM.WORD.H = ((IDE_LAFM)|(STD_LAFM<<2));
                                                  /*   STD_LAFM and IDE_LAFM settings   */
    RCANET0.MB[7].CTRL1.BYTE.H = 0x02;            /*   Set mailbox 7 to receive         */

    /*  Bit rate = 500 kbps  */
    RCANET0.BCR1.WORD = 0x4300;/*  TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, (pϕ = 20 MHz)*/
    RCANET0.BCR0.WORD = 0x0001;                   /*   BRP = 1                          */


}

/*****************************************************************************/
/*      RCAN start routine                                                   */
/*****************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                   /*   Clear MCR0                       */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000); /*   GSR3 = 0?                       */


}


/*****************************************************************************/
/*      RAM area Initialize routine                                          */
/*****************************************************************************/
void Clear_MBbuff(void){

    unsigned int i;

    for(i = 0;i < 16;i++){                        /*   Initialize RAM area for storing receive data*/
        MBbuff[i].ID.LONG = 0;
        MBbuff[i].DATA.LONG[0] = 0;
        MBbuff[i].DATA.LONG[1] = 0;
    }
```

```
}

/*****************************************************************************/
/*      Data Frame Received Interrupt routine(MailBox0)                      */
/*****************************************************************************/
#pragma interrupt(RM0_0)
void RM0_0(void)
{
    unsigned int i;
    unsigned short rcv_id;

    /*  MB0     */
    if(RCANET0.RXPR0.WORD&0x0001){              /*  Set IRR1, MB0 receive                   */
        RCANET0.RXPR0.WORD = 0x0001;      /*  Clear receive-end flag (clearing condition: write 1) */
        while(RCANET0.RXPR0.WORD & 0x0001);   /*  Check flag                              */

        rcv_id = RCANET0.MB[0].CTRL0.WORD.H;
        rcv_id = ((rcv_id>>2)&0x07FF);
        switch(rcv_id){
            case ID_1:                          /*  Receive ID = ID_1 (0x011)               */
                for(i = 0;i < 8;i++){           /*  Store receive data in RAM               */
                    MBbuff[1].DATA.BYTE[i] = RCANET0.MB[0].MSG_DATA[i];
                }
                break;
            case ID_2:                          /*  Receive ID = ID_2 (0x0AA)               */
                for(i = 0;i < 8;i++){           /*  Store receive data in RAM               */
                    MBbuff[2].DATA.BYTE[i] = RCANET0.MB[0].MSG_DATA[i];
                }
                break;
            default:
                break;
        }
    }
}

/*****************************************************************************/
/*      Data Frame Received Interrupt routine(MailBox1 to 7)                 */
/*****************************************************************************/
#pragma interrupt(RM1_0)
void RM1_0(void)
{
    unsigned int i;
    unsigned short rcv_id;

    /*  MB1     */
    if(RCANET0.RXPR0.WORD&0x0002){              /*  Set IRR1, MB1 receive                   */
        RCANET0.RXPR0.WORD = 0x0002;      /*  Clear receive-end flag (clearing condition: write 1) */
        while(RCANET0.RXPR0.WORD & 0x0002);   /*  Check flag                              */

        rcv_id = RCANET0.MB[1].CTRL0.WORD.H;
        rcv_id = ((rcv_id>>2)&0x07FF);
```

```
    switch(rcv_id){
        case ID_3:                          /*  Receive ID = ID_3 (0x122)              */
            for(i = 0;i < 8;i++){           /*  Store receive data in RAM              */
                MBbuff[3].DATA.BYTE[i] = RCANET0.MB[1].MSG_DATA[i];
            }
            break;
        case ID_4:                          /*  Receive ID = ID_4 (0x1BB)              */
            for(i = 0;i < 8;i++){           /*  Store receive data in RAM              */
                MBbuff[4].DATA.BYTE[i] = RCANET0.MB[1].MSG_DATA[i];
            }
            break;
        default:
            break;
    }
}
/*  MB2     */
else if(RCANET0.RXPR0.WORD&0x0004){         /*  Set IRR1, MB2 receive                  */
    RCANET0.RXPR0.WORD = 0x0004;      /*  Clear receive-end flag (clearing condition: write 1) */
    while(RCANET0.RXPR0.WORD & 0x0004);     /*  Check flag                             */

    rcv_id = RCANET0.MB[2].CTRL0.WORD.H;
    rcv_id = ((rcv_id>>2)&0x07FF);
    switch(rcv_id){
        case ID_5:                          /*  Receive ID = ID_5 (0x233)              */
            for(i = 0;i < 8;i++){           /*  Store receive data in RAM              */
                MBbuff[5].DATA.BYTE[i] = RCANET0.MB[2].MSG_DATA[i];
            }
            break;
        case ID_6:                          /*  Receive ID = ID_6 (0x2CC)              */
            for(i = 0;i < 8;i++){           /*  Store receive data in RAM              */
                MBbuff[6].DATA.BYTE[i] = RCANET0.MB[2].MSG_DATA[i];
            }
            break;
        default:
            break;
    }
}
/*  MB3     */
else if(RCANET0.RXPR0.WORD&0x0008){         /*  Set IRR1, MB3 receive                  */
    RCANET0.RXPR0.WORD = 0x0008;      /*  Clear receive-end flag (clearing condition: write 1) */
    while(RCANET0.RXPR0.WORD & 0x0008);     /*  Check flag                             */

    rcv_id = RCANET0.MB[3].CTRL0.WORD.H;
    rcv_id = ((rcv_id>>2)&0x07FF);
```

```
        switch(rcv_id){
            case ID_7:                          /*  Receive ID = ID_7 (0x344)           */
                for(i = 0;i < 8;i++){           /*  Store receive data in RAM           */
                    MBbuff[7].DATA.BYTE[i] = RCANET0.MB[3].MSG_DATA[i];
                }
                break;
            case ID_8:                          /*  Receive ID = ID_8 (0x3DD)           */
                for(i = 0;i < 8;i++){           /*  Store receive data in RAM           */
                    MBbuff[8].DATA.BYTE[i] = RCANET0.MB[3].MSG_DATA[i];
                }
                break;
            default:
                break;
        }
    }
    /*  MB4     */
    else if(RCANET0.RXPR0.WORD&0x0010){         /*  Set IRR1, MB4 receive               */
        RCANET0.RXPR0.WORD = 0x0010;      /*  Clear receive-end flag (clearing condition: write 1) */
        while(RCANET0.RXPR0.WORD & 0x0010);     /*  Check flag                          */

        rcv_id = RCANET0.MB[4].CTRL0.WORD.H;
        rcv_id = ((rcv_id>>2)&0x07FF);
        switch(rcv_id){
            case ID_9:                          /*  Receive ID = ID_9 (0x455)           */
                for(i = 0;i < 8;i++){           /*  Store receive data in RAM           */
                    MBbuff[9].DATA.BYTE[i] = RCANET0.MB[4].MSG_DATA[i];
                }
                break;
            case ID_10:                         /*  Receive ID = ID_10 (0x4EE)          */
                for(i = 0;i < 8;i++){           /*  Store receive data in RAM           */
                    MBbuff[10].DATA.BYTE[i] = RCANET0.MB[4].MSG_DATA[i];
                }
                break;
            default:
                break;
        }
    }
    /*  MB5     */
    else if(RCANET0.RXPR0.WORD&0x0020){         /*  Set IRR1, MB5 receive               */
        RCANET0.RXPR0.WORD = 0x0020;      /*  Clear receive-end flag (clearing condition: write 1) */
        while(RCANET0.RXPR0.WORD & 0x0020);     /*  Check flag                          */

        rcv_id = RCANET0.MB[5].CTRL0.WORD.H;
        rcv_id = ((rcv_id>>2)&0x07FF);
        switch(rcv_id){
            case ID_11:                         /*  Receive ID = ID_11 (0x566)          */
                for(i = 0;i < 8;i++){           /*  Store receive data in RAM           */
                    MBbuff[11].DATA.BYTE[i] = RCANET0.MB[5].MSG_DATA[i];
                }
                break;
```

```
            case ID_12:                         /*  Receive ID = ID_12 (0x5FF)          */
                for(i = 0;i < 8;i++){           /*  Store receive data in RAM           */
                    MBbuff[12].DATA.BYTE[i] = RCANET0.MB[5].MSG_DATA[i];
                }
                break;
            default:
                break;
        }
    }
    /*  MB6     */
    else if(RCANET0.RXPR0.WORD&0x0040){         /*  Set IRR1, MB6 receive               */
        RCANET0.RXPR0.WORD = 0x0040;      /*  Clear receive-end flag (clearing condition: write 1) */
        while(RCANET0.RXPR0.WORD & 0x0040);     /*  Check flag                          */

        rcv_id = RCANET0.MB[6].CTRL0.WORD.H;
        rcv_id = ((rcv_id>>2)&0x07FF);
        switch(rcv_id){
            case ID_13:                         /*  Receive ID = ID_13 (0x677)          */
                for(i = 0;i < 8;i++){           /*  Store receive data in RAM           */
                    MBbuff[13].DATA.BYTE[i] = RCANET0.MB[6].MSG_DATA[i];
                }
                break;
            case ID_14:                         /*  Receive ID = ID_14 (0x6EE)          */
                for(i = 0;i < 8;i++){           /*  Store receive data in RAM           */
                    MBbuff[14].DATA.BYTE[i] = RCANET0.MB[6].MSG_DATA[i];
                }
                break;
            default:
                break;
        }
    }
    /*  MB7     */
    else if(RCANET0.RXPR0.WORD&0x0080){         /*  Set IRR1, MB7 receive               */
        RCANET0.RXPR0.WORD = 0x0080;      /*  Clear receive-end flag (clearing condition: write 1) */
        while(RCANET0.RXPR0.WORD & 0x0080);     /*  Check flag                          */

        rcv_id = RCANET0.MB[7].CTRL0.WORD.H;
        rcv_id = ((rcv_id>>2)&0x07FF);
        switch(rcv_id){
            case ID_15:                         /*  Receive ID = ID_15 (0x788)          */
                for(i = 0;i < 8;i++){           /*  Store receive data in RAM           */
                    MBbuff[15].DATA.BYTE[i] = RCANET0.MB[7].MSG_DATA[i];
                }
                break;
            default:
                break;
        }
    }
}
/******************************************************************************/
```

2.8 Message Filtering (LAFM) and IDE Bit Mask

2.8.1 Specification

Two standard format (STD_format) messages and two extended format (EXT_format) messages are transmitted by node A, as shown in figure 2.8.1. Node B uses the receive message filtering function (LAFM) to set the IDE bit mask and receives all the messages into a single mailbox.

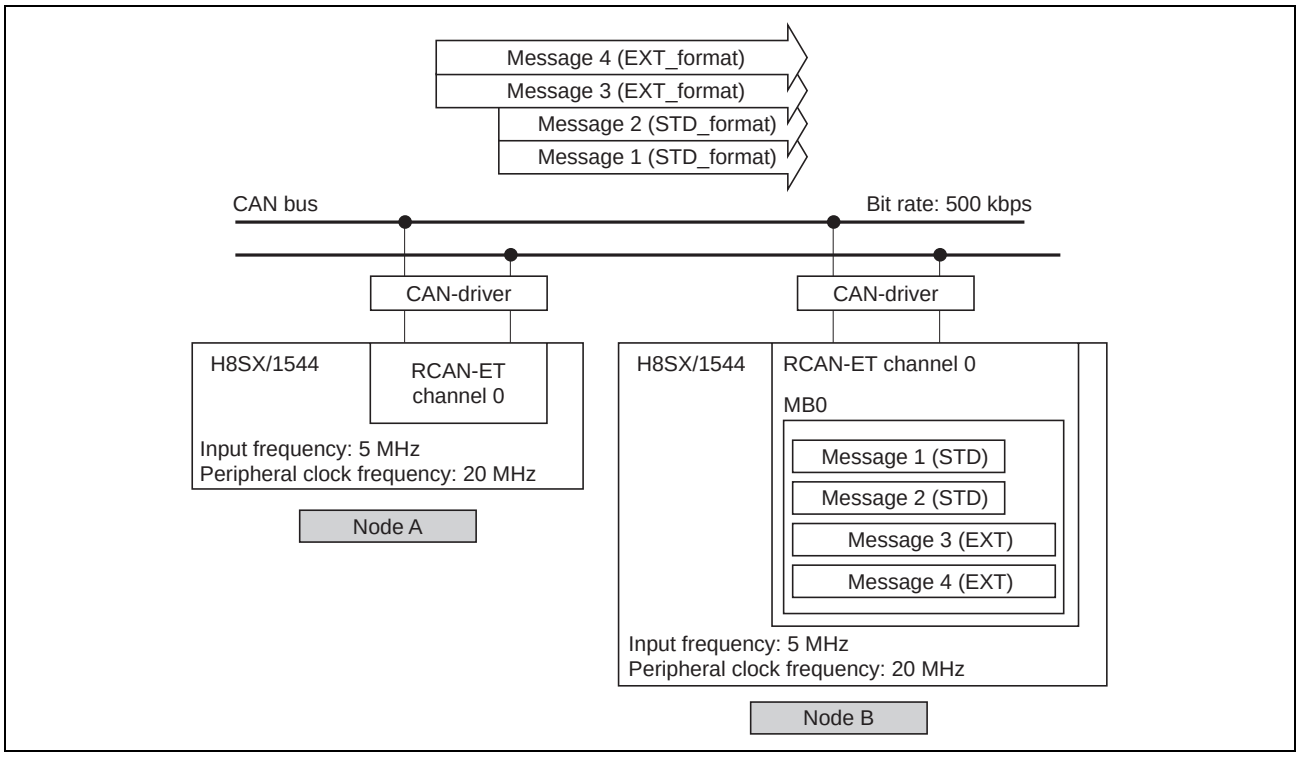


Figure 2.8.1 Communication Specification

Table 2.8.1 lists the message specifications.

Table 2.8.1 Message Specifications

Message	Format	Standard ID	Extended ID	Data
Message 1	Standard	H'0AA	—	H'01 01 01 01 01 01 01 01 (8 bytes)
Message 2	Standard	H'1BB	—	H'02 02 02 02 02 02 02 02 (8 bytes)
Message 3	Extended	H'2CC	H'2AAAA	H'03 03 03 03 03 03 03 03 (8 bytes)
Message 4	Extended	H'3DD	H'25555	H'04 04 04 04 04 04 04 04 (8 bytes)

2.8.2 Software Description

(1) Module Description

Table 2.8.2 Module Description

Module	Label	Function	Settings
H8SX/1544 initial settings	set_1544_init	See 2.1, Initial Settings.	Device-specific
H8SX/1544 interrupt settings	set_1544_INTC_init		
RCAN initial settings	set_RCAN0_init		RCAN-ET common
RCAN mailbox initial settings	set_RCAN0MB_init		
RCAN start	set_RCAN0_start		
CAN message transmit	RCAN0_Tx	Transmits CAN message.	
Mailbox empty interrupt	SLE0_0	Clears CAN message transmit-end flag.	
Data frame receive interrupt	RM0_0	Clears CAN message receive-end flag. Stores receive message in RAM.	
Receive buffer initialization	Clear_MBbuff	Initializes on-chip RAM area for storing CAN message.	—

(2) Description of Registers Used

(a) Transmitting Side

Table 2.8.3 Description of Registers Used (Transmitting Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFEFF	Enables mailbox empty interrupt (IRR8) (IMR8 = 0).
	RCANET0.MBIMR0.WORD	0xFFE1	Enables interrupts for mailboxes 1 to 4.
	RCANET0.MB[1].CTRL0.WORD.H	0x02A8	Sets mailbox 1 to standard format, data frame. Also sets standard ID (H'0AA).
	RCANET0.MB[1].CTRL1.BYTE.H	0x00	Sets mailbox 1 to transmit.
	RCANET0.MB[1].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[1].MSG_DATA[0] to RCANET0.MB[1].MSG_DATA[7]	0x01 × 8	Sets the transmit data.
	RCANET0.MB[2].CTRL0.WORD.H	0x06EC	Sets mailbox 2 to standard format, data frame. Also sets standard ID (H'1BB).
	RCANET0.MB[2].CTRL1.BYTE.H	0x00	Sets mailbox 2 to transmit.
	RCANET0.MB[2].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[2].MSG_DATA[0] to RCANET0.MB[2].MSG_DATA[7]	0x02 × 8	Sets the transmit data.
	RCANET0.MB[3].CTRL0.WORD.H	0x8B32	Sets mailbox 3 to extended format, data frame. Also sets standard ID (H'2CC) and extended ID (H'2AAAA).
	RCANET0.MB[3].CTRL0.WORD.L	0xAAAA	
	RCANET0.MB[3].CTRL1.BYTE.H	0x00	Sets mailbox 3 to transmit.
	RCANET0.MB[3].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[3].MSG_DATA[0] to RCANET0.MB[3].MSG_DATA[7]	0x03 × 8	Sets the transmit data.
	RCANET0.MB[4].CTRL0.WORD.H	0x8F76	Sets mailbox 4 to extended format, data frame. Also sets standard ID (H'3DD) and extended ID (H'25555).
	RCANET0.MB[4].CTRL0.WORD.L	0x5555	
	RCANET0.MB[4].CTRL1.BYTE.H	0x00	Sets mailbox 4 to transmit.
	RCANET0.MB4].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[4].MSG_DATA[0] to RCANET0.MB[4].MSG_DATA[7]	0x04 × 8	Sets the transmit data.
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
H8SX/1544 interrupt settings			
CAN message transmit	RCANET0.TXPR0.LONG	0x0000001E	Sets mailboxes 1 to 4 to transmit-wait status.
Mailbox empty interrupt	RCANET0.TXACK0.WORD	—	Clears transmit-end flag for mailboxes 1 to 4. (Clearing condition: write 1)

Note: The bit names listed in the table correspond to the bit names used in the hardware manual.

(b) Receiving Side

Table 2.8.4 Description of Registers Used (Receiving Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFFFD	Enables data frame receive interrupt (IRR1) (IMR1 = 0).
	RCANET0.MBIMR0.WORD	0xFFFE	Enables interrupts for mailbox 0.
	RCANET0.MB[0].CTRL0.WORD.H	0x0000	Sets mailbox 0 to standard format, data frame.
	RCANET0.MB[0].CTRL0.WORD.L	0x0000	
	RCANET0.MB[0].LAFM.WORD.H	0x9FFF	Disables all ID bits (standard ID, extended ID, IDE bit) for mailbox 0. Enables mailbox 0 to receive messages with any format and ID.
	RCANET0.MB[0].LAFM.WORD.L	0xFFFF	
	RCANET0.MB[0].CTRL1.BYTE.H	0x02	Sets mailbox 0 to receive.
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
H8SX/1544 interrupt settings			
Data frame receive interrupt	RCANET0.RXPR0.WORD	0x0001	Clears mailbox 0 receive-end flag. (Clearing condition: write 1)

Note: The bit names listed in the table correspond to the bit names used in the hardware manual.

(3) Description of RAM Used

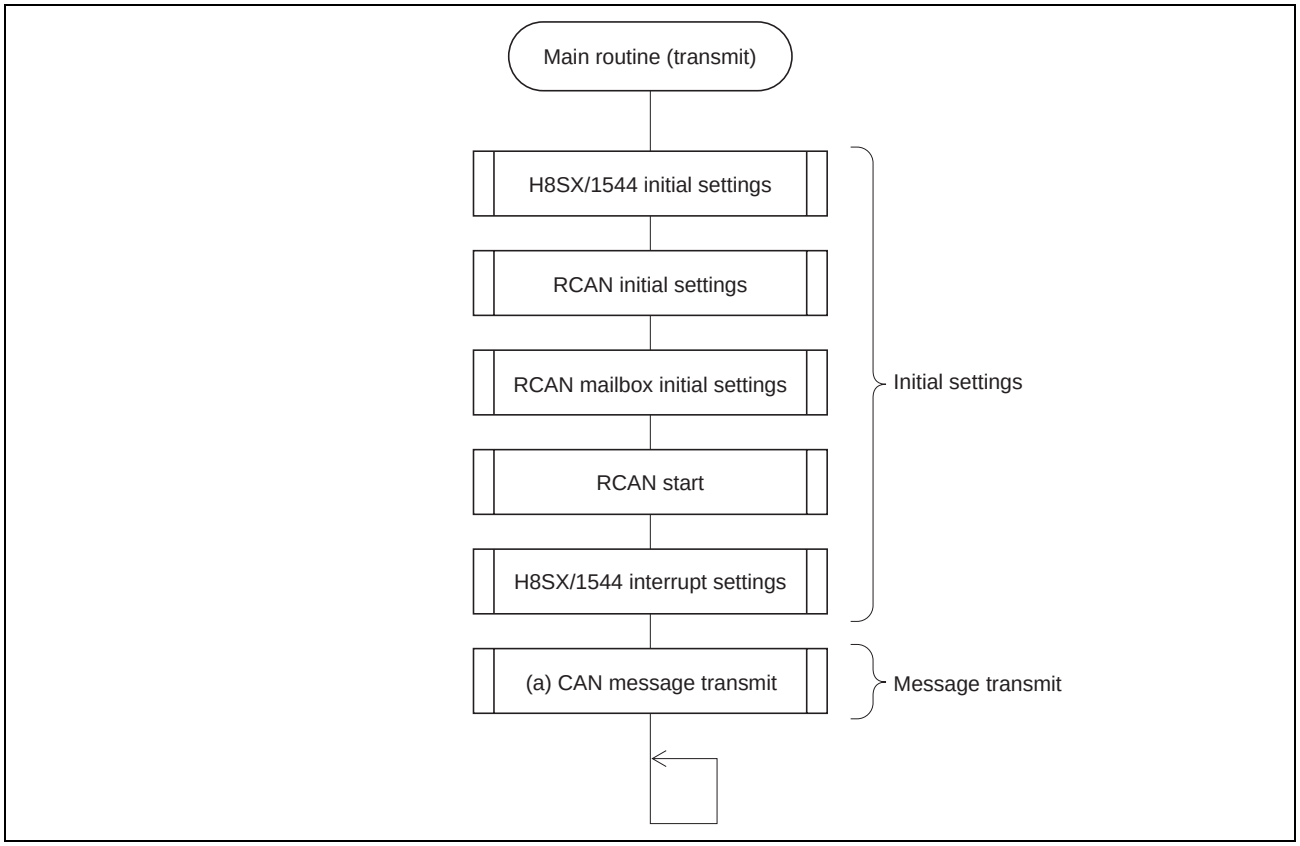
Table 2.8.5 Description of RAM Used

Module	Label	Function
Receive buffer initialization	Mbbuff[4].ID.LONG MBbuff[4].ID.WORD.H MBbuff[4].ID.WORD.L	Stores receive ID.
Data frame receive interrupt	MBbuff[4].DATA.LONG[0] to [1] MBbuff[4].DATA.WORD[0] to [3] MBbuff[4].DATA.BYTE[0] to [7]	Stores receive data.
Main routine Data frame receive interrupt	Rcv_cnt	Indicates number of receive messages.

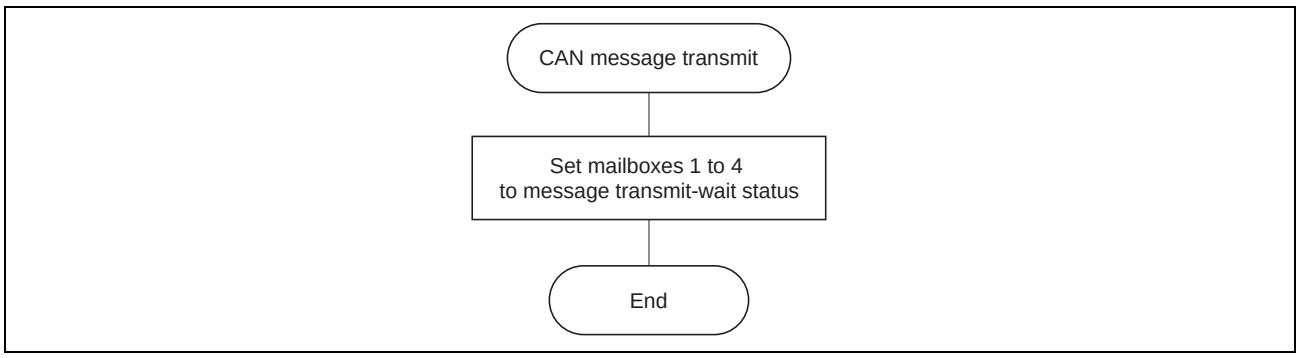
2.8.3 Flowcharts

(1) Transmitting Side Flowcharts

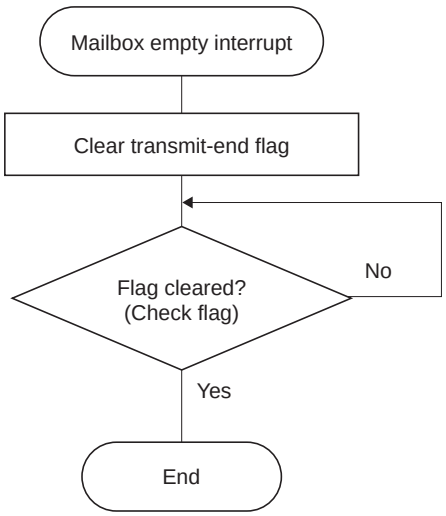
Main Routine



(a) CAN Message Transmit Routine (RCAN-ET Common Settings)



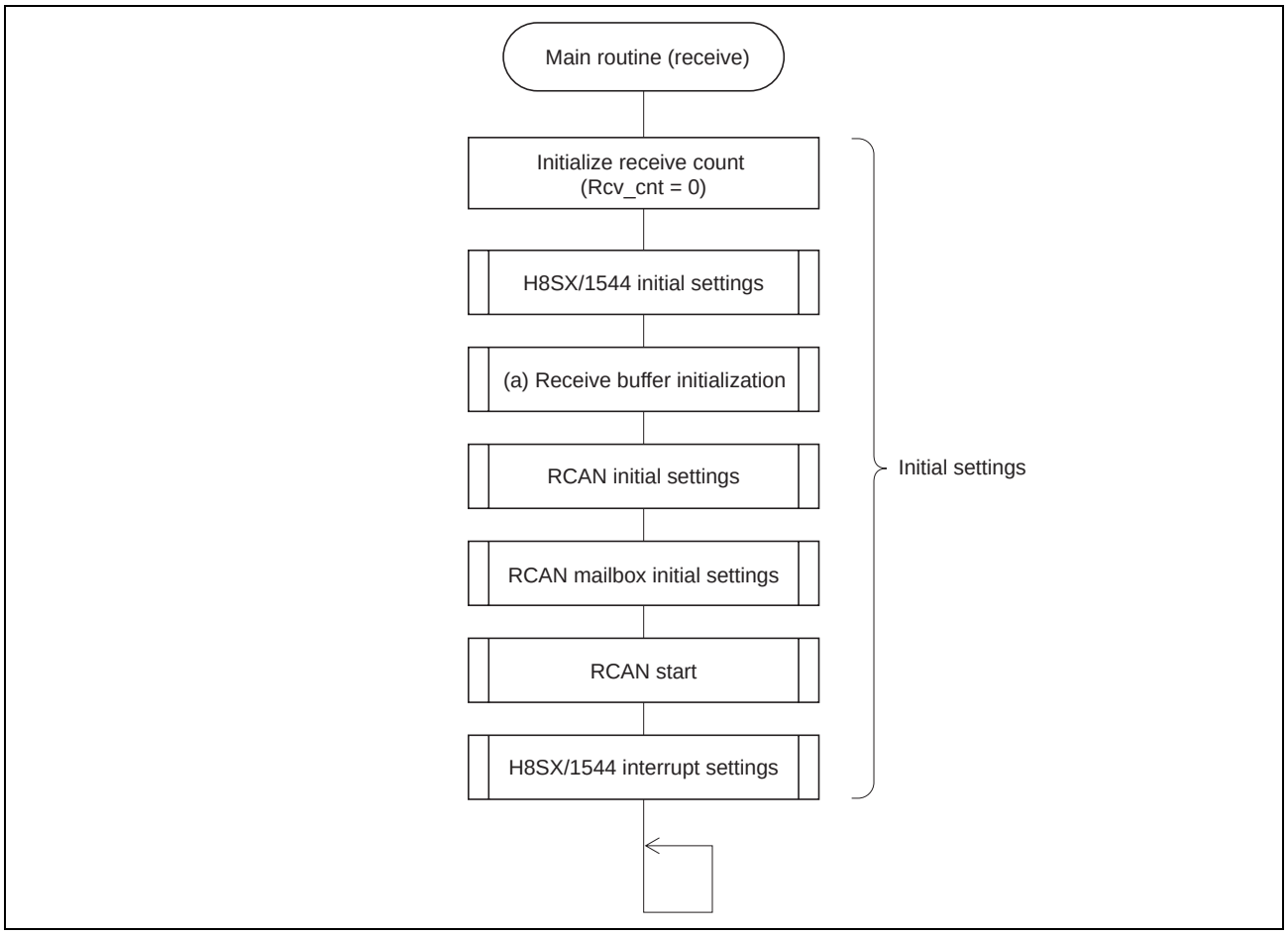
(b) Mailbox Empty Interrupt Routine (RCAN-ET Common Settings)



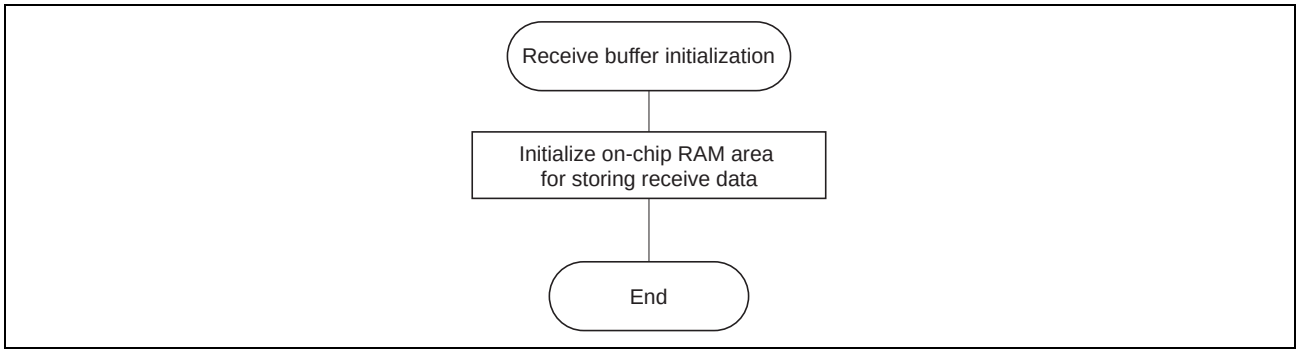
Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

(2) Receiving Side Flowcharts

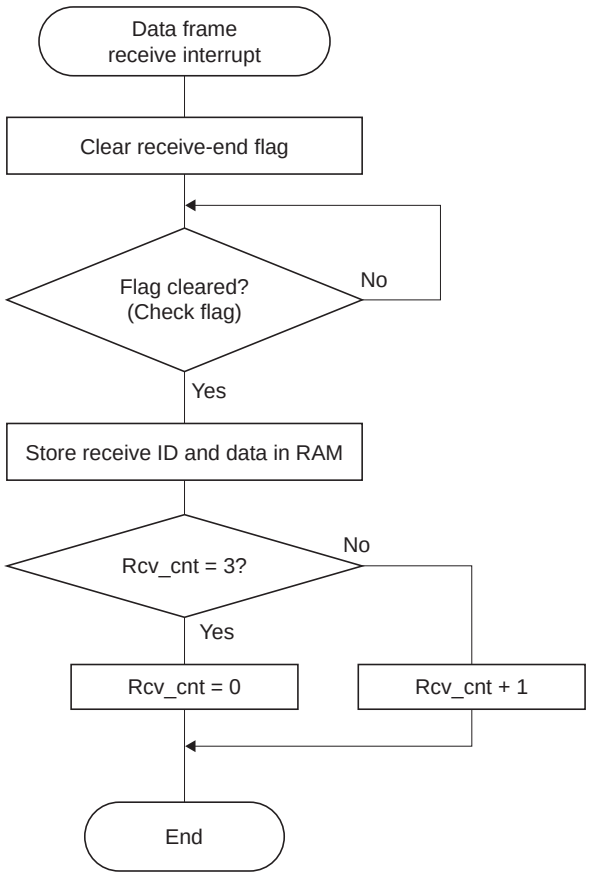
Main Routine



(a) Receive Buffer Initialization Routine (Device-Specific Settings)



(b) Data Frame Receive Interrupt Routine (RCAN-ET Common Settings)



Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

2.8.4 Program Listing

(1) Transmitting Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void RCAN0_Tx(void);
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{

    set_1544_init();                        /*  H8SX/1544 initialization     */
    set_RCAN0_init();                       /*  RCAN initialization          */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();                   /*  H8SX/1544 interrupt settings */
    RCAN0_Tx();                             /*  CAN message transmit         */
    while(1);

}
/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)      */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz)  */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz)*/
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode        */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN*/

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin) */

}
```

```
/*****************************************************************************/
/*      H8SX/1544 INTC Initialize routine                                    */
/*****************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;                /*   Interrupt control mode 2          */
    INTC.IPRQ.BIT._RCAN01 = 7;              /*   Set interrupt priority level: RCAN */
    set_imask_exr(0);                       /*   Interrupt mask level 0            */


}
/*****************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename    :  rcan.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void RCAN0_Tx(void);
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;         /*   Reset request (set automatically by a hardware reset)  */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)    */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)  */
    RCANET0.IRR.WORD = 0x0001;          /*   Clear IRR0 (clearing condition: write 1)               */
    RCANET0.MCR.WORD |= 0x8000;         /*   Reorder ID: Set to MCR15 = 1 (initial setting)         */

    for(i = 0;i < 16;i++){              /*   Initialize mailbox (RAM area)                          */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }
}

/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    unsigned int i;

    RCANET0.IMR.WORD &= 0xFEFF;             /*   Enable mailbox empty interrupt                 */
    RCANET0.MBIMR0.WORD = 0xFFE1;           /*   Enable MB1-4 interrupt                         */

    RCANET0.MB[1].CTRL0.WORD.H = 0x02A8;    /*   Set STDID: H'0AA, standard format, data frame  */
    RCANET0.MB[1].CTRL1.BYTE.H = 0x00;      /*   Set mailbox 1 to transmit                      */
    RCANET0.MB[1].CTRL1.BYTE.L = 0x08;      /*   Data length: 8 bytes                           */
    for(i = 0;i < 8;i++){
        RCANET0.MB[1].MSG_DATA[i] = 0x01;   /*   Transmit data                                  */
    }
```

```
    RCANET0.MB[2].CTRL0.WORD.H = 0x06EC;   /*   Set STDID: H'1BB, standard format, data frame */
    RCANET0.MB[2].CTRL1.BYTE.H = 0x00;     /*   Set mailbox 2 to transmit                     */
    RCANET0.MB[2].CTRL1.BYTE.L = 0x08;     /*   Data length: 8 bytes                          */
    for(i = 0;i < 8;i++){
        RCANET0.MB[2].MSG_DATA[i] = 0x02; /*   Transmit data                                 */
    }

    RCANET0.MB[3].CTRL0.WORD.H = 0x8B32;   /*   Set STDID: H'2CC, extended format, data frame */
    RCANET0.MB[3].CTRL0.WORD.L = 0xAAAA;   /*   EXTID:H'2AAAA                                 */
    RCANET0.MB[3].CTRL1.BYTE.H = 0x00;     /*   Set mailbox 3 to transmit                     */
    RCANET0.MB[3].CTRL1.BYTE.L = 0x08;     /*   Data length: 8 bytes                          */
    for(i = 0;i < 8;i++){
        RCANET0.MB[3].MSG_DATA[i] = 0x03; /*   Transmit data                                 */
    }

    RCANET0.MB[4].CTRL0.WORD.H = 0x8F76;   /*   Set STDID: H'3DD, extended format, data frame */
    RCANET0.MB[4].CTRL0.WORD.L = 0x5555;   /*   EXTID:H'25555                                 */
    RCANET0.MB[4].CTRL1.BYTE.H = 0x00;     /*   Set mailbox 4 to transmit                     */
    RCANET0.MB[4].CTRL1.BYTE.L = 0x08;     /*   Data length: 8 bytes                          */
    for(i = 0;i < 8;i++){
        RCANET0.MB[4].MSG_DATA[i] = 0x04; /*   Transmit data                                 */
    }

    /*  Bit rate = 500 kbps  */
    RCANET0.BCR1.WORD = 0x4300;/*  TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, (pϕ = 20 MHz) */
    RCANET0.BCR0.WORD = 0x0001;            /*   BRP = 1                                       */


}

/*****************************************************************************/
/*      RCAN start routine                                                   */
/*****************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                     /*   Clear MCR0                    */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000); /*   GSR3 = 0?                     */


}

/*****************************************************************************/
/*      RCAN send message routine                                            */
/*****************************************************************************/
void RCAN0_Tx(void){

    RCANET0.TXPR0.LONG = 0x0000001E;          /*   Set MB1-4 to transmit-wait status  */

}
```

```
/************************************************************************/
/*      Mailbox Empty Interrupt routine                                 */
/************************************************************************/
#pragma interrupt(SLE0_0)
void SLE0_0(void)
{
    RCANET0.TXACK0.WORD = RCANET0.TXACK0.WORD;
                                        /*  Clear transmit-end flag (clearing condition: write 1)  */
    while(RCANET0.TXACK0.WORD);         /*  Check flag                                             */

}
/************************************************************************/
```

(2) Receiving Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void Clear_MBbuff(void);
/*---------------------------------------------------------------------------*/
extern unsigned char Rcv_cnt;
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{
    Rcv_cnt = 0;                            /*  Initialize receive counter   */

    set_1544_init();                        /*  H8SX/1544 initialization     */
    Clear_MBbuff();                         /*  Initialize RAM area for storing receive data */
    set_RCAN0_init();                       /*  RCAN initialization          */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();                   /*  H8SX/1544 interrupt settings */
    while(1);

}
/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)  */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz)  */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz)  */
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode        */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN  */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins  */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin)  */
}
```

```
/*****************************************************************************/
/*      H8SX/1544 INTC Initialize routine                                    */
/*****************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;                /*   Interrupt control mode 2          */
    INTC.IPRQ.BIT._RCAN01 = 7;              /*   Set interrupt priority level: RCAN */
    set_imask_exr(0);                       /*   Interrupt mask level 0            */


}
/*****************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename    :  rcan.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void Clear_MBbuff(void);
/*---------------------------------------------------------------------------*/
struct {
    union {
        unsigned long LONG;
        struct{
            unsigned short H;
            unsigned short L;
        }WORD;
    }ID;
    union {
        unsigned char BYTE[8];
        unsigned short WORD[4];
        unsigned long LONG[2];
    }DATA;
} MBbuff[4];
/*---------------------------------------------------------------------------*/
unsigned char Rcv_cnt;
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*  Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)    */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)  */
    RCANET0.IRR.WORD = 0x0001;           /*  Clear IRR0 (clearing condition: write 1)            */
    RCANET0.MCR.WORD |= 0x8000;          /*  Reorder ID: Set to MCR15 = 1 (initial setting)      */

    for(i = 0;i < 16;i++){               /*  Initialize mailbox (RAM area)                       */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }
}
```

```
/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    RCANET0.IMR.WORD &= 0xFFFD;                 /*  Enable data frame receive interrupt     */
    RCANET0.MBIMR0.WORD &= 0xFFFE;              /*  Enable MB0 interrupt                    */

    RCANET0.MB[0].CTRL0.WORD.H = 0x0000;        /*  STD_ID, data frame                      */
    RCANET0.MB[0].CTRL0.WORD.L = 0x0000;        /*  EXT_ID                                  */
    RCANET0.MB[0].LAFM.WORD.H = 0x9FFF;         /*  STD_LAFM and IDE_LAFM settings          */
    RCANET0.MB[0].LAFM.WORD.L = 0xFFFF;         /*  EXT_LAFM settings                       */
    RCANET0.MB[0].CTRL1.BYTE.H = 0x02;          /*  Set mailbox 0 to receive                */

    /*  Bit rate = 500 kbps  */
    RCANET0.BCR1.WORD = 0x4300;/*  TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, (pϕ = 20 MHz)*/
    RCANET0.BCR0.WORD = 0x0001;                 /*  BRP = 1                                 */


}

/*****************************************************************************/
/*      RCAN start routine                                                   */
/*****************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                     /*  Clear MCR0                  */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000); /*  GSR3 = 0?                   */


}


/*****************************************************************************/
/*      RAM area Initialize routine                                          */
/*****************************************************************************/
void Clear_MBbuff(void){

    unsigned int i;

    for(i = 0;i < 4;i++){                       /*  Initialize RAM area for storing receive data*/
        MBbuff[i].ID.LONG = 0;
        MBbuff[i].DATA.LONG[0] = 0;
        MBbuff[i].DATA.LONG[1] = 0;
    }

}
```

```
/*****************************************************************************/
/*      Data Frame Received Interrupt routine(MailBox0)                      */
/*****************************************************************************/
#pragma interrupt(RM0_0)
void RM0_0(void)
{
    unsigned int i;

    RCANET0.RXPR0.WORD = 0x0001;          /*  Clear receive-end flag (clearing condition: write 1) */
    while(RCANET0.RXPR0.WORD & 0x0001);   /*  Check flag                                           */

    MBbuff[Rcv_cnt].ID.WORD.H = RCANET0.MB[0].CTRL0.WORD.H;/*  Store receive ID in RAM */
    MBbuff[Rcv_cnt].ID.WORD.L = RCANET0.MB[0].CTRL0.WORD.L;/*  Store receive ID in RAM */
    for(i = 0;i < 8;i++){                 /*  Store receive data in RAM                            */
        MBbuff[Rcv_cnt].DATA.BYTE[i] = RCANET0.MB[0].MSG_DATA[i];
    }

    if(Rcv_cnt == 3){
        Rcv_cnt = 0;
    }
    else{
        Rcv_cnt++;
    }


}
/*****************************************************************************/
```

2.9 CAN Wake-up Function

2.9.1 Specification

As shown in figure 2.9.1, after initial settings are performed for node A and node B, node B only transitions to software standby mode.

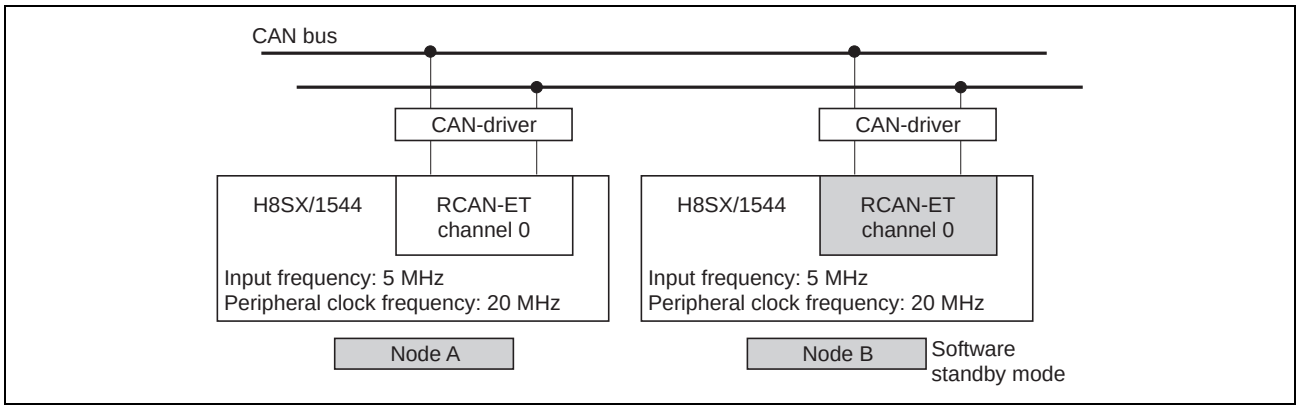


Figure 2.9.1 Communication Specification (1)

Next, a data frame with standard ID H'555 is transmitted by node A, as shown in figure 2.9.2. When port 64 (combined CRx_0/IRQ12) is detected low-level, an interrupt is generated and node B returns from software standby mode. After the return, RCAN-ET initial settings are performed and the data frame is received.

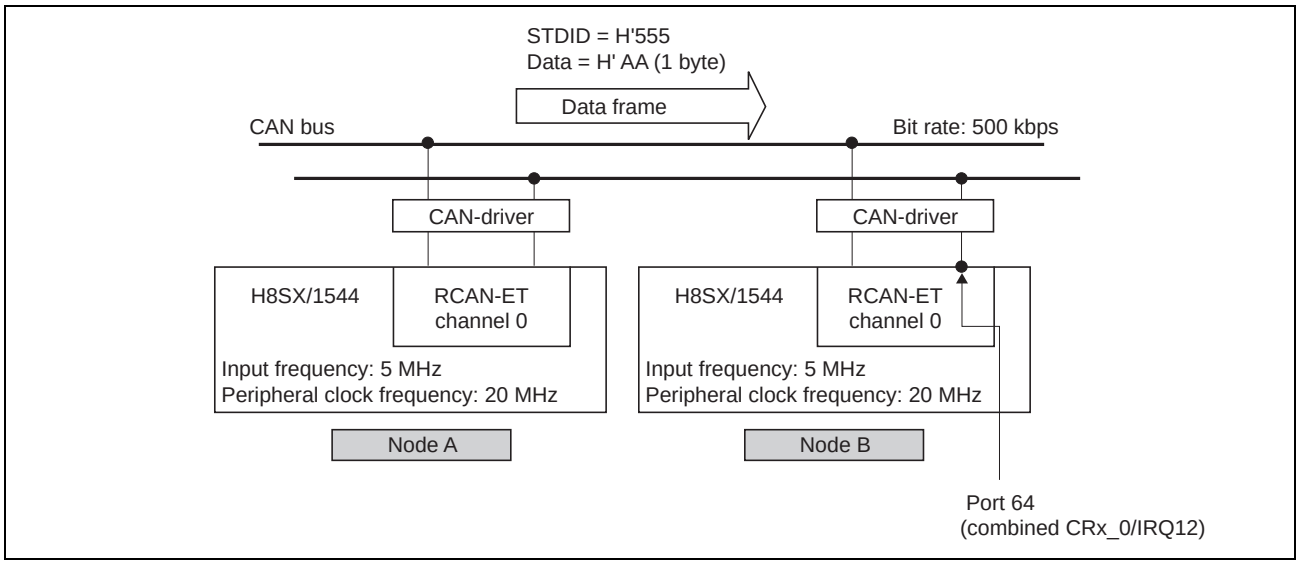


Figure 2.9.2 Communication Specification (2)

Note: In software standby mode RCAN-ET enters reset status. It is therefore necessary to perform RCAN-ET initial settings after returning from software standby mode. See section 24, Power-Down Modes, in the H8SX/1544 Hardware Manual for information on the operating status of the CPU and peripheral modules in software standby mode.

2.9.2 Software Description

(1) Module Description

Table 2.9.1 Module Description

Module	Label	Function	Settings
H8SX/1544 initial settings	set_1544_init	See 2.1, Initial Settings.	Device-specific
H8SX/1544 interrupt settings	set_1544_INTC_init	Performs RCAN-ET interrupt priority level setting for the H8SX/1544. Sets software standby cancel interrupt.	
H8SX/1544 I/O pins settings	set_1544_IO_init	Sets the RCAN-ET transmit and receive pins on the H8SX/1544.	
IRQ Interrupt	IRQ12	Performs I/O and RCAN-ET initial settings for the H8SX/1544.	
RCAN initial settings	set_RCAN0_init	See 2.1, Initial Settings.	RCAN-ET common
RCAN mailbox initial settings	set_RCAN0MB_init		
RCAN start	set_RCAN0_start		
CAN message transmit	RCAN0_Tx	Transmits CAN message.	
Mailbox empty interrupt	SLE0_0	Clears CAN message transmit-end flag.	
Data frame receive interrupt	RM0_0	Clears CAN message receive-end flag. Stores receive message in RAM.	
Receive buffer initialization	Clear_MBbuff	Initializes on-chip RAM area for storing CAN message.	—

(2) Description of Registers Used

(a) Transmitting Side

Table 2.9.2 Description of Registers Used (Transmitting Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFEFF	Enables mailbox empty interrupt (IRR8) (IMR8 = 0).
	RCANET0.MBIMR0.WORD	0xFFFD	Enables mailbox 1 interrupt.
	RCANET0.MB[1].CTRL0.WORD.H	0x1554	Sets mailbox 1 to standard format, data frame. Also sets standard ID (H'555).
	RCANET0.MB[1].CTRL1.BYTE.H	0x00	Sets mailbox 1 to transmit.
	RCANET0.MB[1].CTRL1.BYTE.L	0x01	Sets the data length (1 byte).
	RCANET0.MB[1].MSG_DATA[0]	0xAA	Sets the transmit data.
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
H8SX/1544 interrupt settings			
CAN message transmit	RCANET0.TXPR0.LONG	0x00000002	Sets mailbox 1 to transmit-wait status.
Mailbox empty interrupt	RCANET0.TXACK0.WORD	0x0002	Clears mailbox 1 transmit-end flag. (Clearing condition: write 1)

Note: The bit names listed in the table correspond to the bit names used in the hardware manual.

(b) Receiving Side

Table 2.9.3 Description of Registers Used (Receiving Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFFFD	Enables data frame receive interrupt (IRR1) (IMR1 = 0).
	RCANET0.MBIMR0.WORD	0xFFFE	Enables interrupts for mailbox 0.
	RCANET0.MB[0].CTRL0.WORD.H	0x1554	Sets mailbox 0 to standard format, data frame. Also sets standard ID (H'555).
	RCANET0.MB[0].LAFM.WORD.H	0x0000	Set filter mask for standard ID and IDE bit of mailbox 0.
	RCANET0.MB[0].CTRL1.BYTE.H	0x02	Sets mailbox 0 to receive.
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
H8SX/1544 interrupt settings	INTC.INTCR.BIT.INTM	2	Sets interrupt controller to interrupt control mode 2.
	INTC.IPRD.BIT._IRQ12	7	Sets IRQ12 interrupt priority level.
	INTC.IPRQ.BIT._RCAN01	6	Sets the RCAN-ET interrupt priority level.
	INTC.SSIER.BIT.SSI12	1	Sets software standby cancel interrupt. When IRQ12 occurs in software standby mode, software standby is cancelled after the oscillation stabilization time elapses.
	INTC.IER.BIT.IRQ12E	1	Enables IRQ12 interrupt requests.
IRQ Interrupt	INTC.IER.BIT.IRQ12E	0	Disables IRQ12 interrupt requests.
H8SX/1544 I/O pins settings	RCANET0.RCANMON.BYTE	0x20	Enables RCAN-ET transmit and receive pins.
Data frame receive interrupt	P6.ICR.BIT.B4	1	Sets P64 (pin 67) as RCAN-ET channel 0 receive pin.
	RCANET0.RXPR0.WORD	0x0001	Clears mailbox 0 receive-end flag. (Clearing condition: write 1)

Note: The bit names listed in the table correspond to the bit names used in the hardware manual.

(3) Description of RAM Used

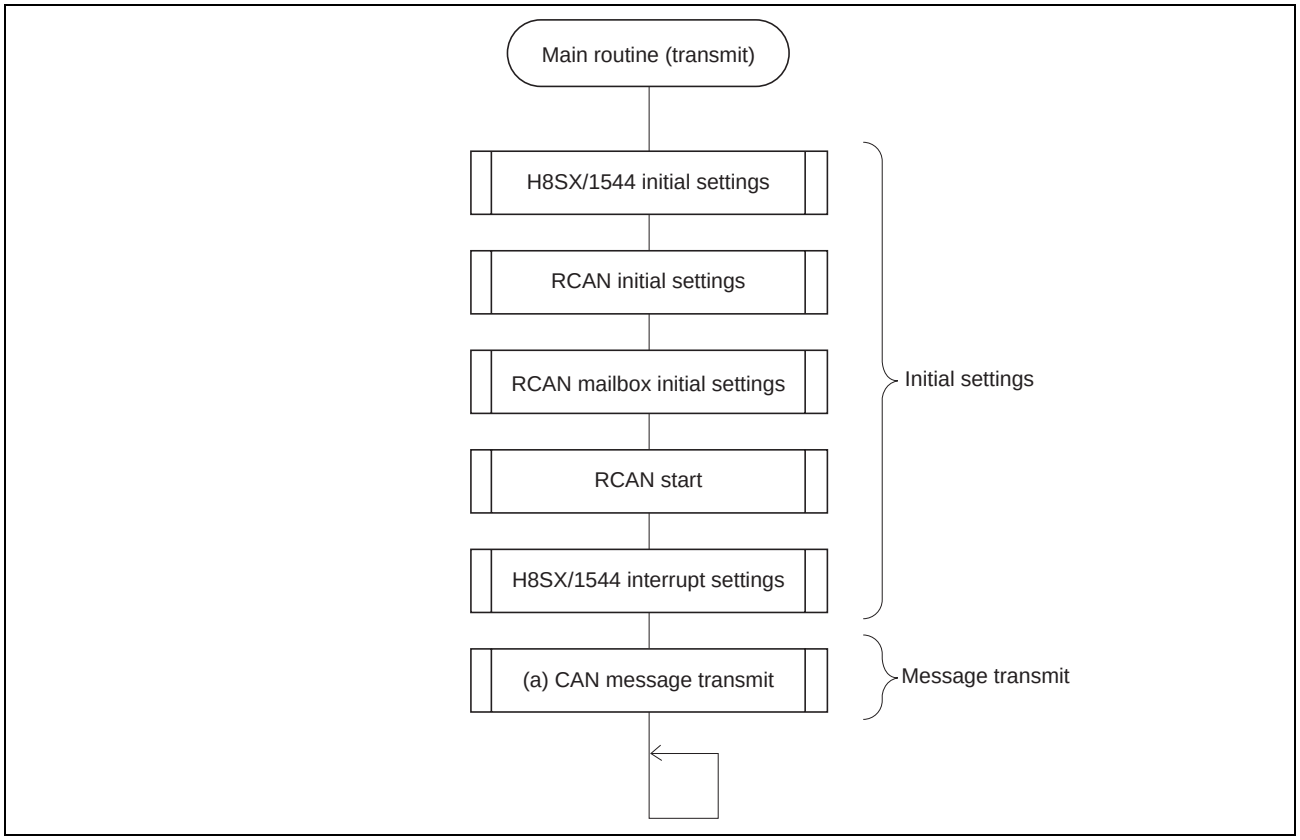
Table 2.9.4 Description of RAM Used

Module	Label	Function
Receive buffer initialization	Mbbuff.ID.LONG MBbuff.ID.WORD.H MBbuff.ID.WORD.L	Stores receive ID.
Data frame receive interrupt	MBbuff.DATA.LONG[0] to [1] MBbuff.DATA.WORD[0] to [3] MBbuff.DATA.BYTE[0] to [7]	Stores receive data.

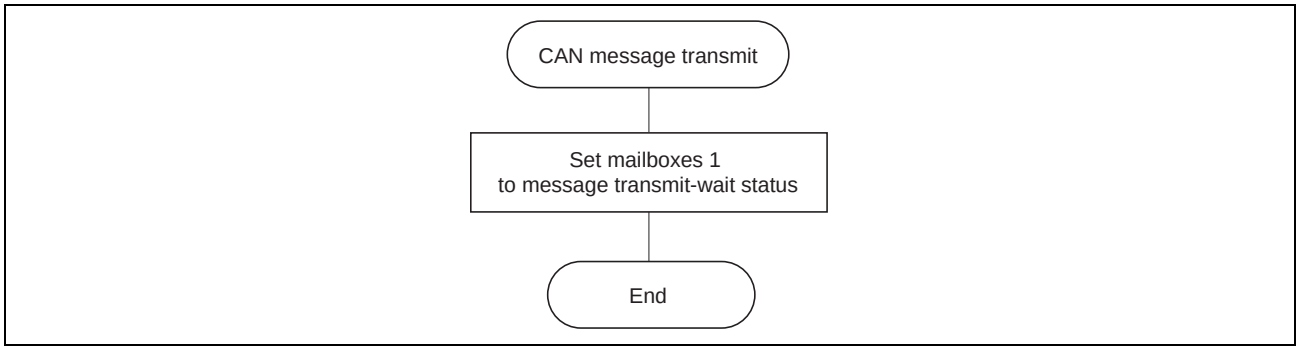
2.9.3 Flowcharts

(1) Transmitting Side Flowcharts

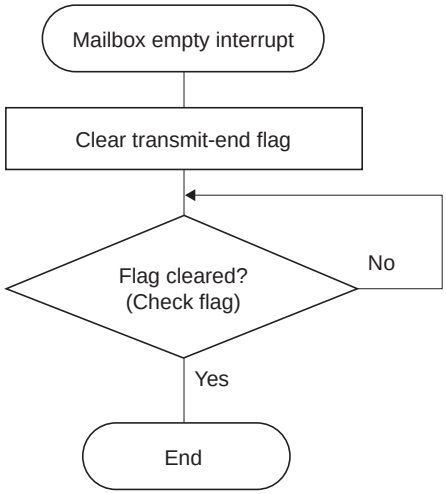
Main Routine



(a) CAN Message Transmit Routine (RCAN-ET Common Settings)



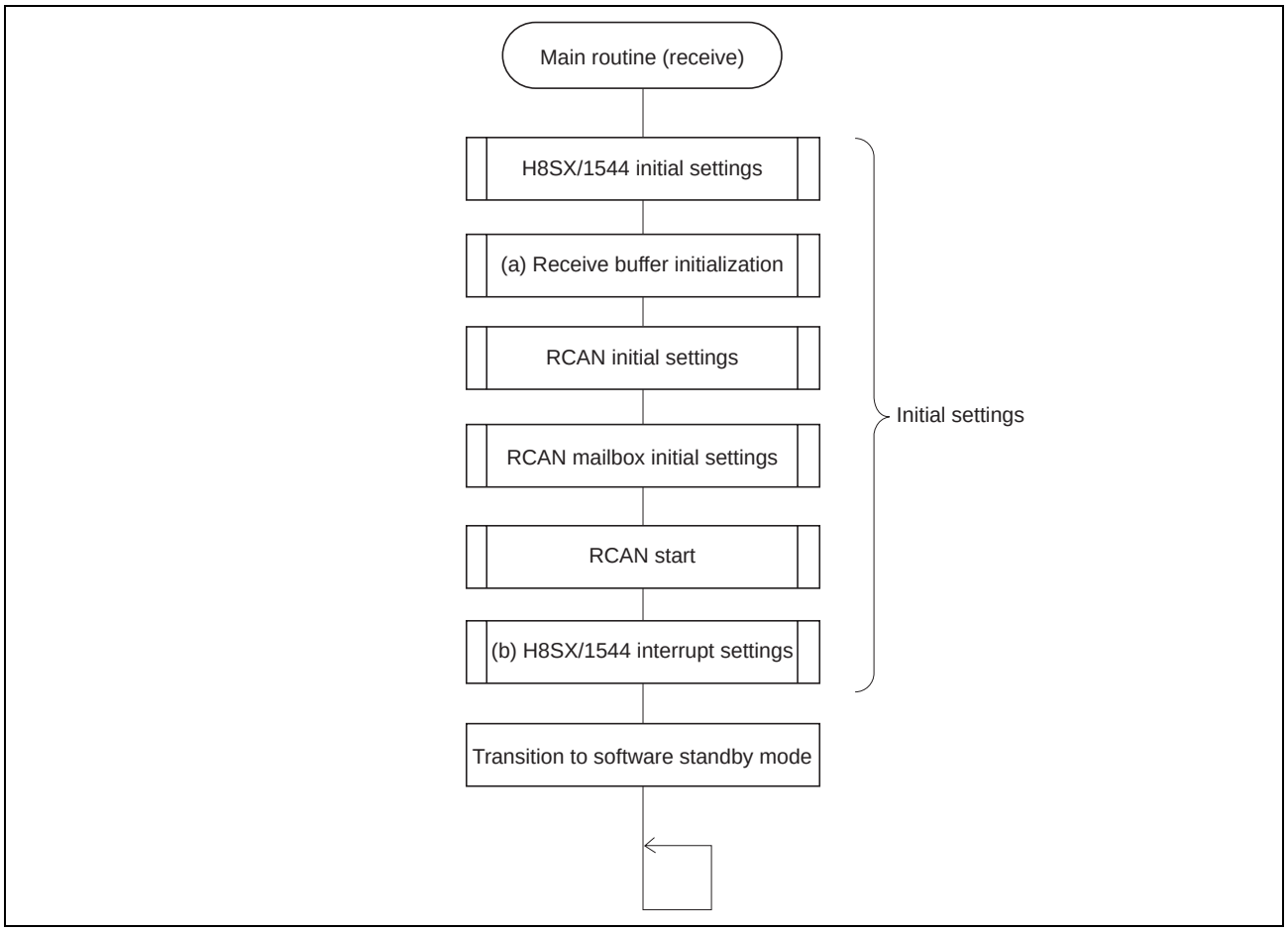
(b) Mailbox Empty Interrupt Routine (RCAN-ET Common Settings)



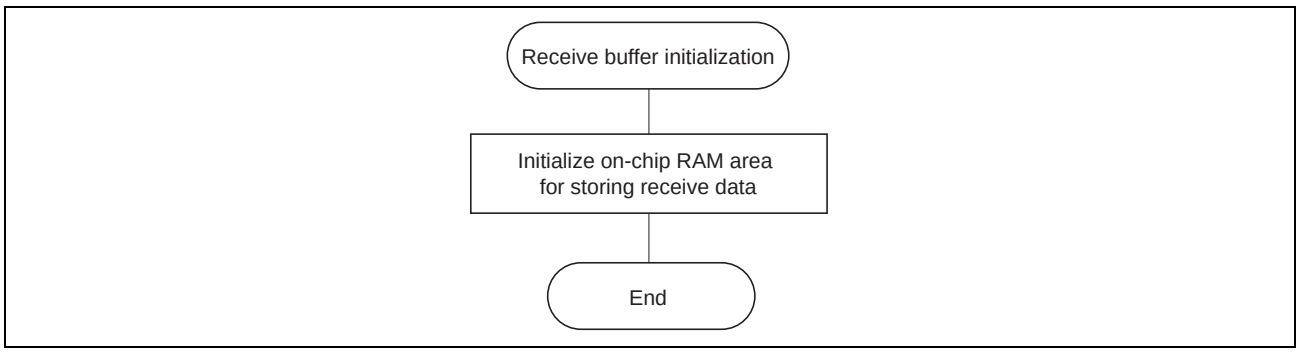
Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

(2) Receiving Side Flowcharts

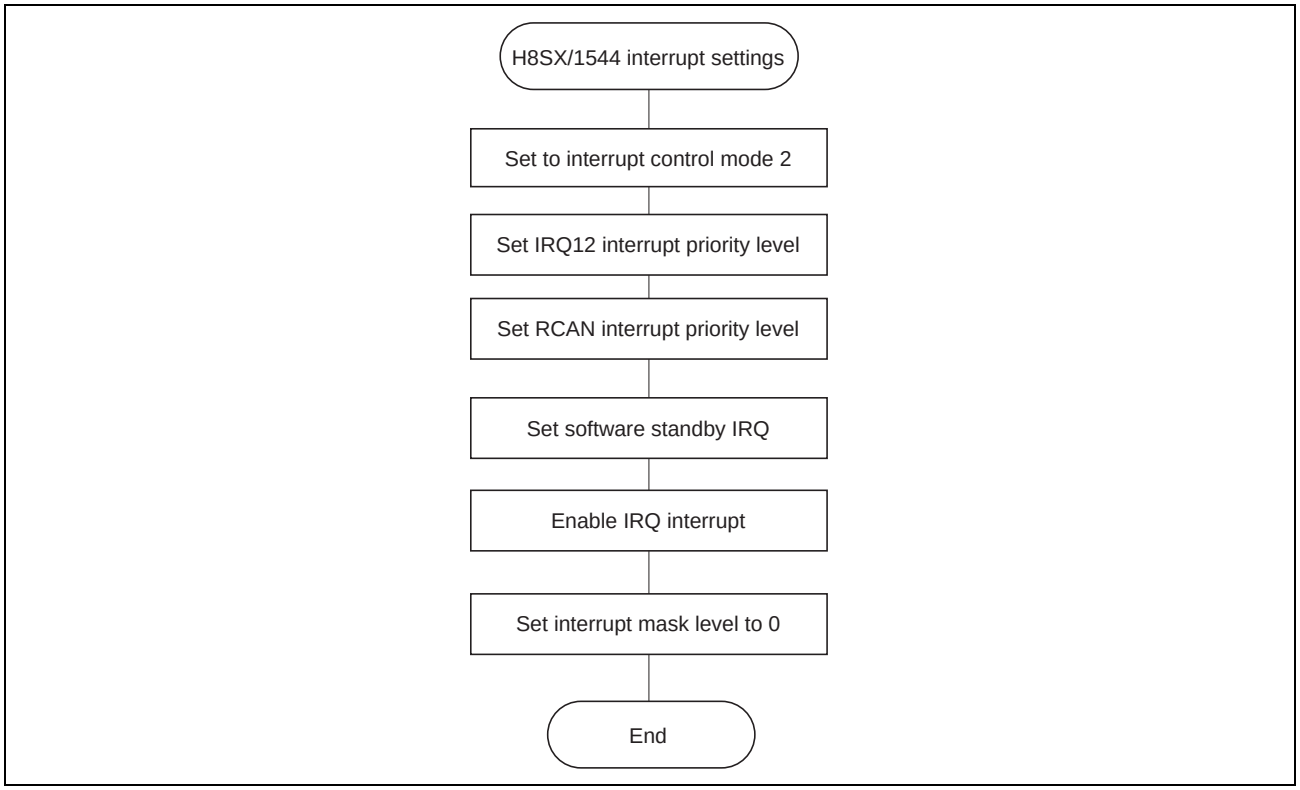
Main Routine



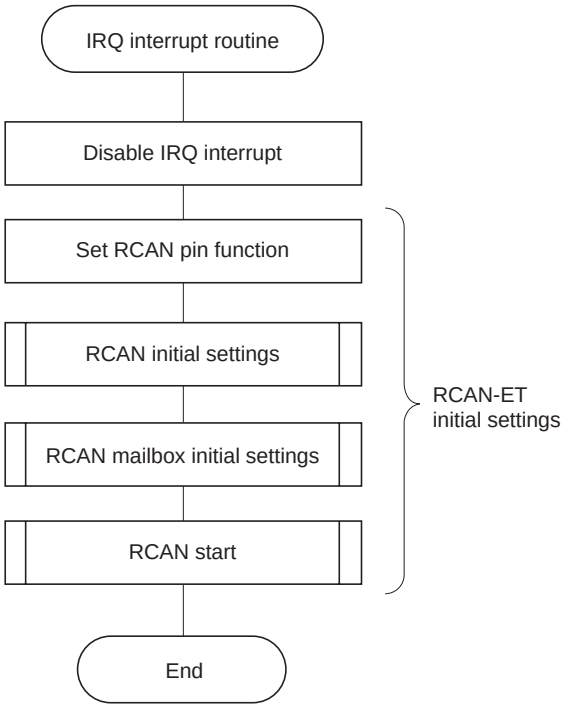
(a) Receive Buffer Initialization Routine (Device-Specific Settings)



(b) H8SX/1544 Interrupt Settings (Device-Specific Settings)

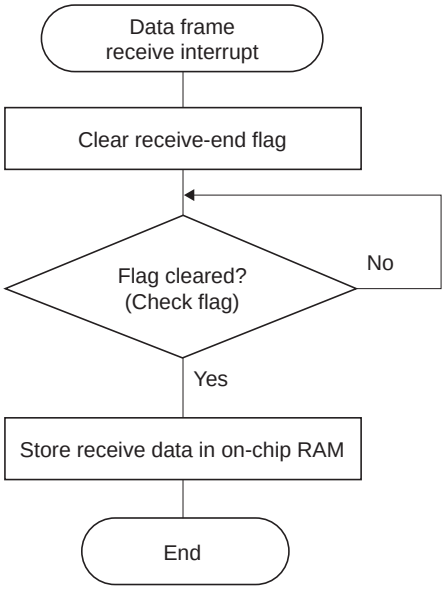


(c) IRQ Interrupt Routine (Device-Specific Settings)



Note: After IRQ interrupt generation, CAN message transmit or receive cannot be performed until RCAN initial settings are completed.

(d) Data Frame Receive Interrupt Routine (RCAN-ET Common Settings)



Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

2.9.4 Program Listing

(1) Transmitting Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void RCAN0_Tx(void);
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{
    set_1544_init();                        /*  H8SX/1544 initialization        */
    set_RCAN0_init();                       /*  RCAN initialization             */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();                   /*  H8SX/1544 interrupt settings    */
    RCAN0_Tx();                             /*  CAN message transmit            */
    while(1);

}
/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)       */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz)   */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz) */
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode           */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN   */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins    */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin)    */

}
```

```
/*****************************************************************************/
/*      H8SX/1544 INTC Initialize routine                                    */
/*****************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;                /*   Interrupt control mode 2          */
    INTC.IPRQ.BIT._RCAN01 = 7;              /*   Set interrupt priority level: RCAN */
    set_imask_exr(0);                       /*   Interrupt mask level 0            */

}
/*****************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename    :  rcan.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void RCAN0_Tx(void);
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*  Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)    */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)  */
    RCANET0.IRR.WORD = 0x0001;           /*  Clear IRR0 (clearing condition: write 1)    */
    RCANET0.MCR.WORD |= 0x8000;          /*  Reorder ID: Set to MCR15 = 1 (initial setting)  */

    for(i = 0;i < 16;i++){               /*  Initialize mailbox (RAM area)               */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }
}

/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    unsigned int i;

    RCANET0.IMR.WORD &= 0xFEFF;          /*  Enable mailbox empty interrupt              */
    RCANET0.MBIMR0.WORD &= 0xFFFD;       /*  Enable MB1 interrupt                        */

    RCANET0.MB[1].CTRL0.WORD.H = 0x1554; /*  Set STDID: H'555, standard format, data frame   */
    RCANET0.MB[1].CTRL1.BYTE.H = 0x00;   /*  Set mailbox 1 to transmit                   */
    RCANET0.MB[1].CTRL1.BYTE.L = 0x01;   /*  Data length: 1 byte                         */
    RCANET0.MB[1].MSG_DATA[0] = 0xAA;    /*  Transmit data: 0xAA                         */
```

```
    /*  Bit rate = 500 kbps  */
    RCANET0.BCR1.WORD = 0x4300;/*  TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, (pϕ = 20 MHz) */
    RCANET0.BCR0.WORD = 0x0001;            /*   BRP = 1                                  */


}

/*****************************************************************************/
/*      RCAN start routine                                                   */
/*****************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                          /*   Clear MCR0          */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000);       /*   GSR3 = 0?           */


}


/*****************************************************************************/
/*      RCAN send message routine                                            */
/*****************************************************************************/
void RCAN0_Tx(void){

    RCANET0.TXPR0.LONG = 0x00000002;      /*   Set MB1 to transmit-wait status     */


}

/*****************************************************************************/
/*      Mailbox Empty Interrupt routine                                      */
/*****************************************************************************/
#pragma interrupt(SLE0_0)
void SLE0_0(void)
{
    RCANET0.TXACK0.WORD = 0x0002;         /*   Clear transmit-end flag (clearing condition: write 1) */
    while(RCANET0.TXACK0.WORD & 0x0002); /*   Check flag                           */

}
/*****************************************************************************/
```

(2) Receiving Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_IO_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void Clear_MBbuff(void);
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{
    set_1544_init();                        /*  H8SX/1544 initialization                    */
    Clear_MBbuff();                         /*  Initialize RAM area for storing receive data */
    set_RCAN0_init();                       /*  RCAN initialization                         */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();                   /*  H8SX/1544 interrupt settings                */
    sleep();                                /*  Transition to software standby mode         */
    while(1);

}

/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)       */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz)   */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz) */
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode                       */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN               */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins    */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin)                */

}
```

```
/*****************************************************************************/
/*      H8SX/1544 I/O Initialize routine                                     */
/*****************************************************************************/
void set_1544_IO_init(void){

    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins  */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin)              */


}


/*****************************************************************************/
/*      H8SX/1544 INTC Initialize routine                                    */
/*****************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;                /*  Interrupt control mode 2                  */
    INTC.IPRD.BIT._IRQ12 = 7;               /*  Set interrupt priority level: IRQ12       */
    INTC.IPRQ.BIT._RCAN01 = 6;              /*  Set interrupt priority level: RCAN        */
    INTC.SSIER.BIT.SSI12 = 1;               /*  Set software standby cancel IRQ: IRQ12    */
    INTC.IER.BIT.IRQ12E = 1;                /*  Enable IRQ interrupt: IRQ12               */
    set_imask_exr(0);                       /*  Interrupt mask level 0                    */

}
/*****************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename    :  rcan.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void Clear_MBbuff(void);
extern void set_1544_IO_init(void);
/*---------------------------------------------------------------------------*/
struct {
    union {
        unsigned long LONG;
        struct{
            unsigned short H;
            unsigned short L;
        }WORD;
    }ID;
    union {
        unsigned char BYTE[8];
        unsigned short WORD[4];
        unsigned long LONG[2];
    }DATA;
} MBbuff;
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*   Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)    */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)  */
    RCANET0.IRR.WORD = 0x0001;           /*   Clear IRR0 (clearing condition: write 1)           */
    RCANET0.MCR.WORD |= 0x8000;          /*   Reorder ID: Set to MCR15 = 1 (initial setting)     */

    for(i = 0;i < 16;i++){               /*   Initialize mailbox (RAM area)                      */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }
}
```

```
/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    RCANET0.IMR.WORD &= 0xFFFD;           /*  Enable data frame receive interrupt      */
    RCANET0.MBIMR0.WORD &= 0xFFFE;        /*  Enable MB0 interrupt                     */

    RCANET0.MB[0].CTRL0.WORD.H = 0x1554; /*  ID: H'555, standard format, data fram    */
    RCANET0.MB[0].LAFM.WORD.H = 0x0000;  /*  STD_LAFM and IDE_LAFM settings           */
    RCANET0.MB[0].CTRL1.BYTE.H = 0x02;   /*  Set mailbox 0 to receive                 */

    /*  Bit rate = 500 kbps  */
    RCANET0.BCR1.WORD = 0x4300;/*  TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, (pϕ = 20 MHz)*/
    RCANET0.BCR0.WORD = 0x0001;           /*  BRP = 1                                  */


}

/*****************************************************************************/
/*      RCAN start routine                                                   */
/*****************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                         /*  Clear MC            */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000);      /*  GSR3 = 0?           */


}


/*****************************************************************************/
/*      RAM area Initialize routine                                          */
/*****************************************************************************/
void Clear_MBbuff(void){

    MBbuff.ID.LONG = 0;                   /*  Initialize RAM area for storing receive data */
    MBbuff.DATA.LONG[0] = 0;
    MBbuff.DATA.LONG[1] = 0;

}
```

```
/*****************************************************************************/
/*      IRQ Interrupt routine                                                */
/*****************************************************************************/
#pragma interrupt(IRQ12)
void IRQ12(void){

    INTC.IER.BIT.IRQ12E = 0;            /*  Disable IRQ interrupt: IRQ12        */

    set_1544_IO_init();                 /*  H8SX/1544 I/O initialization        */

    set_RCAN0_init();                   /*  RCAN initialization                 */
    set_RCAN0MB_init();
    set_RCAN0_start();

}

/*****************************************************************************/
/*      Data Frame Received Interrupt routine(MailBox0)                      */
/*****************************************************************************/
#pragma interrupt(RM0_0)
void RM0_0(void)
{

    RCANET0.RXPR0.WORD = 0x0001;        /*  Clear transmit-end flag (clearing condition: write 1) */
    while(RCANET0.RXPR0.WORD & 0x0001); /*  Check flag                          */

    MBbuff.DATA.BYTE[0] = RCANET0.MB[0].MSG_DATA[0]; /*  Store receive data in RAM  */


}
/*****************************************************************************/
```

2.10 Transfer of Receive Message Using DMAC

2.10.1 Specification

As shown in figure 2.10.1, 15 messages are transmitted by node A and received by node B. Note B receives all the messages in mailbox 0, then uses DMAC to transfer the receive data from mailbox 0 to on-chip RAM.

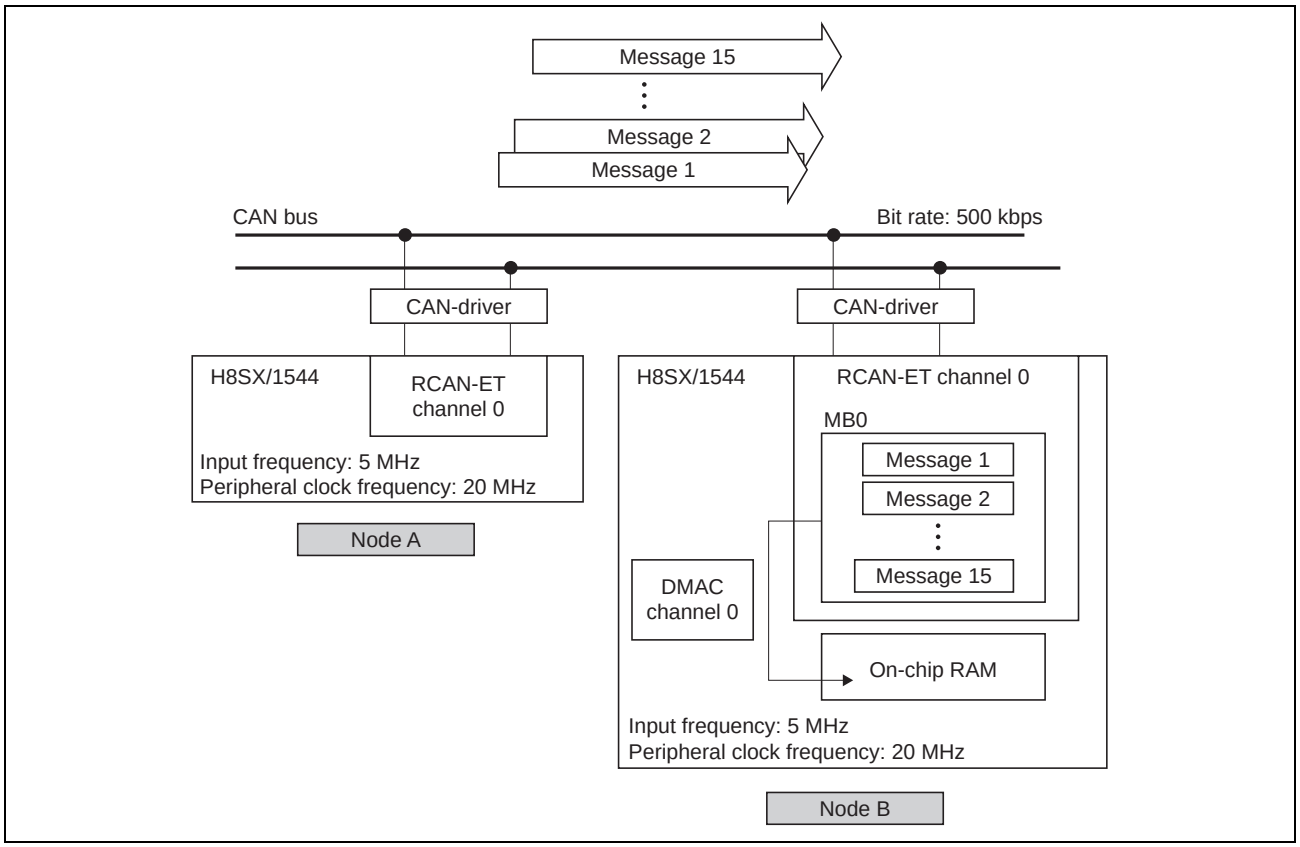


Figure 2.10.1 Communication Specification

Table 2.10.1 lists the message specifications.

Table 2.10.1 Message Specifications

Message	Standard ID	Data
Message 1	H'011	H'01 01 01 01 01 01 01 01 (8 bytes)
Message 2	H'0AA	H'02 02 02 02 02 02 02 02 (8 bytes)
Message 3	H'122	H'03 03 03 03 03 03 03 03 (8 bytes)
Message 4	H'1BB	H'04 04 04 04 04 04 04 04 (8 bytes)
Message 5	H'233	H'05 05 05 05 05 05 05 05 (8 bytes)
Message 6	H'2CC	H'06 06 06 06 06 06 06 06 (8 bytes)
Message 7	H'344	H'07 07 07 07 07 07 07 07 (8 bytes)
Message 8	H'3DD	H'08 08 08 08 08 08 08 08 (8 bytes)
Message 9	H'455	H'09 09 09 09 09 09 09 09 (8 bytes)
Message 10	H'4EE	H'0A 0A 0A 0A 0A 0A 0A 0A (8 bytes)
Message 11	H'566	H'0B 0B 0B 0B 0B 0B 0B 0B (8 bytes)
Message 12	H'5FF	H'0C 0C 0C 0C 0C 0C 0C 0C (8 bytes)
Message 13	H'677	H'0D 0D 0D 0D 0D 0D 0D 0D (8 bytes)
Message 14	H'6EE	H'0E 0E 0E 0E 0E 0E 0E 0E (8 bytes)
Message 15	H'788	H'0F 0F 0F 0F 0F 0F 0F 0F (8 bytes)

Note: This operation example uses standard format. (It is not necessary to set an extended ID.)

2.10.2 Software Description

(1) Module Description

Table 2.10.2 Module Description

Module	Label	Function	Settings
H8SX/1544 initial settings	set_1544_init	Sets H8SX/1544 clock and transmit and receive pins. Cancels RCAN-ET and DMAC module-stop mode.	Device-specific
H8SX/1544 interrupt settings	set_1544_INTC_init	Sets H8SX/1544 RCAN-ET and DMAC interrupt priority levels.	
DMAC initial settings	set_DMAC0_init	Initializes H8SX/1544 DMAC channel 0.	
DMAC transfer-end interrupt	DMTEND0	Clears DMA transfer-end interrupt flag. Clears data frame receive-end flag. Sets transfer destination address and transfer size of next message.	
RCAN initial settings	set_RCAN0_init	See 2.1, Initial Settings.	RCAN-ET common
RCAN mailbox initial settings	set_RCAN0MB_init		
RCAN start	set_RCAN0_start		
CAN message transmit	RCAN0_Tx	Transmits CAN message.	
Mailbox empty interrupt	SLE0_0	Clears CAN message transmit-end flag.	
Receive buffer initialization	Clear_MBbuff	Initializes on-chip RAM area for storing CAN message.	—

(2) Description of Constants Used

Table 2.10.3 Description of Constants Used

Label	Setting Value	Function
ID_1	0x011	CAN message ID
ID_2	0x0AA	
ID_3	0x122	
ID_4	0x1BB	
ID_5	0x233	
ID_6	0x2CC	
ID_7	0x344	
ID_8	0x3DD	
ID_9	0x455	
ID_10	0x4EE	
ID_11	0x566	
ID_12	0x5FF	
ID_13	0x677	
ID_14	0x6EE	
ID_15	0x788	

(3) Description of Registers Used

(a) Transmitting Side

Table 2.10.4 Description of Registers Used (Transmitting Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFEFF	Enables mailbox empty interrupt (IRR8) (IMR8 = 0).
	RCANET0.MBIMR0.WORD	0x0001	Enables interrupts for mailboxes 1 to 15.
	RCANET0.MB[1].CTRL0.WORD.H	(ID_1<<2)	Sets mailbox 1 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[1].CTRL1.BYTE.H	0x00	Sets mailbox 1 to transmit.
	RCANET0.MB[1].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[1].MSG_DATA[0] to RCANET0.MB[1].MSG_DATA[7]	0x01 × 8	Sets the transmit data.
	RCANET0.MB[2].CTRL0.WORD.H	(ID_2<<2)	Sets mailbox 2 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[2].CTRL1.BYTE.H	0x00	Sets mailbox 2 to transmit.
	RCANET0.MB[2].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[2].MSG_DATA[0] to RCANET0.MB[2].MSG_DATA[7]	0x02 × 8	Sets the transmit data.
	RCANET0.MB[3].CTRL0.WORD.H	(ID_3<<2)	Sets mailbox 3 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[3].CTRL1.BYTE.H	0x00	Sets mailbox 3 to transmit.
	RCANET0.MB[3].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[3].MSG_DATA[0] to RCANET0.MB[3].MSG_DATA[7]	0x03 × 8	Sets the transmit data.
	RCANET0.MB[4].CTRL0.WORD.H	(ID_4<<2)	Sets mailbox 4 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[4].CTRL1.BYTE.H	0x00	Sets mailbox 4 to transmit.
	RCANET0.MB[4].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[4].MSG_DATA[0] to RCANET0.MB[4].MSG_DATA[7]	0x04 × 8	Sets the transmit data.
	RCANET0.MB[5].CTRL0.WORD.H	(ID_5<<2)	Sets mailbox 5 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[5].CTRL1.BYTE.H	0x00	Sets mailbox 5 to transmit.
	RCANET0.MB[5].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[5].MSG_DATA[0] to RCANET0.MB[5].MSG_DATA[7]	0x05 × 8	Sets the transmit data.
	RCANET0.MB[6].CTRL0.WORD.H	(ID_6<<2)	Sets mailbox 6 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[6].CTRL1.BYTE.H	0x00	Sets mailbox 6 to transmit.
	RCANET0.MB[6].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[6].MSG_DATA[0] to RCANET0.MB[6].MSG_DATA[7]	0x06 × 8	Sets the transmit data.
	RCANET0.MB[7].CTRL0.WORD.H	(ID_7<<2)	Sets mailbox 7 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[7].CTRL1.BYTE.H	0x00	Sets mailbox 7 to transmit.
	RCANET0.MB[7].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[7].MSG_DATA[0] to RCANET0.MB[7].MSG_DATA[7]	0x07 × 8	Sets the transmit data.

Module	Register	Setting Value	Function
RCAN mailbox initial settings	RCANET0.MB[8].CTRL0.WORD.H	(ID_8<<2)	Sets mailbox 8 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[8].CTRL1.BYTE.H	0x00	Sets mailbox 8 to transmit.
	RCANET0.MB[8].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[8].MSG_DATA[0] to RCANET0.MB[8].MSG_DATA[7]	0x08 × 8	Sets the transmit data.
	RCANET0.MB[9].CTRL0.WORD.H	(ID_9<<2)	Sets mailbox 9 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[9].CTRL1.BYTE.H	0x00	Sets mailbox 9 to transmit.
	RCANET0.MB[9].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[9].MSG_DATA[0] to RCANET0.MB[9].MSG_DATA[7]	0x09 × 8	Sets the transmit data.
	RCANET0.MB[10].CTRL0.WORD.H	(ID_10<<2)	Sets mailbox 10 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[10].CTRL1.BYTE.H	0x00	Sets mailbox 10 to transmit.
	RCANET0.MB[10].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[10].MSG_DATA[0] to RCANET0.MB[10].MSG_DATA[7]	0x0A × 8	Sets the transmit data.
	RCANET0.MB[11].CTRL0.WORD.H	(ID_11<<2)	Sets mailbox 11 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[11].CTRL1.BYTE.H	0x00	Sets mailbox 11 to transmit.
	RCANET0.MB[11].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[11].MSG_DATA[0] to RCANET0.MB[11].MSG_DATA[7]	0x0B × 8	Sets the transmit data.
	RCANET0.MB[12].CTRL0.WORD.H	(ID_12<<2)	Sets mailbox 12 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[12].CTRL1.BYTE.H	0x00	Sets mailbox 12 to transmit.
	RCANET0.MB[12].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[12].MSG_DATA[0] to RCANET0.MB[12].MSG_DATA[7]	0x0C × 8	Sets the transmit data.
	RCANET0.MB[13].CTRL0.WORD.H	(ID_13<<2)	Sets mailbox 13 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[13].CTRL1.BYTE.H	0x00	Sets mailbox 13 to transmit.
	RCANET0.MB[13].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[13].MSG_DATA[0] to RCANET0.MB[13].MSG_DATA[7]	0x0D × 8	Sets the transmit data.
	RCANET0.MB[14].CTRL0.WORD.H	(ID_14<<2)	Sets mailbox 14 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[14].CTRL1.BYTE.H	0x00	Sets mailbox 14 to transmit.
	RCANET0.MB[14].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[14].MSG_DATA[0] to RCANET0.MB[14].MSG_DATA[7]	0x0E × 8	Sets the transmit data.
	RCANET0.MB[15].CTRL0.WORD.H	(ID_15<<2)	Sets mailbox 15 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[15].CTRL1.BYTE.H	0x00	Sets mailbox 15 to transmit.
	RCANET0.MB[15].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[15].MSG_DATA[0] to RCANET0.MB[15].MSG_DATA[7]	0x0F × 8	Sets the transmit data.
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	

Module	Register	Setting Value	Function
RCAN start H8SX/1544 interrupt settings	See 2.1, Initial Settings.		
CAN message transmit	RCANET0.TXPR0.LONG	0x0000FFFE	Sets mailboxes 1 to 15 to transmit-wait status.
Mailbox empty interrupt	RCANET0.TXACK0.WORD	—	Clears transmit-end flag for mailboxes 1 to 15. (Clearing condition: write 1)

Notes: 1. The bit names listed in the table correspond to the bit names used in the hardware manual.
2. ID_x (x = 1 to 15) indicates labels listed in (2), “Description of Constants Used,” in 2.10.2, “Software Description.”

(b) Receiving Side

Table 2.10.5 Description of Registers Used (Receiving Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	SCKCR.BIT.ICK	0	Sets the system clock frequency to 40 MHz (input clock: 5 MHz × 8).
	SCKCR.BIT.BCK	1	Sets the peripheral module clock frequency to 20 MHz (input clock: 5 MHz × 4).
	SCKCR.BIT.PCK	1	Sets the external bus clock frequency to 20 MHz (input clock: 5 MHz × 4).
	SBYCR.BIT.SSBY	1	Enables transition to software standby mode after execution of Sleep instruction.
	MSTP.CRA.BIT._DMAC	0	Cancels DMAC module-stop mode.
	MSTP.CRC.BIT._RCAN01	0	Cancels RCAN-ET module-stop mode.
	RCANET0.RCANMON.BYTE	0x20	Enables RCAN-ET transmit and receive pins.
	P6.ICR.BIT.B4	1	Sets P64 (pin 67) as RCAN-ET channel 0 receive pin.
DMAC initial settings	DMAC0.DMDR.LONG	0x000090A0	Sets the data size to longword and the transfer mode to block transfer mode. Also sets on-chip module interrupt as the DMAC start source.
	DMAC0.DACR.LONG	0x00220000	Sets the address mode to dual address mode and the block area to the transfer source address. Also adds address update mode settings for transfer source and transfer destination addresses.
	DMAC0.DSAR	Note 2.	Specifies transfer source address.
	DMAC0.DDAR	Note 2.	Specifies transfer destination address.
	DMAC0.DOFR	0x00000000	Sets address update offset values (no offset) for transfer source and transfer destination addresses.
	DMAC0.DTCR	0x00000008	Sets total transfer size (8 bytes) of transfer data.
	DMAC0.DBSR	0x00020000	Sets block size (2 longwords).
	DMAC0.DMRSR	0xDC	Specifies vector number of on-chip module interrupt to be used as DMAC transfer start source. • Interrupt source: Message received by mailbox 0 on RCAN-ET channel 0 • Vector number: 220 (= h'DC)

Module	Register	Setting Value	Function
DMAC initial settings	DMAC0.DMDR.BIT.DTIE	1	Enables data transfer-end interrupt.
	DMAC0.DMDR.BIT.DTE	1	Enables data transfer.
RCAN initial settings	See 2.1, Initial Settings.		
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFFFD	Enables data frame receive interrupt (IRR1) (IMR1 = 0).
	RCANET0.MBIMR0.WORD	0xFFFE	Enables interrupts for mailbox 0.
	RCANET0.MB[0].CTRL0.WORD.H	0x0000	Sets mailbox 0 to standard format, data frame.
	RCANET0.MB[0].LAFM.WORD.H	0x1FFC	Sets filter mask for standard ID and IDE bit of mailbox 0. (Enables receive of messages with any standard ID.)
	RCANET0.MB[0].CTRL1.BYTE.H	0x02	Sets mailbox 0 to receive.
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
H8SX/1544 interrupt settings	INTC.INTCR.BIT.INTM	2	Sets interrupt controller to interrupt control mode 2.
	INTC.IPRI.BIT._DMAC0	7	Sets DMAC interrupt priority level.
	INTC.IPRQ.BIT._RCAN01	0	Sets RCAN-ET interrupt priority level.
DMAC transfer-end interrupt	DMAC0.DMDR.LONG	0xFFFEFFFF	Clears data transfer-end interrupt flag (DTIF).
	RCANET0.RXPR0.WORD	0x0001	Clears mailbox 0 receive-end flag. (Clearing condition: write 1)
	DMAC0.DDAR	Note 2.	Specifies transfer destination address.
	DMAC0.DTCR	0x00000008	Sets total transfer size (8 bytes) of transfer data.
	DMAC0.DBSR	0x00020000	Sets block size (2 longwords).
	DMAC0.DMDR.BIT.DTE	1	Enables data transfer.
	DMAC0.DMDR.BIT.DTIE	0	Disables data transfer-end interrupt.

Notes: 1. The bit names listed in the table correspond to the bit names used in the hardware manual.
2. See (2), “Receiving Side Program Listing,” in 2.10.4, “Program Listing” for setting values.

(4) Description of RAM Used

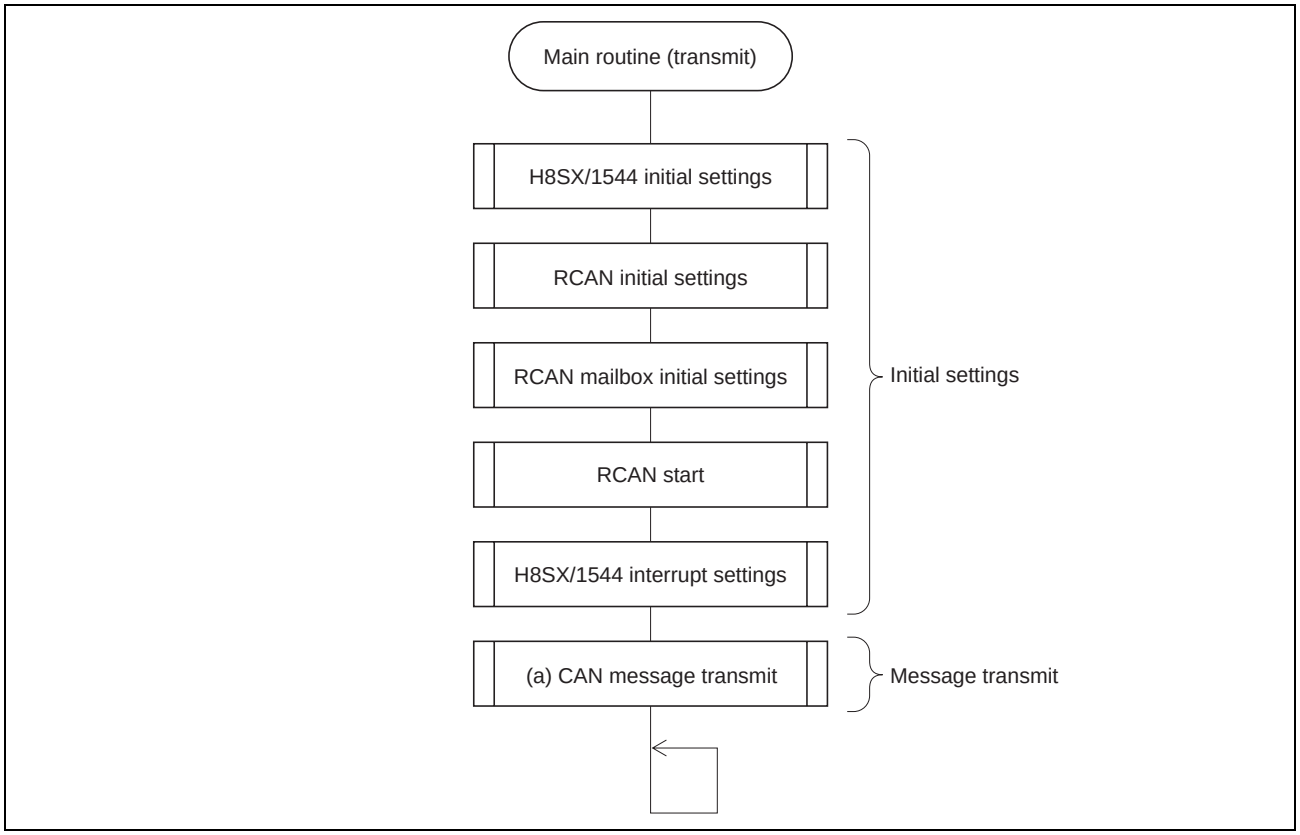
Table 2.10.6 Description of RAM Used

Module	Label	Function
Receive buffer initialization	Mbbuff[15].ID.LONG MBbuff[15].ID.WORD.H MBbuff[15].ID.WORD.L	Stores receive ID.
Data frame receive interrupt	MBbuff[15].DATA.LONG[0] to [1] MBbuff[15].DATA.WORD[0] to [3] MBbuff[15].DATA.BYTE[0] to [7]	Stores receive data.
Main routine DMAC transfer-end interrupt	Rcv_cnt	Indicates number of messages for which DMA transfer is complete.

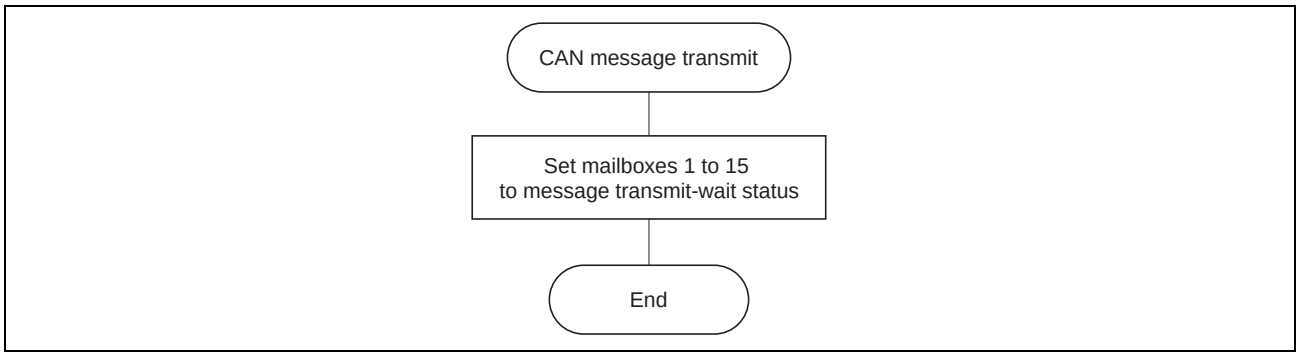
2.10.3 Flowcharts

(1) Transmitting Side Flowcharts

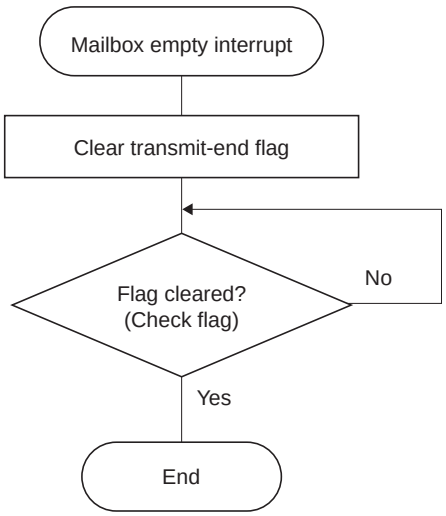
Main Routine



(a) CAN Message Transmit Routine (RCAN-ET Common Settings)



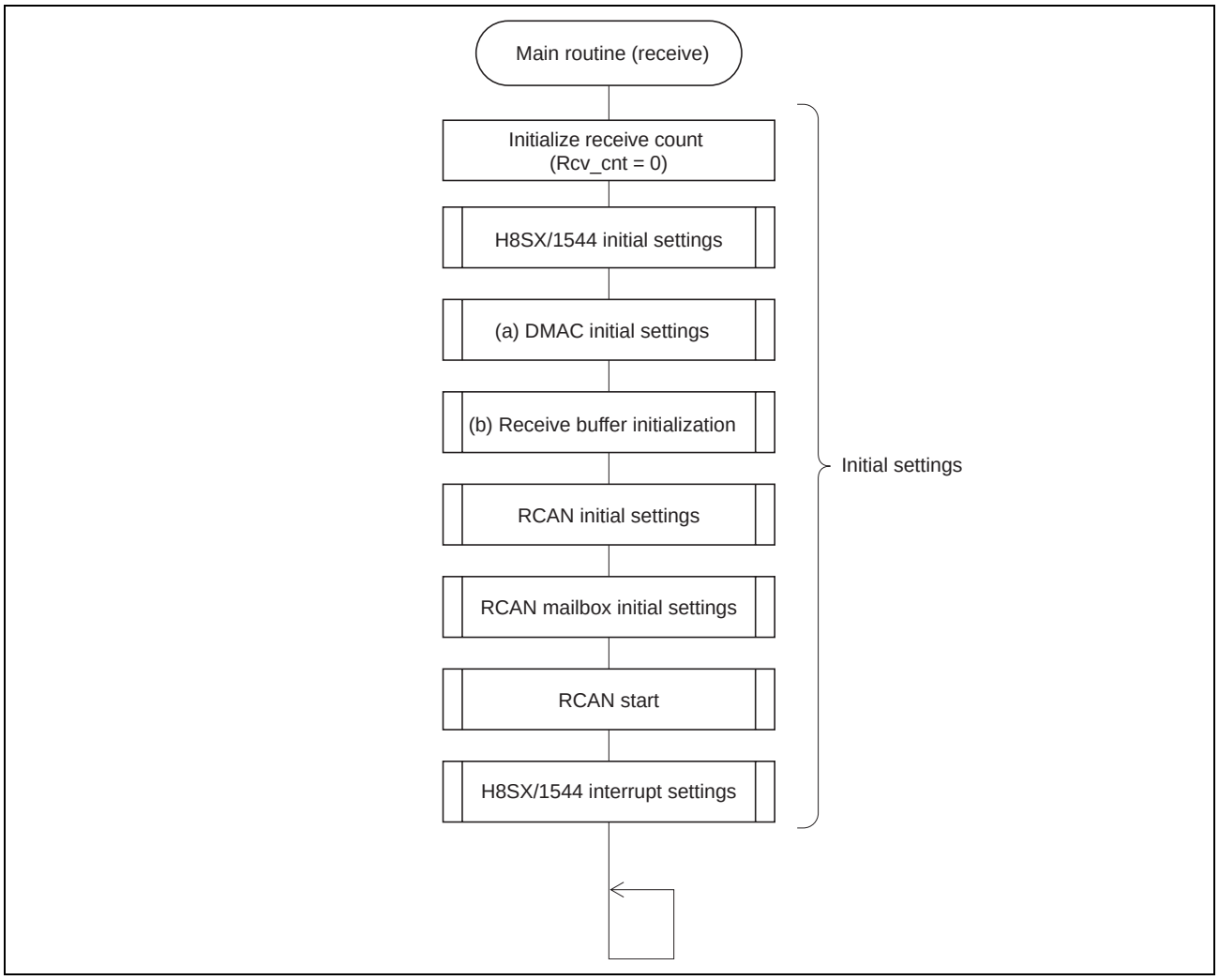
(b) Mailbox Empty Interrupt Routine (RCAN-ET Common Settings)



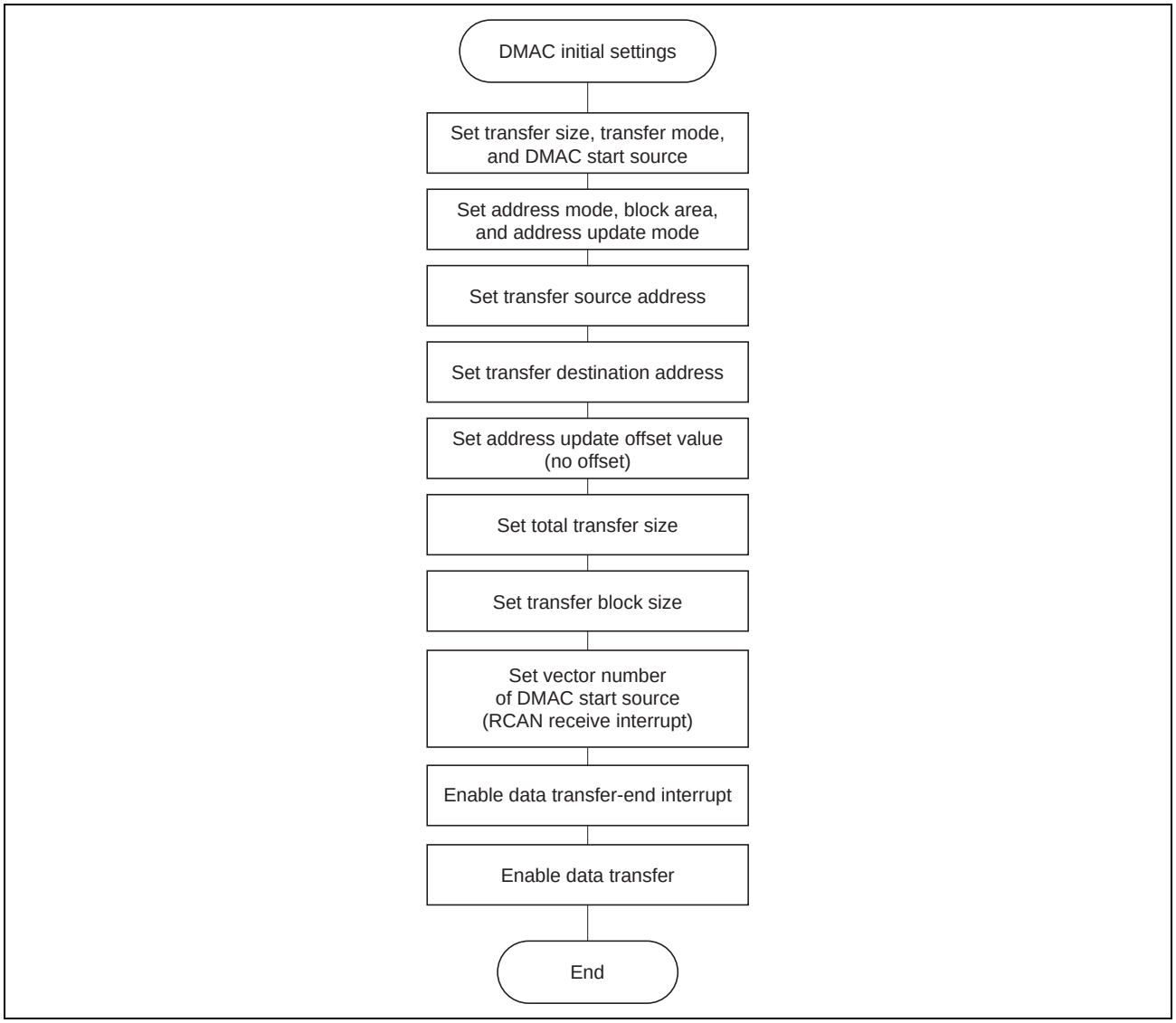
Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

(2) Receiving Side Flowcharts

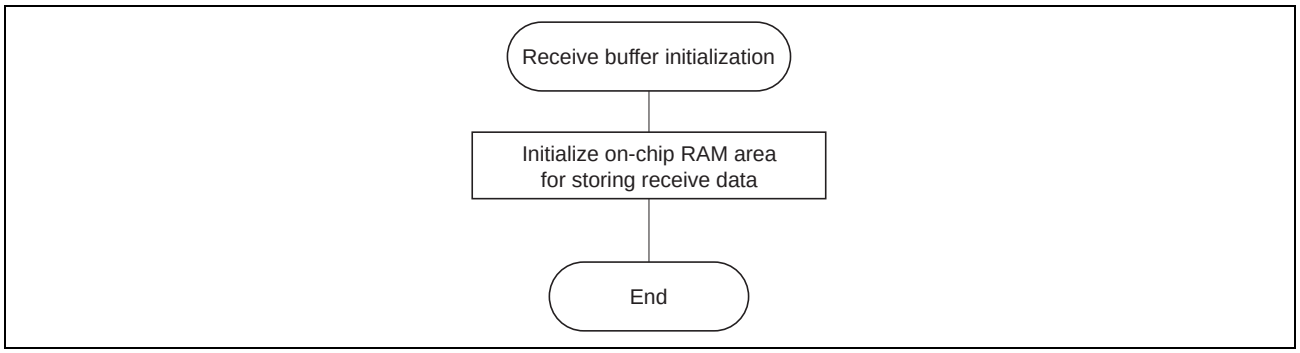
Main Routine



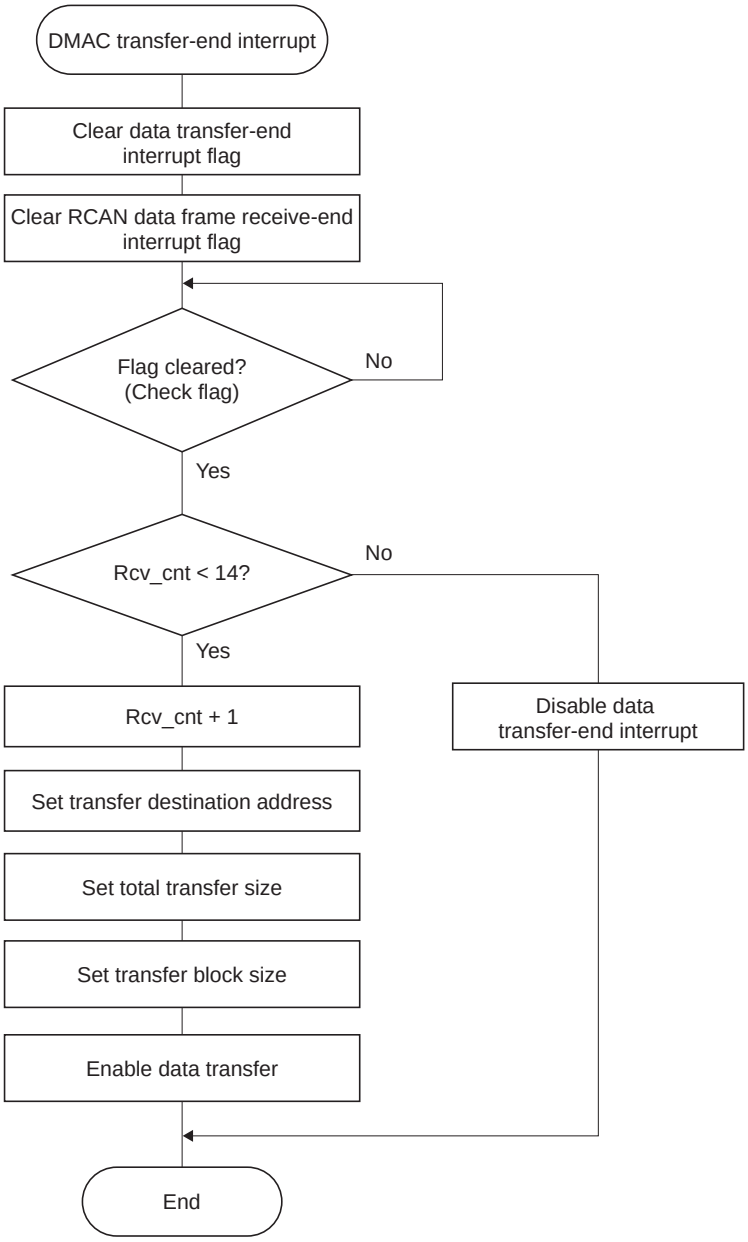
(a) DMAC Initial Settings Routine (Device-Specific Settings)



(b) Receive Buffer Initialization Routine (Device-Specific Settings)



(c) DMAC Transfer-End Interrupt Routine (Device-Specific Settings)



Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

2.10.4 Program Listing

(1) Transmitting Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void RCAN0_Tx(void);
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{
    set_1544_init();                    /*  H8SX/1544 initialization        */
    set_RCAN0_init();                   /*  RCAN initialization             */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();               /*  H8SX/1544 interrupt settings    */
    RCAN0_Tx();                         /*  CAN message transmit            */
    while(1);

}

/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                  /*  Set system clock (5 MHz × 8 = 40 MHz)       */
    SCKCR.BIT.PCK = 1;                  /*  Set peripheral clock (5 MHz × 4 = 20 MHz)   */
    SCKCR.BIT.BCK = 1;                  /*  Set external bus clock (5 MHz × 4 = 20 MHz) */
    SBYCR.BIT.SSBY = 1;                 /*  Software standby mode           */
    MSTP.CRC.BIT._RCAN01 = 0;           /*  Cancel module-stop mode: RCAN   */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;        /*  Enable RCAN-ET transmit and receive pins    */
    P6.ICR.BIT.B4 = 1;                  /*  Set P64 as CRx_0 (input pin)    */

}
```

```
/*****************************************************************************/
/*      H8SX/1544 INTC Initialize routine                                    */
/*****************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;                /*  Interrupt control mode 2        */
    INTC.IPRQ.BIT._RCAN01 = 7;              /*  Set interrupt priority level: RCAN  */
    set_imask_exr(0);                       /*  Interrupt mask level 0          */


}
/*****************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename    :  rcan.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void RCAN0_Tx(void);
/*---------------------------------------------------------------------------*/
#define ID_1        0x011
#define ID_2        0x0AA
#define ID_3        0x122
#define ID_4        0x1BB
#define ID_5        0x233
#define ID_6        0x2CC
#define ID_7        0x344
#define ID_8        0x3DD
#define ID_9        0x455
#define ID_10       0x4EE
#define ID_11       0x566
#define ID_12       0x5FF
#define ID_13       0x677
#define ID_14       0x6EE
#define ID_15       0x788
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*  Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)    */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)  */
    RCANET0.IRR.WORD = 0x0001;           /*  Clear IRR0 (clearing condition: write 1)    */
    RCANET0.MCR.WORD |= 0x8000;          /*  Reorder ID: Set to MCR15 = 1 (initial setting)  */

    for(i = 0;i < 16;i++){               /*  Initialize mailbox (RAM area)               */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }
}
```

```
/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    unsigned int i;

    RCANET0.IMR.WORD &= 0xFEFF;                /*  Enable mailbox empty interrupt      */
    RCANET0.MBIMR0.WORD = 0x0001;              /*  Enable MB1-15 interrupt             */

    RCANET0.MB[1].CTRL0.WORD.H  = (ID_1<<2);   /*  Set STDID, standard format, data frame  */
    RCANET0.MB[1].CTRL1.BYTE.H = 0x00;         /*  Set mailbox 1 to transmit           */
    RCANET0.MB[1].CTRL1.BYTE.L = 0x08;         /*  Data length: 8 bytes                */
    for(i = 0;i < 8;i++){
        RCANET0.MB[1].MSG_DATA[i] = 0x01;      /*  Transmit data                       */
    }

    RCANET0.MB[2].CTRL0.WORD.H  = (ID_2<<2);   /*  Set STDID, standard format, data frame  */
    RCANET0.MB[2].CTRL1.BYTE.H = 0x00;         /*  Set mailbox 2 to transmit           */
    RCANET0.MB[2].CTRL1.BYTE.L = 0x08;         /*  Data length: 8 bytes                */
    for(i = 0;i < 8;i++){
        RCANET0.MB[2].MSG_DATA[i] = 0x02;      /*  Transmit data                       */
    }

    RCANET0.MB[3].CTRL0.WORD.H  = (ID_3<<2);   /*  Set STDID, standard format, data frame  */
    RCANET0.MB[3].CTRL1.BYTE.H = 0x00;         /*  Set mailbox 3 to transmit           */
    RCANET0.MB[3].CTRL1.BYTE.L = 0x08;         /*  Data length: 8 bytes                */
    for(i = 0;i < 8;i++){
        RCANET0.MB[3].MSG_DATA[i] = 0x03;      /*  Transmit data                       */
    }

    RCANET0.MB[4].CTRL0.WORD.H  = (ID_4<<2);   /*  Set STDID, standard format, data frame  */
    RCANET0.MB[4].CTRL1.BYTE.H = 0x00;         /*  Set mailbox 4 to transmit           */
    RCANET0.MB[4].CTRL1.BYTE.L = 0x08;         /*  Data length: 8 bytes                */
    for(i = 0;i < 8;i++){
        RCANET0.MB[4].MSG_DATA[i] = 0x04;      /*  Transmit data                       */
    }

    RCANET0.MB[5].CTRL0.WORD.H  = (ID_5<<2);   /*  Set STDID, standard format, data frame  */
    RCANET0.MB[5].CTRL1.BYTE.H = 0x00;         /*  Set mailbox 5 to transmit           */
    RCANET0.MB[5].CTRL1.BYTE.L = 0x08;         /*  Data length: 8 bytes                */
    for(i = 0;i < 8;i++){
        RCANET0.MB[5].MSG_DATA[i] = 0x05;      /*  Transmit data                       */
    }

    RCANET0.MB[6].CTRL0.WORD.H  = (ID_6<<2);   /*  Set STDID, standard format, data frame  */
    RCANET0.MB[6].CTRL1.BYTE.H = 0x00;         /*  Set mailbox 6 to transmit           */
    RCANET0.MB[6].CTRL1.BYTE.L = 0x08;         /*  Data length: 8 bytes                */
    for(i = 0;i < 8;i++){
        RCANET0.MB[6].MSG_DATA[i] = 0x06;      /*  Transmit data                       */
    }
```

```
RCANET0.MB[7].CTRL0.WORD.H  = (ID_7<<2);  /*   Set STDID, standard format, data frame   */
RCANET0.MB[7].CTRL1.BYTE.H = 0x00;        /*   Set mailbox 7 to transmit                */
RCANET0.MB[7].CTRL1.BYTE.L = 0x08;        /*   Data length: 8 bytes                     */
for(i = 0;i < 8;i++){
    RCANET0.MB[7].MSG_DATA[i] = 0x07;     /*   Transmit data                            */
}

RCANET0.MB[8].CTRL0.WORD.H  = (ID_8<<2);  /*   Set STDID, standard format, data frame   */
RCANET0.MB[8].CTRL1.BYTE.H = 0x00;        /*  Set mailbox 8 to transmit                 */
RCANET0.MB[8].CTRL1.BYTE.L = 0x08;        /*   Data length: 8 bytes                     */
for(i = 0;i < 8;i++){
    RCANET0.MB[8].MSG_DATA[i] = 0x08;     /*   Transmit data                            */
}

RCANET0.MB[9].CTRL0.WORD.H  = (ID_9<<2);  /*   Set STDID, standard format, data frame   */
RCANET0.MB[9].CTRL1.BYTE.H = 0x00;        /*   Set mailbox 9 to transmit                */
RCANET0.MB[9].CTRL1.BYTE.L = 0x08;        /*   Data length: 8 bytes                     */
for(i = 0;i < 8;i++){
    RCANET0.MB[9].MSG_DATA[i] = 0x09;     /*   Transmit data                            */
}

RCANET0.MB[10].CTRL0.WORD.H  = (ID_10<<2);/*   Set STDID, standard format, data frame   */
RCANET0.MB[10].CTRL1.BYTE.H = 0x00;       /*   Set mailbox 10 to transmit               */
RCANET0.MB[10].CTRL1.BYTE.L = 0x08;       /*   Data length: 8 bytes                     */
for(i = 0;i < 8;i++){
    RCANET0.MB[10].MSG_DATA[i] = 0x0A;    /*   Transmit data                            */
}

RCANET0.MB[11].CTRL0.WORD.H  = (ID_11<<2);/*   Set STDID, standard format, data frame   */
RCANET0.MB[11].CTRL1.BYTE.H = 0x00;       /*   Set mailbox 11 to transmit               */
RCANET0.MB[11].CTRL1.BYTE.L = 0x08;       /*   Data length: 8 bytes                     */
for(i = 0;i < 8;i++){
    RCANET0.MB[11].MSG_DATA[i] = 0x0B;    /*   Transmit data                            */
}

RCANET0.MB[12].CTRL0.WORD.H  = (ID_12<<2);/*   Set STDID, standard format, data frame   */
RCANET0.MB[12].CTRL1.BYTE.H = 0x00;       /*   Set mailbox 12 to transmit               */
RCANET0.MB[12].CTRL1.BYTE.L = 0x08;       /*   Data length: 8 bytes                     */
for(i = 0;i < 8;i++){
    RCANET0.MB[12].MSG_DATA[i] = 0x0C;    /*   Transmit data                            */
}

RCANET0.MB[13].CTRL0.WORD.H  = (ID_13<<2);/*   Set STDID, standard format, data frame   */
RCANET0.MB[13].CTRL1.BYTE.H = 0x00;       /*   Set mailbox 13 to transmit               */
RCANET0.MB[13].CTRL1.BYTE.L = 0x08;       /*   Data length: 8 bytes                     */
for(i = 0;i < 8;i++){
    RCANET0.MB[13].MSG_DATA[i] = 0x0D;    /*   Transmit data                            */
}
```

```
    RCANET0.MB[14].CTRL0.WORD.H  = (ID_14<<2);/*  Set STDID, standard format, data frame  */
    RCANET0.MB[14].CTRL1.BYTE.H = 0x00;       /*  Set mailbox 14 to transmit             */
    RCANET0.MB[14].CTRL1.BYTE.L = 0x08;       /*  Data length: 8 bytes                   */
    for(i = 0;i < 8;i++){
        RCANET0.MB[14].MSG_DATA[i] = 0x0E;    /*  Transmit data                          */
    }

    RCANET0.MB[15].CTRL0.WORD.H  = (ID_15<<2);/*  Set STDID, standard format, data frame  */
    RCANET0.MB[15].CTRL1.BYTE.H = 0x00;       /*  Set mailbox 15 to transmit             */
    RCANET0.MB[15].CTRL1.BYTE.L = 0x08;       /*  Data length: 8 bytes                   */
    for(i = 0;i < 8;i++){
        RCANET0.MB[15].MSG_DATA[i] = 0x0F;    /*  Transmit data                          */
    }

    /* Bit rate = 500 kbps */
    RCANET0.BCR1.WORD = 0x4300;/* TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, (pϕ = 20 MHz)*/
    RCANET0.BCR0.WORD = 0x0001;               /*  BRP = 1                                */


}

/*****************************************************************************/
/*      RCAN start routine                                                   */
/*****************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                         /*  Clear MCR0           */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000);      /*  GSR3 = 0?            */

}

/*****************************************************************************/
/*      RCAN send message routine                                            */
/*****************************************************************************/
void RCAN0_Tx(void){

    RCANET0.TXPR0.LONG = 0x0000FFFE;          /*  Set MB1-15 to transmit-wait status     */

}

/*****************************************************************************/
/*      Mailbox Empty Interrupt routine                                      */
/*****************************************************************************/
#pragma interrupt(SLE0_0)
void SLE0_0(void)
{
    RCANET0.TXACK0.WORD = RCANET0.TXACK0.WORD;
                                      /*  Clear transmit-end flag (clearing condition: write 1) */
    while(RCANET0.TXACK0.WORD);       /*  Check flag                             */

}
/*****************************************************************************/
```

(2) Receiving Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void Clear_MBbuff(void);
extern void set_DMAC0_init(void);
/*---------------------------------------------------------------------------*/
unsigned char Rcv_cnt;
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{
    Rcv_cnt = 0;                        /*  Initialize receive counter       */
    set_1544_init();                    /*  H8SX/1544 initialization         */
    set_DMAC0_init();                   /*  Initialize DMAC                  */
    Clear_MBbuff();                     /*  Initialize RAM area for storing receive data */
    set_RCAN0_init();                   /*  RCAN initialization              */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();               /*  H8SX/1544 interrupt settings     */
    while(1);
}
/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                  /*  Set system clock (5 MHz × 8 = 40 MHz)  */
    SCKCR.BIT.PCK = 1;                  /*  Set peripheral clock (5 MHz × 4 = 20 MHz) */
    SCKCR.BIT.BCK = 1;                  /*  Set external bus clock (5 MHz × 4 = 20 MHz) */
    SBYCR.BIT.SSBY = 1;                 /*  Software standby mode            */
    MSTP.CRA.BIT._DMAC = 0;             /*  Cancel module-stop mode: DMAC    */
    MSTP.CRC.BIT._RCAN01 = 0;           /*  Cancel module-stop mode: RCAN    */
    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;        /*  Enable RCAN-ET transmit and receive pins */
    P6.ICR.BIT.B4 = 1;                  /*  Set P64 as CRx_0 (input pin)     */
}
```

```
/************************************************************************/
/*      H8SX/1544 INTC Initialize routine                               */
/************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;            /*  Interrupt control mode 2                    */
    INTC.IPRI.BIT._DMAC0 = 7;           /*  Interrupt priority level: DMAC              */
    INTC.IPRQ.BIT._RCAN01 = 0;          /*  Interrupt priority level: RCAN (interrupt disabled) */
    set_imask_exr(0);                   /*  Interrupt mask level 0                      */


}
/************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename    :  rcan.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void Clear_MBbuff(void);
void set_DMAC0_init(void);
/*---------------------------------------------------------------------------*/
struct {
    union {
        unsigned long LONG;
        struct{
            unsigned short H;
            unsigned short L;
        }WORD;
    }ID;
    union {
        unsigned char BYTE[8];
        unsigned short WORD[4];
        unsigned long LONG[2];
    }DATA;
} MBbuff[15];
/*---------------------------------------------------------------------------*/
extern unsigned char Rcv_cnt;
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*  Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)    */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)  */
    RCANET0.IRR.WORD = 0x0001;           /*  Clear IRR0 (clearing condition: write 1)    */
    RCANET0.MCR.WORD |= 0x8000;          /*  Reorder ID: Set to MCR15 = 1 (initial setting)  */

    for(i = 0;i < 16;i++){               /*  Initialize mailbox (RAM area)               */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }
}
```

```
/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    RCANET0.IMR.WORD &= 0xFFFD;           /*   Enable data frame receive interrupt      */
    RCANET0.MBIMR0.WORD &= 0xFFFE;        /*   Enable MB0 interrupt                     */

    RCANET0.MB[0].CTRL0.WORD.H = 0x0000; /*   Set STDID, standard format, data frame   */
    RCANET0.MB[0].LAFM.WORD.H = 0x1FFC;  /*   Set STD_LAFM                             */
    RCANET0.MB[0].CTRL1.BYTE.H = 0x02;   /*   Set mailbox 0 to receive                 */

    /* Bit rate = 500 kbps */
    RCANET0.BCR1.WORD = 0x4300;/* TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, (pϕ = 20 MHz)*/
    RCANET0.BCR0.WORD = 0x0001;           /*   BRP = 1                                  */


}

/*****************************************************************************/
/*      RCAN start routine                                                   */
/*****************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                         /*   Clear MCR0          */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000);      /*   GSR3 = 0?           */


}


/*****************************************************************************/
/*      RAM area Initialize routine                                          */
/*****************************************************************************/
void Clear_MBbuff(void){

    unsigned int i;

    for(i = 0;i < 15;i++){                /*   Initialize RAM area for storing receive data */
        MBbuff[i].ID.LONG = 0;
        MBbuff[i].DATA.LONG[0] = 0;
        MBbuff[i].DATA.LONG[1] = 0;
    }

}
```

```
/*****************************************************************************/
/*      DMAC Initialize routine                                              */
/*****************************************************************************/
void set_DMAC0_init(void){

    DMAC0.DMDR.LONG = 0x000090A0;
                    /* Transfer size: longword, block transfer, start source: on-chip module */
    DMAC0.DACR.LONG = 0x00220000;
                    /* Dual address, block area: transfer source, add transfer source and destination addresses */
    DMAC0.DSAR = (void*)&RCANET0.MB[0].MSG_DATA[0];    /*  Transfer source address      */
    DMAC0.DDAR = (void*)&MBbuff[Rcv_cnt].DATA.BYTE[0];/*  Transfer destination address */
    DMAC0.DOFR = 0x00000000;                    /*  No address offset                  */
    DMAC0.DTCR = 0x00000008;                    /*  Total transfer size: 8 bytes       */
    DMAC0.DBSR = 0x00020000;                    /*  Block size: 2 longwords            */
    DMAC0.DMRSR = 0xDC;       /*  Interrupt source: RM0_0 (RCAN channel 0 MB0 receive interrupt) */
    DMAC0.DMDR.BIT.DTIE = 1;                    /*  Enable data transfer-end interrupt */
    DMAC0.DMDR.BIT.DTE = 1;                     /*  Enable data transfer               */

}


/*****************************************************************************/
/*      DMAC Transmit End Interrupt routine                                  */
/*****************************************************************************/
#pragma interrupt(DMTEND0)
void DMTEND0(void)
{

    DMAC0.DMDR.LONG &= 0xFFFEFFFF;      /*  Clear data transfer-end interrupt flag (DTIF) */
    RCANET0.RXPR0.WORD = 0x0001;        /*  Clear message receive-end flag                */
    while(RCANET0.RXPR0.WORD & 0x0001); /*  Check flags                                   */

    if(Rcv_cnt < 14){                   /*  Transfer of all messages (15 messages) not complete */
        Rcv_cnt++;
        DMAC0.DDAR = (void*)&MBbuff[Rcv_cnt].DATA.BYTE[0];
                                        /*  Transfer destination address                  */
        DMAC0.DTCR = 0x00000008;        /*  Total transfer size: 8 bytes                  */
        DMAC0.DBSR = 0x00020000;        /*  Block size                                    */
        DMAC0.DMDR.BIT.DTE = 1;         /*  Enable data transfer                          */
    }
    else{                               /*  Transfer of all messages (15 messages) complete */
        DMAC0.DMDR.BIT.DTIE = 0;        /*  Disable transfer-end interrupt                */
    }


}
/*****************************************************************************/
```

2.11 Mailbox Reset

2.11.1 Specification

As shown in figure 2.11.1, a message with standard ID H'1BB is transmitted by node A from mailbox 1 and received by node B in mailbox 1.

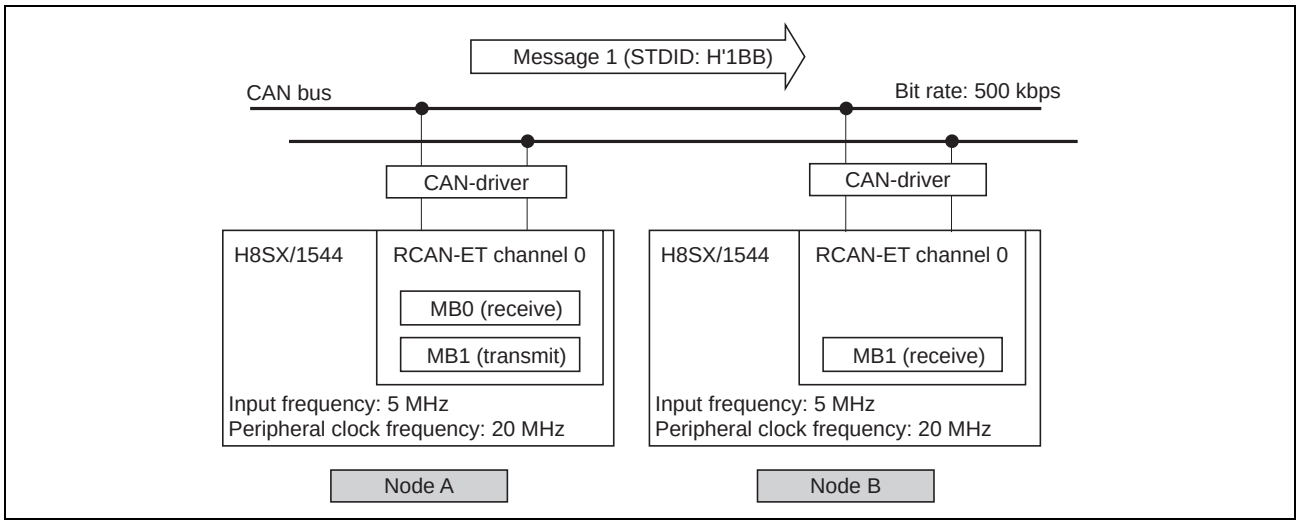


Figure 2.11.1 Communication Specification (1)

Next, as shown in figure 2.11.2, node B changes mailbox 1 from receive to transmit, resets the standard ID to H'0AA, and transmits the message. The message is received by node A in mailbox 0.

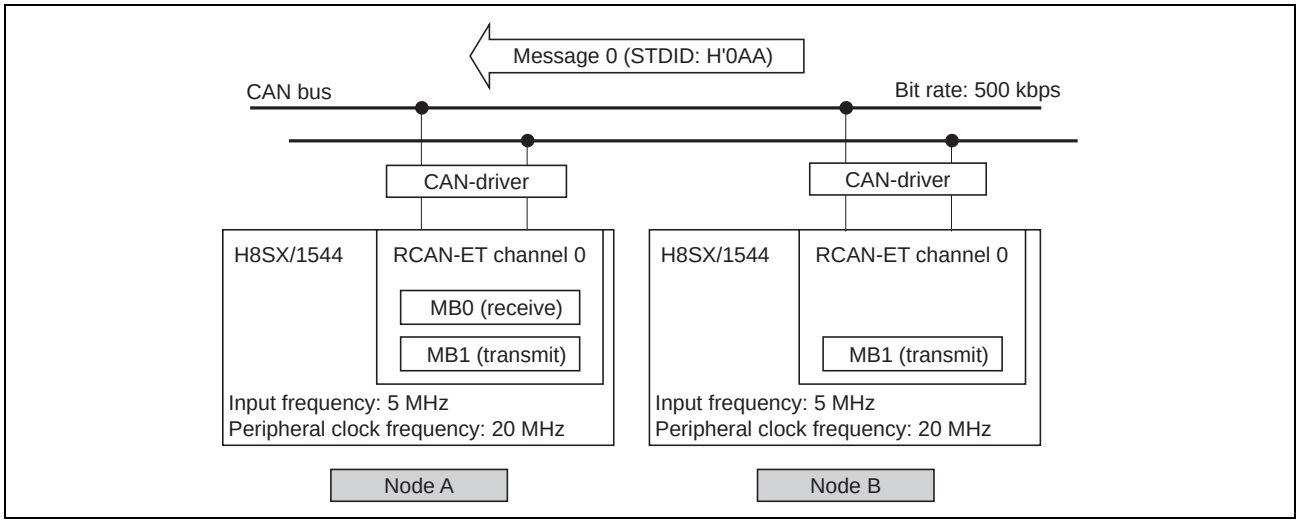


Figure 2.11.2 Communication Specification (2)

2.11.2 Software Description

(1) Module Description

Table 2.11.1 Module Description

Module	Label	Function	Settings
H8SX/1544 initial settings	set_1544_init	See 2.1, Initial Settings.	Device-specific
H8SX/1544 interrupt settings	set_1544_INTC_init		
RCAN initial settings	set_RCAN0_init		RCAN-ET common
RCAN mailbox initial settings	set_RCAN0MB_init		
RCAN start	set_RCAN0_start		
CAN message transmit	RCAN0_Tx	Transmits CAN message.	
Mailbox reset	set_RCAN0MB_change	Resets mailbox from receive to transmit.	
Mailbox empty interrupt	SLE0_0	Clears CAN message transmit-end flag.	
Data frame receive interrupt (mailbox 0)	RM0_0	Clears data frame transmit-end flag. Stores receive data in RAM.	
Data frame receive interrupt* (mailbox 1)	RM1_0	Clears data frame transmit-end flag. Resets mailbox and transmits message. Clears message transmit-end flag.	
Receive buffer initialization	Clear_MBbuff	Initializes on-chip RAM area for storing CAN message.	—

Note: * Common interrupt vector addresses are used for all interrupt sources (RM1, SLE, OVR, and ERS) in the case of mailboxes other than mailbox 0. Therefore, in this operation example the same interrupt function is used for the data frame receive interrupt (IRR1) and the data frame transmit-end interrupt (IRR8).

(2) Description of Constants Used

Table 2.11.2 Description of Constants Used

Label	Setting Value	Function
ID_0	0x0AA	CAN message ID
ID_1	0x1BB	

(3) Description of Registers Used

(a) Transmitting Side

Table 2.11.3 Description of Registers Used (Transmitting Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFEFD	Enables mailbox empty interrupt (IRR8) and data frame receive interrupt (IRR1) (IMR8 = 0, IMR1 = 0).
	RCANET0.MBIMR0.WORD	0xFFFC	Enables interrupts for mailboxes 0 and 1.
	RCANET0.MB[0].CTRL0.WORD.H	(ID_0<< 2)	Sets mailbox 0 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[0].LAFM.WORD.H	0x0000	Sets filter mask for standard ID and IDE bit of mailbox 0.
	RCANET0.MB[0].CTRL1.BYTE.H	0x02	Sets mailbox 0 to receive.
	RCANET0.MB[1].CTRL0.WORD.H	(ID_1<< 2)	Sets mailbox 1 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[1].CTRL1.BYTE.H	0x00	Sets mailbox 1 to transmit.
	RCANET0.MB[1].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[1].MSG_DATA[0] to RCANET0.MB[1].MSG_DATA[7]	0x01 × 8	Sets the transmit data.
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
H8SX/1544 interrupt settings			
CAN message transmit	RCANET0.TXPR0.LONG	0x0000 0002	Sets mailbox 1 to transmit-wait status.
Mailbox empty interrupt	RCANET0.TXACK0.WORD	0x0002	Clears mailbox 1 transmit-end flag. (Clearing condition: write 1)
Data frame receive interrupt (mailbox 0)	RCANET0.RXPR0.WORD	0x0001	Clears mailbox 0 receive-end flag. (Clearing condition: write 1)

Notes: 1. The bit names listed in the table correspond to the bit names used in the hardware manual.
2. ID_0 and ID_1 indicate labels listed in (2), “Description of Constants Used,” in 2.11.2, “Software Description.”

(b) Receiving Side

Table 2.11.4 Description of Registers Used (Receiving Side)

Module	Register	Setting Value	Function
H8SX/1544 initial settings	See 2.1, Initial Settings.		
RCAN initial settings			
RCAN mailbox initial settings	RCANET0.IMR.WORD	0xFEFD	Enables mailbox empty interrupt (IRR8) and data frame receive interrupt (IRR1) (IMR8 = 0, IMR1 = 0).
	RCANET0.MBIMR0.WORD	0xFFFD	Enables interrupts for mailbox 1.
	RCANET0.MB[1].CTRL0.WORD.H	(ID_1<<2)	Sets mailbox 1 to standard format, data frame. Also sets standard ID.
	RCANET0.MB[1].LAFM.WORD.H	0x0000	Sets filter mask for standard ID and IDE bit of mailbox 1.
	RCANET0.MB[1].CTRL1.BYTE.H	0x02	Sets mailbox 1 to receive.
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
H8SX/1544 interrupt settings			
Data frame receive interrupt (mailbox 1)	RCANET0.RXPR0.WORD	0x0002	Clears mailbox 1 receive-end flag. (Clearing condition: write 1)
	RCANET0.TXACK0.WORD	0x0002	Clears mailbox 1 transmit-end flag. (Clearing condition: write 1)
Mailbox reset	RCANET0.MCR.WORD	0x0002	Sets RCAN-ET to halt mode (MCR1 = 1).
	RCANET0.GSR.WORD	—	Confirms that RCAN-ET has transitioned to halt mode. (Confirms that GSR4 = 1.)
	RCANET0.IRR.WORD	—	Confirms that RCAN-ET has transitioned to halt mode. (Confirms that IRR0 = 1.)
	RCANET0.IRR.WORD	0x0001	Clears reset/halt/sleep interrupt flag (IRR0). (Clearing condition: write 1)
	RCANET0.MB[1].CTRL0.WORD.H	(ID_0<<2)	Changes status ID setting of mailbox 1 (ID_1 to ID_0).
	RCANET0.MB[1].CTRL1.BYTE.H	0x00	Changes MBC setting of mailbox 1 (receive to transmit).
	RCANET0.MB[1].CTRL1.BYTE.L	0x08	Sets the data length (8 bytes).
	RCANET0.MB[1].MSG_DATA[0] to RCANET0.MB[1].MSG_DATA[7]	0x02 × 8	Sets the transmit data.
	RCANET0.MCR.WORD	0xFFFD	Cancels RCAN-ET halt mode (MCR1 = 0).
	RCANET0.GSR.WORD	—	Confirms that RCAN-ET halt mode has been cancelled. (Confirms that GSR4 = 0.)
CAN message transmit	RCANET0.TXPR0.LONG	0x00000002	Sets mailbox 1 to transmit-wait status.

Notes: 1. The bit names listed in the table correspond to the bit names used in the hardware manual.
2. ID_0 and ID_1 indicate labels listed in (2), “Description of Constants Used,” in 2.11.2, “Software Description.”

(4) Description of RAM Used

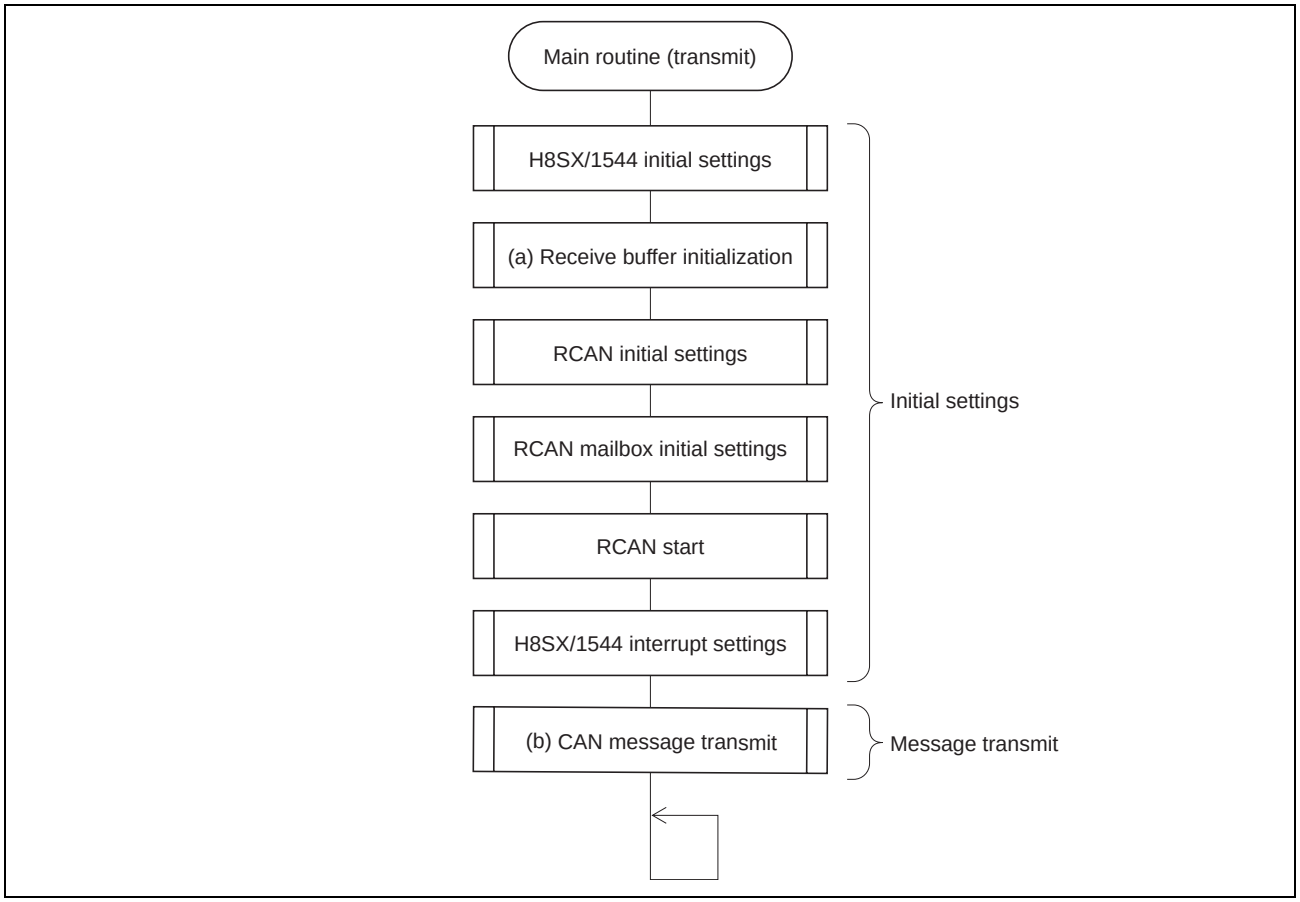
Table 2.11.5 Description of RAM Used

Module	Label	Function
Receive buffer initialization	Mbbuff.ID.LONG MBbuff.ID.WORD.H MBbuff.ID.WORD.L	Stores receive ID.
Data frame receive interrupt	MBbuff.DATA.LONG[0] to [1] MBbuff.DATA.WORD[0] to [3] MBbuff.DATA.BYTE[0] to [7]	Stores receive data.

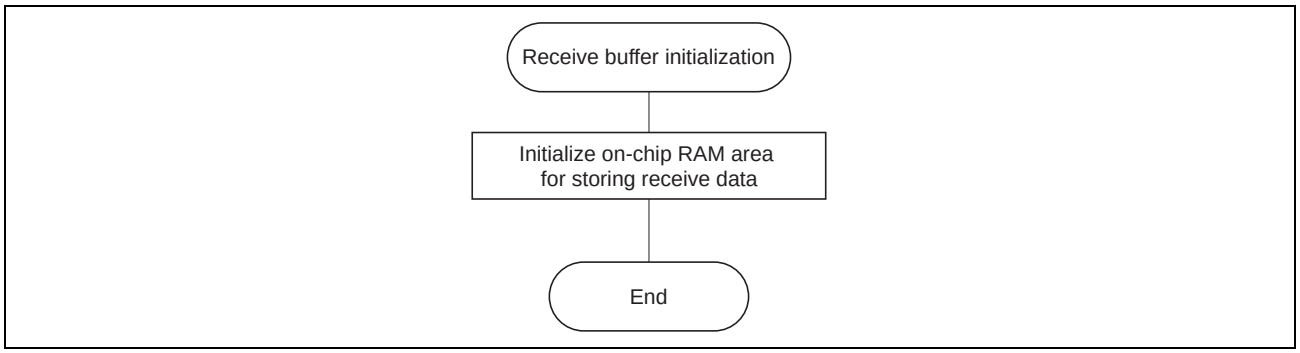
2.11.3 Flowcharts

(1) Transmitting Side Flowcharts

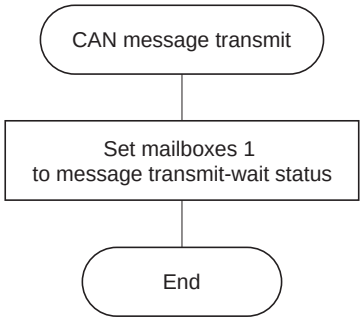
Main Routine



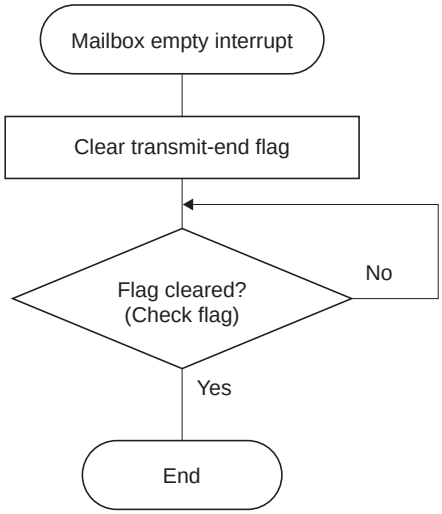
(a) Receive Buffer Initialization Routine (Device-Specific Settings)



(b) CAN Message Transmit Routine (RCAN-ET Common Settings)

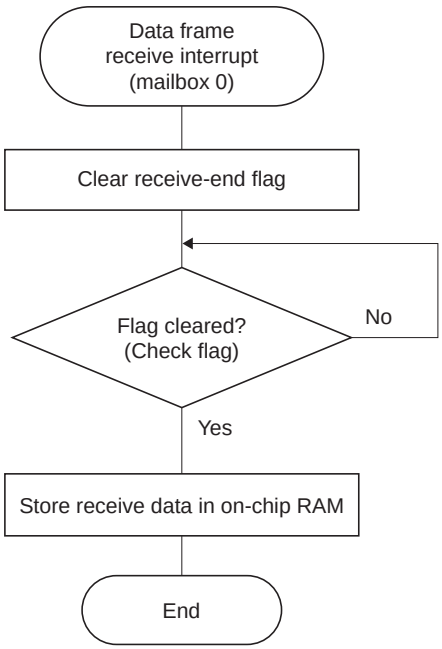


(c) Mailbox Empty Interrupt Routine (RCAN-ET Common Settings)



Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

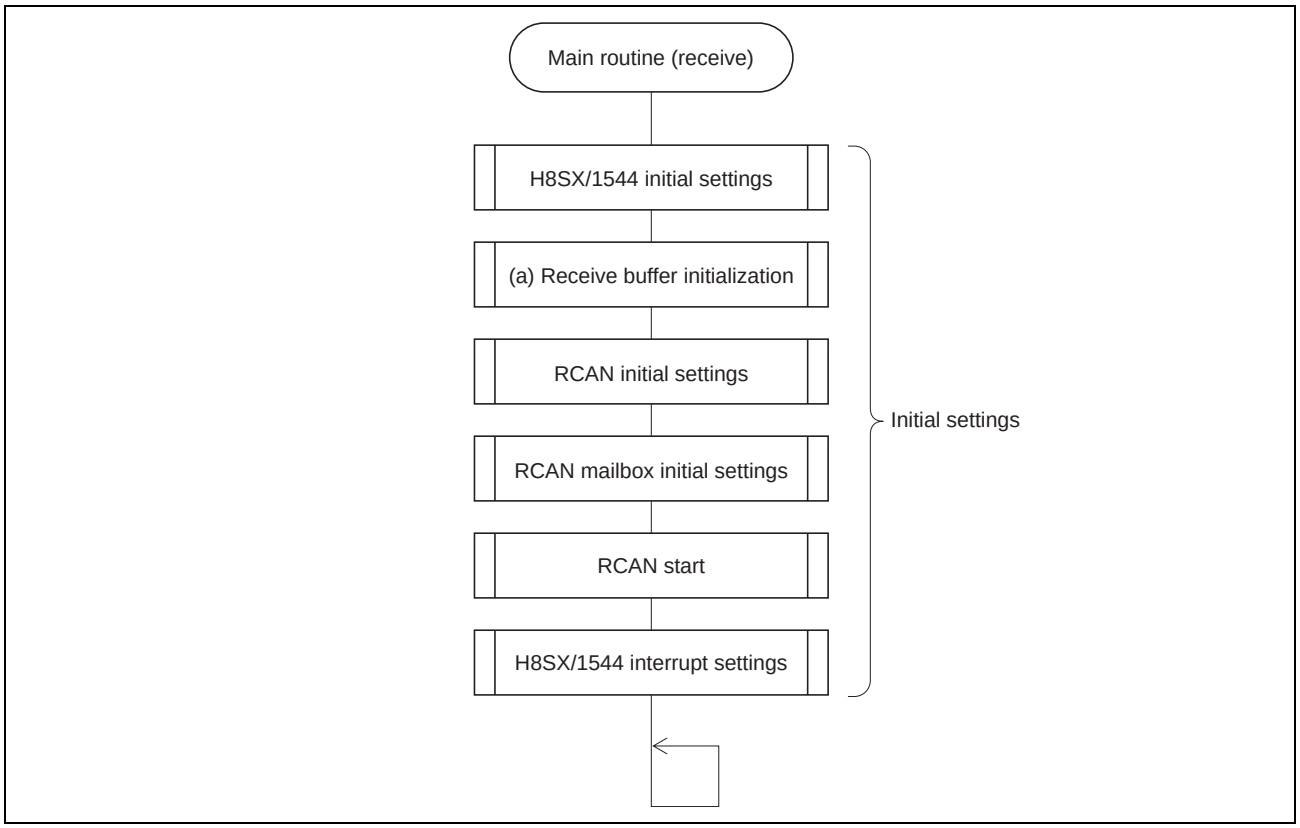
(d) Data Frame Receive Interrupt Routine (Mailbox 0, RCAN-ET Common Settings)



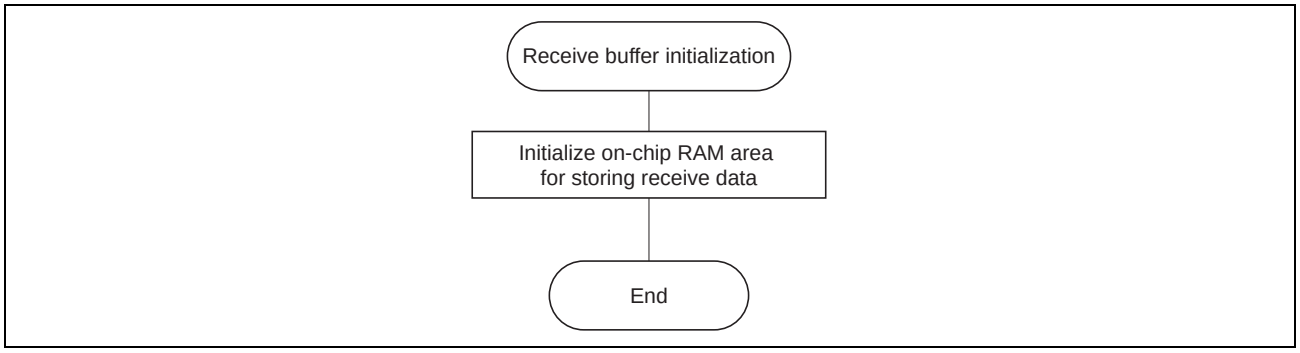
Note: To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

(2) Receiving Side Flowcharts

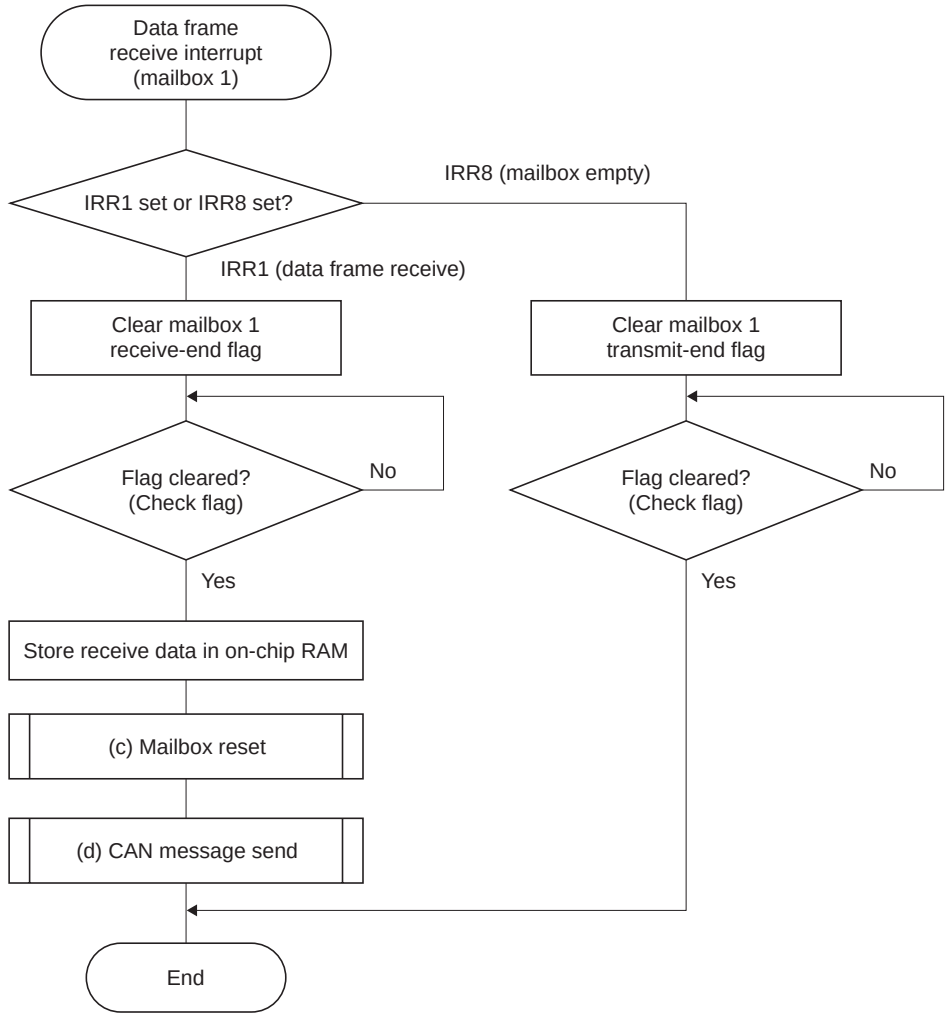
Main Routine



(a) Receive Buffer Initialization Routine (Device-Specific Settings)



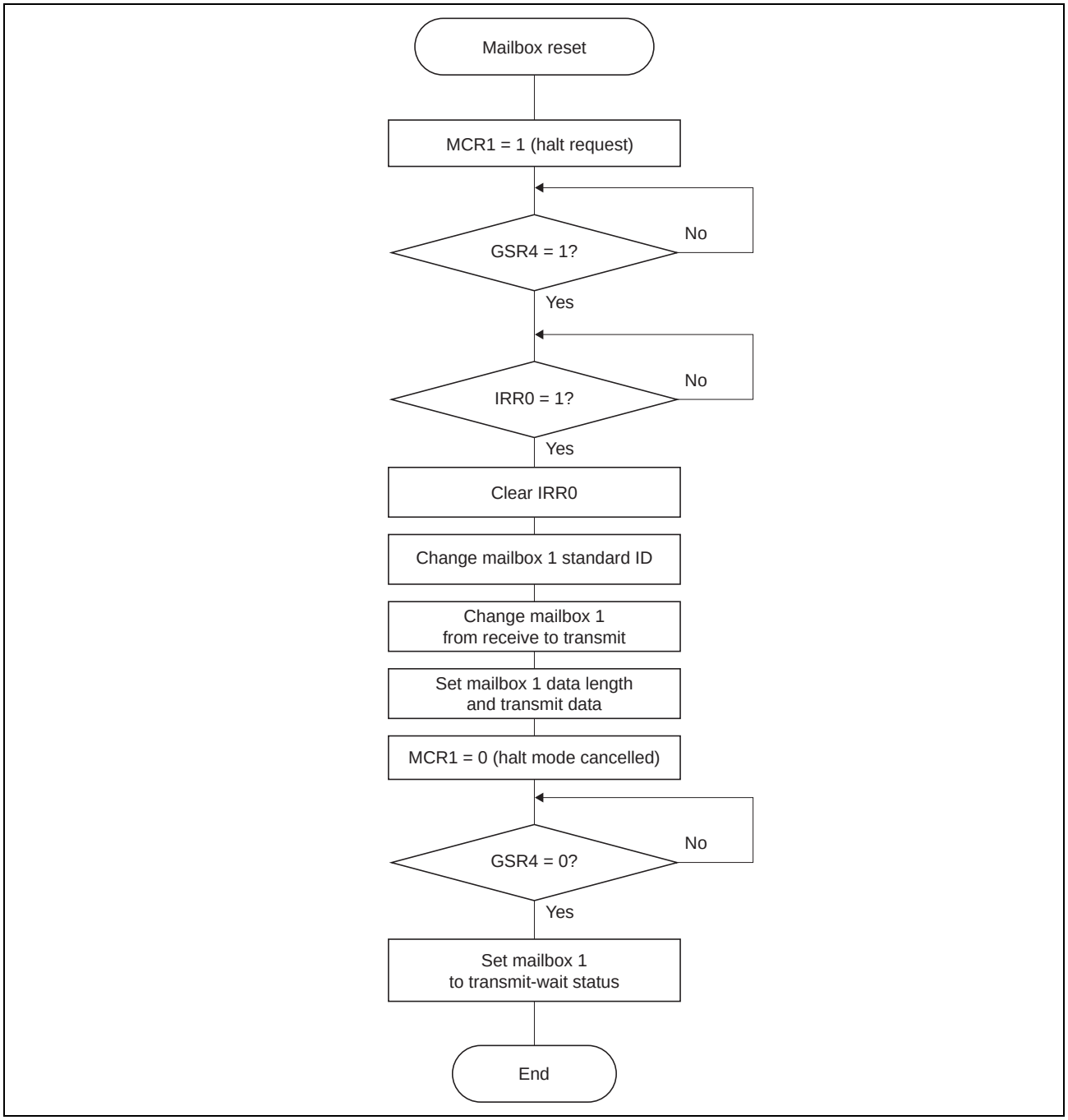
(b) Data Frame Receive Interrupt Routine (Mailbox 1, RCAN-ET Common Settings)



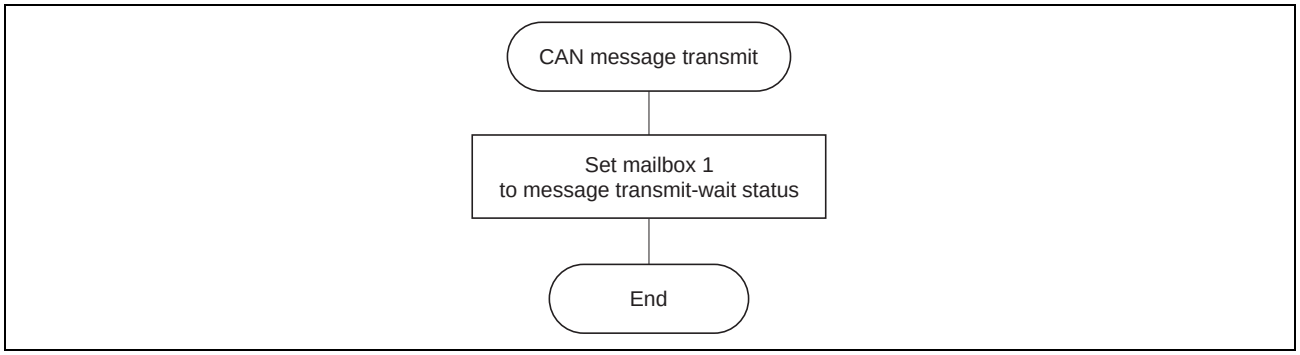
Notes: 1. Common interrupt vector addresses are used for all interrupt sources (RM1, SLE, OVR, and ERS) in the case of mailboxes other than mailbox 0. Therefore, in this operation example the same interrupt function is used for the data frame receive interrupt (IRR1) and the data frame transmit-end interrupt (IRR8).

2. To prevent cleared interrupt sources from erroneously being reaccepted, read the interrupt source flag after clearing it. Only execute the RTE instruction after confirming that the flag has been cleared.

(c) Mailbox Reset Routine (RCAN-ET Common Settings)



(d) CAN Message Transmit Routine (RCAN-ET Common Settings)



2.11.4 Program Listing

(1) Transmitting Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void RCAN0_Tx(void);
extern void Clear_MBbuff(void);
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{
    set_1544_init();                        /*  H8SX/1544 initialization                    */
    Clear_MBbuff();                         /*  Initialize RAM area for storing receive data */
    set_RCAN0_init();                       /*  RCAN initialization                         */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();                   /*  H8SX/1544 interrupt settings                */
    RCAN0_Tx();                             /*  CAN message transmit                        */
    while(1);

}

/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)       */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz)   */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz) */
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode                       */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN               */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins    */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin)                */
}
```

```
/*****************************************************************************/
/*      H8SX/1544 INTC Initialize routine                                    */
/*****************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;                /*   Interrupt control mode 2          */
    INTC.IPRQ.BIT._RCAN01 = 7;              /*   Set interrupt priority level: RCAN */
    set_imask_exr(0);                       /*   Interrupt mask level 0            */


}
/*****************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename    :  rcan.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void RCAN0_Tx(void);
void Clear_MBbuff(void);
/*---------------------------------------------------------------------------*/
struct {
    union {
        unsigned long LONG;
        struct{
            unsigned short H;
            unsigned short L;
        }WORD;
    }ID;
    union {
        unsigned char BYTE[8];
        unsigned short WORD[4];
        unsigned long LONG[2];
    }DATA;
} MBbuff;
/*---------------------------------------------------------------------------*/
#define ID_0        0x0AA
#define ID_1        0x1BB
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){
    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*  Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)    */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)  */
    RCANET0.IRR.WORD = 0x0001;           /*  Clear IRR0 (clearing condition: write 1)    */
    RCANET0.MCR.WORD |= 0x8000;          /*  Reorder ID: Set to MCR15 = 1 (initial setting)  */

    for(i = 0;i < 16;i++){               /*  Initialize mailbox (RAM area)               */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }
}
```

```
/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){

    unsigned int i;

    RCANET0.IMR.WORD &= 0xFEFD;    /*  Enable mailbox empty and data frame receive interrupts  */
    RCANET0.MBIMR0.WORD = 0xFFFC;                /*  Enable MB0-1 interrupt                  */

    RCANET0.MB[0].CTRL0.WORD.H = (ID_0<<2);      /*  Set STDID, standard format, data frame  */
    RCANET0.MB[0].LAFM.WORD.H = 0x0000;          /*  STD_LAFM and IDE_LAFM settings          */
    RCANET0.MB[0].CTRL1.BYTE.H = 0x02;           /*  Set mailbox 0 to receive                */

    RCANET0.MB[1].CTRL0.WORD.H = (ID_1<<2);      /*  Set STDID, standard format, data frame  */
    RCANET0.MB[1].CTRL1.BYTE.H = 0x00;           /*  Set mailbox 1 to transmit               */
    RCANET0.MB[1].CTRL1.BYTE.L = 0x08;           /*  Data length: 8 bytes                    */
    for(i = 0;i < 8;i++){
        RCANET0.MB[1].MSG_DATA[i] = 0x01;        /*  Transmit data                           */
    }

    /*  Bit rate = 500 kbps  */
    RCANET0.BCR1.WORD = 0x4300;/*  TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, (pϕ = 20 MHz) */
    RCANET0.BCR0.WORD = 0x0001;                  /*  BRP = 1                                 */


}

/*****************************************************************************/
/*      RCAN start routine                                                   */
/*****************************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                          /*  Clear MCR0              */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000);       /*  GSR3 = 0?               */


}



/*****************************************************************************/
/*      RCAN send message routine                                            */
/*****************************************************************************/
void RCAN0_Tx(void){

    RCANET0.TXPR0.LONG = 0x00000002;             /*  Set MB1 to transmit-wait status         */


}
```

```
/************************************************************************/
/*      RAM area Initialize routine                                     */
/************************************************************************/
void Clear_MBbuff(void){

    MBbuff.ID.LONG = 0;                         /*  Initialize RAM area for storing receive data*/
    MBbuff.DATA.LONG[0] = 0;
    MBbuff.DATA.LONG[1] = 0;

}


/************************************************************************/
/*      Mailbox Empty Interrupt routine                                 */
/************************************************************************/
#pragma interrupt(SLE0_0)
void SLE0_0(void)
{
    RCANET0.TXACK0.WORD = 0x0002;       /*  Clear transmit-end flag (clearing condition: write 1)   */
    while(RCANET0.TXACK0.WORD & 0x0002);        /*  Check flag                                      */


}


/************************************************************************/
/*      Data Frame Received Interrupt routine(MailBox0)                 */
/************************************************************************/
#pragma interrupt(RM0_0)
void RM0_0(void)
{
    unsigned int i;

    RCANET0.RXPR0.WORD = 0x0001;        /*  Clear receive-end flag (clearing condition: write 1)    */
    while(RCANET0.RXPR0.WORD & 0x0001);         /*  Check flag                                      */

    for(i = 0;i < 8;i++){
        MBbuff.DATA.BYTE[i] = RCANET0.MB[0].MSG_DATA[i];/* Store receive data in RAM */
    }


}
/************************************************************************/
```

(2) Receiving Side Program Listing

(a) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void main(void);
void set_1544_init(void);
void set_1544_INTC_init(void);
extern void set_RCAN0_init(void);
extern void set_RCAN0MB_init(void);
extern void set_RCAN0_start(void);
extern void Clear_MBbuff(void);
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma entry main(sp=0xFFC000)
void main(void)
{
    set_1544_init();                        /*  H8SX/1544 initialization                    */
    Clear_MBbuff();                         /*  Initialize RAM area for storing receive data */
    set_RCAN0_init();                       /*  RCAN initialization                         */
    set_RCAN0MB_init();
    set_RCAN0_start();
    set_1544_INTC_init();                   /*  H8SX/1544 interrupt settings                */
    while(1);

}

/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 0;                      /*  Set system clock (5 MHz × 8 = 40 MHz)       */
    SCKCR.BIT.PCK = 1;                      /*  Set peripheral clock (5 MHz × 4 = 20 MHz)   */
    SCKCR.BIT.BCK = 1;                      /*  Set external bus clock (5 MHz × 4 = 20 MHz) */
    SBYCR.BIT.SSBY = 1;                     /*  Software standby mode                       */
    MSTP.CRC.BIT._RCAN01 = 0;               /*  Cancel module-stop mode: RCAN               */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;            /*  Enable RCAN-ET transmit and receive pins    */
    P6.ICR.BIT.B4 = 1;                      /*  Set P64 as CRx_0 (input pin)                */

}
```

```
/*****************************************************************************/
/*      H8SX/1544 INTC Initialize routine                                    */
/*****************************************************************************/
void set_1544_INTC_init(void){

    INTC.INTCR.BIT.INTM = 2;                /*   Interrupt control mode 2          */
    INTC.IPRQ.BIT._RCAN01 = 7;              /*   Set interrupt priority level: RCAN */
    set_imask_exr(0);                       /*   Interrupt mask level 0            */


}
/*****************************************************************************/
```

(b) rcan.c

```
/*****************************************************************************/
/*  Filename    :  rcan.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include"iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
void set_RCAN0MB_change(void);
void RCAN0_Tx(void);
void Clear_MBbuff(void);
/*---------------------------------------------------------------------------*/
struct {
    union {
        unsigned long LONG;
        struct{
            unsigned short H;
            unsigned short L;
        }WORD;
    }ID;
    union {
        unsigned char BYTE[8];
        unsigned short WORD[4];
        unsigned long LONG[2];
    }DATA;
} MBbuff;
/*---------------------------------------------------------------------------*/
#define ID_0        0x0AA
#define ID_1        0x1BB
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*   Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)     */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt)   */
    RCANET0.IRR.WORD = 0x0001;           /*   Clear IRR0 (clearing condition: write 1)            */
    RCANET0.MCR.WORD |= 0x8000;          /*   Reorder ID: Set to MCR15 = 1 (initial setting)      */
```

```
    for(i = 0;i < 16;i++){              /*  Initialize mailbox (RAM area)                */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }


}

/***********************************************************************/
/*      RCAN Mailbox Initialize routine                                */
/***********************************************************************/
void set_RCAN0MB_init(void){

    unsigned int i;

    RCANET0.IMR.WORD &= 0xFEFD;      /*  Enable mailbox empty and data frame receive interrupts  */
    RCANET0.MBIMR0.WORD &= 0xFFFD;           /*  Enable MB1 interrupt                    */

    RCANET0.MB[1].CTRL0.WORD.H = (ID_1<<2);  /*  Set STDID, standard format, data frame  */
    RCANET0.MB[1].LAFM.WORD.H = 0x0000;      /*  STD_LAFM and IDE_LAFM settings          */
    RCANET0.MB[1].CTRL1.BYTE.H = 0x02;       /*  Set mailbox 1 to receive                */

    /*  Bit rate = 500 kbps  */
    RCANET0.BCR1.WORD = 0x4300;/*  TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, (pϕ = 20 MHz) */
    RCANET0.BCR0.WORD = 0x0001;              /*  BRP = 1                                 */


}

/***********************************************************************/
/*      RCAN start routine                                             */
/***********************************************************************/
void set_RCAN0_start(void){

    RCANET0.MCR.WORD &= 0xFFFE;                         /*  Clear MCR0          */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000);      /*  GSR3 = 0?           */


}
```

```
/*****************************************************************************/
/*      RCAN Mailbox Change routine                                          */
/*****************************************************************************/
void set_RCAN0MB_change(void){

    unsigned int i;

    /*   Mailbox reset   */
    RCANET0.MCR.WORD |= 0x0002;                    /*   Set MCR1 (halt request)            */
    while((RCANET0.GSR.WORD & 0x0010) != 0x0010);/*   GSR4 = 1? (transition to halt mode)  */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*   IRR0 = 1? (reset/halt/sleep interrupt) */
    RCANET0.IRR.WORD = 0x0001;                  /*   Clear IRR0 (clearing condition: write 1) */

    RCANET0.MB[1].CTRL0.WORD.H = (ID_0<<2);       /*   Change STDID setting               */
    RCANET0.MB[1].CTRL1.BYTE.H = 0x00;            /*   Change MBC setting (set to transmit)*/
    RCANET0.MB[1].CTRL1.BYTE.L = 0x08;            /*   Data length: 8 bytes               */
    for(i = 0;i < 8;i++){
        RCANET0.MB[1].MSG_DATA[i] = 0x02;         /*   Transmit data                      */
    }

    RCANET0.MCR.WORD &= 0xFFFD;                 /*   Clear MCR1 (halt mode cancel request)  */
    while ((RCANET0.GSR.WORD & 0x0010) != 0x0000);/*   GSR4 = 0? (halt mode cancelled)    */


}


/*****************************************************************************/
/*      RCAN send message routine                                            */
/*****************************************************************************/
void RCAN0_Tx(void){

    RCANET0.TXPR0.LONG = 0x00000002;          /*   Set MB1 to transmit-wait status        */


}


/*****************************************************************************/
/*      RAM area Initialize routine                                          */
/*****************************************************************************/
void Clear_MBbuff(void){

    MBbuff.ID.LONG = 0;                       /*   Initialize RAM area for storing receive data*/
    MBbuff.DATA.LONG[0] = 0;
    MBbuff.DATA.LONG[1] = 0;

}
```

```
/*****************************************************************************/
/*      Data Frame Received Interrupt routine(MailBox1-15)                   */
/*****************************************************************************/
#pragma interrupt(RM1_0)
void RM1_0(void)
{
    unsigned int i;

    if(RCANET0.RXPR0.WORD){                     /*   Set IRR1, MB1 receive                  */
        RCANET0.RXPR0.WORD = 0x0002;      /*   Clear receive-end flag (clearing condition: write 1) */
        while(RCANET0.RXPR0.WORD & 0x0002);   /*   Check flag                             */

        MBbuff.ID.WORD.H = RCANET0.MB[1].CTRL0.WORD.H;  /*   Store receive ID in RAM    */
        for(i = 0;i < 8;i++){                           /*   Store receive data in RAM  */
            MBbuff.DATA.BYTE[i] = RCANET0.MB[1].MSG_DATA[i];
        }

        set_RCAN0MB_change();                   /*   Reset mailbox                          */

        RCAN0_Tx();                             /*   Transmit CAN message                   */

    }
    else if(RCANET0.TXACK0.WORD){               /*   Set IRR8, MB1 transmit-end             */
        RCANET0.TXACK0.WORD = 0x0002;     /*   Clear transmit-end flag (clearing condition: write 1)*/
        while(RCANET0.TXACK0.WORD & 0x0002);  /*   Check flag                             */
    }

}
/*****************************************************************************/
```

2.12 Flash Memory Programming via CAN

2.12.1 Specification

As shown in figure 2.12.1, the slave device (H8SX/1544) receives data transferred from the master device via CAN, and the entire flash memory area of the slave device (512 Kbytes) is reprogrammed. User program mode is used to reprogram flash memory.

Note: This operation example describes the operation of the slave device.

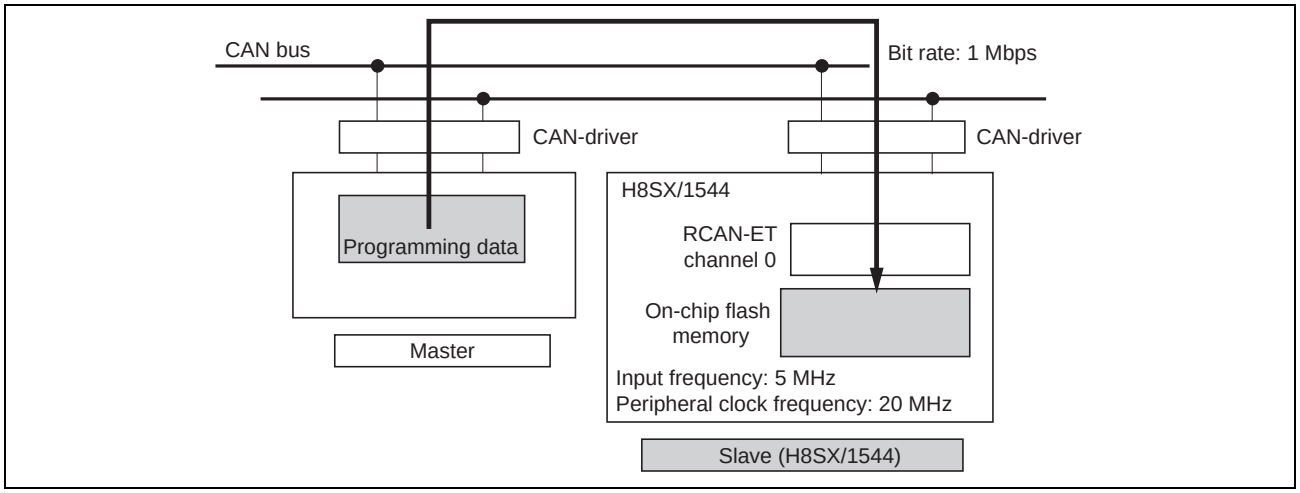


Figure 2.12.1 Specification

RCAN-ET Settings

- Bit rate: 1 Mbps (system clock = 20 MHz, peripheral module clock = 20 MHz)
- Interrupts: Not used
- Number of mailboxes used: 2
- Mailbox specifications: See table 2.12.1.

Table 2.12.1 Mailbox Specifications

Mailbox	Standard ID	Transmit/Receive Direction	Function
Mailbox 0	H'2AA	Receive	Receives commands and data.
Mailbox 1	H'555	Transmit	Transmits commands.

Note: This operation example uses commands (4-byte fixed data) to perform erase and programming operations. See (5), “Description of Constants Used,” in 2.12.3, “Software Description,” for information on command functions.

2.12.2 Operation Description

(1) Erase and Programming Procedure

Figures 2.12.2 to 2.12.7 show the flash memory erase and programming procedure used in this operation example.

1. The application program is started in user program mode. (The reprogramming control program must be prepared on the slave device, and the reprogramming data on the master device, beforehand.)

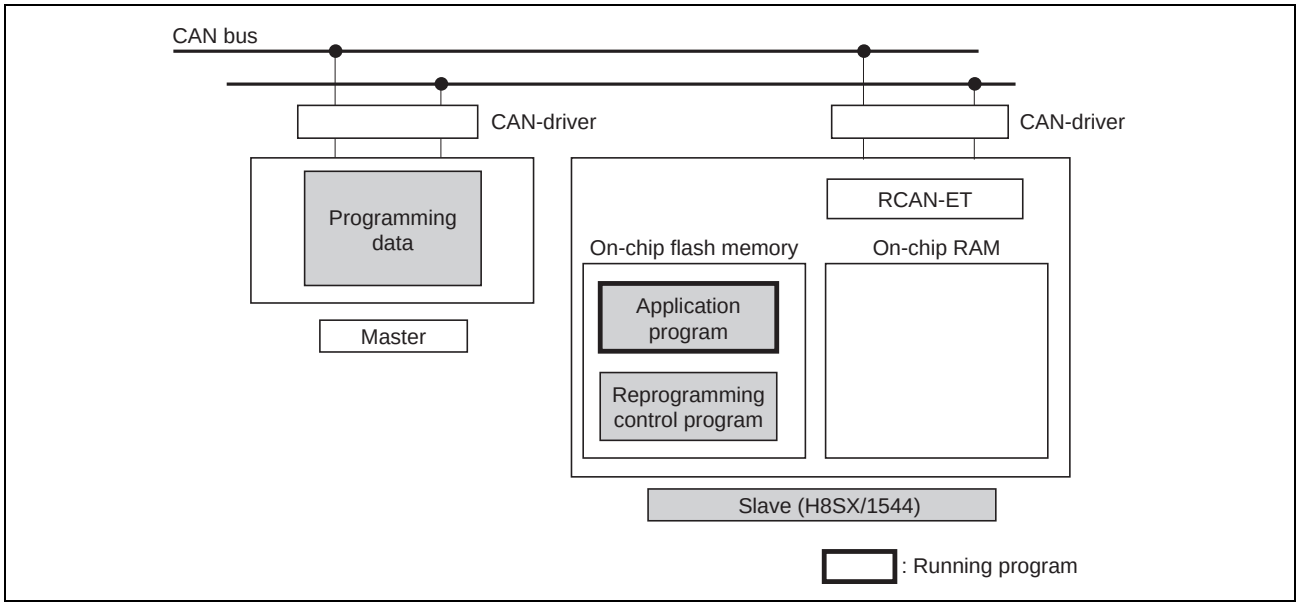


Figure 2.12.2 Erase and Programming Procedure (1)

2. The reprogramming control program is transferred to on-chip RAM.

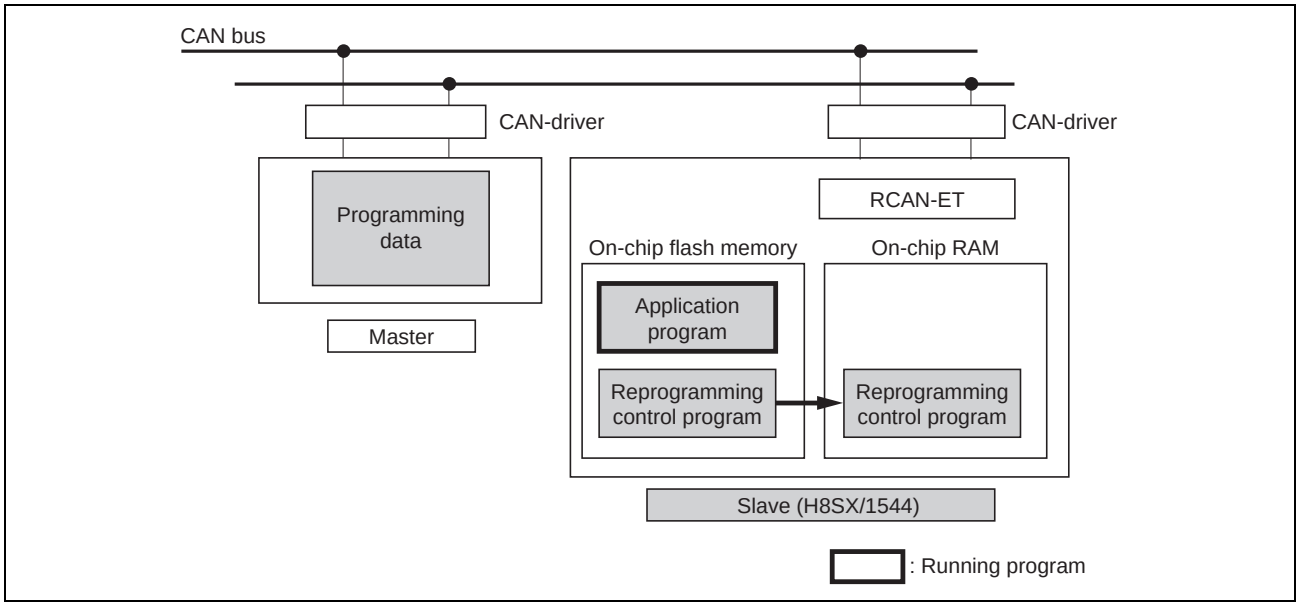


Figure 2.12.3 Erase and Programming Procedure (2)

3. The program in on-chip RAM is executed, and all programs in on-chip flash memory are erased.

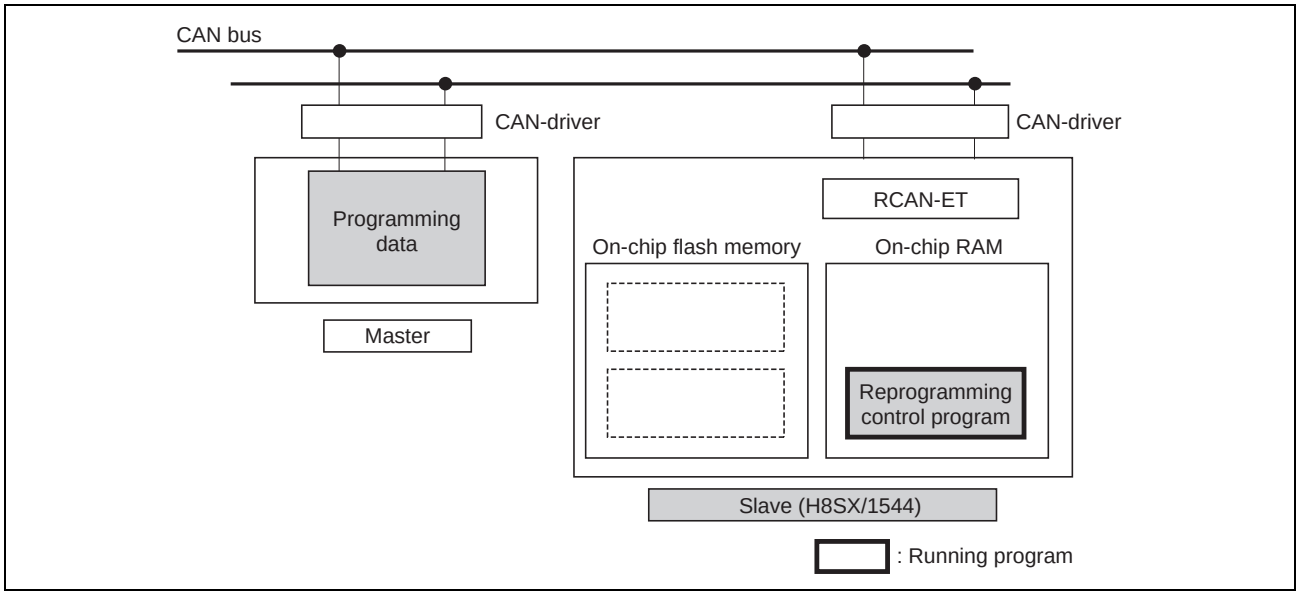


Figure 2.12.4 Erase and Programming Procedure (3)

4. Programming data (128 bytes) is transferred from the master to the slave device via CAN and stored in on-chip RAM.

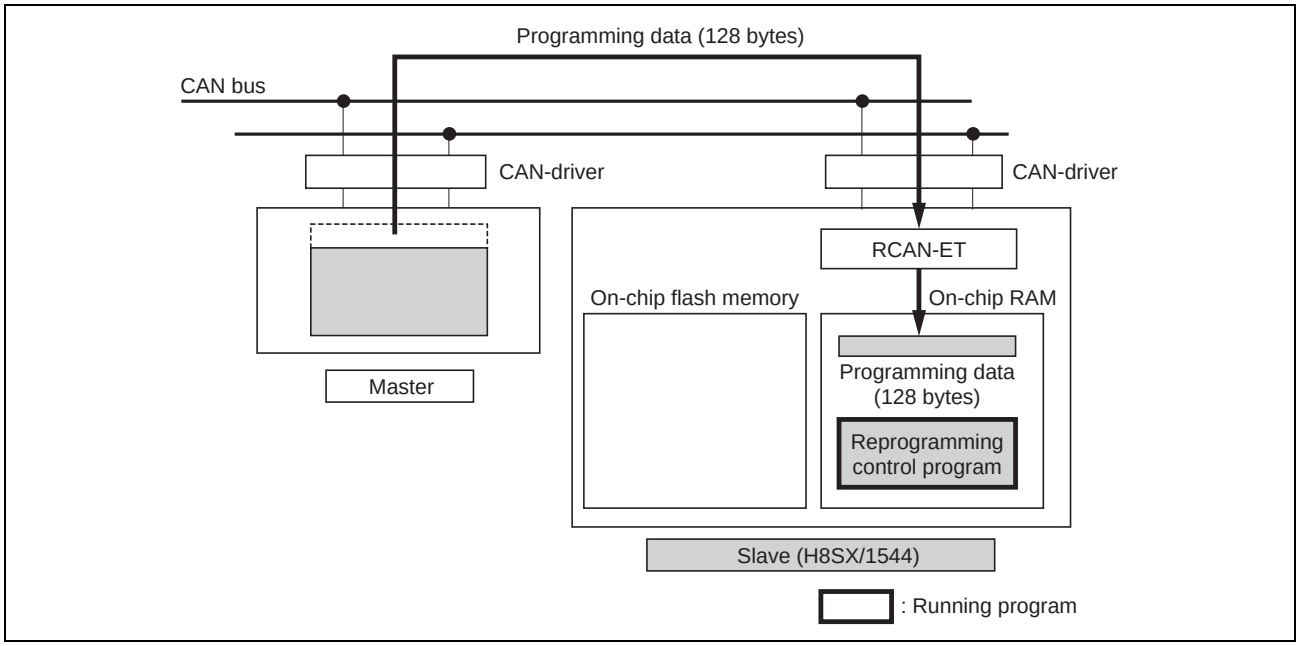


Figure 2.12.5 Erase and Programming Procedure (4)

5. The programming data (128 bytes) in on-chip RAM is written to on-chip flash memory.

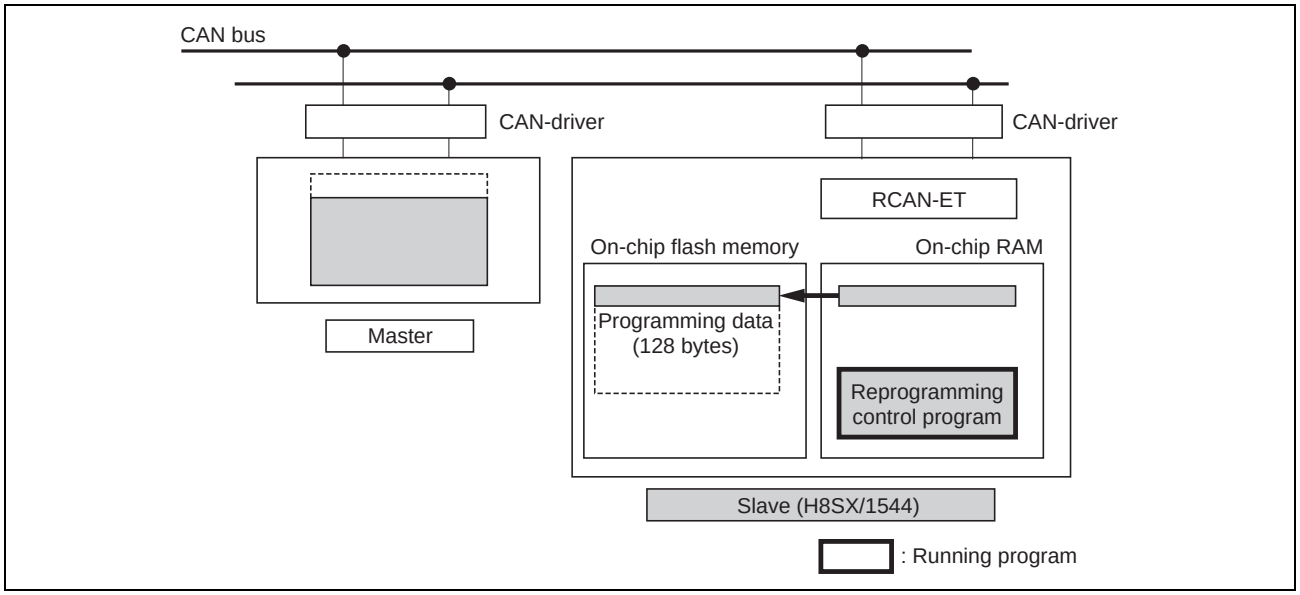


Figure 2.12.6 Erase and Programming Procedure (5)

6. Steps 4. and 5. are repeated until the entire flash memory area has been reprogrammed.

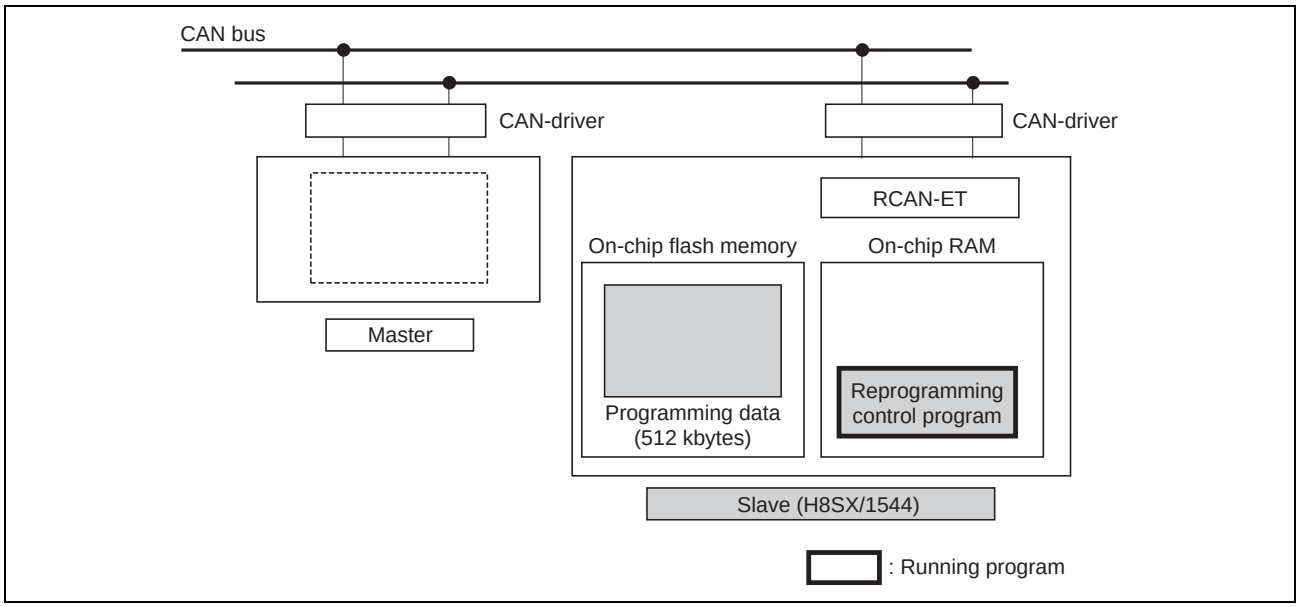


Figure 2.12.7 Erase and Programming Procedure (6)

(2) Command and Data Communication Operations during Erase and Programming

Figures 2.12.8 and 2.12.9 show the command and data communication operations during erase and programming. By writing a program for the master device that is tailored to the specifications of the slave device, it is possible to reprogram the flash memory of the slave device.

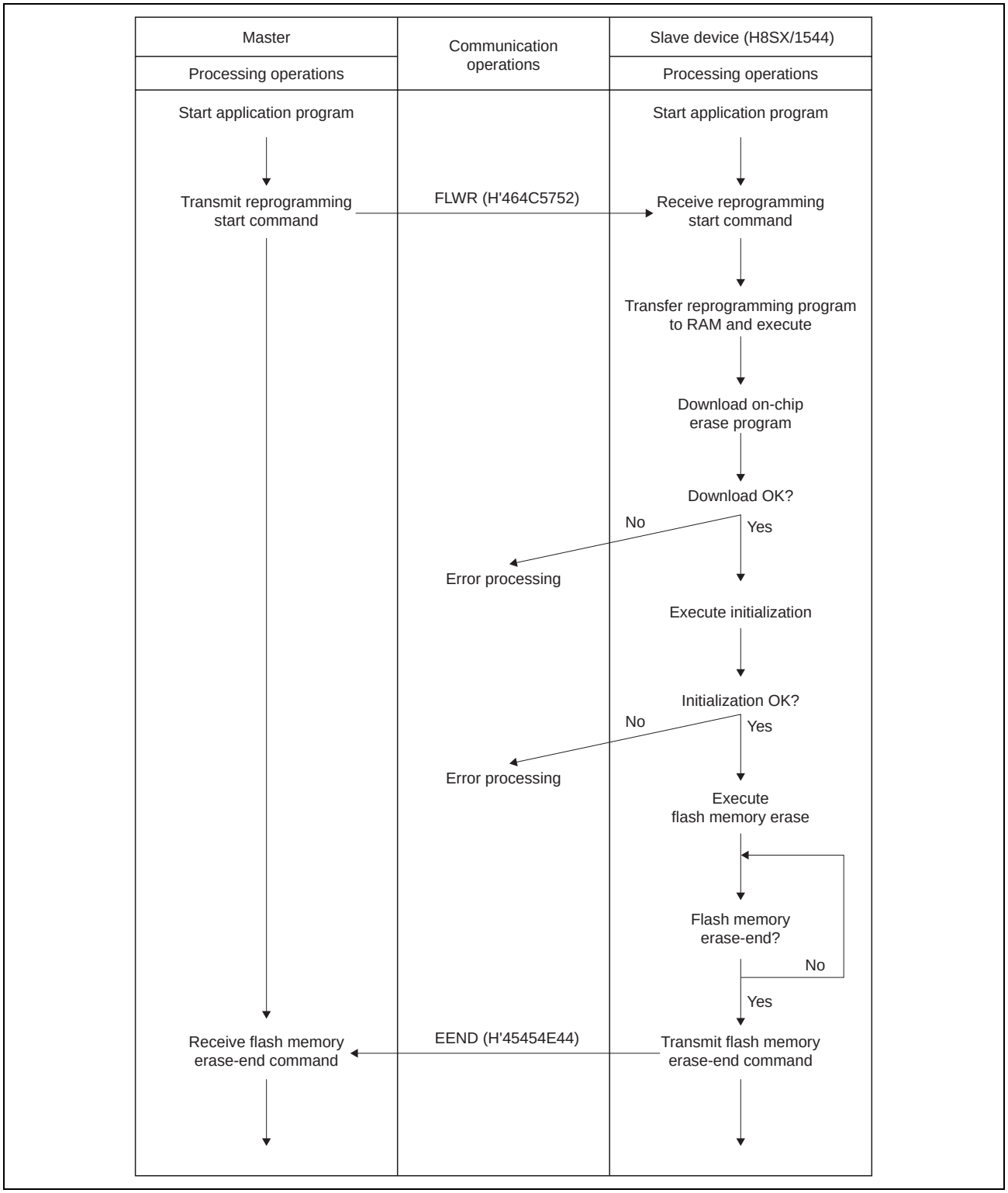


Figure 2.12.8 Communication Operations

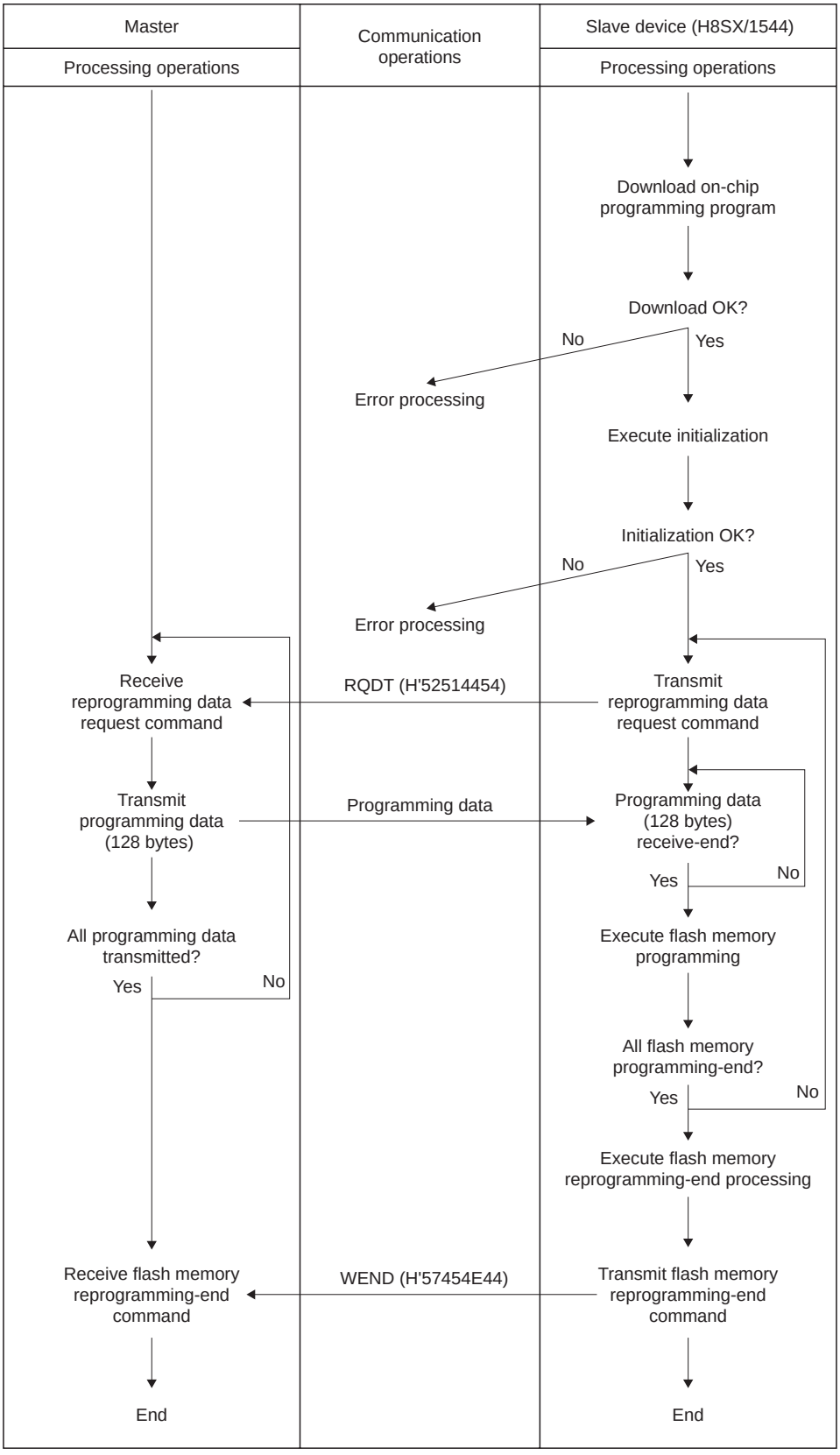


Figure 2.12.8 Communication Operations (cont)

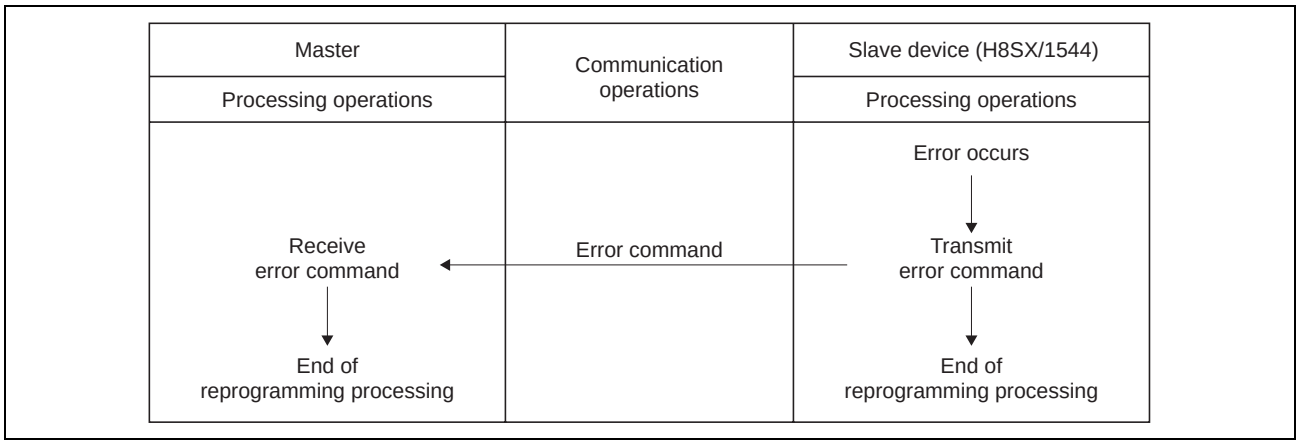


Figure 2.12.9 Communication Operations (Error Processing)

2.12.3 Software Description

(1) Module Description

Table 2.12.2 Module Description

Module	Label	Function	Settings
H8SX/1544 initial settings	set_1544_init	See 2.1, Initial Settings.	Device-specific
RCAN initial settings	set_RCAN0_init		RCAN-ET common
RCAN mailbox initial settings	set_RCAN0MB_init		
RCAN start	set_RCAN0_start		
Reprogramming start command receive routine	CAN_RV_MB0	Receives flash memory reprogramming start command via CAN.	
Programming data receive routine	CAN_RV_MB0_128B	Receives flash memory programming data via CAN and stores it in on-chip RAM.	
Reprogramming control command transmit routine	CAN_TR_MB1	Transmits flash memory reprogramming control commands via CAN.	
Flash memory erase/programming processing routine	procedure	Erases and programs flash memory.	Device-specific
Dummy routine for erase/programming initialization	FLASH_INIT	Dummy routine to jump to on-chip initialization program.	
Dummy routine for erase	FLASH_ERASE	Dummy routine to jump to on-chip erase program.	
Dummy routine for programming	FLASH_WRITE	Dummy routine to jump to on-chip programming program.	

(2) Description of Arguments

Table 2.12.3 Description of Arguments

Label	Function	Size	Module Used
FPEFEQ	Sets CPU operating frequency.	unsigned long	FLASH_INIT
FEBS	Specifies erase block number	unsigned long	FLASH_ERASE
FMPAR	Programming destination start address	unsigned long	FLASH_WRITE
FMPDR	Data storage area (programming source) start address	unsigned long	
work	Sets transmit command	unsigned long	CAN_TR_MB1

(3) Description of Return Values

Table 2.12.4 Description of Return Values

Label	Function	Size	Module Used
ERR_CHECK	Result confirmation parameters for erase/programming initialization, erase processing, and programming processing	unsigned char	FLASH_INIT FLASH_ERASE FLASH_WRITE

(4) Description of Variables Used

Table 2.12.5 Description of Variables Used

Label	Function	Size	Module Used
_BMAIN_BGN	Start address of flash memory reprogramming control routine	unsigned char*	main
_BMAIN_END	End address of flash memory reprogramming control routine	unsigned char*	
_BMAIN_RAM	Transfer destination address of flash memory reprogramming control routine	unsigned char*	
Command	Stores receive command.	unsigned long	
ERASE_block	Sets erase block.	unsigned short	main procedure

(5) Description of Constants Used

Table 2.12.6 Description of Constants Used

Label	Function	Setting Value	Module Used
RAM90	Download destination start address of on-chip erase/programming program (FTDAR register setting value)	0x00	procedure
RAM90_TOP	Download destination start area of on-chip erase/programming program	0xFF9000	
FLWR	Flash memory reprogramming start command	0x464C5752	main procedure
EDER	Download error command of on-chip erase program	0x45444552	
EIER	Error command at erase start setting	0x45494552	
ERER	Error command at erase execute	0x45524552	
EEND	Flash memory erase-end command	0x45454E44	
WDER	Download error command of on-chip programming program	0x57444552	
WIER	Error command at programming start setting	0x57494552	
WRER	Error command at programming execute	0x57524552	
RQDT	Programming data request command	0x52514454	
WEND	Programming-end command	0x57454E44	

(6) Description of Registers Used

Table 2.12.7 Description of Registers Used

Module	Register	Setting Value	Function
H8SX/1544 initial settings	SCKCR.BIT.ICK	1	Sets the system clock frequency to 20 MHz (input clock: 5 MHz × 4).
	SCKCR.BIT.BCK	1	Sets the peripheral module clock frequency to 20 MHz (input clock: 5 MHz × 4).
	SCKCR.BIT.PCK	1	Sets the external bus clock frequency to 20 MHz (input clock: 5 MHz × 4).
	SBYCR.BIT.SSBY	1	Enables transition to software standby mode after execution of Sleep instruction.
	MSTP.CRC.BIT._RCAN01	0	Cancels RCAN-ET module-stop mode.
	RCANET0.RCANMON.BYTE	0x20	Enables RCAN-ET transmit and receive pins.
	P6.ICR.BIT.B4	1	Sets P64 (pin 67) as RCAN-ET channel 0 receive pin.
RCAN initial settings	See 2.1, Initial Settings.		
RCAN mailbox initial settings	RCANET0.MB[0].CTRL0.WORD.H	0x0AA8	Sets mailbox 0 to standard format, data frame. Also sets standard ID (H'2AA).
	RCANET0.MB[0].LAFM.WORD.H	0x0000	Sets filter mask for standard ID and IDE bit of mailbox 0.
	RCANET0.MB[0].CTRL1.BYTE.H	0x02	Sets mailbox 0 to receive.
	RCANET0.MB[1].CTRL0.WORD.H	0x1554	Sets mailbox 1 to standard format, data frame. Also sets standard ID (H'555).
	RCANET0.MB[1].CTRL1.BYTE.H	0x00	Sets mailbox 1 to transmit.
	RCANET0.MB[1].CTRL1.BYTE.L	0x04	Sets the data length (4 bytes)
	RCANET0.BCR1.WORD	0x4300	Sets to 500 kbps when Pϕ = 20 MHz. (TSEG1 = 4 (5 tq), TSEG2 = 3 (4 tq), SJW = 0, BSP = 0, BRP = 1)
	RCANET0.BCR0.WORD	0x0001	
RCAN start	See 2.1, Initial Settings.		
Reprogramming start command receive routine	RCANET0.RXPR0.WORD	—	Mailbox 0 waits for receive-end. (Flag is polled until bit 0 in RXPR0 is set to 1.)
	RCANET0.RXPR0.WORD	0x0001	Clears mailbox 0 receive-end flag. (Clearing condition: write 1)

Module	Register	Setting Value	Function
Flash memory erase/programming processing routine	FLASH.FTDAR.BYTE	RAM90 (= 0x00)	Sets download destination start address of on-chip program to H'FF9000.
	FLASH.FECS.BYTE	0x01	Selects erase program to download. (Sets EPVB bit.)
	FLASH.FPCS.BYTE	0x01	Selects programming program to download. (Sets PPVS bit.)
	FLASH.FCCS.BYTE	0x01	Enables download of erase/programming program. (Sets SC0.)
	FLASH.FKEY	0xA5	Enables read of SC0 bit in FCCS.
		0x5A	Enables flash memory erase/programming.
		0x00	Software protect.
	RAM90_TOP (= DPFR)	—	Parameter for checking for on-chip erase/programming program error (0: normal operation, other than 0: error).
	FPEFEQ	0x000007D0	Parameter for setting CPU operating frequency. (Sets operating frequency to 20 MHz.)
	ERR_CHECK (= FPFR)	—	Parameter for checking result of erase/programming initialization, erase processing, and programming processing (0: normal operation, other than 0: error).
	FMPAR	0x00000000	Parameter for setting programming destination start address. (Sets start address to H'0.)
	FMPDR	0xFFB100	Parameter for setting programming source start address. (Sets start address to H'FFB100.)
	FEBS	0x00000000	Parameter for setting erase block number.
Reprogramming start command transmit routine	RCANET0.TXPR0.LONG	0x00000002	Sets mailbox 1 to transmit-wait status.
	RCANET0.TXACK0.WORD	—	Mailbox 1 waits for transmit-end. (Flag is polled until bit 1 in TXACK0 is set to 1.)
	RCANET0.TXACK0.WORD	0x0002	Clears mailbox 1 transmit-end flag. (Clearing condition: write 1)

Note: The bit names listed in the table correspond to the bit names used in the hardware manual.

(7) Memory Map

Figure 2.12.10 shows the memory map for the this operation example.

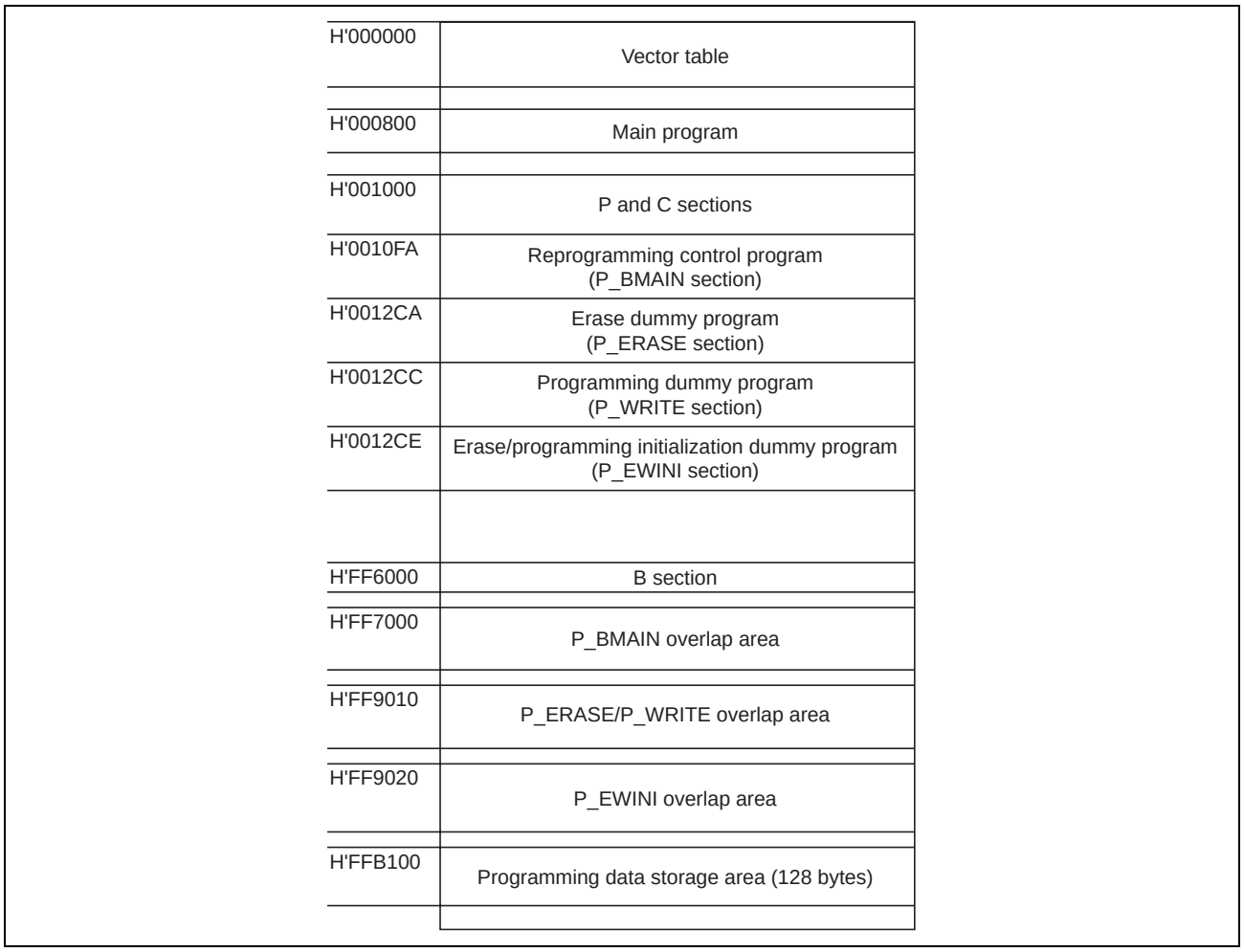
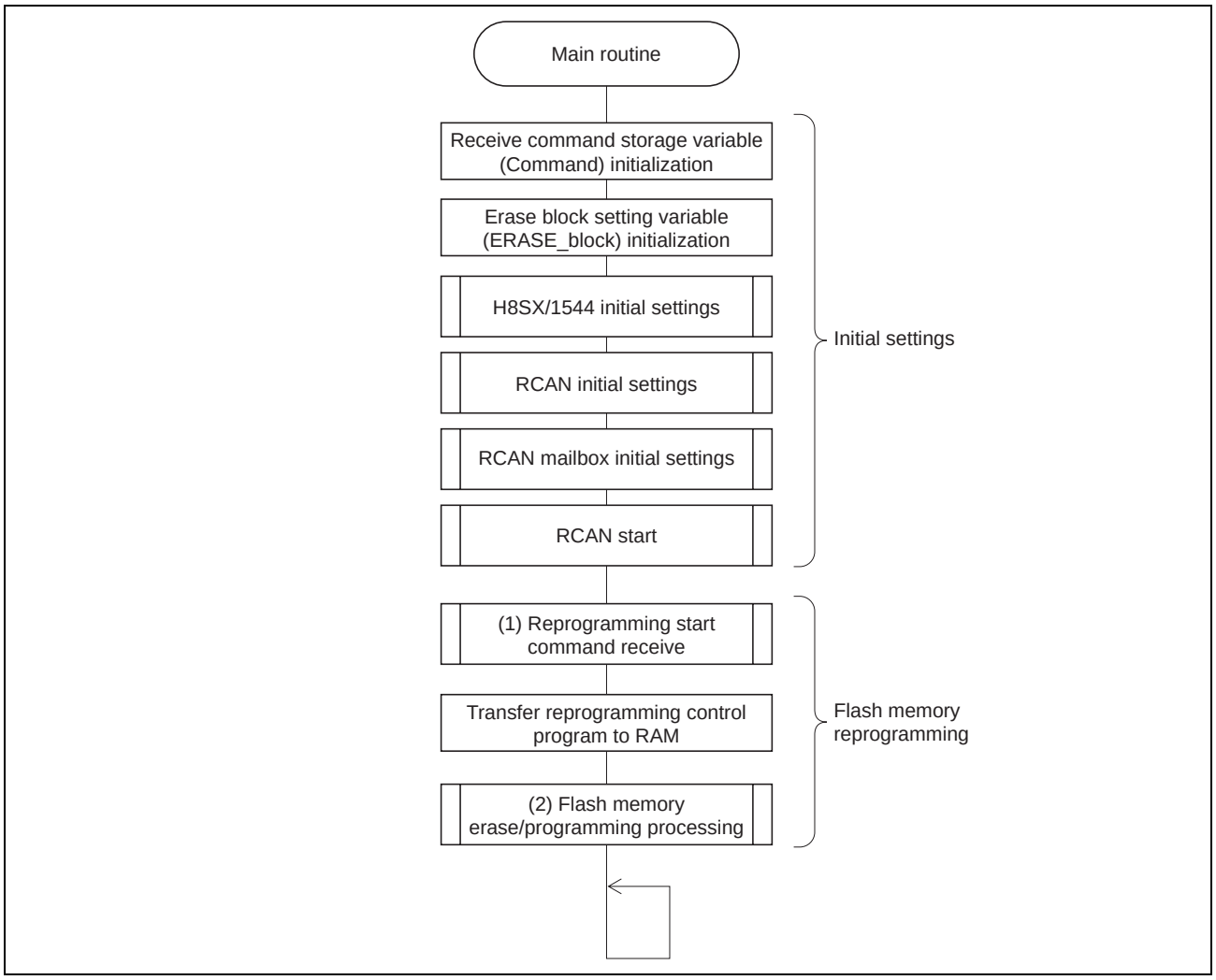


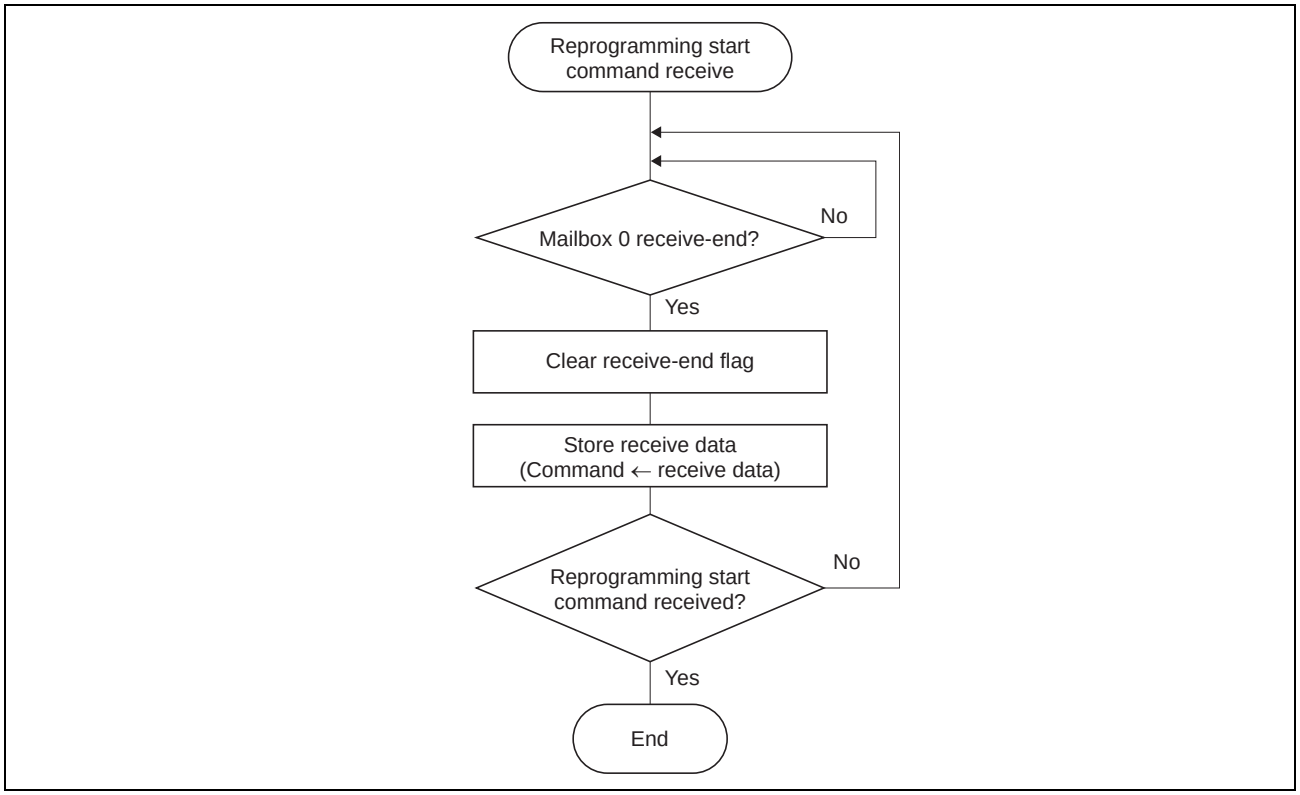
Figure 2.12.10 Memory Map

2.12.4 Flowcharts

Main Routine

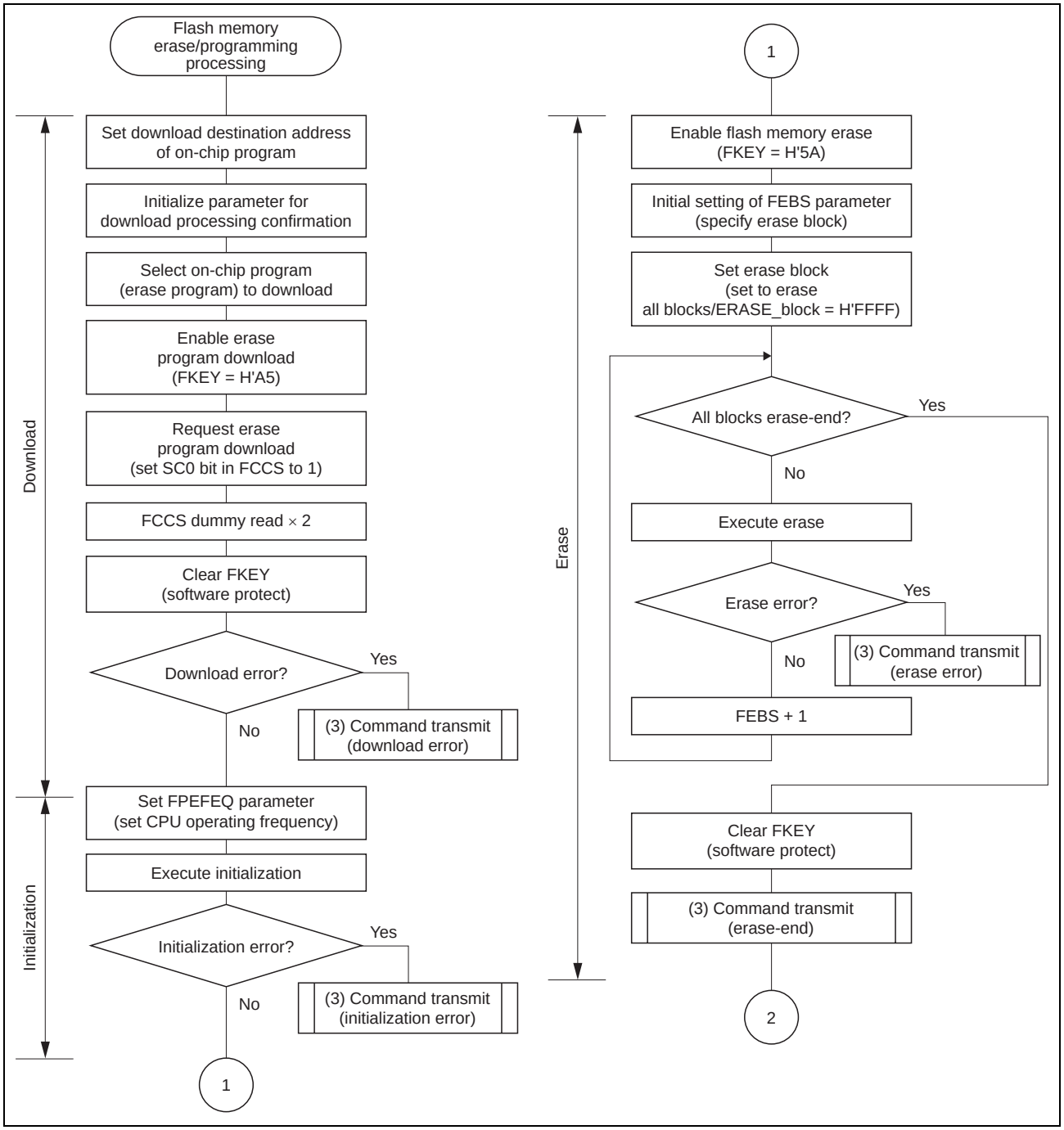


(1) Reprogramming Start Command Receive Routine (RCAN-ET Common Settings)

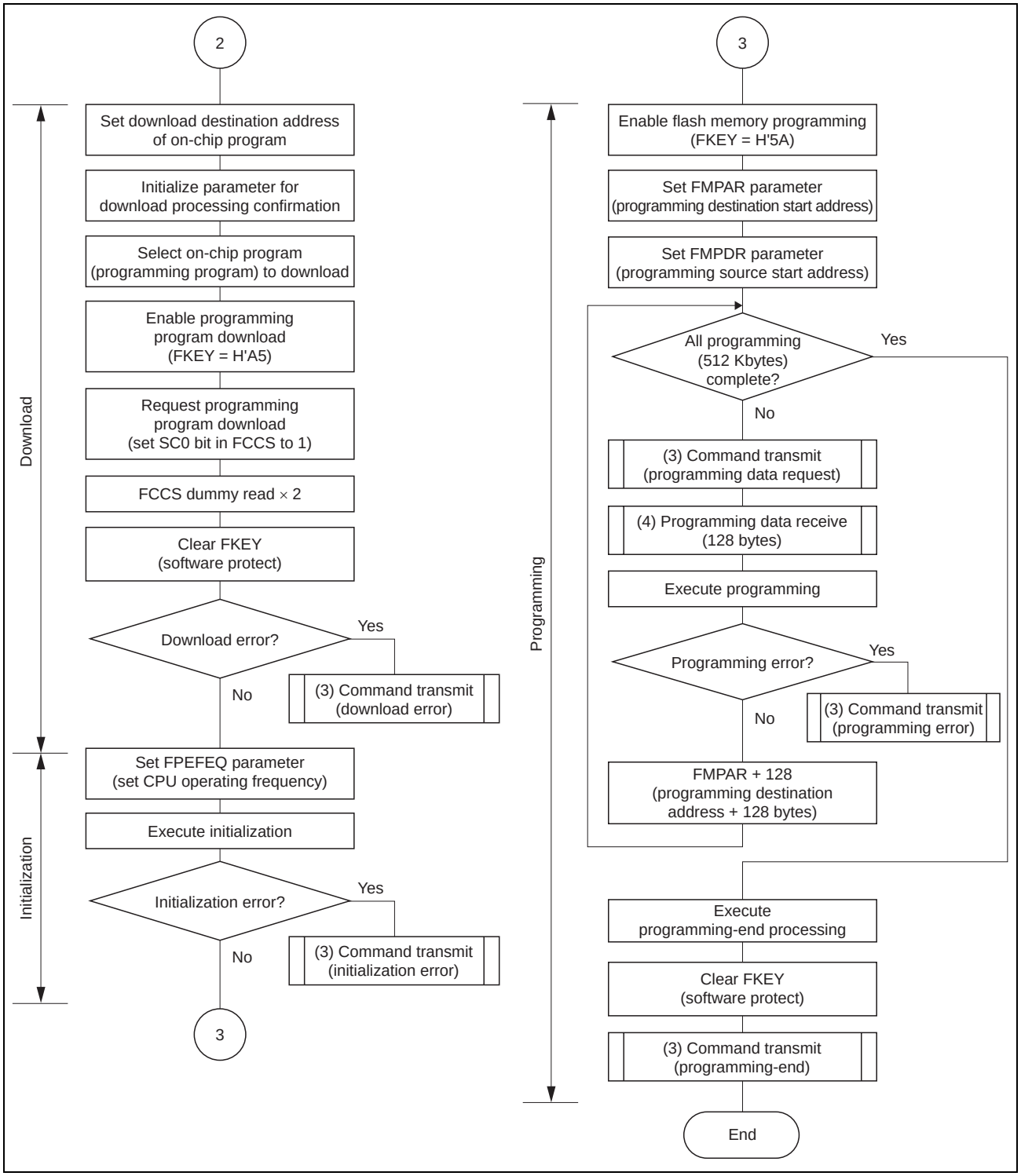


(2) Flash Memory Erase/Programming Processing Routine (Device-Specific Settings)

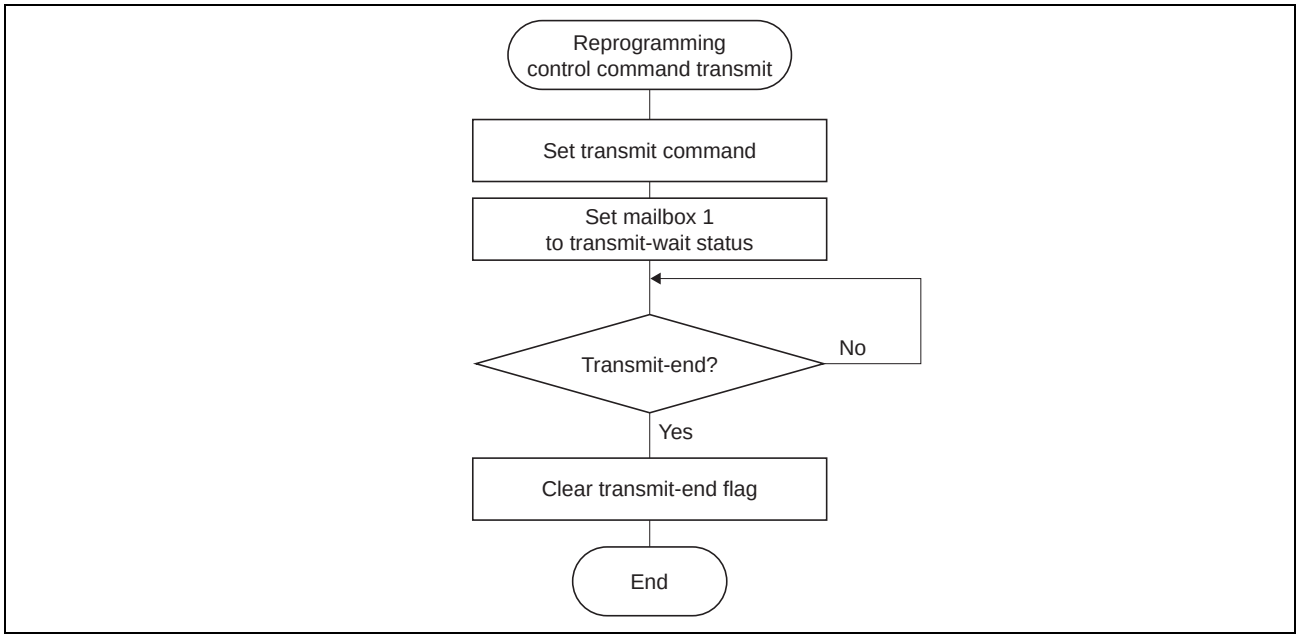
(a) Erase



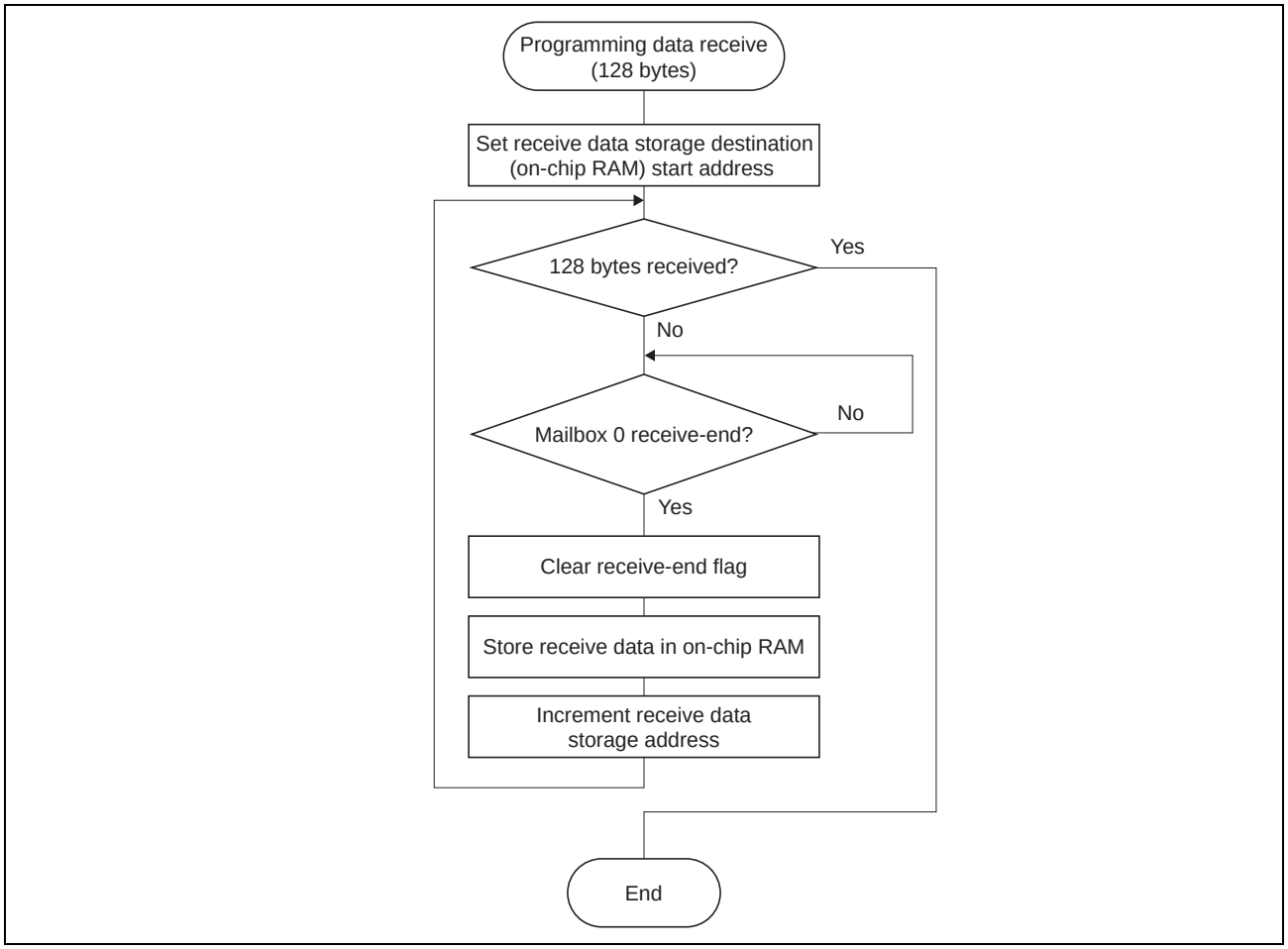
(b) Programming



(3) Reprogramming Control Command Transmit Routine (RCAN-ET Common Settings)



(4) Programming Data Receive Routine (RCAN-ET Common Settings)



2.12.5 Program Listing

(1) main.c

```
/*****************************************************************************/
/*  Filename    :  main.c                                                    */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/
#include <machine.h>
#include "iodefine.h"
#include "reg.h"
/*---------------------------------------------------------------------------*/
void main(void);                        /*  main                             */
void set_1544_init(void);               /*  H8SX/1544 initialization         */
void CAN_RV_MB0(void);                  /*  Receive reprogramming start command */
extern void set_RCAN0_init(void);       /*  RCAN initialization              */
extern void set_RCAN0MB_init(void);     /*  RCAN initialization              */
extern void set_RCAN0_start(void);      /*  RCAN initialization              */
extern void procedure(void);            /*  Programming control program      */
/*---------------------------------------------------------------------------*/
unsigned long Command;                  /*  Receive command storage variable */
unsigned short ERASE_block;             /*  Erase block setting variable     */
extern unsigned char *_BMAIN_BGN;       /*  Programming control program start address */
extern unsigned char *_BMAIN_END;       /*  Programming control program end address */
extern unsigned char *_BMAIN_RAM;       /*  Programming control program transfer destination address */
/*****************************************************************************/
/*      Main program                                                         */
/*****************************************************************************/
#pragma section _MAIN
#pragma entry main(sp=0xFFC000)
void main(void)
{

    unsigned int i;
    unsigned char *src, *dst;

    /*** INIT Value ***/
    Command = 0;                        /*  Receive command storage variable initialization */
    ERASE_block = 0;                    /*  Erase block setting variable initialization */

    /*** H8SX/1544 init ***/
    set_1544_init();                    /*  H8SX/1544 initialization         */

    /*** RCAN init ***/
    set_RCAN0_init();                   /*  RCAN initialization              */
    set_RCAN0MB_init();
    set_RCAN0_start();

    /*** Receive reprogramming start command ***/
    CAN_RV_MB0();
```

```
    /*** Transfer to RAM ***/
    for(src=_BMAIN_BGN, dst=_BMAIN_RAM; src<=_BMAIN_END; src++, dst++) {
        *dst = *src;                            /*  Transfer reprogramming program to RAM    */
    }

    /*** Program flash memory ***/
    procedure();                                /*  Execute programming control program      */

    while(1);

}


/*****************************************************************************/
/*      H8SX/1544 Initialize routine                                         */
/*****************************************************************************/
void set_1544_init(void){

    /*** SYSTEM ***/
    SCKCR.BIT.ICK = 1;                          /*  Set system clock (5 MHz × 4 = 20 MHz)    */
    SCKCR.BIT.PCK = 1;                          /*  Set peripheral clock (5 MHz × 4 = 20 MHz)*/
    SCKCR.BIT.BCK = 1;                          /*  Set external bus clock (5 MHz × 4 = 20 MHz)*/
    SBYCR.BIT.SSBY = 1;                         /*  Software standby mode                    */
    MSTP.CRC.BIT._RCAN01 = 0;                   /*  Cancel module-stop mode: RCAN            */

    /*** IO ***/
    RCANET0.RCANMON.BYTE = 0x20;                /*  Enable RCAN-ET transmit and receive pins */
    P6.ICR.BIT.B4 = 1;                          /*  Set P64 as CRx_0 (input pin)             */

}


/*****************************************************************************/
/*  receive command routine                                                  */
/*****************************************************************************/
void CAN_RV_MB0(void){

    unsigned int i;

    do {
        while ((RCANET0.RXPR0.WORD & 0x0001) != 0x0001); /*  Wait for receive-end    */
        RCANET0.RXPR0.WORD = 0x0001;      /*  Clear receive-end flag (clearing condition: write 1) */
        for(i = 0;i < 4;i++){
            Command <<= 8;
            Command |= RCANET0.MB[0].MSG_DATA[i];       /*  Store receive data       */
        }
    } while(Command != FLWR);         /*  Programming start command (FLWR command) received? */
}
/*****************************************************************************/
```

(2) rcan.c

```
/*****************************************************************************/
/*  Filename    :   rcan.c                                                   */
/*  Written     :   '06/06/01   REV.1.00                                     */
/*  Purpose     :   for H8SX/1544   RCAN-ET                                  */
/*****************************************************************************/
#include "iodefine.h"
/*---------------------------------------------------------------------------*/
void set_RCAN0_init(void);
void set_RCAN0MB_init(void);
void set_RCAN0_start(void);
/*****************************************************************************/
/*      RCAN Initialize routine                                              */
/*****************************************************************************/
void set_RCAN0_init(void){

    unsigned int i,j;

    RCANET0.MCR.WORD |= 0x0001;          /*  Reset request (set automatically by a hardware reset) */
    while((RCANET0.GSR.WORD & 0x0008) != 0x0008);/*  GSR3 = 1? (RCAN-ET reset status)  */
    while((RCANET0.IRR.WORD & 0x0001) != 0x0001);/*  IRR0 = 1? (reset/halt/sleep interrupt) */
    RCANET0.IRR.WORD = 0x0001;           /*  Clear IRR0 (clearing condition: write 1)  */
    RCANET0.MCR.WORD |= 0x8000;          /*  Reorder ID: Set to MCR15 = 1 (initial setting)  */

    for(i = 0;i < 16;i++){               /*  Initialize mailbox (RAM area)  */
        RCANET0.MB[i].CTRL0.WORD.H = 0;
        RCANET0.MB[i].CTRL0.WORD.L = 0;
        RCANET0.MB[i].LAFM.WORD.H = 0;
        RCANET0.MB[i].LAFM.WORD.L = 0;
        for(j = 0;j < 8;j++){
            RCANET0.MB[i].MSG_DATA[j] = 0;
        }
    }

}
/*****************************************************************************/
/*      RCAN Mailbox Initialize routine                                      */
/*****************************************************************************/
void set_RCAN0MB_init(void){


    RCANET0.MB[0].CTRL0.WORD.H = 0x0AA8;    /*  ID: H'2AA, standard format, data frame  */
    RCANET0.MB[0].LAFM.WORD.H = 0x0000;     /*  STD_LAFM and IDE_LAFM settings  */
    RCANET0.MB[0].CTRL1.BYTE.H = 0x02;      /*  Set mailbox 0 to receive  */

    RCANET0.MB[1].CTRL0.WORD.H = 0x1554;    /*  ID: H'555, standard format, data frame  */
    RCANET0.MB[1].CTRL1.BYTE.H = 0x00;      /*  Set mailbox 1 to transmit  */
    RCANET0.MB[1].CTRL1.BYTE.L = 0x04;      /*  Data length: 4 bytes  */
```

```
    /*  Bit rate = 1Mbps  */
    RCANET0.BCR1.WORD = 0x4300;/*  TSEG1 = 3 (4 tq), TSEG2 = 2 (3 tq), SJW = 0, BSP = 0, (pϕ = 20 MHz) */
    RCANET0.BCR0.WORD = 0x0000;                /*    BRP = 0                                          */

}

/*****************************************************************************/
/*      RCAN start routine                                                   */
/*****************************************************************************/
void set_RCAN0_start(void){


    RCANET0.MCR.WORD &= 0xFFFE;                          /*   Clear MCR0          */
    while ((RCANET0.GSR.WORD & 0x0008) != 0x0000);       /*   GSR3 = 0?           */


}
/*****************************************************************************/
```

(3) procedure.c

```
/*****************************************************************************/
/*  Filename    :  procedure.c                                                */
/*  Written     :  '06/06/01   REV.1.00                                       */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                    */
/*****************************************************************************/
#include "iodefine.h"
#include "reg.h"
/*---------------------------------------------------------------------------*/
void procedure(void);                       /*  Programming control program          */
void CAN_TR_MB1(unsigned long work);        /*  Transmit programming/erase control commands */
void CAN_RV_MB0_128B(void);                 /*  Receive programming data (128 bytes) */
extern unsigned char FLASH_INIT(unsigned long FPEFEQ);  /*  Dummy function      */
extern unsigned char FLASH_ERASE(unsigned long FEBS);   /*  Dummy function      */
extern unsigned char FLASH_WRITE(unsigned long FMPDR,unsigned long FMPAR);
                                                        /*  Dummy function      */
/*---------------------------------------------------------------------------*/
extern unsigned long Command;               /*  Receive command storage variable    */
extern unsigned short ERASE_block;          /*  Erase block setting variable        */
/*****************************************************************************/
/*  procedure program routine                                                */
/*****************************************************************************/
#pragma section _BMAIN
void procedure(void){

    unsigned long FPEFEQ;               /*  Programming/erase parameter         */
    unsigned long FEBS;                 /*  Programming/erase parameter         */
    unsigned long FMPAR;                /*  Programming/erase parameter         */
    unsigned long FMPDR;                /*  Programming/erase parameter         */
    unsigned char ERR_CHECK;            /*  Programming/erase parameter         */
    unsigned char buff;                 /*  FCCS dummy read variable            */

/*** Erase ***/
    /*  Download processing     */
    FLASH.FTDAR.BYTE = RAM90;           /*  Set download destination address (H'FF9000)     */
    RAM90_TOP = 0xFF;                   /*  Clear download processing confirmation parameter */
    FLASH.FECS.BYTE = 0x01;             /*  Select erase program (set EPVB bit)             */
    FLASH.FKEY = 0xA5;                  /*  Enable erase program download                   */
    FLASH.FCCS.BYTE |= 0x01;            /*  Request download (set SC0 bit)                  */
    buff = FLASH.FCCS.BYTE;             /*  FCCS dummy read × 2                             */
    buff = FLASH.FCCS.BYTE;
    FLASH.FKEY = 0x00;                  /*  Clear FKEY (software protect)                   */
    if(RAM90_TOP != 0x00){              /*  Download error?                                 */
        CAN_TR_MB1(EDER);               /*  Transmit download error command (EDER command)  */
        while(1);
    }
    /*  Initial settings    */
    FPEFEQ = 0x000007D0;                /*  Set CPU operating frequency: 20 MHz (H'07D0)    */
    ERR_CHECK = FLASH_INIT(FPEFEQ);     /*  Execute initialization                          */
    if(ERR_CHECK != 0){                 /*  Initial setting error?                          */
        CAN_TR_MB1(EIER);               /*  Transmit initial setting error command (EIER command) */
        while(1);
    }
```

```
    /*   Erase processing      */
    FLASH.FKEY = 0x5A;                          /*   Enable flash memory erase                    */
    FEBS = 0x00000000;                          /*   Set parameter specifying erase program       */
    ERASE_block = 0xFFFF;                       /*   Set erase program: Erase all blocks          */
    while(ERASE_block != 0x0000){               /*   All blocks erased?                           */
        if((ERASE_block & 0x0001)==0x0001){
            ERR_CHECK = FLASH_ERASE(FEBS);/*   Execute erase                                */
            if(ERR_CHECK != 0){                 /*   Erase error?                                 */
                CAN_TR_MB1(ERER);               /*   Transmit erase error command (ERER command)*/
                while(1);
            }
        }
        ERASE_block = (ERASE_block>>1);
        FEBS++;
    }
    FLASH.FKEY = 0x00;                          /*   Clear FKEY (software protect)                */
    CAN_TR_MB1(EEND);                           /*   Transmit erase-end command (EEND command)    */
/*** Programming ***/
    /* Download processing */
    FLASH.FTDAR.BYTE = RAM90;                   /*   Set download destination address (H'FF9000)  */
    RAM90_TOP = 0xFF;                           /*   Clear download processing confirmation parameter*/
    FLASH.FPCS.BYTE = 0x01;                     /*   Select programming program (set PPVS bit)    */
    FLASH.FKEY = 0xA5;                          /*   Enable programming program download           */
    FLASH.FCCS.BYTE |= 0x01;                    /*   Request download (set SC0 bit)               */
    buff = FLASH.FCCS.BYTE;                     /*   FCCS dummy read × 2                          */
    buff = FLASH.FCCS.BYTE;
    FLASH.FKEY = 0x00;                          /*   Clear FKEY (software protect)                */
    if(RAM90_TOP != 0x00){                      /*   Download error?                              */
        CAN_TR_MB1(WDER);               /*   Transmit download error command (WDER command)  */
        while(1);
    }
    /* Initial settings */
    FPEFEQ = 0x000007D0;                        /*   Set CPU operating frequency: 20 MHz (H'07D0) */
    ERR_CHECK = FLASH_INIT(FPEFEQ);             /*   Execute initialization                       */
    if(ERR_CHECK != 0){                         /*   Initial setting error?                       */
        CAN_TR_MB1(WIER);               /*   Transmit initial setting error command (WIER command)*/
        while(1);
    }
    /* Programming processing */
    FLASH.FKEY = 0x5A;                          /*   Enable flash memory programming              */
    FMPAR = 0x00000000;                         /*   Programming destination start address        */
    FMPDR = 0xFFB100;                           /*   Programming source start address             */
    while(FMPAR < 0x00080000){                  /*   All programming complete (512 Kbytes)?       */
        CAN_TR_MB1(RQDT);           /* Transmit programming data request command (RQDT command) */
        CAN_RV_MB0_128B();                      /*   Receive programming data (128 bytes)         */
        ERR_CHECK = FLASH_WRITE(FMPDR,FMPAR);/*   Execute programming                          */
        if(ERR_CHECK != 0){                     /*   Programming error?                           */
            CAN_TR_MB1(WRER);       /* Transmit programming error command (WRER command)        */
            while(1);
        }
        FMPAR += 0x80;
    }
```

```
    FLASH_WRITE(0xF0F0F0F0,0x0F0F0F0F);          /*  Execute programming-end processing     */
    FLASH.FKEY = 0x00;                          /*  Clear FKEY (software protect)          */

    CAN_TR_MB1(WEND);                  /*  Transmit programming-end command (WEND command) */

    while(1);
}


/*************************************************************************/
/*  send command routine                                                 */
/*************************************************************************/
void CAN_TR_MB1(unsigned long work){

    unsigned int i;

    for(i = 4;i > 0;i--){                       /*  Set transmit data                      */
        RCANET0.MB[1].MSG_DATA[i-1] = (unsigned char)(work&0x000000FF);
        work >>= 8;
    }
    RCANET0.TXPR0.LONG = 0x00000002;            /*  Set MB1 to transmit-wait status        */
    while ((RCANET0.TXACK0.WORD & 0x0002) != 0x0002); /*  Wait for transmit-end          */
    RCANET0.TXACK0.WORD = RCANET0.TXACK0.WORD;
                                        /*  Clear transmit-end flag (clearing condition: write 1) */


}


/*************************************************************************/
/*  receive write data routine                                           */
/*************************************************************************/
void CAN_RV_MB0_128B(void){

    unsigned char *buff;
    unsigned int i,j;

    buff = (unsigned char*)0xFFB100;            /*  Receive data storage destination address */
    for(i = 0;i < 16; i++){                     /*  8 bytes × 16 = 128 bytes               */
        while (RCANET0.RXPR0.WORD != 0x0001); /*  Wait for receive-end                   */
        RCANET0.RXPR0.WORD = 0x0001;     /*  Clear receive-end flag (clearing condition: write 1) */
        for(j = 0;j < 8; j++){
            *buff = RCANET0.MB[0].MSG_DATA[j];/*  Store receive data                     */
            buff++;
        }
    }

}
/*************************************************************************/
```

(4) flash.c

```
/*****************************************************************************/
/*  Filename    :   flash.c                                                  */
/*  Written     :   '06/06/01   REV.1.00                                     */
/*  Purpose     :   for H8SX/1544   RCAN-ET                                  */
/*****************************************************************************/
unsigned char FLASH_INIT(unsigned long FPEFEQ);
                                          /*  Erase/programming initialization dummy function */
unsigned char FLASH_ERASE(unsigned long FEBS);   /*  Erase dummy function            */
unsigned char FLASH_WRITE(unsigned long FMPDR,unsigned long FMPAR);
                                                  /*  Programming dummy function      */
/*****************************************************************************/
/*  flash initialize dummy routine                                           */
/*****************************************************************************/
#pragma section _EWINI
unsigned char FLASH_INIT(unsigned long FPEFEQ){

}

/*****************************************************************************/
/*  flash erase dummy routine                                                */
/*****************************************************************************/
#pragma section _ERASE
unsigned char FLASH_ERASE(unsigned long FEBS){

}

/*****************************************************************************/
/*  flash write dummy routine                                                */
/*****************************************************************************/
#pragma section _WRITE
unsigned char FLASH_WRITE(unsigned long FMPDR,unsigned long FMPAR){

}
/*****************************************************************************/
```

(5) reg.h

```
/*****************************************************************************/
/*  Filename    :  reg.h                                                     */
/*  Written     :  '06/06/01   REV.1.00                                      */
/*  Purpose     :  for H8SX/1544   RCAN-ET                                   */
/*****************************************************************************/

/***  Programming/erase control commands  ***/
#define FLWR  0x464C5752
#define EDER  0x45444552
#define EIER  0x45494552
#define ERER  0x45524552
#define EEND  0x45454E44
#define WDER  0x57444552
#define WIER  0x57494552
#define WRER  0x57524552
#define RQDT  0x52514454
#define WEND  0x57454E44

/***  On-chip program download destination addresses  ***/
#define RAM90  0x00
#define RAM90_TOP   (*(volatile unsigned char  *)0xFF9000)
#define RAMA0  0x01
#define RAMA0_TOP   (*(volatile unsigned char  *)0xFFA000)
#define RAMB0  0x02
#define RAMB0_TOP   (*(volatile unsigned char  *)0xFFB000)
```

Website and Support

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

Inquiries
http://www.renesas.com/inquiry
csc@renesas.com

Revision Record

		Description	
Revision History	**Date**	**Page**	**Summary**
1.00	July 24, 2007	—	First edition issued

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, 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 products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life
   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas 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 applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.