

H8S/2623F-ZTAT™ On-Chip HCAN (Hitachi Controller Area Network)

Application Note

HITACHI

ADE-502-071

Rev. 1.0

1/28/2000

Hitachi, Ltd.

Cautions

1. Hitachi neither warrants nor grants licenses of any rights of Hitachi's or any third party's patent, copyright, trademark, or other intellectual property rights for information contained in this document. Hitachi bears no responsibility for problems that may arise with third party's rights, including intellectual property rights, in connection with use of the information contained in this document.
2. Products and product specifications may be subject to change without notice. Confirm that you have received the latest product standards or specifications before final design, purchase or use.
3. Hitachi makes every attempt to ensure that its products are of high quality and reliability. However, contact Hitachi's sales office before using the product in an application that demands especially high quality and reliability or where its failure or malfunction may directly threaten human life or cause risk of bodily injury, such as aerospace, aeronautics, nuclear power, combustion control, transportation, traffic, safety equipment or medical equipment for life support.
4. Design your application so that the product is used within the ranges guaranteed by Hitachi particularly for maximum rating, operating supply voltage range, heat radiation characteristics, installation conditions and other characteristics. Hitachi bears no responsibility for failure or damage when used beyond the guaranteed ranges. Even within the guaranteed ranges, consider normally foreseeable failure rates or failure modes in semiconductor devices and employ systemic measures such as fail-safes, so that the equipment incorporating Hitachi product does not cause bodily injury, fire or other consequential damage due to operation of the Hitachi product.
5. This product is not designed to be radiation resistant.
6. No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without written approval from Hitachi.
7. Contact Hitachi's sales office for any questions regarding this document or Hitachi semiconductor products.

Preface

This Application Note gives examples of operation in cases where the H8S Series on-chip HCAN (Hitachi Controller Area Network) is used.

(The program examples in the text refer to the H8S/2623 on-chip HCAN.)

The operation of programs, circuit examples, and so forth appearing in this Application Note has been confirmed, but correct operation must be reconfirmed before any of these examples are actually used.

Contents

Section 1	HCAN Transmission/Reception (Example 1): Standard Format, 1-Byte Data	1
1.1	Specifications	1
1.2	Functions	3
1.3	Operation	4
1.4	Software	6
1.5	Transmission Flowchart	8
1.6	Transmission Program List	9
1.7	Reception Flowchart	11
1.8	Reception Program List	12
1.9	Notes	14
Section 2	HCAN Transmission/Reception (Example 2): Standard Format, 8-Byte Data, Using DTC	21
2.1	Specifications	21
2.2	Functions	24
2.3	Operation	26
2.4	Software	28
2.5	Transmission Flowchart	32
2.6	Transmission Program List	33
2.7	Reception Flowcharts	35
2.8	Reception Program List	38
2.9	Notes	41
Section 3	HCAN Transmission/Reception (Example 3): Extended Format, 1-Byte Data	47
3.1	Specifications	47
3.2	Functions	50
3.3	Operation	52
3.4	Software	54
3.5	Transmission Flowchart	56
3.6	Transmission Program List	57
3.7	Reception Flowcharts	59
3.8	Reception Program List	62
3.9	Notes	64

Section 4 HCAN Transmission/Reception (Example 4): Standard Format, 8-Byte Data, Prioritized..... 71

4.1 Specifications..... 71

4.2 Functions..... 74

4.3 Operation 76

4.4 Software 78

4.5 Transmission Flowchart..... 84

4.6 Transmission Program List 85

4.7 Reception Flowcharts..... 91

4.8 Reception Program List 94

4.9 Notes 99

Section 5 HCAN Transmission/Reception (Example 5): Remote Frame..... 105

5.1 Specifications..... 105

5.2 Functions..... 107

5.3 Operation 108

5.4 Software 110

5.5 Transmission Flowchart..... 112

5.6 Transmission Program List 113

5.7 Reception Flowchart 115

5.8 Reception Program List 116

5.9 Notes 118

Section 1 HCAN Transmission/Reception (Example 1): Standard Format, 1-Byte Data

MCU: H8S/2623F-ZTAT	Function Used: HCAN
----------------------------	----------------------------

1.1 Specifications

1. Data frame*¹ transmission/reception (using two H8S/2623F-ZTATs).

Data frame specifications are shown in figure 1.1.

- a. SOF: Indicates start of data frame.
- b. Arbitration field*²: Set to 101010101010.
 - RTR = 0: Select data frame
- c. Control field*³: Set to 000001.
 - IDE = 0: Select standard format
 - R0 = 0: Reserved bit
 - DLC = 0001: Set data length of 1 byte
- d. Data field*⁴: Set to 10101010.
- e. CRC field*⁵: CRC is generated automatically within the HCAN.
- f. ACK field*⁶: 11 is output on the transmitting side, and 01 (in normal operation) on the receiving side.
- g. EOF: Indicates the end of a transmit/receive data frame.

2. A communication speed of 250 kbps (when operating at 20 MHz) is set.

3. The data length is set to 1 byte.

4. Message transmission uses mailbox 1.

5. Message reception uses mailbox 0. The message reception method is to mask the identifier and receive the message in case of a match.

6. Receive data is stored in on-chip RAM.

7. Figure 1.2 shows an example of CAN bus connection.

Note: * See 1.9 Notes.

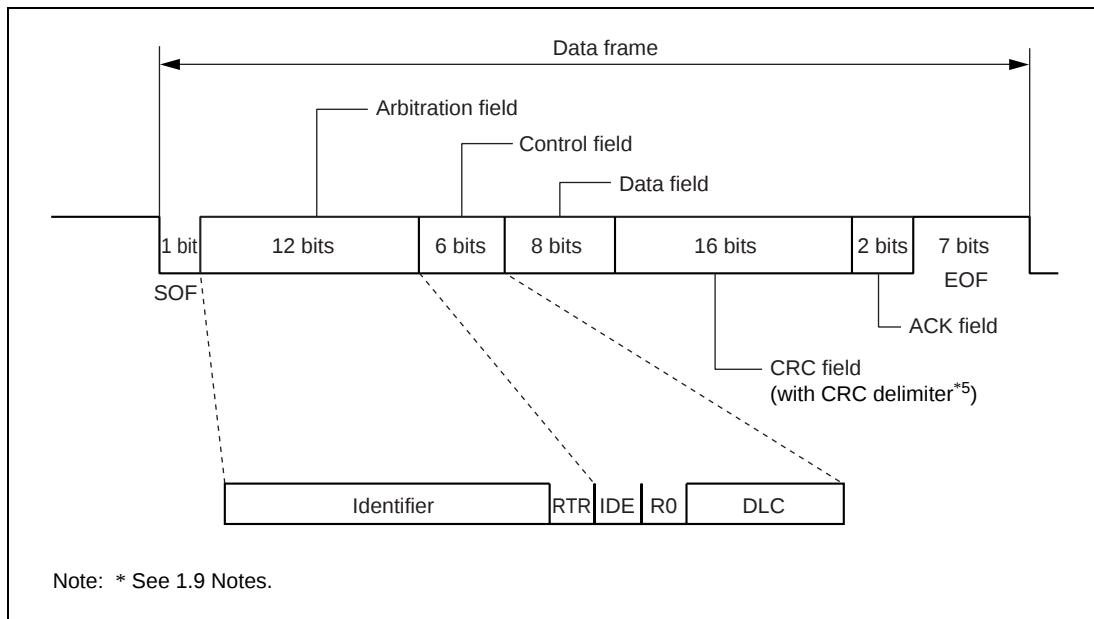


Figure 1.1 Data Frame Specifications

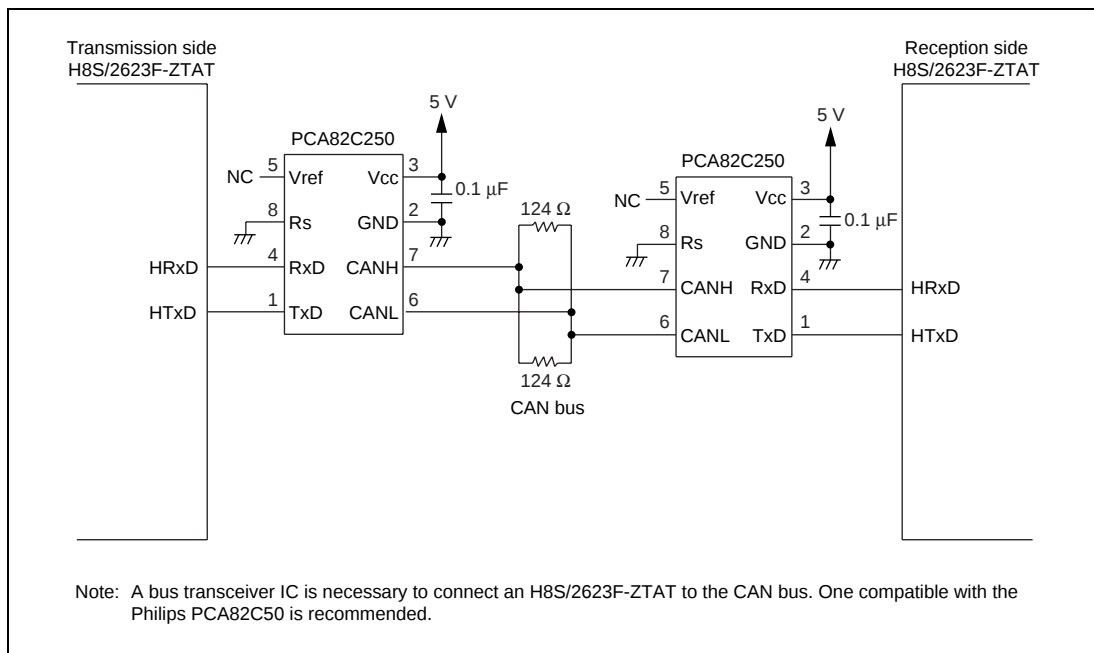


Figure 1.2 CAN Interface Using H8S/2623F-ZTATs

1.2 Functions

Tables 1.1 and 1.2 show the function allocation of this sample task. This sample task allocates H8S/2623F-ZTAT on-chip HCAN functions as shown in these tables, and carries out HCAN transmission and reception.

Table 1.1 HCAN Function Allocation

HCAN Register		Function
Pins	HTxD	Transmits messages.
	HRxD	Receives messages.
Transmission/ reception registers	IRR	Displays status of each interrupt source.
	BCR	Sets CAN baud rate prescaler and bit timing parameters.
	MBCR	Sets mailbox transmission/reception.
	MCR	Controls CAN interface.
	MC0_1 to MC15_8	Arbitration field and control field settings.
	MD0_1 to MD15_8	Data field settings.
Transmission registers	TXPR	Sets transmission wait state after transmit messages are stored in the mailbox.
	TXACK	Indicates that the transmit message of the corresponding mailbox was transmitted normally.
Reception registers	LAFMH	Sets filter mask for reception mailbox 0 identifier.
	RXPR	Indicates that data has been received normally by the corresponding mailbox.

Table 1.2 MSTPCR Function Allocation

MSTPCR Register	Function
MSTPCRC	Controls module stop mode.

1.3 Operation

Operation (Transmission)

Figure 1.3 shows the principle of operation during transmission. HCAN transmission is carried out by H8S/2623F-ZTAT hardware processing and software processing as shown in the figure.

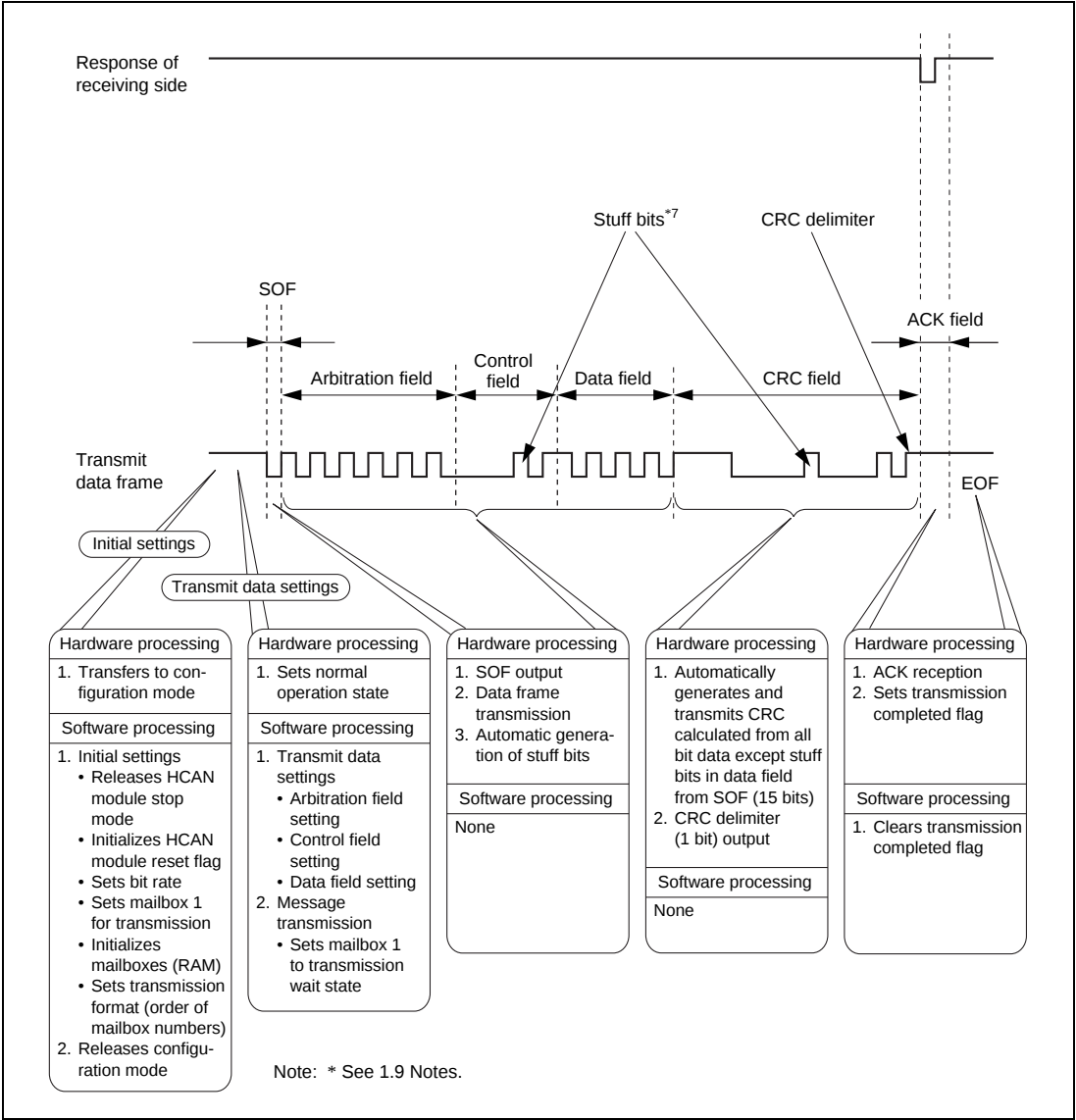


Figure 1.3 Operation During HCAN Transmission

Operation (Reception)

Figure 1.4 shows the principle of operation during reception. HCAN reception is carried out by H8S/2623F-ZTAT hardware processing and software processing as shown in the figure.

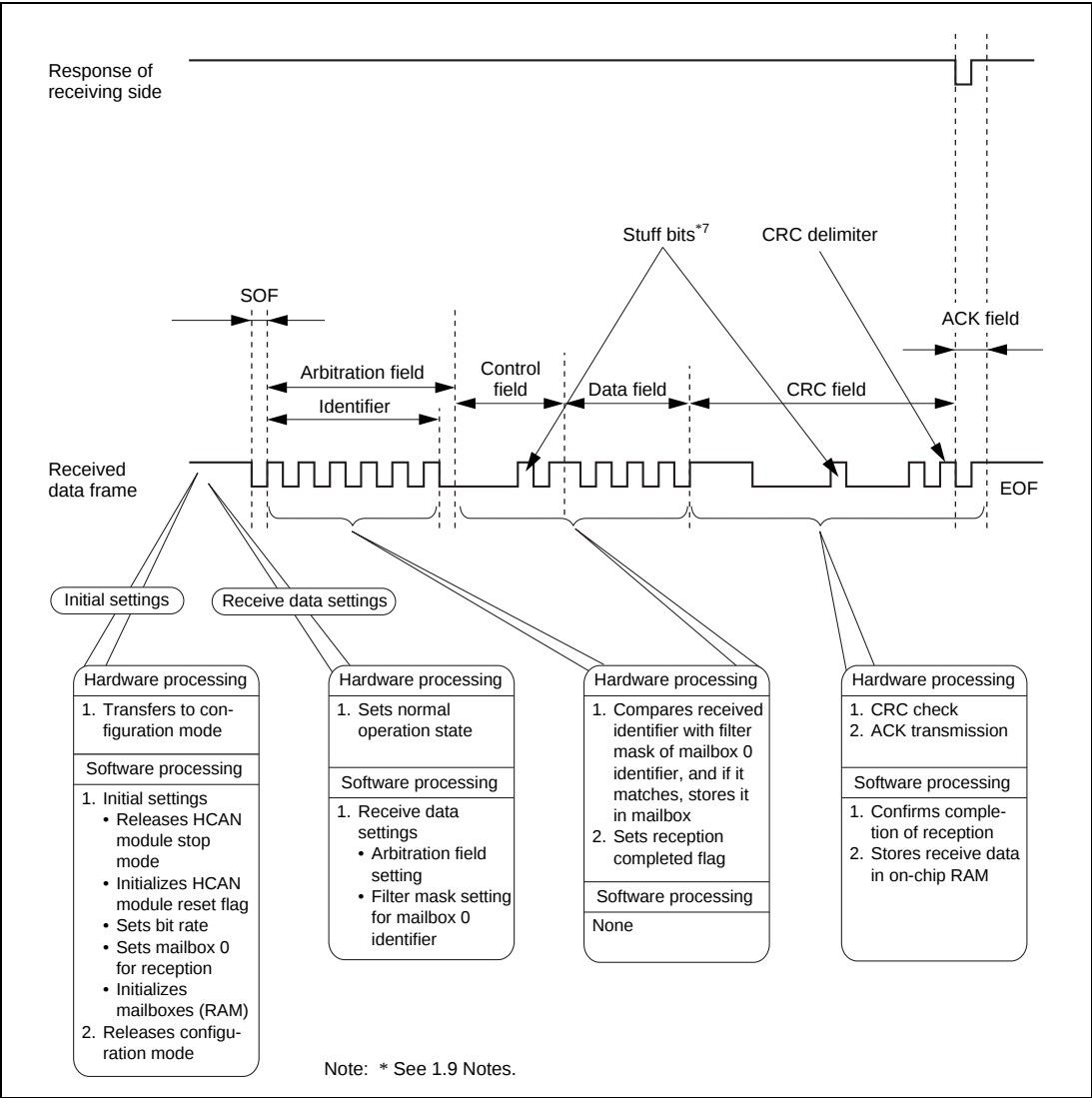


Figure 1.4 Operation During HCAN Reception

1.4 Software

1. Modules

Module Name	Label	Function
Main routine	main	HCAN initial settings and transmission/reception settings.

2. Variables Used

Label	Function	Data Length	Module
COUNT	Initializes HCAN_MC0_1 to MC15_8, and HCAN_MD0_1 to MD15_8.	Unsigned short	Main routine

3. Internal Registers Used

Register Name	Function	Setting	Module
Settings Common to Transmission/Reception			
MSTPCRC	Releases HCAN module stop mode.	0xF7	Main routine
HCAN_IRR	Initializes HCAN module reset flag.	0x0100	
HCAN_BCR	Sets HCAN bit rate to 250 kbps.	0x0334	
Settings for Transmission			
HCAN_MBCR	Sets mailbox 1 for transmission.	0xFDFF	Main routine
HCAN_MCR	Sets transmission in order of mailbox numbers, and clears reset request bit.	0x04	
HCAN_MC1_1	Sets 1-byte data length.	0x01	
HCAN_MC1_5	Selects mailbox 1 data frame and standard format, and sets identifier.	0xA0	
HCAN_MC1_6	Sets mailbox 1 identifier.	0xAA	
HCAN_MD1_1	Sets transmit data for mailbox 1.	0xAA	
HCAN_TXPR	Sets mailbox 1 to transmission wait state.	0x0200	
HCAN_TXACK	Clears data frame transmission completed flag.	0x0200	
Settings for Reception			
HCAN_MBCR	Sets mailbox 0 for reception.	0x0100	Main routine
HCAN_MCR	Clears reset request bit.	0xFE	
HCAN_MC0_5	Selects mailbox 0 data frame and standard format, and sets identifier.	0xA0	
HCAN_MC0_6	Sets mailbox 0 identifier.	0xAA	
HCAN_LAFMH	Sets filter mask for mailbox 0 identifier.	0x0000	

4. RAM Used

Symbol	Function	Address	Module
MAIL_BOX0	Stores HCAN_MD0_1 data.	H'FFC100	Main routine

1.5 Transmission Flowchart

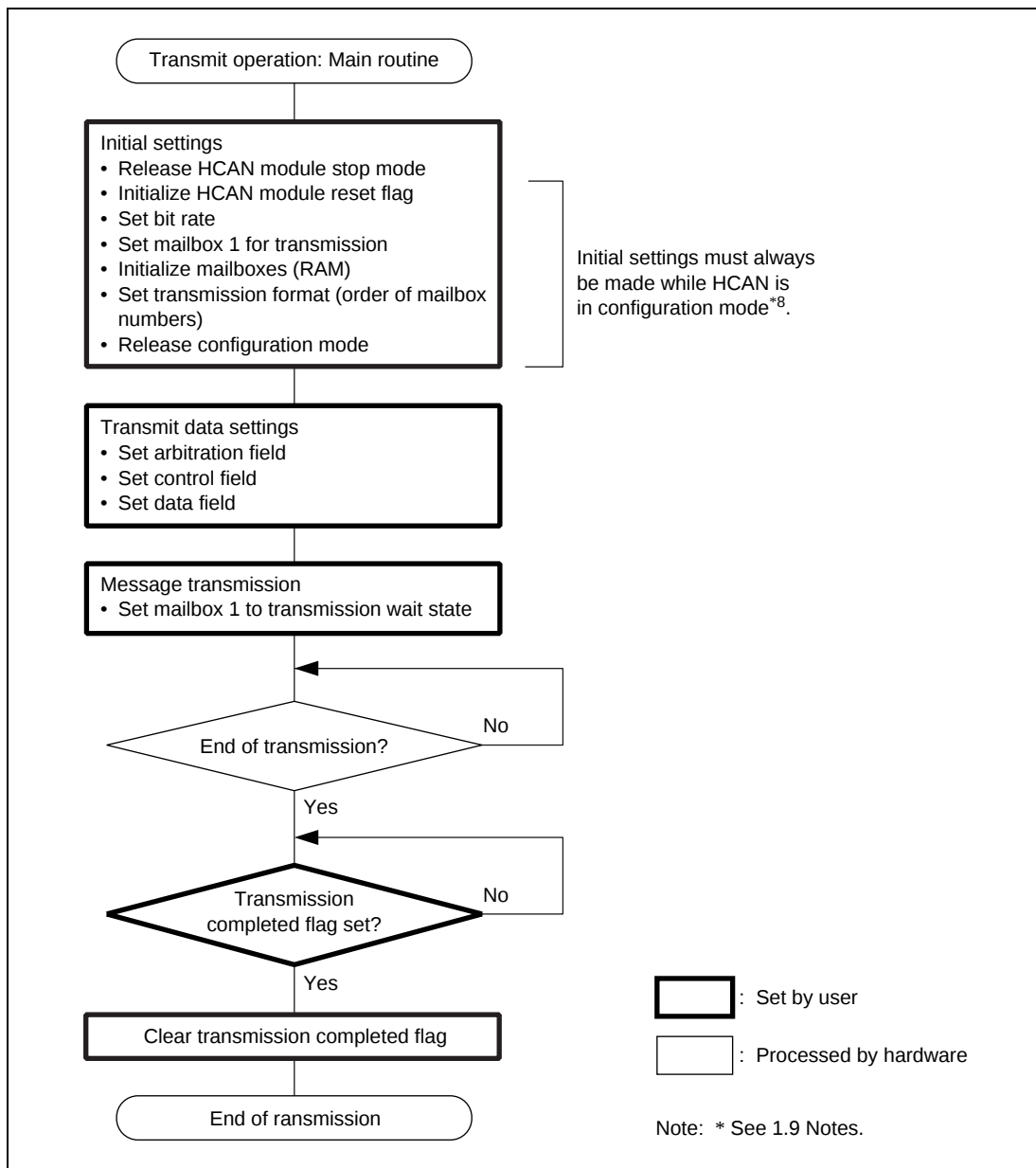


Figure 1.5 Transmission Flowchart

1.6 Transmission Program List

```

/*****
/*
HCAN Transmission Program
*/
*****/
#include <stdio.h>          /* Library function header file */
#include <machine.h>        /* Library function header file */
#include "2623.h"           /* Peripheral register definition header file */
/*****
/*
Function Protocol Declaration
*/
*****/
void main( void );
/*****
/*
Definition of Constants
*/
*****/
#define COUNT  (*(unsigned short *)0xFFC000)
/*****
/*
Main Routine
*/
*****/
void main(void)
{
/* Initial Settings */
    MSTPCRC = 0xF7;          /* Release HCAN module stop mode */
    HCAN_IRR = 0x0100;       /* Initialize HCAN module reset flag */
    HCAN_BCR = 0x0334;       /* Bit rate: 250 kbps */
    HCAN_MBCR = 0xFDFF;      /* Set mailbox 1 for transmission */
    for( COUNT = 0; COUNT < 128; COUNT++ ) /* Initialize mailbox (RAM) */
    {
        *(char*)&HCAN_MC0_1 + COUNT = 0x00;
    }
    for( COUNT = 0; COUNT < 128; COUNT++ ) /* Initialize mailbox (RAM) */
    {
        *(char*)&HCAN_MD0_1 + COUNT = 0x00;
    }
    HCAN_MCR = 0x04;
    /* Set transmission in order of mailbox numbers, and release configuration
    mode*/
/* Transmit data settings */
    HCAN_MC1_5 = 0xA0;
        /* Select data frame and standard format, and set identifier */
    HCAN_MC1_6 = 0xAA;          /* Set identifier */
}

```

```

    HCAN_MC1_1 = 0x01;          /* Data length: 1 byte          */
    HCAN_MD1_1 = 0xAA;          /* Transmit data: 10101010      */

/* Message transmission */
    HCAN_TXPR = 0x0200;          /* Set mailbox 1 to transmission wait state */
    while((HCAN_TXACK & 0x0200) != 0x0200);
                                /* Wait for completion of transmission */

/* Clear transmission completed flag */
    HCAN_TXACK &= 0x0200;          /* Clear transmission completed flag */
    while(1);
}

```

1.7 Reception Flowchart

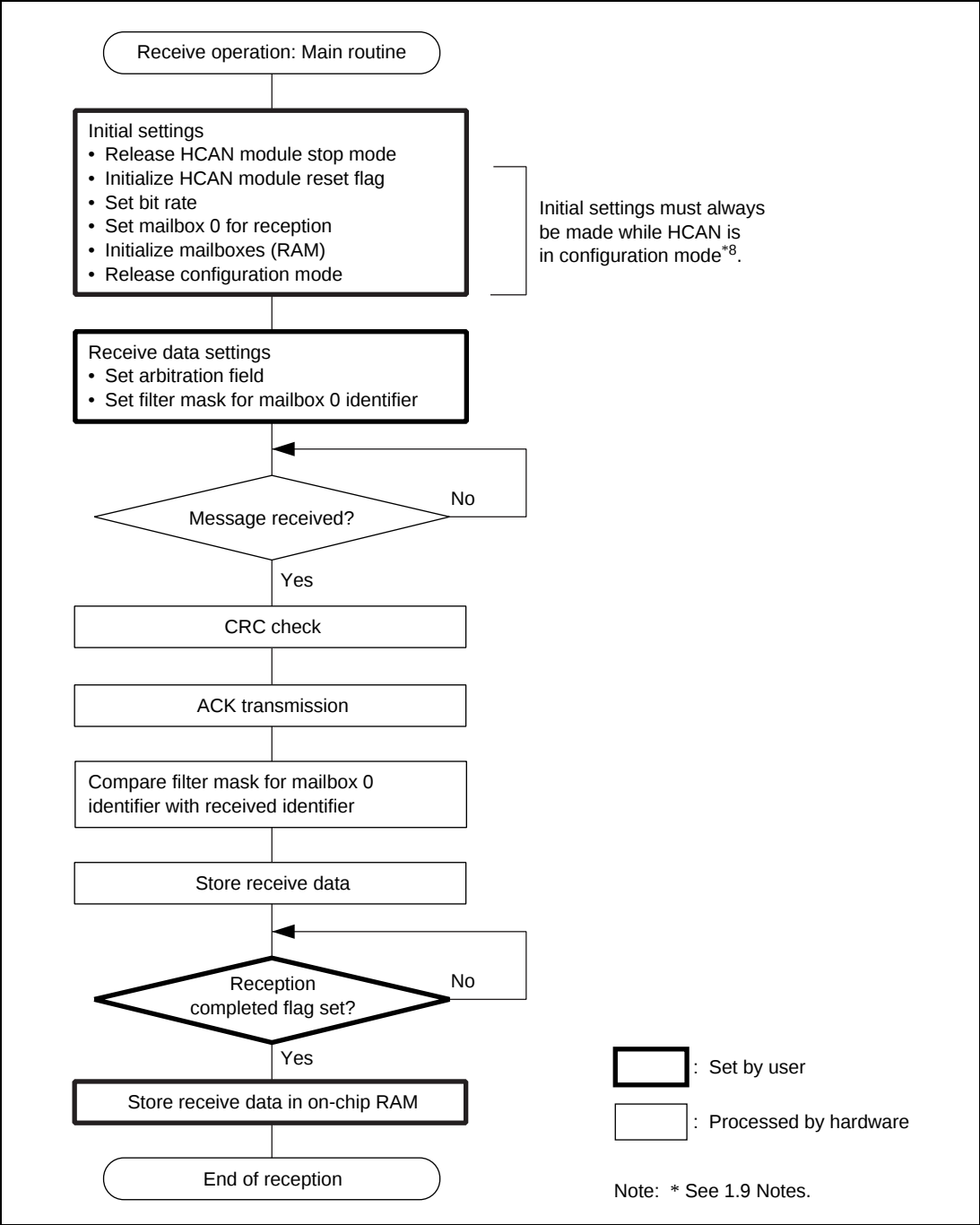


Figure 1.6 Reception Flowchart

1.8 Reception Program List

```

/*****
/*
                                HCAN Reception Program                                */
/*****
#include <stdio.h>                /* Library function header file                */
#include <machine.h>              /* Library function header file                */
#include "2623.h"                /* Peripheral register definition header file */
/*****
/*
                                Function Protocol Declaration                        */
/*****

void main( void );
/*****
/*
                                Definition of Constants                            */
/*****
#define COUNT                    (*(unsigned short *)0xFFC000)
#define MAIL_BOX0                (*(unsigned char *) 0xFFC100)
                                /* Store receive data for mailbox 0 */
/*****
/*
                                Main Routine                                        */
/*****

void main(void)
{
/* Initial settings */
    MSTPCR = 0xF7;                /* Release HCAN module stop mode                */
    HCAN_IRR = 0x0100;            /* Initialize HCAN module reset flag            */
    HCAN_BCR = 0x0334;            /* Bit rate: 250 kbps                            */
    HCAN_MBCR = 0x0100;           /* Set mailbox 0 for reception                    */
    for( COUNT = 0; COUNT < 128; COUNT++ ) /* Initialize mailbox (RAM)                    */
    {
        *(char*)&HCAN_MC0_1 + COUNT) = 0x00;
    }
    for( COUNT = 0; COUNT < 128; COUNT++ ) /* Initialize mailbox (RAM)                    */
    {
        *(char*)&HCAN_MD0_1 + COUNT) = 0x00;
    }
    HCAN_MCR &= 0xFE;            /* Release configuration mode                    */
/* Receive data settings */
    HCAN_MC0_5 = 0xA0;           /* Set data frame, standard format, and identifier */
    HCAN_MC0_6 = 0xAA;           /* Set identifier                                */

```

```

    HCAN_LAFMH = 0x0000;          /* Set filter mask for mailbox 0 identifier */
    while((HCAN_RXPR & 0x0100) != 0x0100);
                                   /* Wait for completion of reception */
/* Store receive data in on-chip RAM */
    MAIL_BOX0 = HCAN_MD0_1;       /* Store receive data */
    while(1);
}

```

1.9 Notes

1. **Data frame:** Data to be transferred from the transmission source to the transmission destination.
2. **Arbitration field:** Set unique ID for message and data frame or remote frame.
3. **Control field:** Set the data length to be transmitted, and standard format or extended format.
4. **Data field:** Set message contents (data to be transmitted).
5. **CRC field:** A CRC is generated automatically in the HCAN from all bit data except stuff bits in the data field from SOF, and is used to detect transmit message errors. The CRC field comprises a 15-bit CRC and a 1-bit delimiter. The CRC delimiter is always output as a 1 after the CRC.

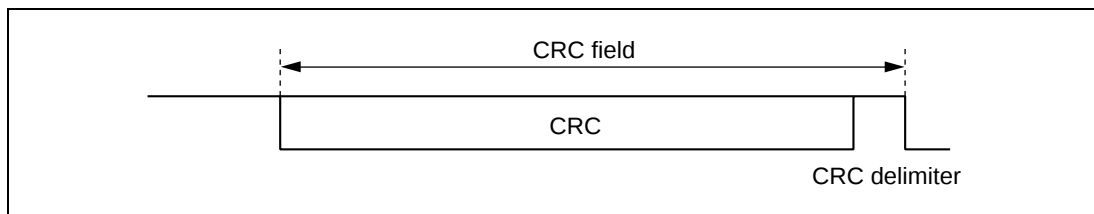


Figure 1.7 CRC Field

About the CRC:

Transmit data polynomial $P(X)$ is multiplied by X^R , then $X^R \cdot P(X)$ is divided by generating polynomial $G(X)$ to give a remainder, $R(X)$. On the side transmitting information, $R(X)$ found from $X^R \cdot P(X)$ is added as check bits, and the result is sent as transmit data $Tx(X)$.

On the side receiving the information, receive data $Rx(X)$ is divided by generating polynomial $G(X)$ to give a remainder. If this remainder is zero, information transmission is regarded as having been completed normally. If the remainder is nonzero, an error is judged to have occurred in the information transmitted.

$$P(X) = \{\text{SOF through data field, excluding stuff bits}\}$$

$$G(X) = X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$$

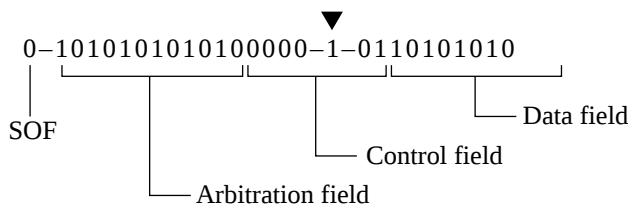
$$R = 15$$

Note: $G(X)$ is stipulated in the CAN protocol as the polynomial that generates the CRC.

As an example, the procedure is described below for a data frame transmitted using the following settings.

Settings SOF: 0
 Arbitration field: 101010101010
 Control field: 000001
 Data field: 10101010

The bit pattern from SOF through the data field in the transmitted data is as follows (▼ indicates the stuff bit):



Excluding the stuff bit gives the following data subject to CRC computation:

010101010101000000110101010

Thus, $P(X) = X^{25} + X^{23} + X^{21} + X^{19} + X^{17} + X^{15} + X^8 + X^7 + X^5 + X^3 + X^1$

$P(X)$ is multiplied by X^{15} , giving:

$$X^{15} \cdot P(X) = X^{40} + X^{38} + X^{36} + X^{34} + X^{32} + X^{30} + X^{23} + X^{22} + X^{20} + X^{18} + X^{16}$$

and this value is divided by

$$G(X) = X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$$

$$\left(\begin{array}{l} X^{15} \cdot P(X) = X^{40} + X^{38} + X^{36} + X^{34} + X^{32} + X^{30} + X^{23} + X^{22} + X^{20} + X^{18} + X^{16} \\ \downarrow \\ P[40:0] = 1010101010100000011010101000000000000000 \\ G(X) = X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1 \\ \downarrow \\ G[15:0] = 1100010110011001 \end{array} \right)$$

		Quotient
1100010110011001	1010101010100000011010101000000000000000	
	1100010110011001	
	1101111001110010	
	1100010110011001	
	1101111101011110	
	1100010110011001	
	1101011000111101	
	1100010110011001	
	1001110100100010	
	1100010110011001	
	1011000101110110	
	1100010110011001	
	1110100111011110	
	1100010110011001	
	1011000100011100	
	1100010110011001	
	1110100100001010	
	1100010110011001	
	1011001001001100	
	1100010110011001	
	1110111110101010	
	1100010110011001	
	1010100011001100	
	1100010110011001	
	1101101010101010	
	1100010110011001	
	1111100110011000	
	1100010110011001	
	111110000000010	

← Remainder:
R[14:0]

The following value is obtained from the calculation.

$$R[14:0] = 111100000000010$$

In other words, this is the CRC value, added after the data field to give transmit data $T_x(X)$, which is transmitted.

In practice, a stuff bit (▲) and CRC delimiter (△) are added, so that 11110000010000101▲△ is transmitted in the CRC field.

Next, the calculation is shown which determines whether there is an error in receive data $Rx(X)$.

$Rx(X)$ is the value obtained by adding the remainder given by the previous calculation to $X^{15} \cdot P(X)$. This value is divided by $G(X)$.

$$\left(\begin{array}{r} P[40:0] = 10101010101000000110101010000000000000000 \\ + R[14:0] = 111100000000010 \\ \hline 10101010101000000110101010111100000000010 \end{array} \right)$$

This value is divided by $G[15:0]$.

1100010110011001

EOR of divisor
and dividend is
taken.

Quotient

10101010101000000110101010111100000000010	
1100010110011001	
1101111001110010	
1100010110011001	
1101111101011110	
1100010110011001	
1101011000111101	
1100010110011001	
1001110100100010	
1100010110011001	
1011000101110111	
1100010110011001	
1110100111011101	
1100010110011001	
1011000100010011	
1100010110011001	
1110100100010100	
1100010110011001	
1011001000110100	
1100010110011001	
1110111101011010	
1100010110011001	
1010101100001100	
1100010110011001	
1101110100101010	
1100010110011001	
1100010110011001	
1100010110011001	

0

 ← Remainder

The remainder from the above calculation is zero, so the transmission is considered to be error-free.

- 6. **ACK field:** For confirmation of normal reception.
Comprises a 1-bit ACK slot and a 1-bit ACK delimiter.

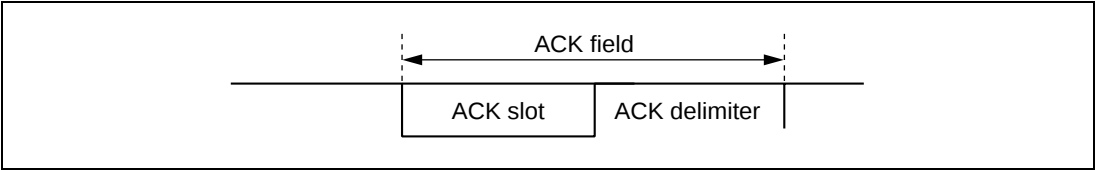


Figure 1.8 ACK Field

- The receiving H8S/2623F-ZTAT outputs a high-level ACK slot if it finds an error in the CRC check, and a low-level ACK slot if it finds no error.
- 7. **Stuff bits:** If there are five consecutive low bits in the data frame, the output is always high for the next bit. Similarly, if there are five consecutive high bits, the output is always low for the next bit.

When stuff bits are output in this way, the bit length of the data frame is increased by the number of stuff bits.

As an example, consider the case where the following settings are made:

Arbitration field: 101010101010
Control field: 000001

The bit pattern in this case is thus 101010101010000001, and so a stuff bit (▼) is output as the second-but-lowest bit (after the five consecutive 0s).

The value transmitted on the CAN bus is therefore 1010101010100000▼101, and the data frame length is increased by the one stuff bit.

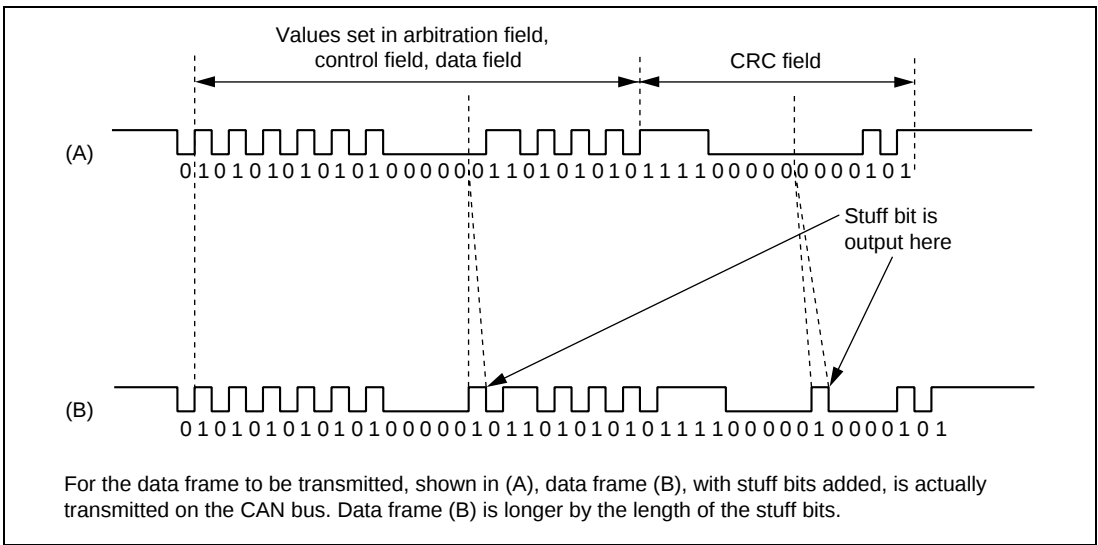


Figure 1.9 Stuff Bits

8. **Configuration mode:** In this mode, the HCAN module is in the reset state. This mode is released by clearing the reset request bit (MCR0) in the master control register (MCR).

Section 2 HCAN Transmission/Reception (Example 2): Standard Format, 8-Byte Data, Using DTC

MCU: H8S/2623F-ZTAT	Function Used: HCAN, DTC
----------------------------	---------------------------------

2.1 Specifications

1. Data frame*¹ transmission/reception (using two H8S/2623F-ZTATs).

Data frame specifications are shown in figure 2.1.

- a. SOF: Indicates start of data frame.
- b. Arbitration field*²: Set to 101010101010.
 - RTR = 0: Select data frame
- c. Control field*³: Set to 001000.
 - IDE = 0: Select standard format
 - R0 = 0: Reserved bit
 - DLC = 1000: Set data length of 8 bytes
- d. Data field*⁴
 - 1st byte: Set to 01010101 (H'55).
 - 2nd byte: Set to 01100110 (H'66).
 - 3rd byte: Set to 01110111 (H'77).
 - 4th byte: Set to 10001000 (H'88).
 - 5th byte: Set to 10011001 (H'99).
 - 6th byte: Set to 10101010 (H'AA).
 - 7th byte: Set to 10111011 (H'BB).
 - 8th byte: Set to 11111111 (H'FF).
- e. CRC field*⁵: CRC is generated automatically within the HCAN.
- f. ACK field*⁶: 11 is output on the transmitting side, and 01 (in normal operation) on the receiving side.
- g. EOF: Indicates the end of a transmit/receive data frame.

2. A communication speed of 250 kbps (when operating at 20 MHz) is set.

3. The data length is set to 8 bytes.

4. Message transmission uses mailbox 1.

5. Message reception uses mailbox 0. The message reception method is to mask the identifier and receive the message in case of a match.

6. Select interrupt control mode 2, and
 - a. Use bus operation interrupt (OVR0).
 - b. Use DTC interrupts by message reception and the message reception interrupt (RM0).
7. Receive messages are stored in on-chip RAM using the DTC.
 - a. The DTC is started up by a message being received.
 - b. DTC specifications are as follows.
 - Block transmission mode is used.
 - After transmission ends, the message reception interrupt (RM0) is used.
8. Use HCAN sleep mode.
 - a. After initial settings, transfer to HCAN sleep mode (released by a CAN bus operation).
 - b. After reception processing, transfer to HCAN sleep mode (released by a CAN bus operation).
9. Figure 2.2 shows an example of CAN bus connection.

Note: * See 2.9 Notes.

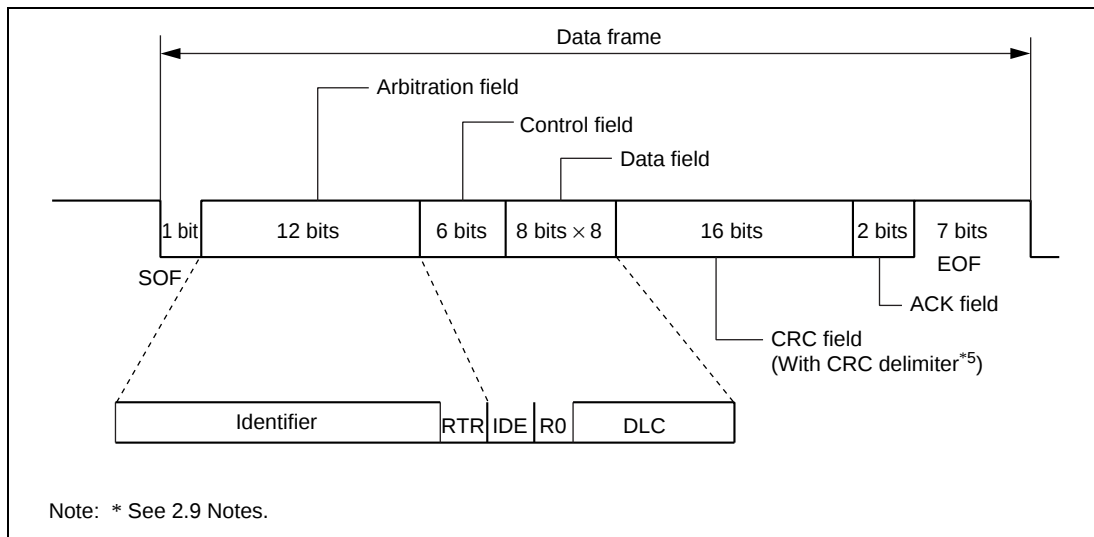


Figure 2.1 Data Frame Specifications

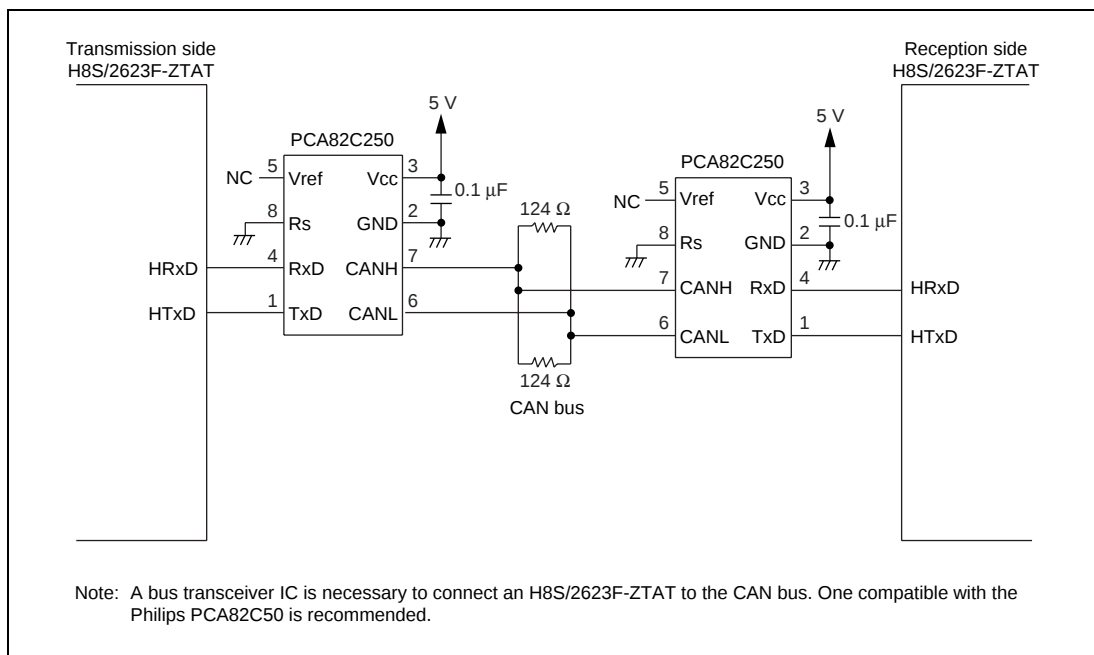


Figure 2.2 CAN Interface Using H8S/2623F-ZTATs

2.2 Functions

Tables 2.1 to 2.4 show the function allocation of this sample task. This sample task allocates H8S/2623F-ZTAT on-chip HCAN functions as shown in these tables, and carries out HCAN transmission and reception.

Table 2.1 HCAN Function Allocation

HCAN Register		Function
Pins	HTxD	Transmits messages.
	HRxD	Receives messages.
Transmission/ reception registers	IRR	Displays status of each interrupt source.
	BCR	Sets CAN baud rate prescaler and bit timing parameters.
	MBCR	Sets mailbox transmission/reception.
	MCR	Controls CAN interface.
	MC0_1 to MC15_8	Arbitration field and control field settings.
	MD0_1 to MD15_8	Data field settings.
Transmission registers	TXPR	Sets transmission wait state after transmit messages are stored in the mailbox.
	TXACK	Indicates that the transmit message of the corresponding mailbox was transmitted normally.
Reception registers	MBIMR	Enables/disables interrupt requests for each mailbox.
	IMR	Enables/disables requests for each interrupt source.
	LAFMH	Sets filter mask for reception mailbox 0 identifier.

Table 2.2 MSTPCR Function Allocation

MSTPCR Register	Function
MSTPCRC	Control module stop mode.
MSTPCRA	

Table 2.3 Interrupt Controller Allocation

IPR Register	Function
IPRM	Controls interrupt priority level.
SYSCR	Sets interrupt control mode.
exr	Specifies interrupt request mask level.

Table 2.4 DTC Function Allocation

DTC Register	Function
SAR (On-chip RAM)	Sets source address (where receive messages are stored from).
DAR (On-chip RAM)	Sets destination address (where receive messages are stored to).
MAR (On-chip RAM)	Sets block transfer mode, byte size transfer, etc.
MRB (On-chip RAM)	Sets interrupt to CPU after DTC data transfer.
CRA (On-chip RAM)	Sets number of DTC data transfers.
CRB (On-chip RAM)	Sets number of DTC block data transfers.
DTCERG (Register)	Sets DTC start by interrupt.

2.3 Operation

Operation (Transmission)

Figure 2.3 shows the principle of operation during transmission. HCAN transmission is carried out by H8S/2623F-ZTAT hardware processing and software processing as shown in the figure.

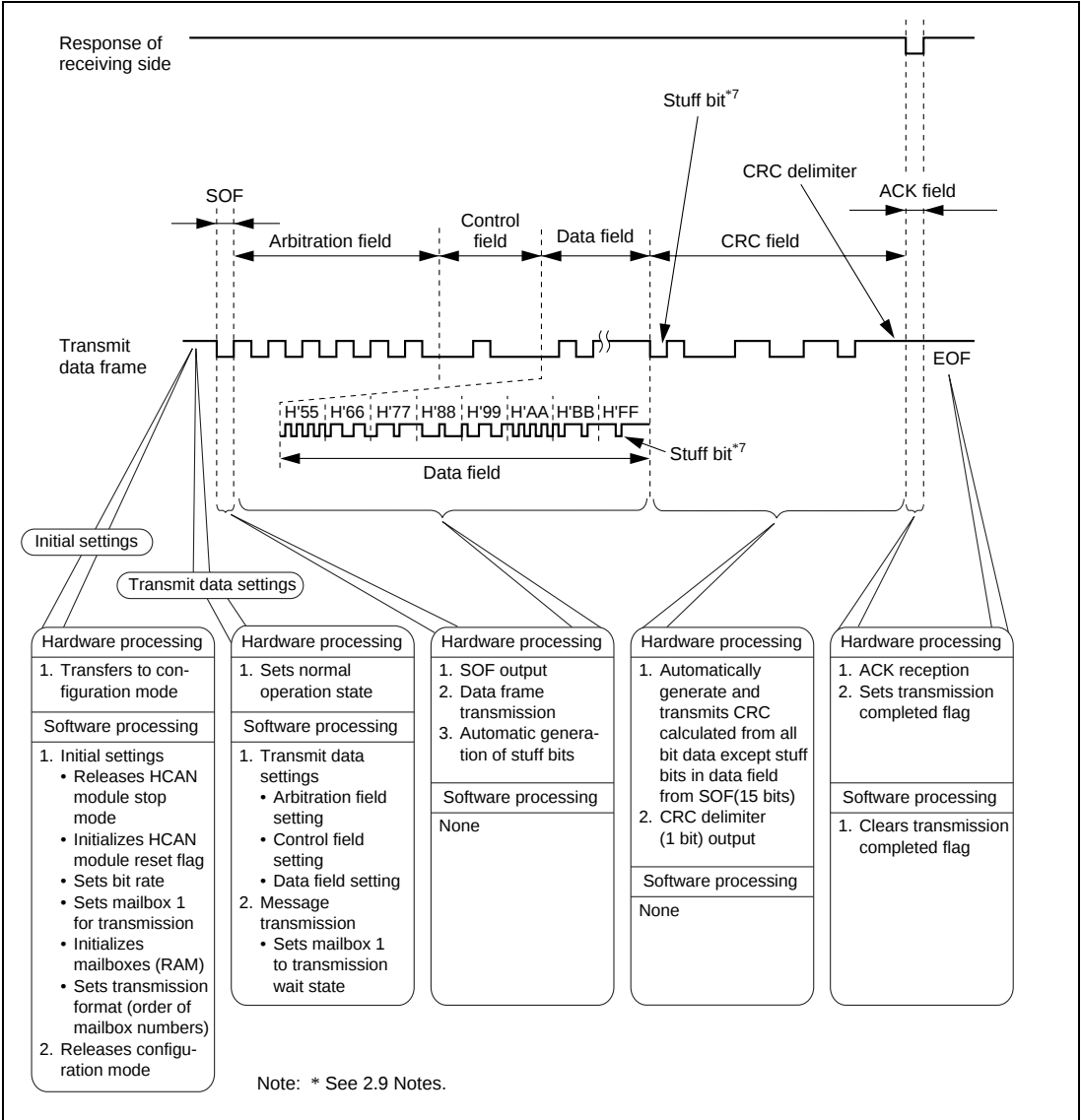


Figure 2.3 Operation During HCAN Transmission

Operation (Reception)

Figure 2.4 shows the principle of operation during reception. HCAN reception is carried out by H8S/2623F-ZTAT hardware processing and software processing as shown in the figure.

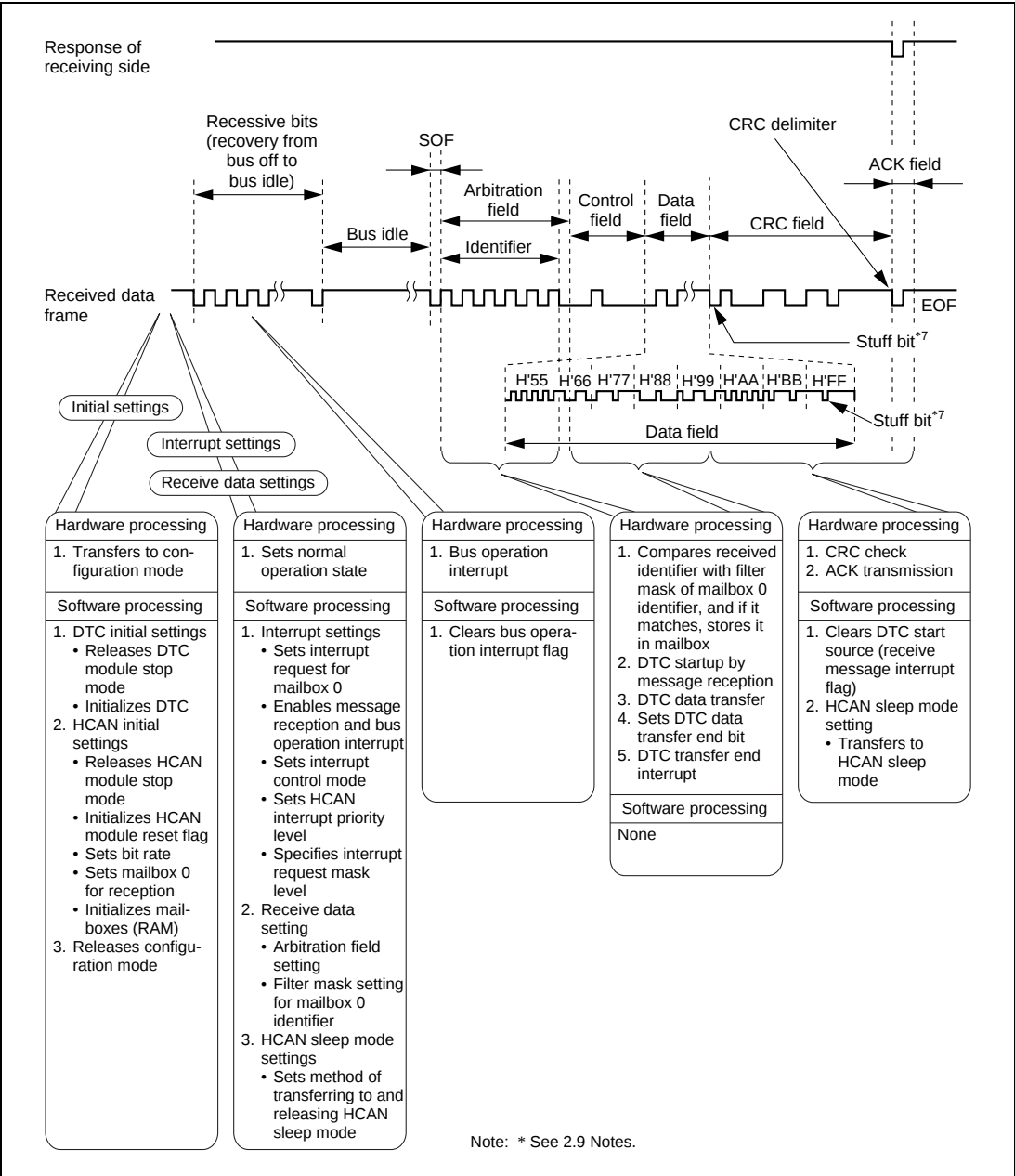


Figure 2.4 Operation During HCAN Reception

2.4 Software

1. Modules

Module Name	Label	Function
Main routine	main	HCAN initial settings and transmission/reception settings.
Bus operation interrupt routine	OVR0_IRR12	Clears bus operation interrupt flag.
DTC transfer end interrupt routine	DTCend_RM0	Clears DTC start source and transfers to HCAN sleep mode.

2. Variables Used

Label	Function	Data Length	Module
COUNT	Initializes HCAN_MC0_1 to MC15_8, and HCAN_MD0_1 to MD15_8.	Unsigned short	Main routine

3. Internal Registers Used

Register Name	Function	Setting	Module
Settings Common to Transmission/Reception			
MSTPCRC	Release HCAN module stop mode.	0xF7	Main routine
HCAN_BCR	Sets HCAN bit rate to 250 kbps.	0x0334	
Settings for Transmission			
HCAN_IRR	Initializes HCAN module reset flag.	0x0100	Main routine
HCAN_MBCR	Sets mailbox 1 for transmission.	0xFDFF	
HCAN_MCR	Sets transmission in order of mailbox numbers, and clears reset request bit.	0x04	
HCAN_MC1_1	Sets 8-byte data length.	0x08	
HCAN_MC1_5	Selects mailbox 1 data frame and standard format, and sets identifier	0xA0	
HCAN_MC1_6	Sets mailbox 1 identifier.	0xAA	
HCAN_MD1_1	Sets transmit data for 1 st byte of mailbox 1.	0x55	
HCAN_MD1_2	Sets transmit data for 2 nd byte of mailbox 1.	0x66	
HCAN_MD1_3	Sets transmit data for 3 rd byte of mailbox 1.	0x77	
HCAN_MD1_4	Sets transmit data for 4 th byte of mailbox 1.	0x88	
HCAN_MD1_5	Sets transmit data for 5 th byte of mailbox 1.	0x99	
HCAN_MD1_6	Sets transmit data for 6 th byte of mailbox 1.	0xAA	
HCAN_MD1_7	Sets transmit data for 7 th byte of mailbox 1.	0xBB	
HCAN_MD1_8	Sets transmit data for 8 th byte of mailbox 1.	0xFF	
HCAN_TXPR	Sets mailbox 1 to transmission wait state.	0x0200	
HCAN_TXACK	Clears data frame transmission completed flag.	0x0200	

Register Name	Function	Setting	Module
Settings for Reception			
HCAN_IRR	Initializes HCAN module reset flag.	0x0100	Main routine
	Clears bus operation interrupt flag.	0x0010	Bus operation
HCAN_MCR	Clears reset request bit.	0xFE	Main routine
	Sets transfer to HCAN sleep mode and release by bus operation.	0xA0	
	Transfers to HCAN sleep mode.	0x20	DTC end
HCAN_RXPR	Clears receive message interrupt.	0xFFFF	
HCAN_MBCR	Sets mailbox 0 for reception.	0x0100	Main routine
HCAN_MC0_5	Selects mailbox 0 data frame and standard format, and sets identifier.	0xA0	
HCAN_MC0_6	Sets identifier for mailbox 0.	0xAA	
HCAN_LAFMH	Sets filter mask for mailbox 0 identifier.	0x0000	
HCAN_MBIMR	Sets enabling of mailbox 0 interrupt requests.	0xFEFF	
HCAN_IMR	Sets enabling of message reception and bus operation interrupt requests.	0xFCEF	
INTC. IPRM	Sets HCAN interrupt priority level to 7.	0x70	
SYSCR	Sets interrupt control mode 2.	0x20	
exr	Specifies interrupt request mask level.	0x00	
MSTPCRA	Releases DTC module stop mode.	0x3F	
DTC_DTCERG	Sets DTC start by HCAN receive message.	0x20	

Note: Bus operation: Bus operation interrupt routine
DTC end: DTC transfer end interrupt routine

4. RAM Used

Symbol	Function	Address	Module
Message_Box1 to 8	Storage destination address for mailbox 0 data (8-byte)	0xFFC000 to 7 (Undefined)	Main routine
SAR	Sets source address (where receive messages are stored from).	0xFFEC01 to 3 (0xFFFF8B0)	
MRA	Sets block transfer mode, byte size transfer, etc.	0xFFEC00 (0xA8)	
DAR	Sets destination address (where receive message are stored to).	0xFFEC05 to 7 (0xFFC000)	
MRB	Enables message reception interrupt (RM0) after DTC data transfer ends.	0xFFEC04 (0x00)	
CRA	Sets transfer block size retained value for DTC data transfer (upper byte) and transfer block size counter (lower byte).	0xFFEC08, 9 (0x0808)	
CRB	Sets transfer number of DTC block transfers.	0xFFEC0A, B (0x0001)	

Figures in parentheses are settings.

2.5 Transmission Flowchart

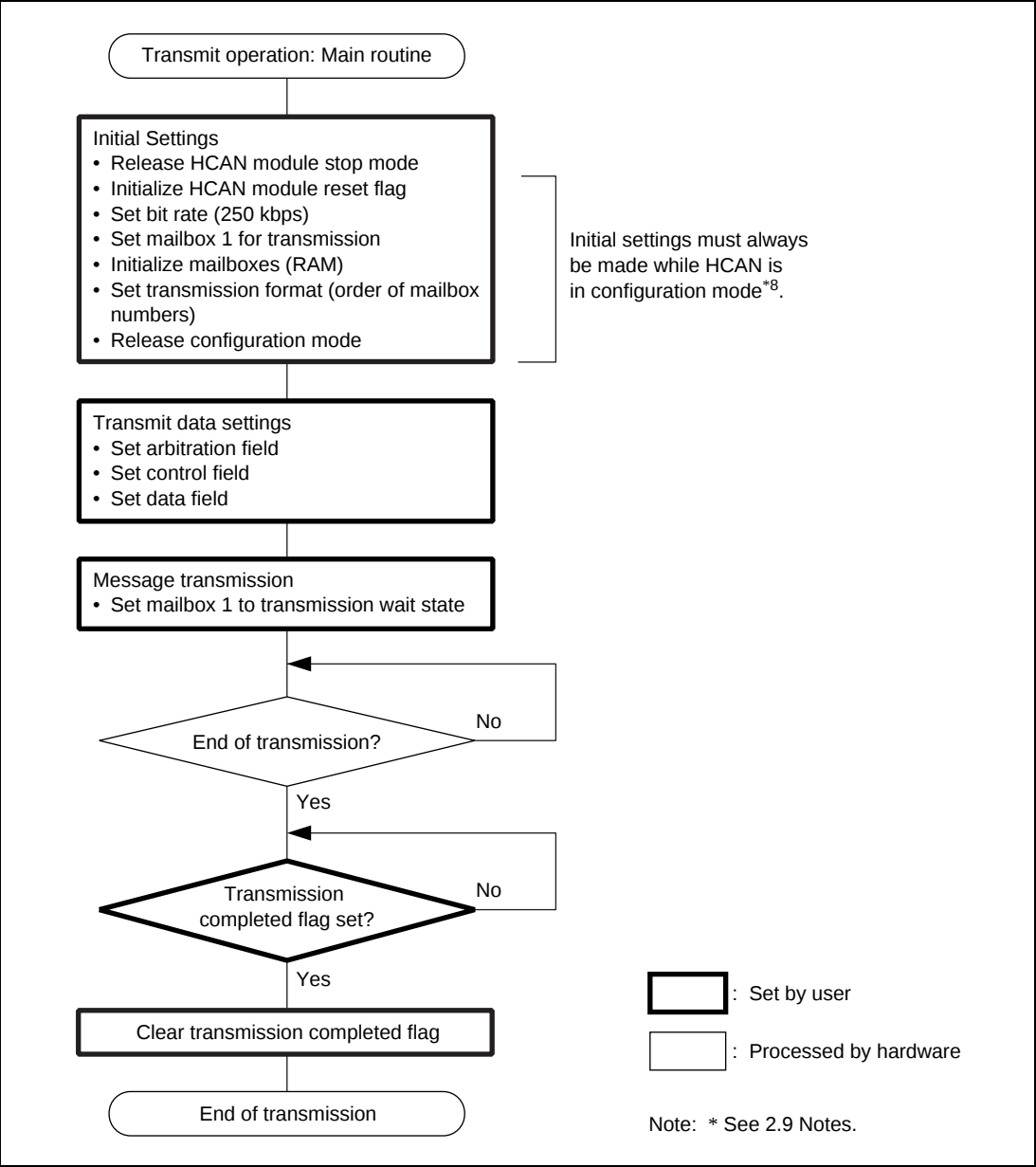


Figure 2.5 Transmission Flowchart

2.6 Transmission Program List

```

/*****
/*          HCAN Transmission Program          */
*****/
#include <stdio.h>          /* Library function header file          */
#include <machine.h>        /* Library function header file          */
#include "2623.h"           /* Peripheral register definition header file */
/*****
/*          Function Protocol Declaration          */
*****/
void main( void );
/*****
/*          Definition of Constants          */
*****/
#define COUNT (*(unsigned long *)0xFFC000)
/*****
/*          Main Routine          */
*****/
void main(void)
{
/* Initial Settings */
    MSTPCRC = 0xF7;          /* Release HCAN module stop mode          */
    HCAN_IRR = 0x0100;       /* Initialize HCAN module reset flag          */
    HCAN_BCR = 0x0334;       /* Bit rate: 250 kbps          */
    HCAN_MBCR = 0xFDFF;      /* Set mailbox 1 for transmission          */
    for( COUNT = 0; COUNT < 128; COUNT++ ) /* Initialize mailbox (RAM)          */
    {
        *(char*)&HCAN_MC0_1 + COUNT = 0x00;
    }
    for( COUNT = 0; COUNT < 128; COUNT++ ) /* Initialize mailbox (RAM)          */
    {
        *(char*)&HCAN_MD0_1 + COUNT = 0x00;
    }
    HCAN_MCR = 0x04;
    /* Set transmission in order of mailbox numbers, and release configuration
       mode */
/* Transmit data settings */
    HCAN_MC1_5 = 0xA0;
        /* Select data frame and standard format, and set identifier */
    HCAN_MC1_6 = 0xAA;      /* Set identifier          */

```

```

HCAN_MC1_1 = 0x08;          /* Data length: 8 bytes          */
HCAN_MD1_1 = 0x55;          /* Message contents: 01010101    */
HCAN_MD1_2 = 0x66;          /* Message contents: 01100110    */
HCAN_MD1_3 = 0x77;          /* Message contents: 01110111    */
HCAN_MD1_4 = 0x88;          /* Message contents: 10001000    */
HCAN_MD1_5 = 0x99;          /* Message contents: 10011001    */
HCAN_MD1_6 = 0xAA;          /* Message contents: 10101010    */
HCAN_MD1_7 = 0xBB;          /* Message contents: 10111011    */
HCAN_MD1_8 = 0xFF;          /* Message contents: 11111111    */

/* Message transmission */
    HCAN_TXPR = 0x0200;      /* Set mailbox 1 to transmission wait state */
    while((HCAN_TXACK & 0x0200) != 0x0200);

/* Clear transmission completed flag */
    HCAN_TXACK &= 0x0200;    /* Clear transmission completed flag */
    while(1);
}

```

2.7 Reception Flowcharts

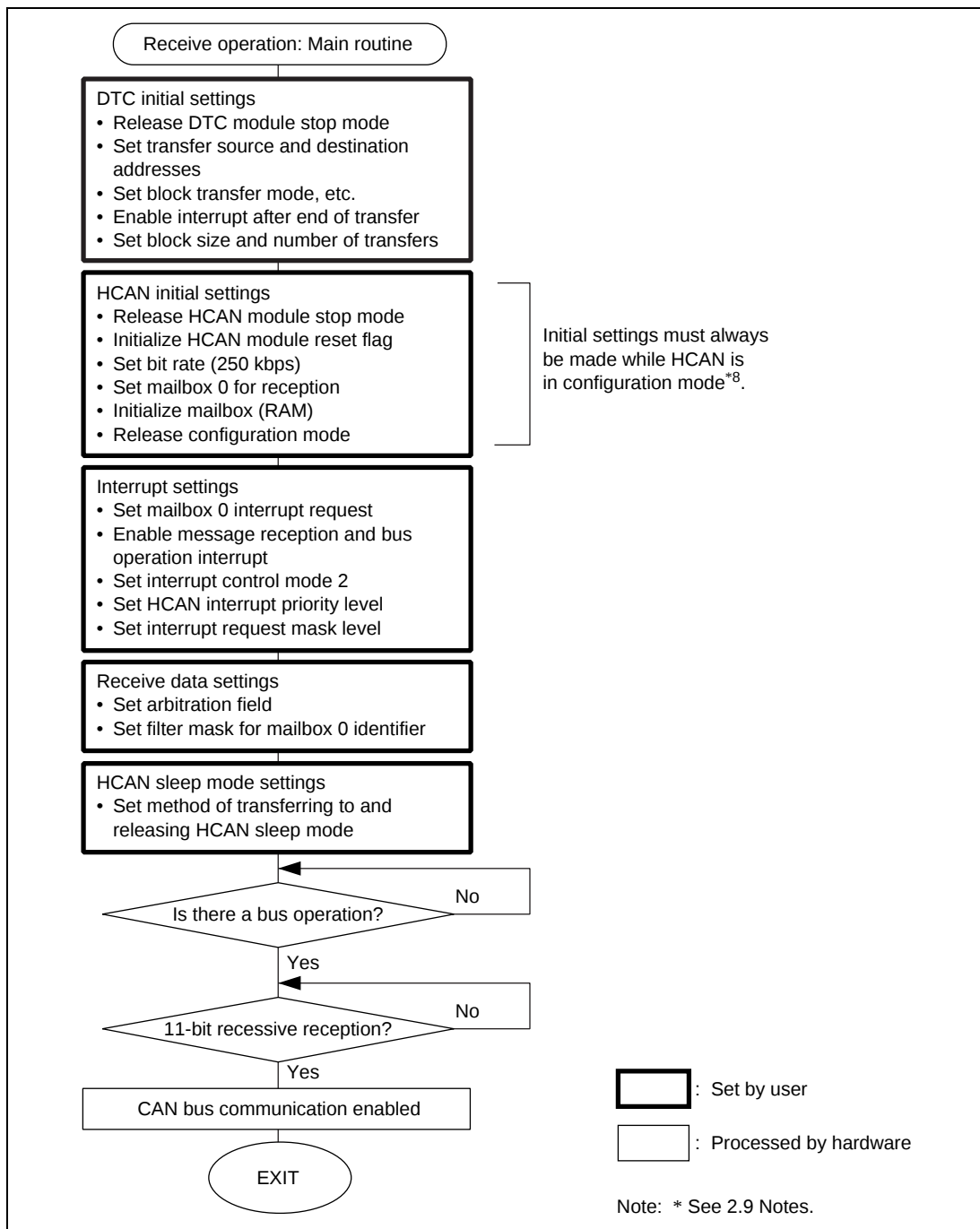


Figure 2.6 Reception Flowchart (1)

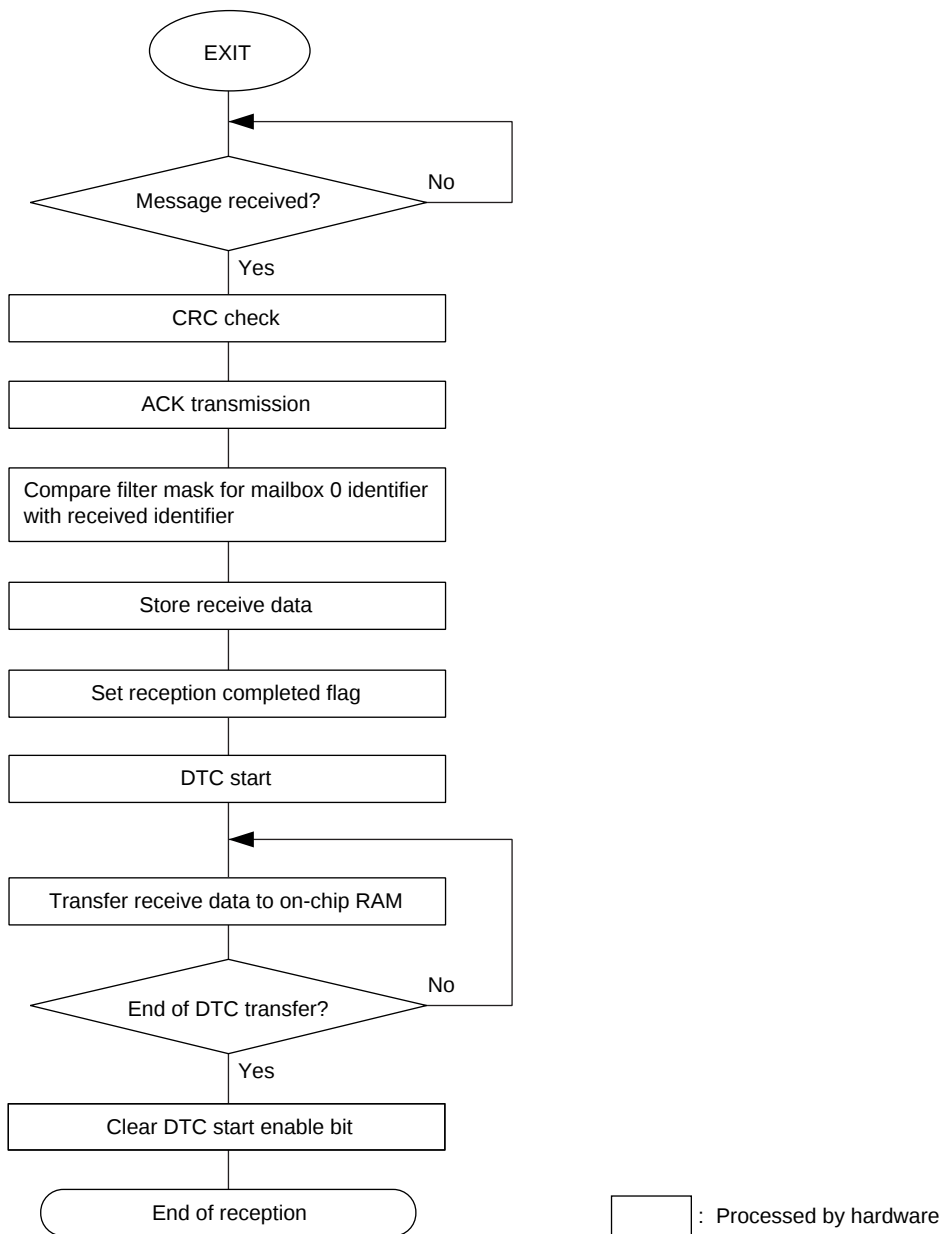


Figure 2.6 Reception Flowchart (2)

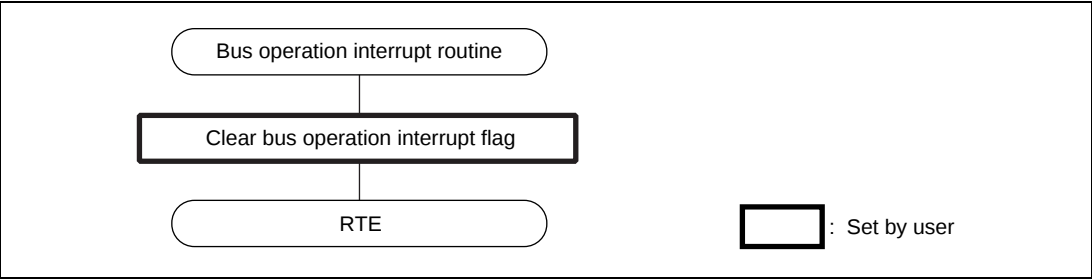


Figure 2.7 Bus Operation Interrupt Flowchart

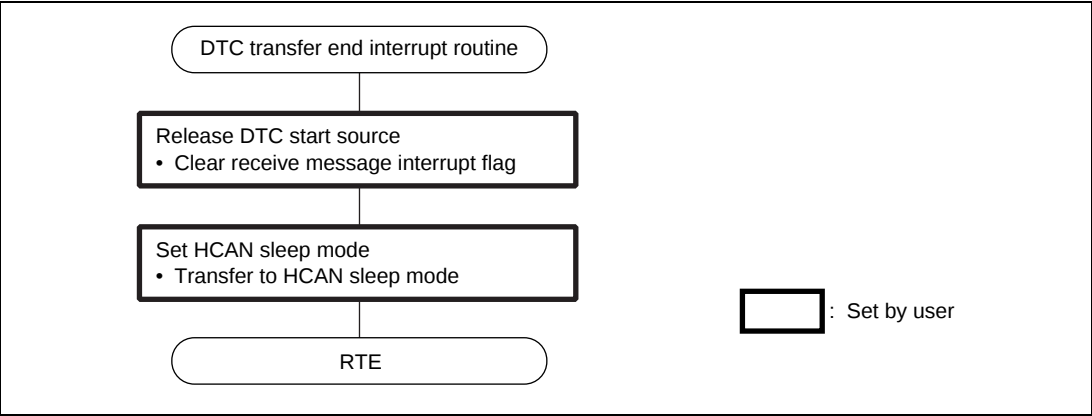


Figure 2.8 DTC Transfer End Interrupt Flowchart

2.8 Reception Program List

```

/*****
/*          HCAN Reception Program          */
*****/

#include <stdio.h>          /* Library function header file          */
#include <machine.h>        /* Library function header file          */
#include "2623.h"          /* Peripheral register definition header file */
*****/

/*          Function Protocol Declaration          */
*****/

void main( void );

/*****
/*          Definition of Constants          */
*****/

#define COUNT      (*(unsigned short *)0xFFC100)
#define Message_Box1 (*(volatile unsigned char *)0xFFC000)
/* Store receive data 1 */
#define Message_Box2 (*(volatile unsigned char *)0xFFC001)
/* Store receive data 2 */
#define Message_Box3 (*(volatile unsigned char *)0xFFC002)
/* Store receive data 3 */
#define Message_Box4 (*(volatile unsigned char *)0xFFC003)
/* Store receive data 4 */
#define Message_Box5 (*(volatile unsigned char *)0xFFC004)
/* Store receive data 5 */
#define Message_Box6 (*(volatile unsigned char *)0xFFC005)
/* Store receive data 6 */
#define Message_Box7 (*(volatile unsigned char *)0xFFC006)
/* Store receive data 7 */
#define Message_Box8 (*(volatile unsigned char *)0xFFC007)
/* Store receive data 8 */
#define SAR          (*(volatile unsigned long *)0xFFEC00)
/* Set DTC register information */
#define MRA          (*(volatile unsigned char *)0xFFEC00)
/* Set DTC register information */
#define DAR          (*(volatile unsigned long *)0xFFEC04)
/* Set DTC register information */
#define MRB          (*(volatile unsigned char *)0xFFEC04)
/* Set DTC register information */
#define CRA          (*(volatile unsigned short *)0xFFEC08)
/* Set DTC register information */
#define CRB          (*(volatile unsigned short *)0xFFEC0A)
/* Set DTC register information */

```

```

/*****
/*          Main Routine          */
*****/

void main(void)
{
/* DTC initial settings */
    MSTPCRA = 0x3F;                /* Release DTC module stop mode */
    SAR = (long>(&HCAN_MD0_1);    /* Set transfer source address */
    MRA = 0xA8;
    /* Set SAR and DAR incrementing after transfer, and block transfer mode */
    DAR = (long>(&Message_Box1);
    /* Set transfer destination address (on-chip RAM) */
    MRB = 0x00;                    /* Enable interrupt after end of data transfer by DTC */
    CRA = 0x0808;                  /* Set block transfer to 8-byte units */
    CRB = 0x0001;                  /* Set number of block transfers to 1 */
    DTC_DTCERG |= 0x20; Enable DTC start by HCAN interrupt (RM0) */

/* HCAN initial settings */
    MSTPCRC = 0xF7                 /* Release HCAN module stop mode */
    HCAN_IRR = 0x0100;             /* Initialize HCAN module reset flag */
    HCAN_BCR = 0x0334;             /* Bit rate: 250 kbps */
    HCAN_MBCR = 0x0100;            /* Set mailbox 0 for reception */
    for( COUNT = 0; COUNT < 128; COUNT++ )    /* Initialize mailbox (RAM) */
    {
        *(char*)&HCAN_MC0_1 + COUNT) = 0x00;
    }
    for( COUNT = 0; COUNT < 128; COUNT++ )    /* Initialize mailbox (RAM) */
    {
        *(char*)&HCAN_MD0_1 + COUNT) = 0x00;
    }
    HCAN_MCR &= 0xFE;              /* Release configuration mode */

/* Interrupt settings */
    HCAN_MBIMR = 0xFEFF;           /* Enable mailbox 0 interrupt request */
    HCAN_IMR = 0xFCEF; /* Enable message reception and bus operation interrupt */
    SYSCR |= 0x20;                 /* Set interrupt control mode 2 */
    INTC.IPRM = 0x70;              /* Set HCAN interrupt priority level to 7 */
    set_imask_exr(0);              /* Specify interrupt request mask level */

/* Receive data settings */
    HCAN_MC0_5 = 0xA0;             /* Set standard format, data frame, and identifier */
    HCAN_MC0_6 = 0xAA;             /* Set identifier */
    HCAN_LAFMH = 0x0000;           /* Mailbox 0 stores data in case of bit match */

/* HCAN sleep mode setting */

```

```

    HCAN_MCR |= 0xA0;
                /* Enable transfer to HCAN mode and release by bus operation */
    while(1);
}
/*****
/*          Bus Operation Interrupt Routine          */
*****/
#pragma interrupt(OVR0_IRR12)
void OVR0_IRR12(void)
{
    HCAN_IRR &= 0x0010;          /* Clear IRR12 (bus operation interrupt flag) */
}
/*****
/*          DTC Transfer End Interrupt Routine          */
*****/
#pragma interrupt(DTCend_RM0)
void DTCend_RM0(void)
{
    HCAN_RXPR &= 0xFFFF;          /* Clear IRR1 (receive message interrupt flag) */
    /* HCAN sleep mode setting */
    HCAN_MCR |= 0x20;          /* Transfer to HCAN sleep mode */
}

```

2.9 Notes

1. **Data frame:** Data to be transferred from the transmission source to the transmission destination.
2. **Arbitration field:** Set unique ID for message and data frame or remote frame.
3. **Control field:** Set the data length to be transmitted, and standard format or extended format.
4. **Data field:** Set message contents (data to be transmitted).
5. **CRC field:** A CRC is generated automatically in the HCAN from all bit data except stuff bits in the data field from SOF, and is used to detect transmit message errors.
The CRC field comprises a 15-bit CRC and a 1-bit delimiter.
The CRC delimiter is always output as a 1 after the CRC.

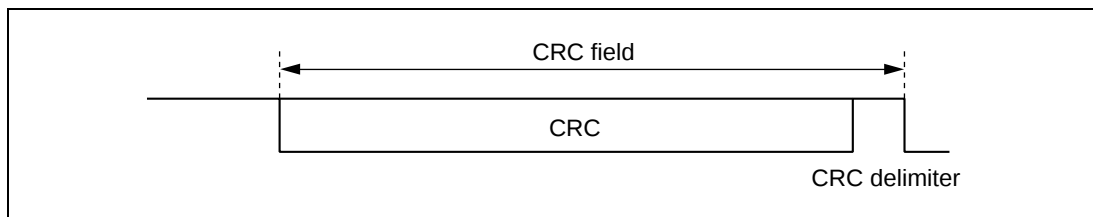


Figure 2.9 CRC Field

About the CRC:

Transmit data polynomial $P(X)$ is multiplied by X^R , then $X^R \cdot P(X)$ is divided by generating polynomial $G(X)$ to give a remainder, $R(X)$. On the side transmitting information, $R(X)$ found from $X^R \cdot P(X)$ is added as check bits, and the result is sent as transmit data $Tx(X)$.

On the side receiving the information, receive data $Rx(X)$ is divided by generating polynomial $G(X)$ to give a remainder. If this remainder is zero, information transmission is regarded as having been completed normally. If the remainder is nonzero, an error is judged to have occurred in the information transmitted.

$$P(X) = \{\text{SOF through data field, excluding stuff bits}\}$$

$$G(X) = X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$$

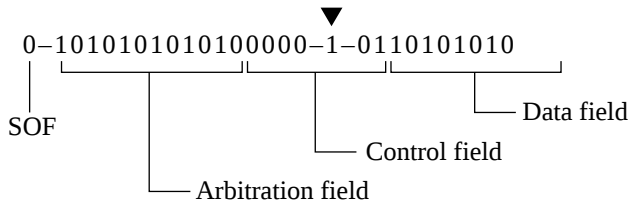
$$R = 15$$

Note: $G(X)$ is stipulated in the CAN protocol as the polynomial that generates the CRC.

As an example, the procedure is described below for a data frame transmitted using the following settings.

```
Settings  SOF: 0
          Arbitration field: 101010101010
          Control field: 000001
          Data field: 10101010
```

The bit pattern from SOF through the data field in the transmitted data is as follows (▼ indicates the stuff bit):



Excluding the stuff bit gives the following data subject to CRC computation:

010101010101000000110101010

Thus, $P(X) = X^{25} + X^{23} + X^{21} + X^{19} + X^{17} + X^{15} + X^8 + X^7 + X^5 + X^3 + X^1$

$P(X)$ is multiplied by X^{15} , giving:

$$X^{15} \cdot P(X) = X^{40} + X^{38} + X^{36} + X^{34} + X^{32} + X^{30} + X^{23} + X^{22} + X^{20} + X^{18} + X^{16}$$

and this value is divided by

$$G(X) = X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$$

$$\left(\begin{array}{l} X^{15} \cdot P(X) = X^{40} + X^{38} + X^{36} + X^{34} + \overset{\downarrow}{X^{32}} + X^{30} + X^{23} + X^{22} + X^{20} + X^{18} + X^{16} \\ P[40:0] = 1010101010100000011010101000000000000000 \\ G(X) = X^{15} + X^{14} + \overset{\downarrow}{X^{10}} + X^8 + X^7 + X^4 + X^3 + 1 \\ G[15:0] = 1100010110011001 \end{array} \right)$$

Quotient

```

00010110011001 ) 10101010101000000110101010000000000000000
                    1100010110011001
                    -----
                    1101111001110010
                    1100010110011001
                    -----
                    110111101011110
                    1100010110011001
                    -----
                    1101011000111101
                    1100010110011001
                    -----
                    1001110100100010
                    1100010110011001
                    -----
                    1011000101110110
                    1100010110011001
                    -----
                    1110100111011110
                    1100010110011001
                    -----
                    1011000100011100
                    1100010110011001
                    -----
                    1110100100001010
                    1100010110011001
                    -----
                    1011001001001100
                    1100010110011001
                    -----
                    1110111110101010
                    1100010110011001
                    -----
                    1010100011001100
                    1100010110011001
                    -----
                    1101101010101010
                    1100010110011001
                    -----
                    1111100110011000
                    1100010110011001
                    -----
                    111100000000010

```

EOR of divisor and dividend is taken.

← Remainder: R[14:0]

The following value is obtained from the calculation.

$$R[14:0] = 111100000000010$$

In other words, this is the CRC value, added after the data field to give transmit data $Tx(X)$, which is transmitted.

In practice, a stuff bit ($\blacktriangle$) and CRC delimiter ($\triangle$) are added, so that 11110000010000101 is transmitted in the CRC field.

Next, the calculation is shown which determines whether there is an error in receive data $Rx(X)$.

$Rx(X)$ is the value obtained by adding the remainder given by the previous calculation to $X^{15} \cdot P(X)$. This value is divided by $G(X)$.

$$\left(\begin{array}{r} P[40:0] = 1010101010100000011010101000000000000000 \\ + R[14:0] = 111100000000010 \\ \hline 10101010101000000110101010111100000000010 \end{array} \right)$$

This value is divided by $G[15:0]$.

EOR of divisor and dividend is taken.

Quotient	
	10101010101000000110101010111100000000010
1100010110011001	<u>1100010110011001</u>
	1101111001110010
1100010110011001	<u>1100010110011001</u>
	1101111101011110
1100010110011001	<u>1100010110011001</u>
	1101011000111101
1100010110011001	<u>1100010110011001</u>
	1001110100100010
1100010110011001	<u>1100010110011001</u>
	1011000101110111
1100010110011001	<u>1100010110011001</u>
	1110100111011101
1100010110011001	<u>1100010110011001</u>
	1011000100010011
1100010110011001	<u>1100010110011001</u>
	1110100100010100
1100010110011001	<u>1100010110011001</u>
	1011001000110100
1100010110011001	<u>1100010110011001</u>
	1110111101011010
1100010110011001	<u>1100010110011001</u>
	1010101100001100
1100010110011001	<u>1100010110011001</u>
	1101110100101010
1100010110011001	<u>1100010110011001</u>
	1100010110011001
1100010110011001	<u>1100010110011001</u>
	1100010110011001
	<u>1100010110011001</u>
	0

0 ← Remainder

The remainder from the above calculation is zero, so the transmission is considered to be error-free.

6. **ACK field:** For confirmation of normal reception.
Comprises a 1-bit ACK slot and a 1-bit ACK delimiter.

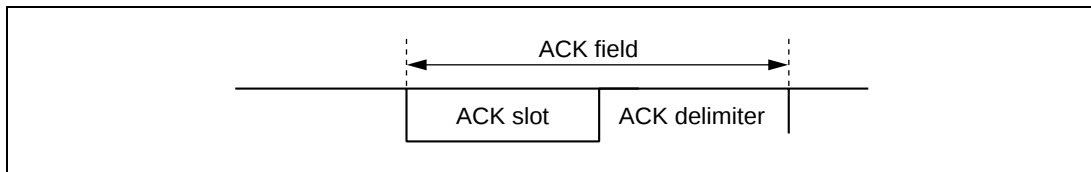


Figure 2.10 ACK Field

The receiving H8S/2623F-ZTAT outputs a high-level ACK slot if it finds an error in the CRC check, and a low-level ACK slot if it finds no error.

7. **Stuff bits:** If there are five consecutive low bits in the data frame, the output is always high for the next bit. Similarly, if there are five consecutive high bits, the output is always low for the next bit.

When stuff bits are output in this way, the bit length of the data frame is increased by the number of stuff bits.

As an example, consider the case where the following settings are made:

Arbitration field: 101010101010

Control field: 000001

The bit pattern in this case is thus 101010101010000001, and so a stuff bit (▼) is output as the second-but-lowest bit (after the five consecutive 0s).

The value transmitted on the CAN bus is therefore 10101010101000001▼01, and the data frame length is increased by the one stuff bit.

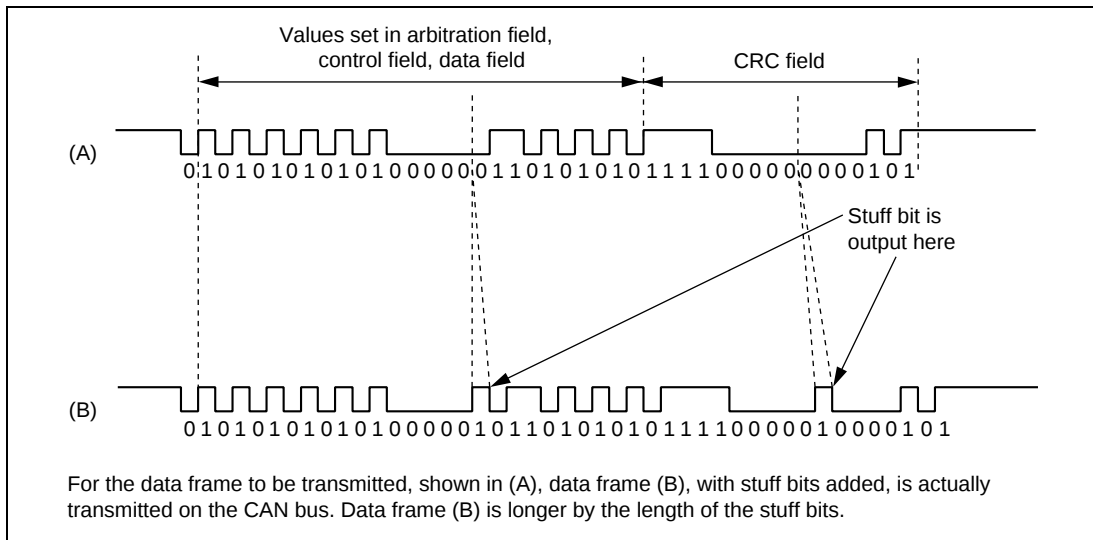


Figure 2.11 Stuff Bits

8. **Configuration mode:** In this mode, the HCAN module is in the reset state. This mode is released by clearing the reset request bit (MCR0) in the master control register (MCR).

Section 3 HCAN Transmission/Reception (Example 3): Extended Format, 1-Byte Data

MCU: H8S/2623F-ZTAT	Function Used: HCAN
----------------------------	----------------------------

3.1 Specifications

1. Data frame*¹ transmission/reception (using two H8S/2623F-ZTATs).
Data frame specifications are shown in figure 3.1.
 - a. SOF: Indicates start of data frame.
 - b. Arbitration field*²
 - 11-bit identifier = 10101010101
 - SRR (Substitute Remote Request) = 1: Select extended format
 - IDE (ID Extension) = 1: Select extended format
 - 18-bit identifier = 1010101010101010
 - RTR = 0: Select data frame
 - c. Control field*³: Set to 000001.
 - R0 = 0: Reserved bit
 - R1 = 0: Reserved bit
 - DLC = 0001: Set data length of 1 byte
 - d. Data field*⁴: Set to 10101010.
 - e. CRC field*⁵: CRC is generated automatically within the HCAN.
 - f. ACK field*⁶: 11 is output on the transmitting side, and 01 (in normal operation) on the receiving side.
 - g. EOF: Indicates the end of a transmit/receive data frame.
2. A communication speed of 1 Mbps (when operating at 20 MHz) is set.
3. The data length is set to 1 byte.
4. Message transmission uses mailbox 1.
5. Message reception uses mailbox 0. The message reception method is to mask the identifier and receive the message in case of a match.
6. Select interrupt control mode 2, and
 - a. Use bus operation interrupt (OVR0).
 - b. Use message reception interrupt (RM0).

7. Receive messages are stored in on-chip RAM.
8. Use HCAN sleep mode.
 - a. After initial settings, transfer to HCAN sleep mode (released by CAN bus operation).
 - b. After reception processing, transfer to HCAN sleep mode (released by CAN bus operation).
9. Figure 3.2 shows an example of CAN bus connection.

Note: * See 3.9 Notes.

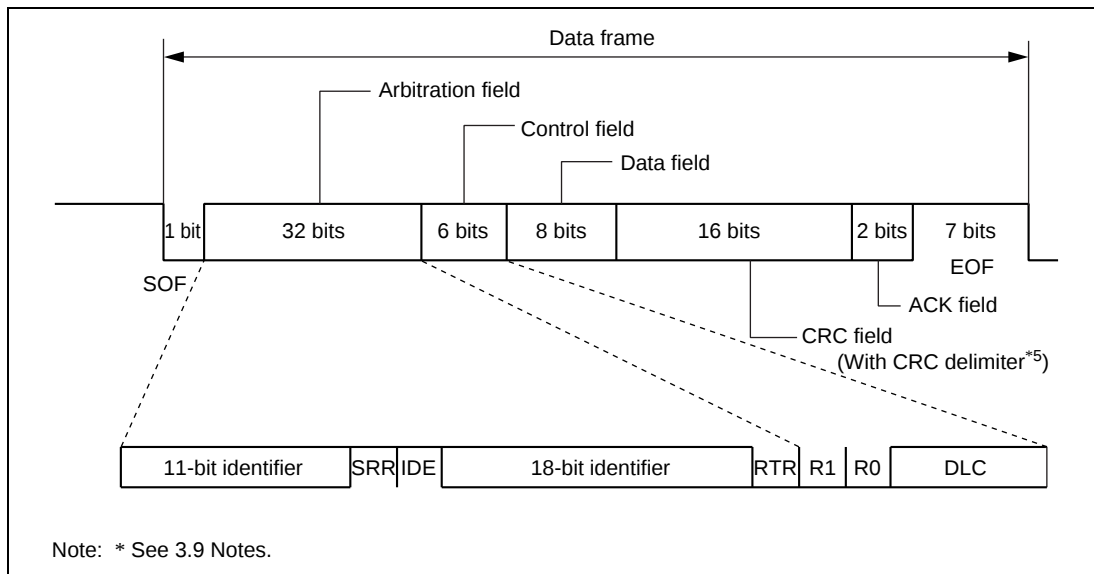


Figure 3.1 Data Frame Specifications

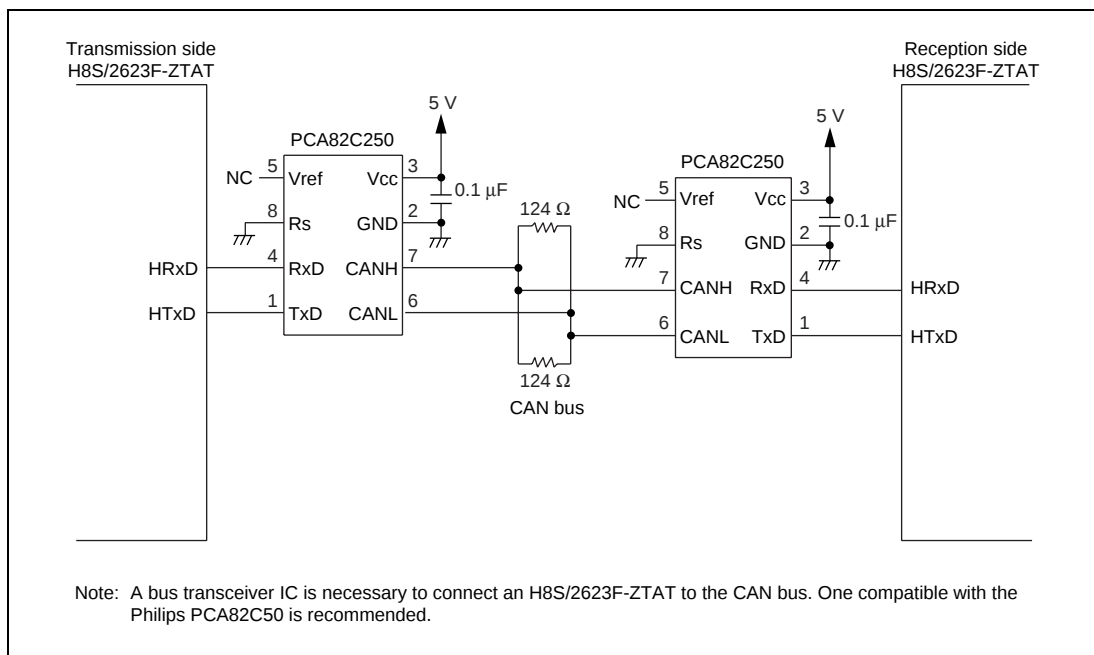


Figure 3.2 CAN Interface Using H8S/2623F-ZTATs

3.2 Functions

Tables 3.1 to 3.3 show the function allocation of this sample task. This sample task allocates H8S/2623F-ZTAT on-chip HCAN functions as shown in these tables, and carries out HCAN transmission and reception.

Table 3.1 HCAN Function Allocation

HCAN Register		Function
Pins	HTxD	Transmits messages.
	HRxD	Receives messages.
Transmission/ reception registers	IRR	Displays status of each interrupt source.
	BCR	Sets CAN baud rate prescaler and bit timing parameters.
	MBCR	Sets mailbox transmission/reception.
	MCR	Controls CAN interface.
	MC0_1 to MC15_8	Arbitration field and control field settings.
	MD0_1 to MD15_8	Data field settings.
Transmission registers	TXPR	Sets transmission wait state after transmit messages are stored in the mailbox.
	TXACK	Indicates that the transmit message of the corresponding mailbox was transmitted normally.
Reception registers	MBIMR	Sets enabling of mailbox 0 interrupt requests.
	IMR	Sets enabling of receive message and bus operation interrupts.
	LAFMH, L	Sets filter mask for reception mailbox 0 identifier.
	RXPR	Clears receive message interrupt flag.

Table 3.2 MSTPCR Function Allocation

MSTPCR Register	Function
MSTPCRC	Controls module stop mode.

Table 3.3 Interrupt Controller Allocation

IPR Register	Function
IPRM	Controls interrupt priority level.
SYSCR	Sets interrupt control mode.
exr	Specifies interrupt request mask level.

3.3 Operation

Operation (Transmission)

Figure 3.3 shows the principle of operation during transmission. HCAN transmission is carried out by H8S/2623F-ZTAT hardware processing and software processing as shown in the figure.

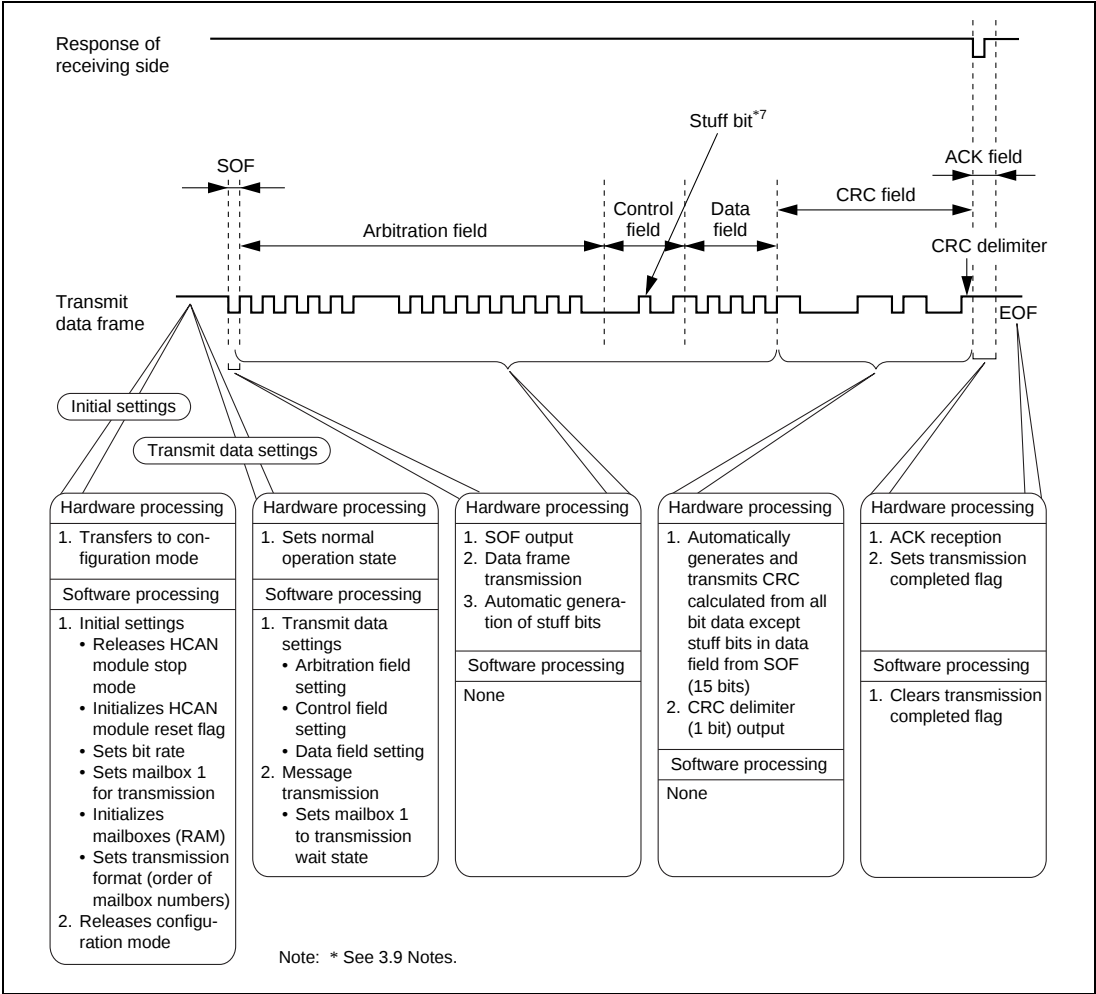


Figure 3.3 Operation During HCAN Transmission

Operation (Reception)

Figure 3.4 shows the principle of operation during reception. HCAN reception is carried out by H8S/2623F-ZTAT hardware processing and software processing as shown in the figure.

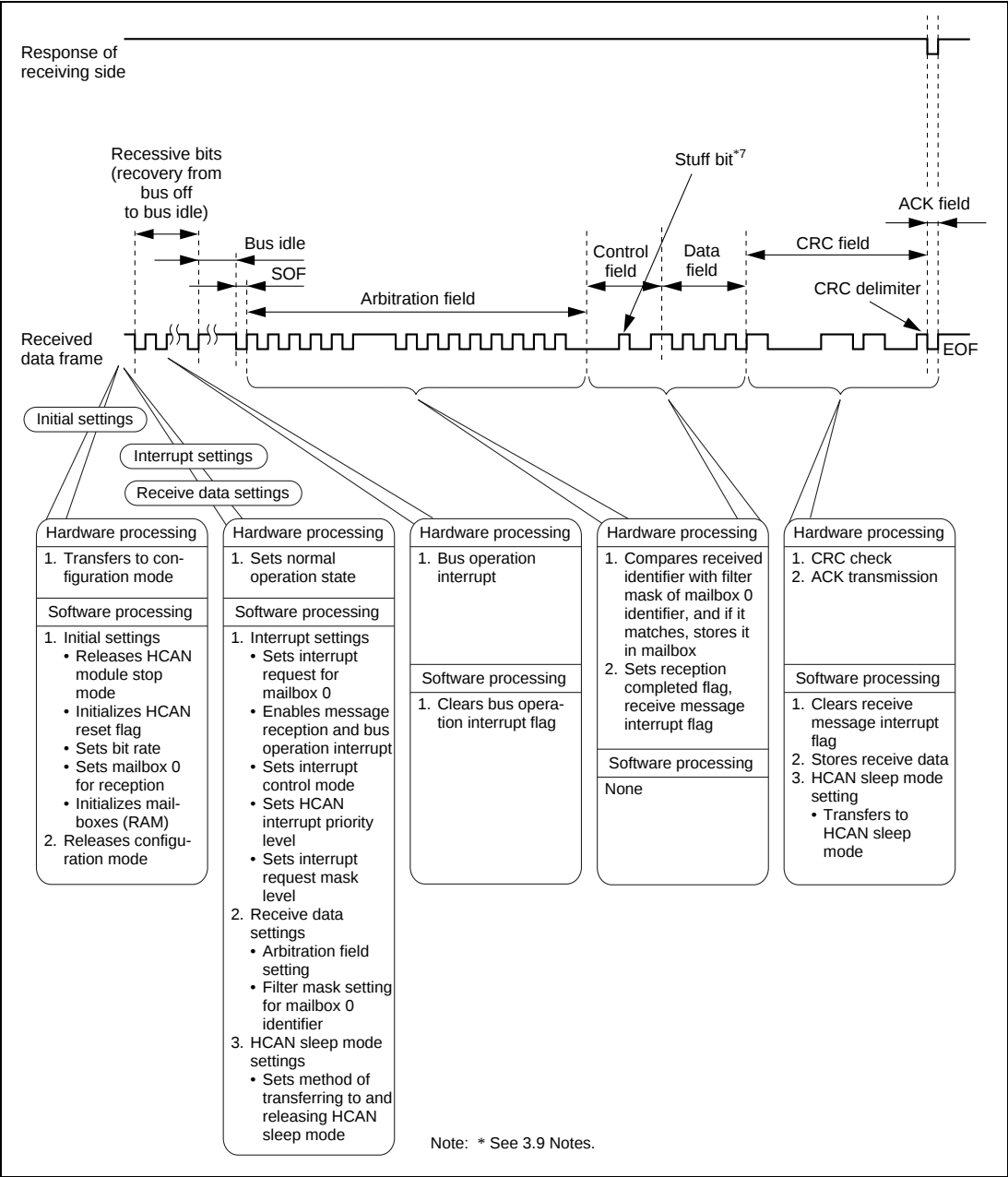


Figure 3.4 Operation During HCAN Reception

3.4 Software

1. Modules

Module Name	Label	Function
Main routine	main	HCAN initial settings and transmission/reception settings.
Bus operation interrupt routine	OVR0_IRR12	Clears bus operation interrupt flag.
Message reception interrupt routine	RM0	Transfers messages stored in mailbox 0 to on-chip RAM.

2. Variables Used

Label	Function	Data Length	Module
COUNT	Initializes HCAN_MC0_1 to MC15_8, and HCAN_MD0_1 to MD15_8.	Unsigned short	Main routine

3. Internal Registers Used

Register Name	Function	Setting	Module
Settings Common to Transmission/Reception			
MSTPCRC	Releases HCAN module stop mode.	0xF7	Main routine
HCAN_BCR	Sets HCAN bit rate to 1 Mbps.	0x0025	
Settings for Transmission			
HCAN_IRR	Initializes HCAN module reset flag.	0x0100	Main routine
HCAN_MBCR	Sets mailbox 1 for transmission.	0xFDFF	
HCAN_MCR	Sets transmission in order of mailbox numbers, and clears reset request bit.	0x04	
HCAN_MC1_1	Sets 1-byte data length.	0x01	
HCAN_MC1_5	Selects mailbox 1 data frame and extended format, and sets identifier	0xAA	
HCAN_MC1_6	Sets mailbox 1 identifier.	0xAA	
HCAN_MC1_7	Sets mailbox 1 identifier.	0xAA	
HCAN_MC1_8	Sets mailbox 1 identifier.	0xAA	
HCAN_MD1_1	Sets transmit data for mailbox 1.	0xAA	
HCAN_TXPR	Sets mailbox 1 to transmission wait state.	0x0200	
HCAN_TXACK	Clears data frame transmission completed flag.	0x0200	

Register Name	Function	Setting	Module
Settings for Reception			
HCAN_IRR	Initializes HCAN module reset flag.	0x0100	Main routine
	Clears bus operation interrupt flag.	0x0010	Bus operation interrupt
HCAN_MCR	Clears reset request bit.	0xFE	Main routine
	Sets transfer to HCAN sleep mode and release by bus operation.	0xA0	
	Sets transfer to HCAN sleep mode.	0x20	Reception interrupt
HCAN_MBCR	Sets mailbox 0 for reception.	0x0100	Main routine
HCAN_MC0_5	Selects mailbox 0 data frame and extended format, and sets identifier.	0xAA	
HCAN_MC0_6	Sets mailbox 0 identifier.	0xAA	
HCAN_MC0_7	Sets mailbox 0 identifier.	0xAA	
HCAN_MC0_8	Sets mailbox 0 identifier.	0xAA	
HCAN_LAFMH	Sets filter mask for mailbox 0 identifier.	0x0000	
HCAN_LAFML	Sets filter mask for mailbox 0 identifier.	0x0000	
HCAN_MBIMR	Sets enabling of mailbox 0 interrupt requests.	0xFEFF	
HCAN_IMR	Sets enabling of message reception and bus operation interrupt requests.	0xFCEF	
INTC.IPRM	Sets HCAN interrupt priority level to 7.	0x70	
SYSCR	Sets interrupt control mode 2.	0x20	
exr	Specifies interrupt request mask level.	0x00	
HCAN_RXPR	Clears receive message interrupt flag.	0xFFFF	Reception interrupt

Note: Reception interrupt: Message reception interrupt routine
Bus operation: Bus operation interrupt routine

4. RAM Used

Symbol	Function	Address	Module
Settings for Reception			
Message_DATA	Storage destination address for mailbox 0 data.	0xFFC100	Main routine

3.5 Transmission Flowchart

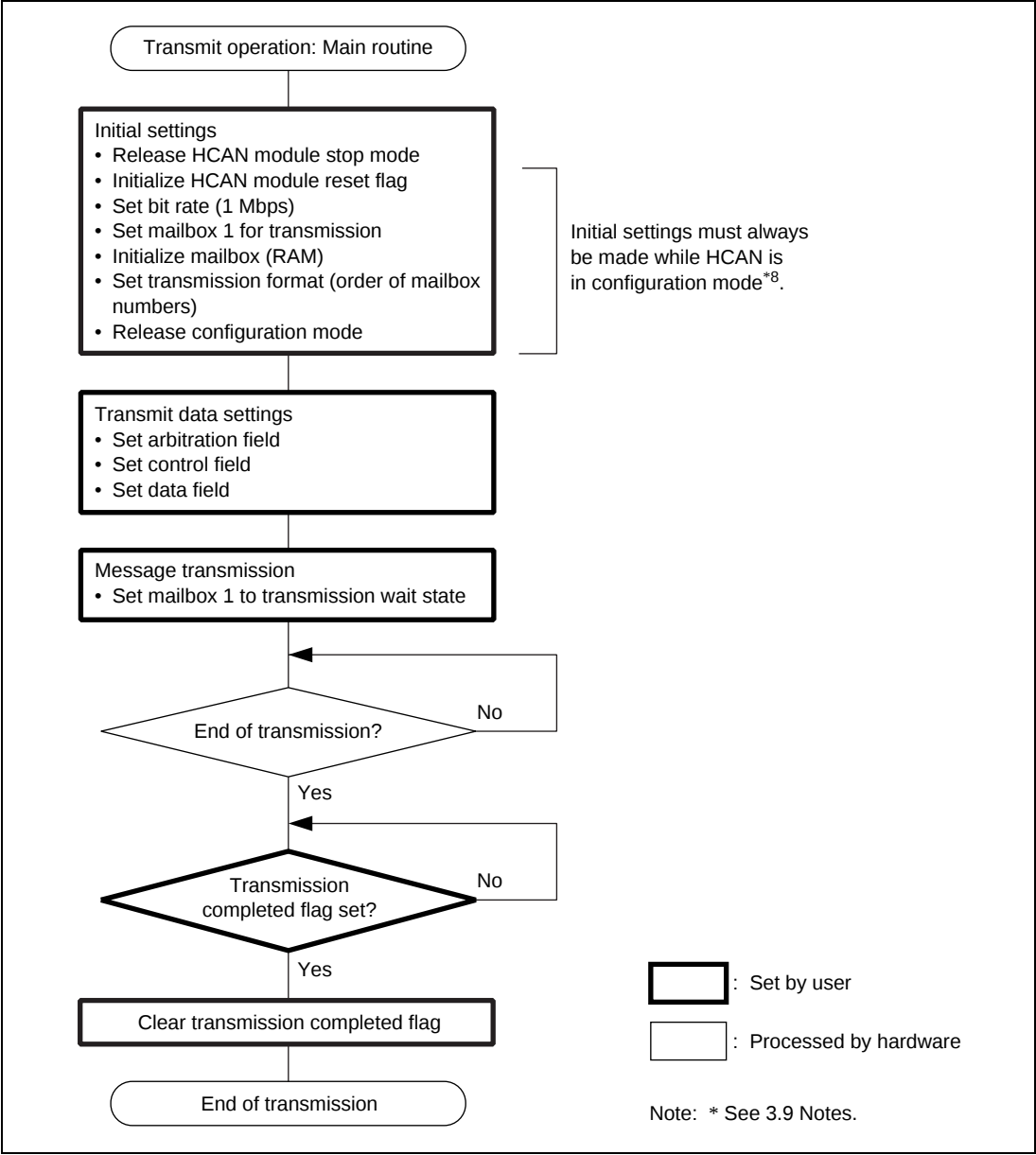


Figure 3.5 Transmission Flowchart

3.6 Transmission Program List

```

/*****
/*          HCAN Transmission Program          */
*****/
#include <stdio.h>                /* Library function header file          */
#include <machine.h>              /* Library function header file          */
#include "2623.h"                /* Peripheral register definition header file */
/*****
/*          Function Protocol Declaration          */
*****/
void main( void );
/*****
/*          Definition of Constants          */
*****/
#define COUNT (*(unsigned short *)0xFFC000)
/*****
/*          Main Routine          */
*****/
void main(void)
{
/* Initial Settings */
    MSTPCRC = 0xF7;                /* Release HCAN module stop mode          */
    HCAN_IRR = 0x0100;            /* Initialize HCAN module reset flag      */
    HCAN_BCR = 0x0025;            /* Bit rate: 1 Mbps                      */
    HCAN_MBCR = 0xFDF;           /* Set mailbox 1 for transmission          */
    for( COUNT = 0; COUNT < 128; COUNT++ ) /* Initialize mailbox (RAM)          */
    {
        *(char*)&HCAN_MC0_1 + COUNT = 0x00;
    }
    for( COUNT = 0; COUNT < 128; COUNT++ ) /* Initialize mailbox (RAM)          */
    {
        *(char*)&HCAN_MD0_1 + COUNT = 0x00;
    }
    HCAN_MCR = 0x04;
    /* Set transmission in order of mailbox numbers, and release configuration
       mode */
/* Transmit data settings */
    HCAN_MC1_5 = 0xAA;
        /* Select data frame and extended format, and set identifier */
    HCAN_MC1_6 = 0xAA;                /* Set identifier          */

```

```

HCAN_MC1_7 = 0xAA;                /* Set identifier */
HCAN_MC1_8 = 0xAA;                /* Set identifier */
HCAN_MC1_1 = 0x01;                /* Data length: 1 byte */
HCAN_MD1_1 = 0xAA;                /* Message contents: 10101010 */

/* Message transmission */
HCAN_TXPR = 0x0200;                /* Set mailbox 1 to transmission wait state */
while((HCAN_TXACK & 0x0200) != 0x0200);

/* Clear transmission completed flag */
HCAN_TXACK &= 0x0200;              /* Clear transmission completed flag */
while(1);

}

```

3.7 Reception Flowcharts

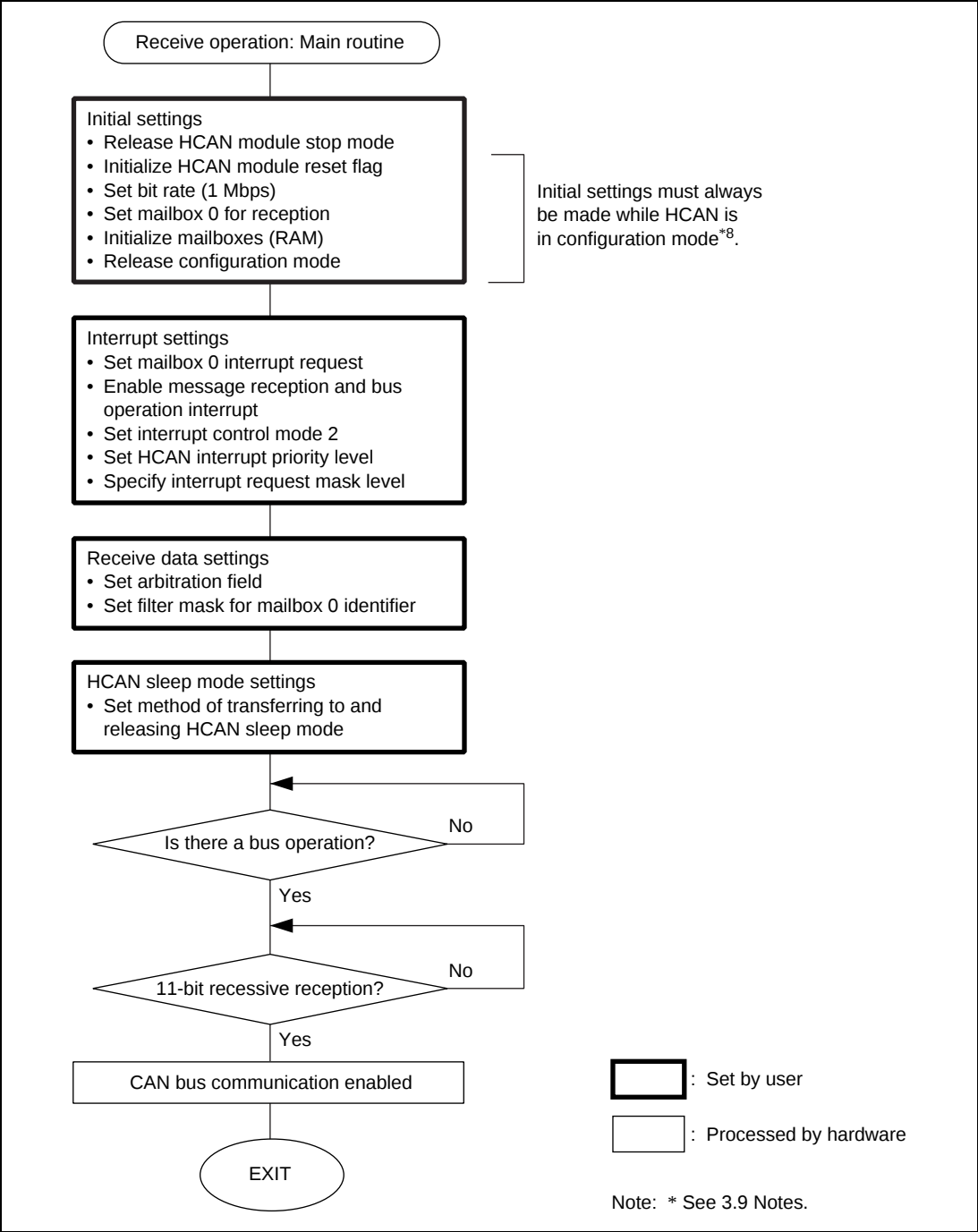


Figure 3.6 Reception Flowchart (1)

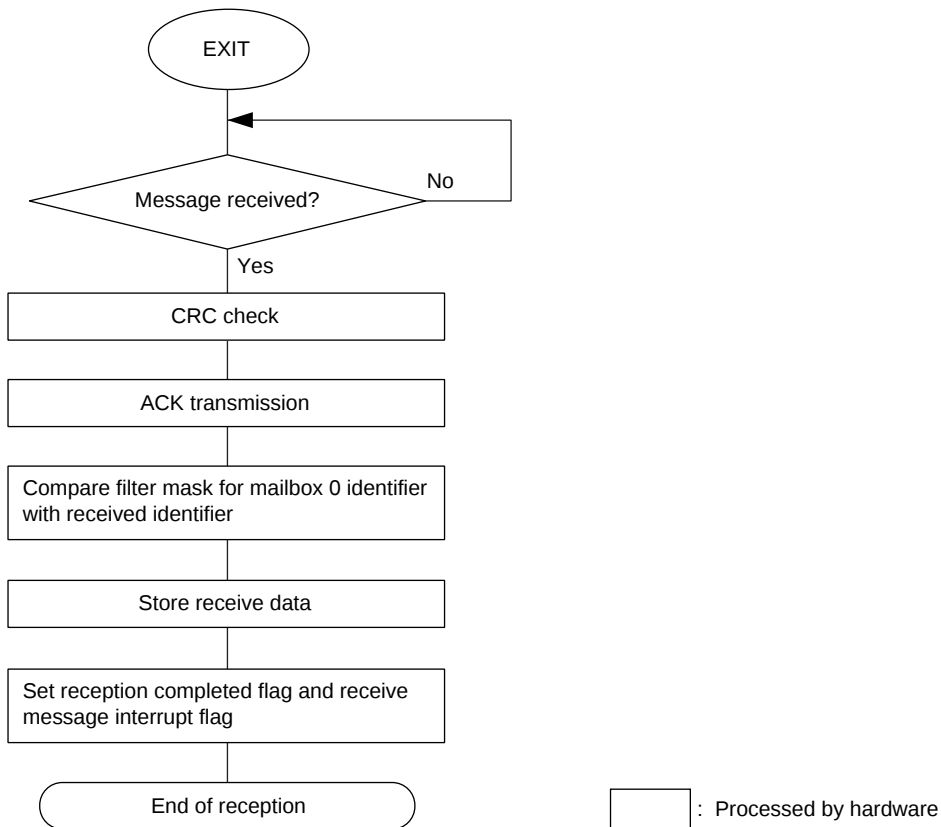


Figure 3.6 Reception Flowchart (2)

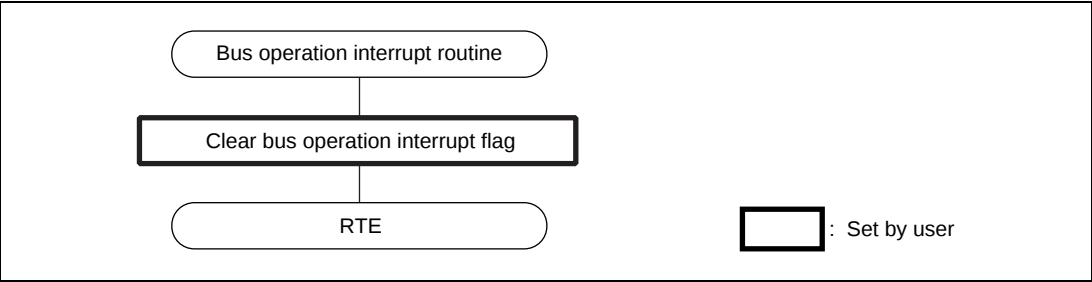


Figure 3.7 Bus Operation Interrupt Flowchart

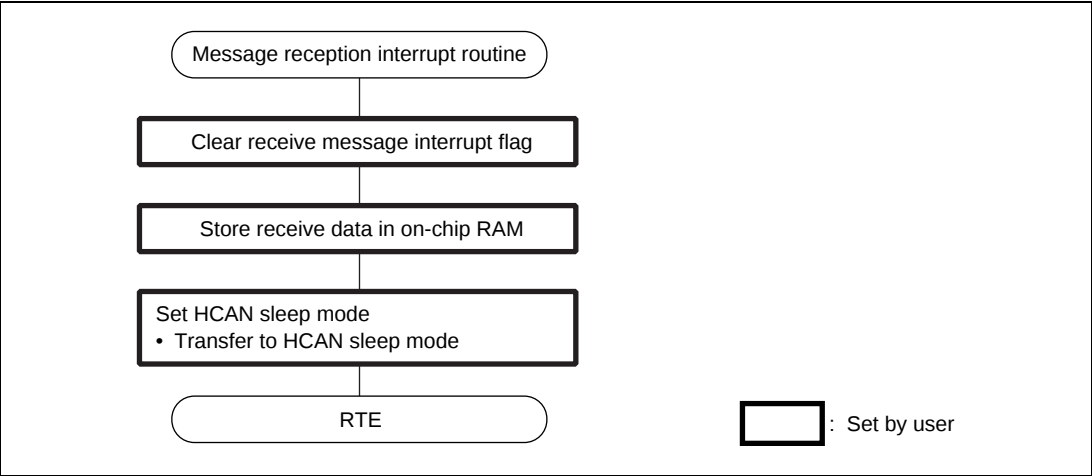


Figure 3.8 Message Reception Interrupt Flowchart

3.8 Reception Program List

```

/*****
/*          HCAN Reception Program          */
*****/
#include <stdio.h>          /* Library function header file          */
#include <machine.h>        /* Library function header file          */
#include "2623.h"           /* Peripheral register definition header file */
/*****
/*          Function Protocol Declaration    */
*****/
void main( void );
/*****
/*          Definition of Constants          */
*****/
#define COUNT      (*(unsigned short *)0xFFC000)
#define Message_DATA (*(unsigned char *)0xFFC100) /* Store receive data */
/*****
/*          Main Routine                    */
*****/
void main(void)
{
/* Initial settings */
    MSTPCRC = 0xF7;          /* Release HCAN module stop mode          */
    HCAN_IRR = 0x0100;       /* Initialize HCAN module reset flag      */
    HCAN_BCR = 0x0025;       /* Bit rate: 1 Mbps                      */
    HCAN_MBCR = 0x0100;      /* Set mailbox 0 for reception            */
    for( COUNT = 0; COUNT < 128; COUNT++ ) /* Initialize mailbox (RAM)              */
    {
        *(char*)&HCAN_MC0_1 + COUNT = 0x00;
    }
    for( COUNT = 0; COUNT < 128; COUNT++ ) /* Initialize mailbox (RAM)              */
    {
        *(char*)&HCAN_MD0_1 + COUNT = 0x00;
    }
    HCAN_MCR &= 0xFE;        /* Release configuration mode            */
/* Interrupt settings */
    HCAN_MBIMR = 0xFEFF;     /* Enable mailbox 0 interrupt request     */
    HCAN_IMR = 0xFCEF; /* Enable message reception and bus operation interrupt */
    SYSCR |= 0x20;          /* Set interrupt control mode 2          */

```

```

    INTC.IPRM = 0x70;                /* Set HCAN interrupt priority level to 7 */
    set_imask_exr(0);                /* Specify interrupt request mask level */
/* Reception data settings */
    HCAN_MC0_5 = 0xAA;               /* Set extended format, data frame, and identifier */
    HCAN_MC0_6 = 0xAA;               /* Set identifier */
    HCAN_MC0_7 = 0xAA;               /* Set identifier */
    HCAN_MC0_8 = 0xAA;               /* Set identifier */
    HCAN_LAFMH = 0x0000;             /* Mailbox 0 stores data in case of bit match */
    HCAN_LAFML = 0x0000;             /* Mailbox 0 stores data in case of bit match */
/* HCAN sleep mode setting */
    HCAN_MCR |= 0xA0;                /* Enable transfer to HCAN mode and release by bus operation */
    while(1);
}
/*****
/*
    Bus Operation Interrupt Routine
*****/
#pragma interrupt(OVR0_IRR12)
void OVR0_IRR12(void)
{
    HCAN_IRR &= 0x0010;              /* Clear IRR12 (bus operation interrupt flag) */
}
/*****
/*
    Message Reception Interrupt Routine
*****/
#pragma interrupt(RM0)
void RM0(void)
{
    HCAN_RXPR &= 0xFFFF;             /* Clear IRR1 (receive message interrupt flag) */
    Message_DATA = HCAN_MD0_1;        /* Store receive data */
/* HCAN sleep mode setting */
    HCAN_MCR |= 0x20;                /* Transfer to HCAN sleep mode */
}

```

3.9 Notes

1. **Data frame:** Data to be transferred from the transmission source to the transmission destination.
2. **Arbitration field:** Set unique ID for message and data frame or remote frame.
3. **Control field:** Set the data length to be transmitted, and standard format or extended format.
4. **Data field:** Set message contents (data to be transmitted).
5. **CRC field:** A CRC is generated automatically in the HCAN from all bit data except stuff bits in the data field from SOF, and is used to detect transmit message errors. The CRC field comprises a 15-bit CRC and a 1-bit delimiter. The CRC delimiter is always output as a 1 after the CRC.

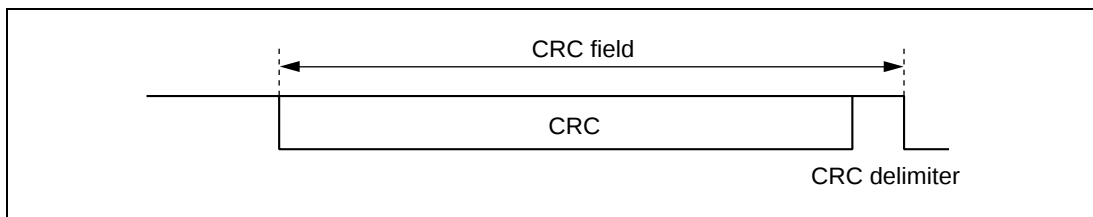


Figure 3.9 CRC Field

About the CRC:

Transmit data polynomial $P(X)$ is multiplied by X^R , then $X^R \cdot P(X)$ is divided by generating polynomial $G(X)$ to give a remainder, $R(X)$. On the side transmitting information, $R(X)$ found from $X^R \cdot P(X)$ is added as check bits, and the result is sent as transmit data $Tx(X)$.

On the side receiving the information, receive data $Rx(X)$ is divided by generating polynomial $G(X)$ to give a remainder. If this remainder is zero, information transmission is regarded as having been completed normally. If the remainder is nonzero, an error is judged to have occurred in the information transmitted.

$$P(X) = \{\text{SOF through data field, excluding stuff bits}\}$$

$$G(X) = X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$$

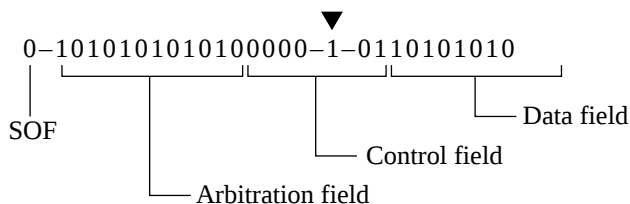
$$R = 15$$

Note: $G(X)$ is stipulated in the CAN protocol as the polynomial that generates the CRC.

As an example, the procedure is described below for a data frame transmitted using the following settings.

Settings SOF: 0
 Arbitration field: 101010101010
 Control field: 000001
 Data field: 10101010

The bit pattern from SOF through the data field in the transmitted data is as follows (▼ indicates the stuff bit):



Excluding the stuff bit gives the following data subject to CRC computation:

010101010101000000110101010

Thus, $P(X) = X^{25} + X^{23} + X^{21} + X^{19} + X^{17} + X^{15} + X^8 + X^7 + X^5 + X^3 + X^1$

$P(X)$ is multiplied by X^{15} , giving:

$$X^{15} \cdot P(X) = X^{40} + X^{38} + X^{36} + X^{34} + X^{32} + X^{30} + X^{23} + X^{22} + X^{20} + X^{18} + X^{16}$$

and this value is divided by

$$G(X) = X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$$

$$\left(\begin{array}{l} X^{15} \cdot P(X) = X^{40} + X^{38} + X^{36} + X^{34} + X^{32} + X^{30} + X^{23} + X^{22} + X^{20} + X^{18} + X^{16} \\ \downarrow \\ P[40:0] = 1010101010100000011010101000000000000000 \\ G(X) = X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1 \\ \downarrow \\ G[15:0] = 1100010110011001 \end{array} \right)$$

	Quotient	
1100010110011001	1010101010100000011010101000000000000000	
	1100010110011001	
	1101111001110010	
	1100010110011001	
	1101111101011110	
	1100010110011001	
	1101011000111101	
	1100010110011001	
	1001110100100010	
	1100010110011001	
	1011000101110110	
	1100010110011001	
	1110100111011110	
	1100010110011001	
	1011000100011100	
	1100010110011001	
	1110100100001010	
	1100010110011001	
	1011001001001100	
	1100010110011001	
	1110111110101010	
	1100010110011001	
	1010100011001100	
	1100010110011001	
	1101101010101010	
	1100010110011001	
	1111100110011000	
	1100010110011001	
	11110000000010	← Remainder: R[14:0]

The following value is obtained from the calculation.

$$R[14:0] = 111100000000010$$

In other words, this is the CRC value, added after the data field to give transmit data $T_x(X)$, which is transmitted.

In practice, a stuff bit (▲) and CRC delimiter (△) are added, so that 11110000010000101▲△ is transmitted in the CRC field.

$Rx(X)$ is the value obtained by adding the remainder given by the previous calculation to $X^{15} \cdot P(X)$. This value is divided by $G(X)$.

$$\left(\begin{array}{r} P[40:0] = 101010101010000001101010100000000000000 \\ + R[14:0] = 111100000000010 \\ \hline 10101010101000000110101010111100000000010 \end{array} \right)$$

This value is divided by G[15:0].

	Quotient
1100010110011001	$\overline{)10101010101000000110101010111100000000010}$
	$\underline{1100010110011001}$
	1101111001110010
	$\underline{1100010110011001}$
	1101111101011110
	$\underline{1100010110011001}$
	1101011000111101
	$\underline{1100010110011001}$
	1001110100100010
	$\underline{1100010110011001}$
	1011000101110111
	$\underline{1100010110011001}$
	1110100111011101
	$\underline{1100010110011001}$
	1011000100010011
	$\underline{1100010110011001}$
	1110100100010100
	$\underline{1100010110011001}$
	1011001000110100
	$\underline{1100010110011001}$
	1110111101011010
	$\underline{1100010110011001}$
	1010101100001100
	$\underline{1100010110011001}$
	1101110100101010
	$\underline{1100010110011001}$
	1100010110011001
	$\underline{\hspace{1cm}}$
	$[0]$ ← Remainder

The remainder from the above calculation is zero, so the transmission is considered to be error-free.

6. **ACK field:** For confirmation of normal reception.
Comprises a 1-bit ACK slot and a 1-bit ACK delimiter.

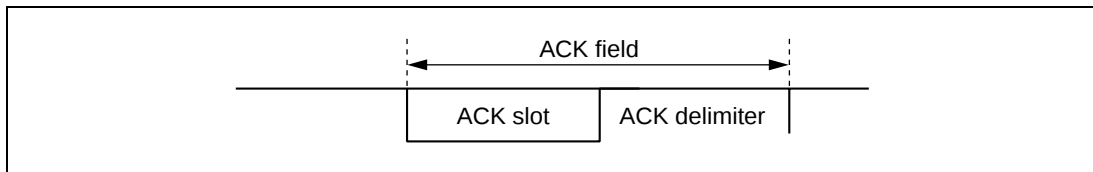


Figure 3.10 ACK Field

The receiving H8S/2623F-ZTAT outputs a high-level ACK slot if it finds an error in the CRC check, and a low-level ACK slot if it finds no error.

7. **Stuff bits:** If there are five consecutive low bits in the data frame, the output is always high for the next bit. Similarly, if there are five consecutive high bits, the output is always low for the next bit.

When stuff bits are output in this way, the bit length of the data frame is increased by the number of stuff bits.

As an example, consider the case where the following settings are made:

Arbitration field: 101010101010

Control field: 000001

The bit pattern in this case is thus 101010101010000001, and so a stuff bit (▼) is output as the second-but-lowest bit (after the five consecutive 0s).

The value transmitted on the CAN bus is therefore 1010101010100000▼101, and the data frame length is increased by the one stuff bit.

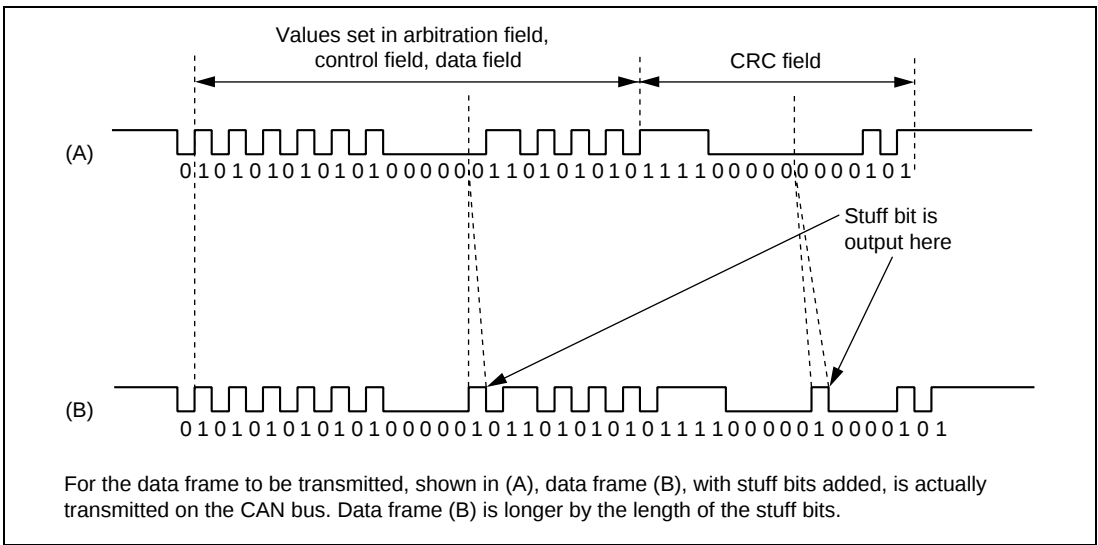


Figure 3.11 Stuff Bits

8. **Configuration mode:** In this mode, the HCAN module is in the reset state. This mode is released by clearing the reset request bit (MCR0) in the master control register (MCR).

Section 4 HCAN Transmission/Reception (Example 4): Standard Format, 8-Byte Data, Prioritized

MCU: H8S/2623F-ZTAT	Function Used: HCAN, DTC
----------------------------	---------------------------------

4.1 Specifications

1. Data frame*¹ transmission/reception (using two H8S/2623F-ZTATs).

Data frame specifications are shown in figure 4.1.

a. SOF: Indicates start of data frame.

b. Arbitration field*²

- Mailbox 1: 110011001100 (H'CCC) → Priority is 12th highest.
- Mailbox 2: 010101010100 (H'554) → Priority is 5th highest.
- Mailbox 3: 111011101110 (H'EEE) → Priority is 14th highest.
- Mailbox 4: 011001100110 (H'666) → Priority is 6th highest.
- Mailbox 5: 000100010000 (H'110) → Priority is highest.
- Mailbox 6: 100110011000 (H'998) → Priority is 9th highest.
- Mailbox 7: 001100110010 (H'332) → Priority is 3rd highest.
- Mailbox 8: 111111111110 (H'FFE) → Priority is lowest.
- Mailbox 9: 001000100010 (H'222) → Priority is 2nd highest.
- Mailbox 10: 100010001000 (H'888) → Priority is 8th highest.
- Mailbox 11: 101010101010 (H'AAA) → Priority is 10th highest.
- Mailbox 12: 110111011100 (H'DDC) → Priority is 13th highest.
- Mailbox 13: 011101110110 (H'776) → Priority is 7th highest.
- Mailbox 14: 010001000100 (H'444) → Priority is 4th highest.
- Mailbox 15: 101110111010 (H'BBA) → Priority is 11th highest.

c. Control field*³: Set to 00100. (Mailboxes 1 to 15)

- IDE = 0: Select standard format
- RO = 0: Reserved bit
- DLC = 1000: Set data length of 8 bytes

d. Data field*⁴: Set to 10101010.

- Mailbox 1: H'1111111111111111
- Mailbox 2: H'2222222222222222
- Mailbox 3: H'3333333333333333
- Mailbox 4: H'4444444444444444
- Mailbox 5: H'5555555555555555
- Mailbox 6: H'6666666666666666

- Mailbox 7: H'7777777777777777
 - Mailbox 8: H'8888888888888888
 - Mailbox 9: H'9999999999999999
 - Mailbox 10: H'AAAAAAAAAAAAAAAAAAAA
 - Mailbox 11: H'BBBBBBBBBBBBBBBBBB
 - Mailbox 12: H'CCCCCCCCCCCCCCCC
 - Mailbox 13: H'DDDDDDDDDDDDDDDDD
 - Mailbox 14: H'EEEEEEEEEEEEEEEE
 - Mailbox 15: H'FFFFFFFFFFFFFFFF
- e. CRC field*⁵: CRC is generated automatically within the HCAN. (Mailboxes 1 to 15)
 - f. ACK field*⁶: 11 is output on the transmitting side, and 01 (in normal operation) on the receiving side. (Mailboxes 1 to 15)
 - g. EOF: Indicates the end of a transmit/receive data frame. (Mailboxes 1 to 15)
2. A communication speed of 1 Mbps (when operating at 20 MHz) is set.
 3. The data length is set to 8 bytes. (Mailboxes 1 to 15)
 4. Message transmission uses mailboxes 1 to 15.
 5. Message reception uses mailboxes 0 to 15. The message reception method for mailbox 0 is to mask the identifier and receive the message in case of a match.
 6. Messages are transmitted in message identifier priority order.
 7. Receive messages are stored in on-chip RAM using the DTC.
 - a. After all messages have been received, the DTC is started by software.
 - b. Use block transfer mode.
 8. After a DTC transfer end interrupt, transfer to HCAN sleep mode.
 9. Figure 4.2 shows an example of CAN bus connection.

Note: * See 4.9 Notes.

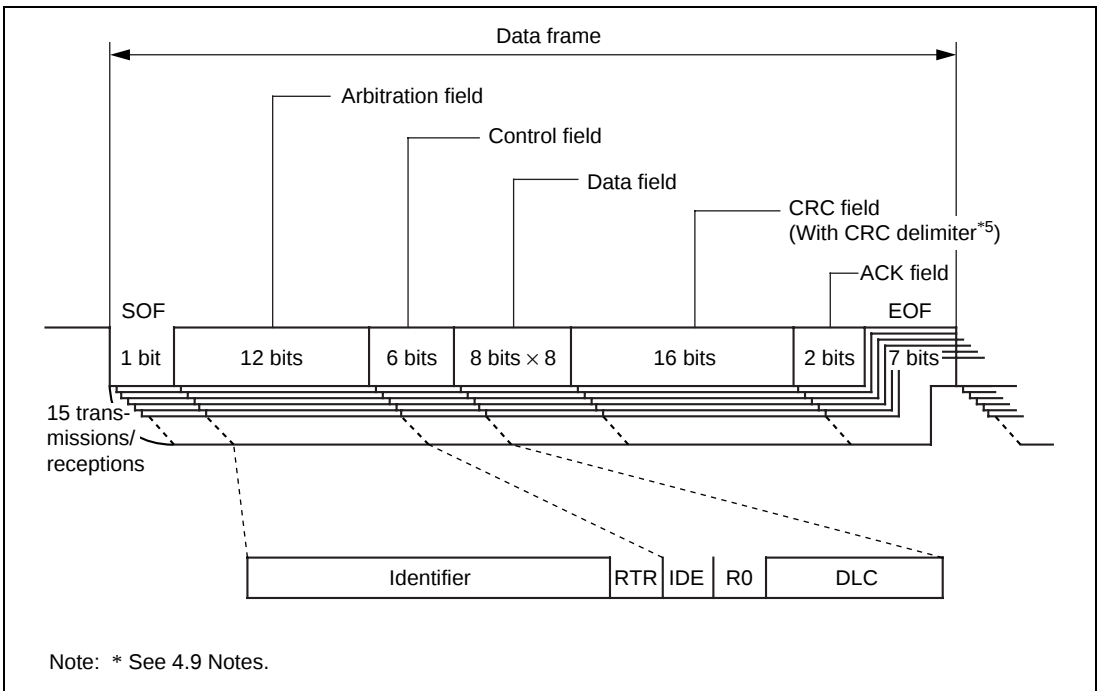


Figure 4.1 Data Frame Specifications

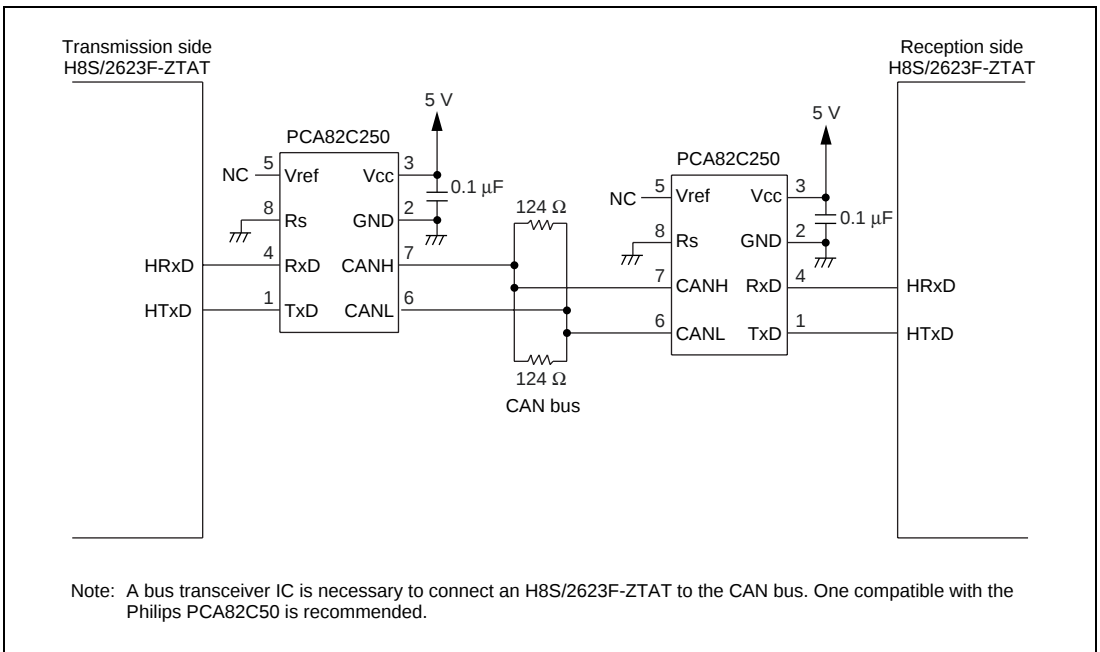


Figure 4.2 CAN Interface Using H8S/2623F-ZTATs

4.2 Functions

Tables 4.1 to 4.4 show the function allocation of this sample task. This sample task allocates H8S/2623F-ZTAT on-chip HCAN functions as shown in these tables, and carries out HCAN transmission and reception.

Table 4.1 HCAN Function Allocation

HCAN Register		Function
Pins	HTxD	Transmits messages.
	HRxD	Receives messages.
Transmission/ reception registers	IRR	Displays status of each interrupt source.
	BCR	Sets CAN baud rate prescaler and bit timing parameters.
	MBCR	Sets mailbox transmission/reception.
	MCR	Controls CAN interface.
	MC0_1 to MC15_8	Arbitration field and control field settings.
	MD0_1 to MD15_8	Data field settings.
Transmission registers	TXPR	Sets transmission wait state after transmit messages are stored in the mailbox.
	TXACK	Indicates that the transmit message of the corresponding mailbox was transmitted normally.
Reception registers	LAFMH	Sets filter mask for reception mailbox 0 identifier.
	RXPR	Indicates that data has been received normally by the corresponding mailbox.

Table 4.2 MSTPCR Function Allocation

MSTPCR Register	Function
MSTPCRC	Controls module stop mode.
MSTPCRA	

Table 4.3 Interrupt Controller Allocation

IPR Register	Function
IPRC	Controls interrupt priority level.
SYSCR	Sets interrupt control mode.
exr	Specifies interrupt request mask level.

Table 4.4 DTC Function Allocation

DTC Register	Function
SAR (On-chip RAM)	Sets source address (where receive messages are stored from).
DAR (On-chip RAM)	Sets destination address (where receive messages are stored to).
MAR (On-chip RAM)	Sets block transfer mode, byte size transfer, etc.
MRB (On-chip RAM)	Sets interrupt to CPU after DTC data transfer.
CRA (On-chip RAM)	Sets number of DTC data transfers.
CRB (On-chip RAM)	Sets number of DTC block data transfers.
DTVECR (Register)	Sets DTC start by software, and vector number.

4.3 Operation

Operation (Transmission)

Figure 4.3 shows the principle of operation during transmission. HCAN transmission is carried out by H8S/2623F-ZTAT hardware processing and software processing as shown in the figure.

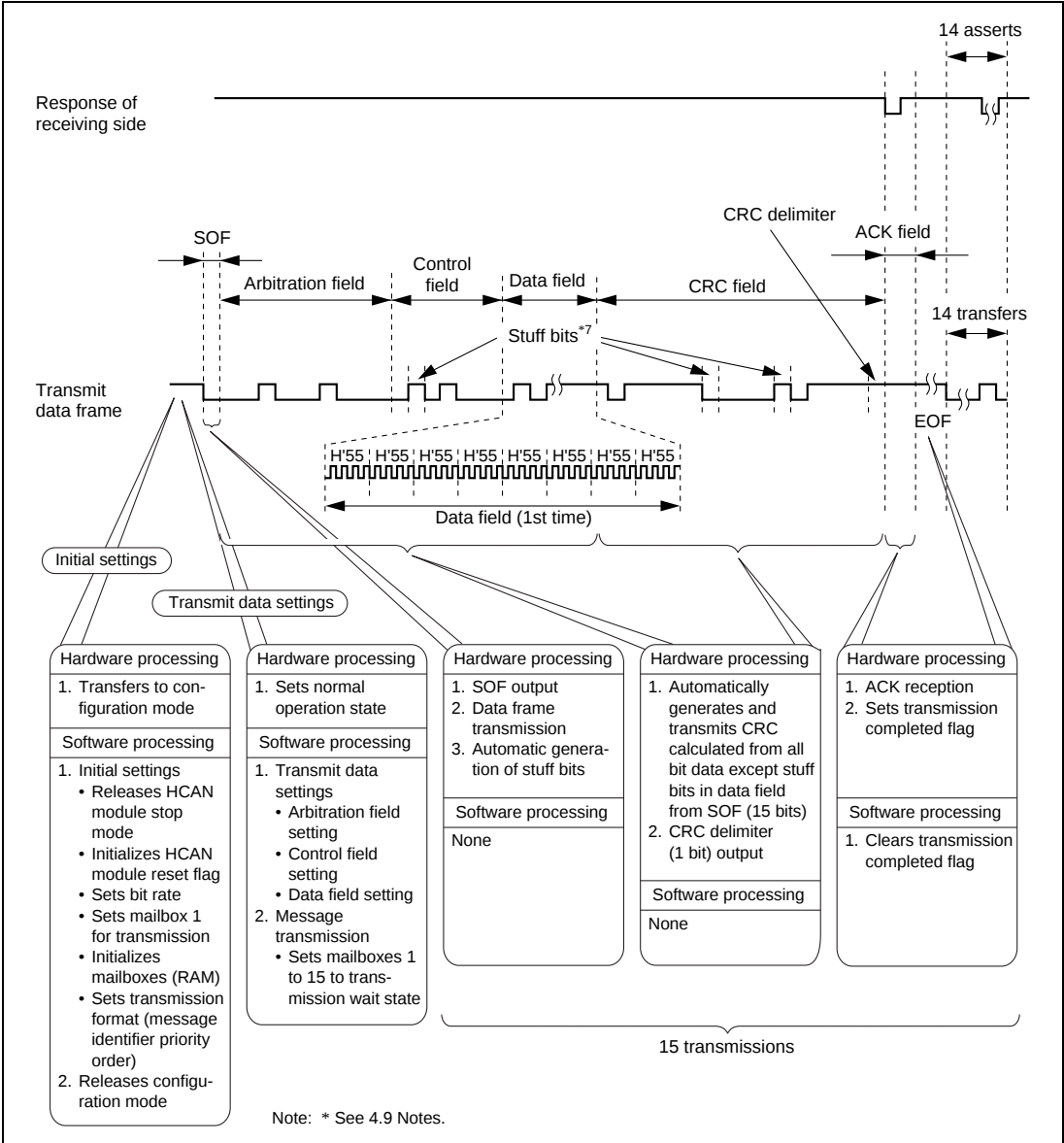


Figure 4.3 Operation During HCAN Transmission

Operation (Reception)

Figure 4.4 shows the principle of operation during reception. HCAN reception is carried out by H8S/2623F-ZTAT hardware processing and software processing as shown in the figure.

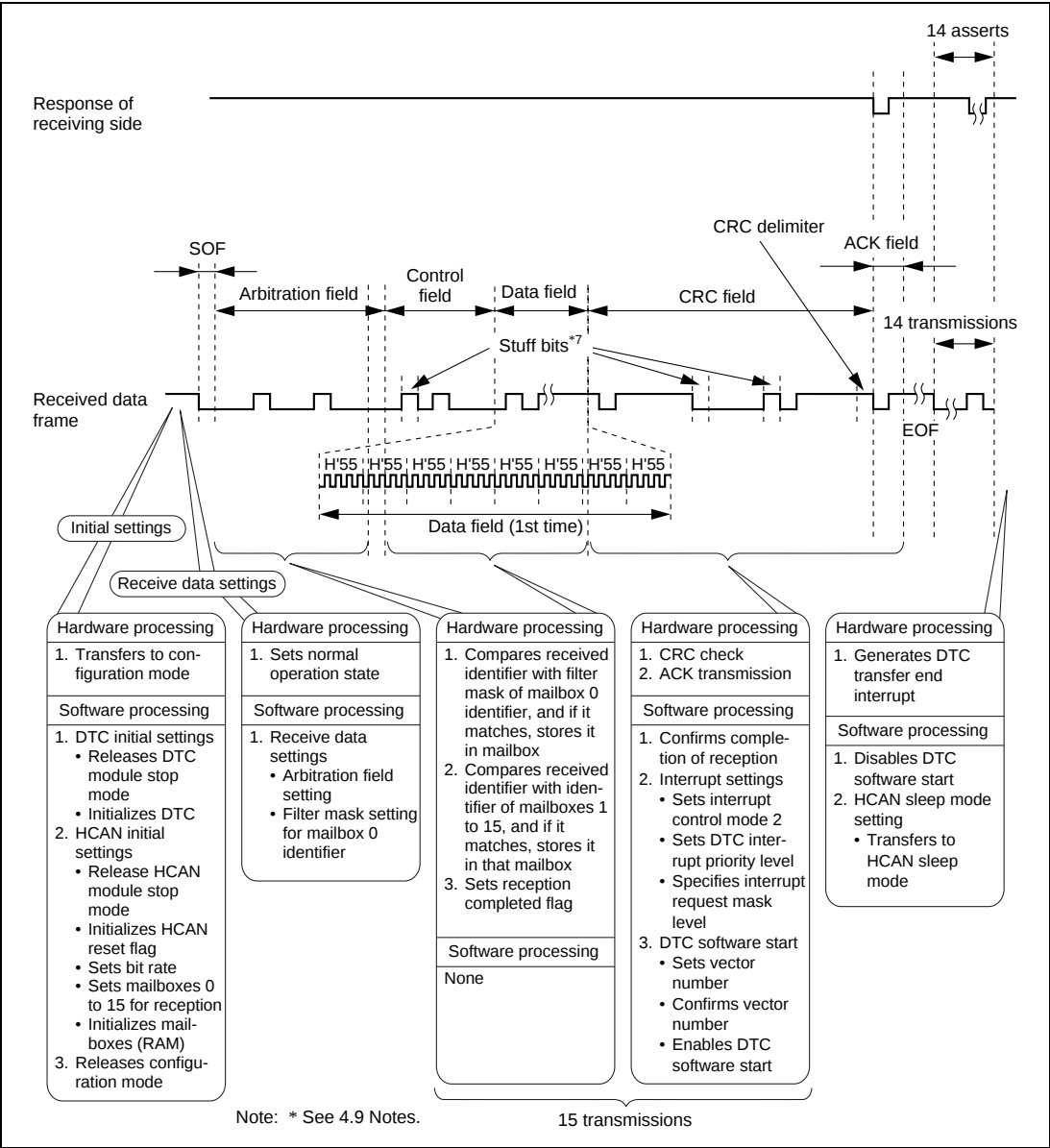


Figure 4.4 Operation During HCAN Reception

4.4 Software

1. Modules

Module Name	Label	Function
Main routine	main	HCAN initial settings and transmission/reception settings.
DTC transfer end interrupt routine	SWDTEND	Disables DTC software startup and transfers to HCAN sleep mode.

2. Variables Used

Label	Function	Data Length	Module
COUNT	Initializes HCAN_MCO_1 to MC15_8, and HCAN_MDO_1 to MD15_8.	Unsigned short	Main routine

3. Internal Registers Used

Register Name	Function	Setting	Module
Settings Common to Transmission/Reception			
MSTPCRC	Releases HCAN module stop mode.	0xF7	Main routine
HCAN_BCR	Sets HCAN bit rate to 1 Mbps.	0x0025	
HCAN_IRR	Initializes HCAN module reset flag.	0x0100	
Settings for Transmission			
HCAN_MBCR	Sets mailboxes 1 to 15 for transmission.	0x0100	Main routine
HCAN_MCR	Sets transmission in order of priority of message identifier, and clears reset request bit.	0x00	
HCAN_MC1 to 15_1	Sets 8-byte data length.	All 0x08	
HCAN_MC1_5	Select mailbox 1 data frame and standard format, and set identifier.	0xC0	
HCAN_MC1_6		0xCC	
HCAN_MC2_5	Select mailbox 2 data frame and standard format, and set identifier.	0x40	
HCAN_MC2_6		0x55	
HCAN_MC3_5	Select mailbox 3 data frame and standard format, and set identifier.	0xE0	
HCAN_MC3_6		0xEE	

Register Name	Function	Setting	Module
Settings for Transmission			
HCAN_MC4_5	Select mailbox 4 data frame and standard format, and set identifier.	0x60	Main routine
HCAN_MC4_6		0x66	
HCAN_MC5_5	Select mailbox 5 data frame and standard format, and set identifier.	0x00	
HCAN_MC5_6		0x11	
HCAN_MC6_5	Select mailbox 6 data frame and standard format, and set identifier.	0x80	
HCAN_MC6_6		0x99	
HCAN_MC7_5	Select mailbox 7 data frame and standard format, and set identifier.	0x20	
HCAN_MC7_6		0x33	
HCAN_MC8_5	Select mailbox 8 data frame and standard format, and set identifier.	0xE0	
HCAN_MC8_6		0xFF	
HCAN_MC9_5	Select mailbox 9 data frame and standard format, and set identifier.	0x20	
HCAN_MC9_6		0x22	
HCAN_MC10_5	Select mailbox 10 data frame and standard format, and set identifier.	0x80	
HCAN_MC10_6		0x88	
HCAN_MC11_5	Select mailbox 11 data frame and standard format, and set identifier.	0xA0	
HCAN_MC11_6		0xAA	
HCAN_MC12_5	Select mailbox 12 data frame and standard format, and set identifier.	0xC0	
HCAN_MC12_6		0xDD	
HCAN_MC13_5	Select mailbox 13 data frame and standard format, and set identifier.	0x60	
HCAN_MC13_6		0x77	
HCAN_MC14_5	Select mailbox 14 data frame and standard format, and set identifier.	0x40	
HCAN_MC14_6		0x44	
HCAN_MC15_5	Select mailbox 15 data frame and standard format, and set identifier.	0xA0	
HCAN_MC15_6		0xBB	
HCAN_MD1_1 to 8	Sets transmit data for 1 st to 8 th bytes of mailbox 1.	All 0x11	
HCAN_MD2_1 to 8	Sets transmit data for 1 st to 8 th bytes of mailbox 2.	All 0x22	
HCAN_MD3_1 to 8	Sets transmit data for 1 st to 8 th bytes of mailbox 3.	All 0x33	
HCAN_MD4_1 to 8	Sets transmit data for 1 st to 8 th bytes of mailbox 4.	All 0x44	
HCAN_MD5_1 to 8	Sets transmit data for 1 st to 8 th bytes of mailbox 5.	All 0x55	

Register Name	Function	Setting	Module
Settings for Transmission			
HCAN_MD6_1 to 8	Sets transmit data for 1 st to 8 th bytes of mailbox 6.	All 0x66	Main routine
HCAN_MD7_1 to 8	Sets transmit data for 1 st to 8 th bytes of mailbox 7.	All 0x77	
HCAN_MD8_1 to 8	Sets transmit data for 1 st to 8 th bytes of mailbox 8.	All 0x88	
HCAN_MD9_1 to 8	Sets transmit data for 1 st to 8 th bytes of mailbox 9.	All 0x99	
HCAN_MD10_1 to 8	Sets transmit data for 1 st to 8 th bytes of mailbox 10.	All 0xAA	
HCAN_MD11_1 to 8	Sets transmit data for 1 st to 8 th bytes of mailbox 11.	All 0xBB	
HCAN_MD12_1 to 8	Sets transmit data for 1 st to 8 th bytes of mailbox 12.	All 0xCC	
HCAN_MD13_1 to 8	Sets transmit data for 1 st to 8 th bytes of mailbox 13.	All 0xDD	
HCAN_MD14_1 to 8	Sets transmit data for 1 st to 8 th bytes of mailbox 14.	All 0xEE	
HCAN_MD15_1 to 8	Sets transmit data for 1 st to 8 th bytes of mailbox 15.	All 0xFF	
HCAN_TXPR	Sets mailboxes 1 to 15 to transmission wait state.	0xFEFF	
HCAN_TXACK	Clears data frame transmission completed flag.	0xFEFF	
Settings for Reception			
MSTPCRA	Releases DTC module stop mode.	0x3F	Main routine
HCANMCR	Clears reset request bit.	0xFE	
	Transfers to HCAN sleep mode.	0x20	
HCAN_MBCR	Sets mailboxes 0 to 15 for reception.	0xFFFF	
HCAN_LAFMH	Sets filter mask for mailbox 0 identifier.	0x0000	
HCAN_RXPR	Clears message reception completed flag.	0xFFFF	
SYSCR	Sets interrupt control mode 2.	0x20	
INTC. IPRC	Sets DTC interrupt priority level to 7.	0x07	
exr	Specifies interrupt request mask level.	0x00	

Register Name	Function	Setting	Module
Settings for Reception			
DTC_DTVECR	Sets vector number.	0x60	Main routine
	Enables DTC software startup.	0x80	
	Disables DTC software startup.	0x7F	End interrupt
HCAN_MC0_5	Select mailbox 0 data frame and standard format, and set identifier.	0xA0	Main routine
HCAN_MC0_6		0xAA	
HCAN_MC1_5	Select mailbox 1 data frame and standard format, and set identifier.	0x00	
HCAN_MC1_6		0x11	
HCAN_MC2_5	Select mailbox 2 data frame and standard format, and set identifier.	0x20	
HCAN_MC2_6		0x22	
HCAN_MC3_5	Select mailbox 3 data frame and standard format, and set identifier.	0x20	
HCAN_MC3_6		0x33	
HCAN_MC4_5	Select mailbox 4 data frame and standard format, and set identifier.	0x40	
HCAN_MC4_6		0x44	
HCAN_MC5_5	Select mailbox 5 data frame and standard format, and set identifier.	0x40	
HCAN_MC5_6		0x55	
HCAN_MC6_5	Select mailbox 6 data frame and standard format, and set identifier.	0x60	
HCAN_MC6_6		0x66	
HCAN_MC7_5	Select mailbox 7 data frame and standard format, and set identifier.	0x60	
HCAN_MC7_6		0x77	
HCAN_MC8_5	Select mailbox 8 data frame and standard format, and set identifier.	0x80	
HCAN_MC8_6		0x88	
HCAN_MC9_5	Select mailbox 9 data frame and standard format, and set identifier.	0x80	
HCAN_MC9_6		0x99	
HCAN_MC10_5	Select mailbox 10 data frame and standard format, and set identifier.	0xA0	
HCAN_MC10_6		0xAA	
HCAN_MC11_5	Select mailbox 11 data frame and standard format, and set identifier.	0xA0	
HCAN_MC11_6		0xBB	
HCAN_MC12_5	Select mailbox 12 data frame and standard format, and set identifier.	0xC0	
HCAN_MC12_6		0xCC	
HCAN_MC13_5	Select mailbox 13 data frame and standard format, and set identifier.	0xC0	
HCAN_MC13_6		0xDD	

Note: End interrupt: DTC transfer end interrupt routine

Register Name	Function	Setting	Module
Settings for Reception			
HCAN_MC14_5	Select mailbox 14 data frame and standard format, and set identifier.	0xE0	Main routine
HCAN_MC14_6		0xEE	
HCAN_MC15_5	Select mailbox 15 data frame and standard format, and set identifier.	0xE0	
HCAN_MC15_6		0xFF	

4. RAM Used

Symbol	Function	Address	Module
Settings for Reception			
MAILBOX0. Message_DATA1 to 8	Store data for HCAN_MD0_1 to 8.	0xFFC100 to 7	Main routine
MAILBOX1. Message_DATA1 to 8	Store data for HCAN_MD1_1 to 8.	0xFFC108 to F	
MAILBOX2. Message_DATA1 to 8	Store data for HCAN_MD2_1 to 8.	0xFFC110 to 7	
MAILBOX3. Message_DATA1 to 8	Store data for HCAN_MD3_1 to 8.	0xFFC118 to F	
MAILBOX4. Message_DATA1 to 8	Store data for HCAN_MD4_1 to 8.	0xFFC120 to 7	
MAILBOX5. Message_DATA1 to 8	Store data for HCAN_MD5_1 to 8.	0xFFC128 to F	
MAILBOX6. Message_DATA1 to 8	Store data for HCAN_MD6_1 to 8.	0xFFC130 to 7	
MAILBOX7. Message_DATA1 to 8	Store data for HCAN_MD7_1 to 8.	0xFFC138 to F	
MAILBOX8. Message_DATA1 to 8	Store data for HCAN_MD8_1 to 8.	0xFFC140 to 7	
MAILBOX9. Message_DATA1 to 8	Store data for HCAN_MD9_1 to 8.	0xFFC148 to F	
MAILBOX10. Message_DATA1 to 8	Store data for HCAN_MD10_1 to 8.	0xFFC150 to 7	
MAILBOX11. Message_DATA1 to 8	Store data for HCAN_MD11_1 to 8.	0xFFC158 to F	
MAILBOX12. Message_DATA1 to 8	Store data for HCAN_MD12_1 to 8.	0xFFC160 to 7	

Symbol Name	Function	Address	Module
Settings for Reception			
MAILBOX13. Message_DATA1 to 8	Store data for HCAN_MD13_1 to 8.	0xFFC168 to F	Main routine
MAILBOX14. Message_DATA1 to 8	Store data for HCAN_MD14_1 to 8.	0xFFC170 to 7	
MAILBOX15. Message_DATA1 to 8	Store data for HCAN_MD15_1 to 8.	0xFFC178 to F	
SAR	Sets source address (where receive messages are stored from).	0xFFEC01 to 3 (0xFFF8B0)	
MRA	Sets block transfer mode, byte size transfer, etc.	0xFFEC00 (0xA8)	
DAR	Sets destination address (where receive messages are stored to).	0xFFEC05 to 7 (0xFFC100)	
MRB	Enables interrupt to CPU after DTC data transfer ends.	0xFFEC04 (0x00)	
CRA	Sets transfer block size retained value for DTC data transfer (upper byte) and transfer block size counter (lower byte).	0xFFEC08, 9 (0x8080)	
CRB	Sets number of DTC block transfers.	0xFFEC0A, B (0x0001)	

Figures in parentheses are settings.

4.5 **Transmission Flowchart**

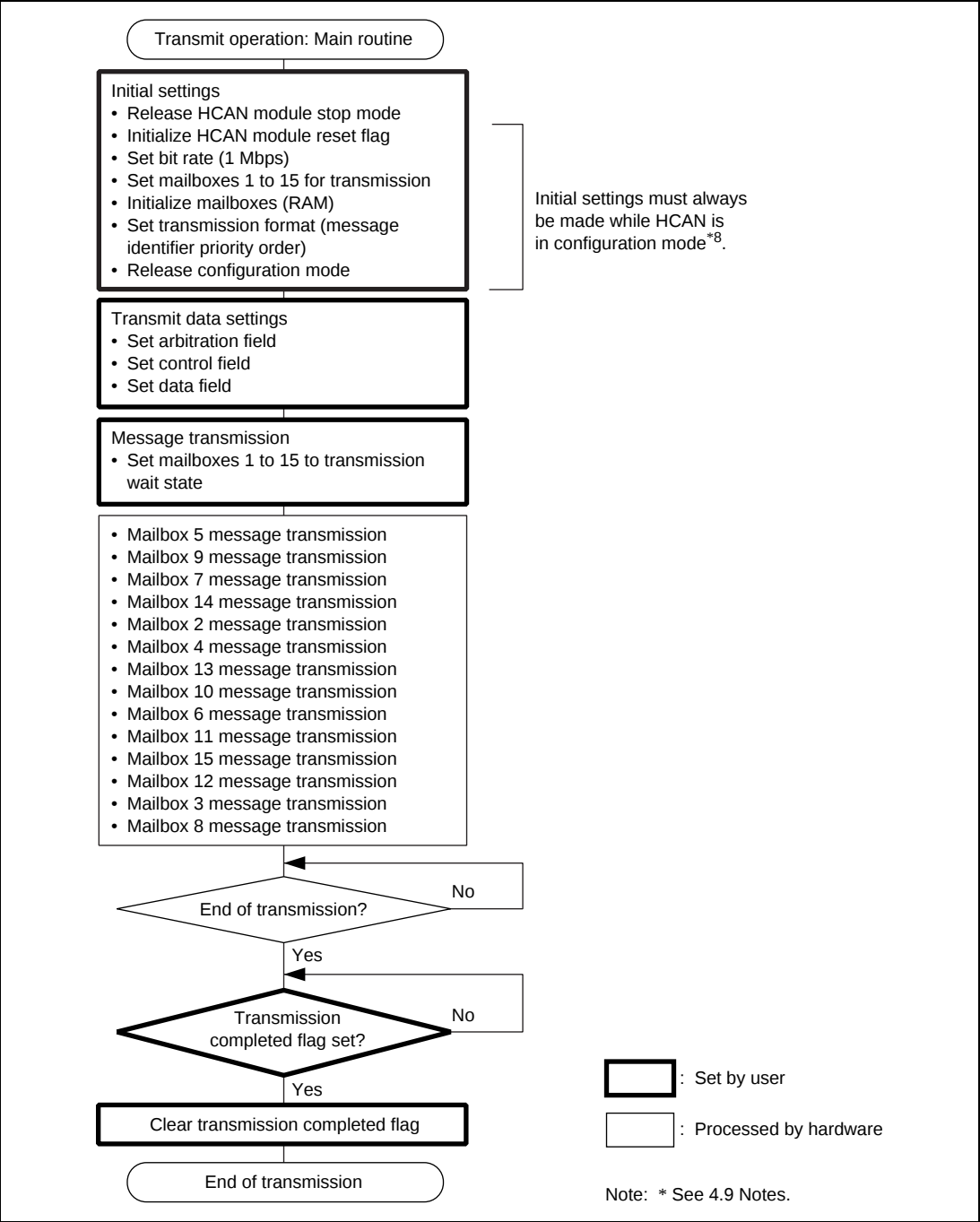


Figure 4.5 Transmission Flowchart

4.6 Transmission Program List

```

/*****
/*
          HCAN Transmission Program
*/
*****/
#include <stdio.h>          /* Library function header file */
#include <machine.h>        /* Library function header file */
#include "2623.h"           /* Peripheral register definition header file */
/*****
/*
          Function Protocol Declaration
*/
*****/
void main( void );
/*****
/*
          Definition of Constants
*/
*****/
#define COUNT (*(unsigned short *)0xFFC000)
/*****
/*
          Main Routine
*/
*****/
void main(void)
{
/* Initial Settings */
    MSTPCRC = 0xF7;          /* Release HCAN module stop mode */
    HCAN_IRR = 0x0100;       /* Initialize HCAN module reset flag */
    HCAN_BCR = 0x0025;       /* Bit rate: 1 Mbps */
    HCAN_MBCR = 0x0100;      /* Set mailboxes 1 to 15 for transmission */
    for( COUNT = 0; COUNT < 128; COUNT++ )
        /* Initialize mailboxes (RAM) */
        {
            *(char*)&HCAN_MC0_1 + COUNT) = 0x00;
        }
    for( COUNT = 0; COUNT < 128; COUNT++ )
        /* Initialize mailboxes (RAM) */
        {
            *(char*)&HCAN_MD0_1 + COUNT) = 0x00;
        }
    HCAN_MCR = 0x00; /* Set transmission in message identifier priority order */
/* Transmit data settings */
/***** Mail Box 1 *****/
    HCAN_MC1_5 = 0xC0;
        /* Set standard format, data frame, and identifier */

```

```

HCAN_MC1_6 = 0xCC;          /* Set identifier          */
HCAN_MC1_1 = 0x08;          /* Data length: 8 bytes    */
HCAN_MD1_1 = 0x11;          /* Message contents: 00010001 */
HCAN_MD1_2 = 0x11;          /* Message contents: 00010001 */
HCAN_MD1_3 = 0x11;          /* Message contents: 00010001 */
HCAN_MD1_4 = 0x11;          /* Message contents: 00010001 */
HCAN_MD1_5 = 0x11;          /* Message contents: 00010001 */
HCAN_MD1_6 = 0x11;          /* Message contents: 00010001 */
HCAN_MD1_7 = 0x11;          /* Message contents: 00010001 */
HCAN_MD1_8 = 0x11;          /* Message contents: 00010001 */

/***** Mail Box 2 *****/

HCAN_MC2_5 = 0x40;          /* Set standard format, data frame, and identifier */
HCAN_MC2_6 = 0x55;          /* Set identifier          */
HCAN_MC2_1 = 0x08;          /* Data length: 8 bytes    */
HCAN_MD2_1 = 0x22;          /* Message contents: 00100010 */
HCAN_MD2_2 = 0x22;          /* Message contents: 00100010 */
HCAN_MD2_3 = 0x22;          /* Message contents: 00100010 */
HCAN_MD2_4 = 0x22;          /* Message contents: 00100010 */
HCAN_MD2_5 = 0x22;          /* Message contents: 00100010 */
HCAN_MD2_6 = 0x22;          /* Message contents: 00100010 */
HCAN_MD2_7 = 0x22;          /* Message contents: 00100010 */
HCAN_MD2_8 = 0x22;          /* Message contents: 00100010 */

/***** Mail Box 3 *****/

HCAN_MC3_5 = 0xE0;          /* Set standard format, data frame, and identifier */
HCAN_MC3_6 = 0xEE;          /* Set identifier          */
HCAN_MC3_1 = 0x08;          /* Data length: 8 bytes    */
HCAN_MD3_1 = 0x33;          /* Message contents: 00110011 */
HCAN_MD3_2 = 0x33;          /* Message contents: 00110011 */
HCAN_MD3_3 = 0x33;          /* Message contents: 00110011 */
HCAN_MD3_4 = 0x33;          /* Message contents: 00110011 */
HCAN_MD3_5 = 0x33;          /* Message contents: 00110011 */
HCAN_MD3_6 = 0x33;          /* Message contents: 00110011 */
HCAN_MD3_7 = 0x33;          /* Message contents: 00110011 */
HCAN_MD3_8 = 0x33;          /* Message contents: 00110011 */

/***** Mail Box 4 *****/

HCAN_MC4_5 = 0x60;          /* Set standard format, data frame, and identifier */
HCAN_MC4_6 = 0x66;          /* Set identifier          */
HCAN_MC4_1 = 0x08;          /* Data length: 8 bytes    */
HCAN_MD4_1 = 0x44;          /* Message contents: 01000100 */
HCAN_MD4_2 = 0x44;          /* Message contents: 01000100 */

```

```

HCAN_MD4_3 = 0x44;          /* Message contents: 01000100 */
HCAN_MD4_4 = 0x44;          /* Message contents: 01000100 */
HCAN_MD4_5 = 0x44;          /* Message contents: 01000100 */
HCAN_MD4_6 = 0x44;          /* Message contents: 01000100 */
HCAN_MD4_7 = 0x44;          /* Message contents: 01000100 */
HCAN_MD4_8 = 0x44;          /* Message contents: 01000100 */
/***** Mail Box 5 *****/
HCAN_MC5_5 = 0x00;          /* Set standard format, data frame, and identifier */
HCAN_MC5_6 = 0x11;          /* Set identifier */
HCAN_MC5_1 = 0x08;          /* Data length: 8 bytes */
HCAN_MD5_1 = 0x55;          /* Message contents: 01010101 */
HCAN_MD5_2 = 0x55;          /* Message contents: 01010101 */
HCAN_MD5_3 = 0x55;          /* Message contents: 01010101 */
HCAN_MD5_4 = 0x55;          /* Message contents: 01010101 */
HCAN_MD5_5 = 0x55;          /* Message contents: 01010101 */
HCAN_MD5_6 = 0x55;          /* Message contents: 01010101 */
HCAN_MD5_7 = 0x55;          /* Message contents: 01010101 */
HCAN_MD5_8 = 0x55;          /* Message contents: 01010101 */
/***** Mail Box 6 *****/
HCAN_MC6_5 = 0x80;          /* Set standard format, data frame, and identifier */
HCAN_MC6_6 = 0x99;          /* Set identifier */
HCAN_MC6_1 = 0x08;          /* Data length: 8 bytes */
HCAN_MD6_1 = 0x66;          /* Message contents: 01100110 */
HCAN_MD6_2 = 0x66;          /* Message contents: 01100110 */
HCAN_MD6_3 = 0x66;          /* Message contents: 01100110 */
HCAN_MD6_4 = 0x66;          /* Message contents: 01100110 */
HCAN_MD6_5 = 0x66;          /* Message contents: 01100110 */
HCAN_MD6_6 = 0x66;          /* Message contents: 01100110 */
HCAN_MD6_7 = 0x66;          /* Message contents: 01100110 */
HCAN_MD6_8 = 0x66;          /* Message contents: 01100110 */
/***** Mail Box 7 *****/
HCAN_MC7_5 = 0x20;          /* Set standard format, data frame, and identifier */
HCAN_MC7_6 = 0x33;          /* Set identifier */
HCAN_MC7_1 = 0x08;          /* Data length: 8 bytes */
HCAN_MD7_1 = 0x77;          /* Message contents: 01110111 */
HCAN_MD7_2 = 0x77;          /* Message contents: 01110111 */
HCAN_MD7_3 = 0x77;          /* Message contents: 01110111 */
HCAN_MD7_4 = 0x77;          /* Message contents: 01110111 */
HCAN_MD7_5 = 0x77;          /* Message contents: 01110111 */
HCAN_MD7_6 = 0x77;          /* Message contents: 01110111 */

```

```

HCAN_MD7_7 = 0x77;          /* Message contents: 01110111 */
HCAN_MD7_8 = 0x77;          /* Message contents: 01110111 */
/***** Mail Box 8 *****/
HCAN_MC8_5 = 0xE0;          /* Set standard format, data frame, and identifier */
HCAN_MC8_6 = 0xFF;          /* Set identifier */
HCAN_MC8_1 = 0x08;          /* Data length: 8 bytes */
HCAN_MD8_1 = 0x88;          /* Message contents: 10001000 */
HCAN_MD8_2 = 0x88;          /* Message contents: 10001000 */
HCAN_MD8_3 = 0x88;          /* Message contents: 10001000 */
HCAN_MD8_4 = 0x88;          /* Message contents: 10001000 */
HCAN_MD8_5 = 0x88;          /* Message contents: 10001000 */
HCAN_MD8_6 = 0x88;          /* Message contents: 10001000 */
HCAN_MD8_7 = 0x88;          /* Message contents: 10001000 */
HCAN_MD8_8 = 0x88;          /* Message contents: 10001000 */
/***** Mail Box 9 *****/
HCAN_MC9_5 = 0x20;          /* Set standard format, data frame, and identifier */
HCAN_MC9_6 = 0x22;          /* Set identifier */
HCAN_MC9_1 = 0x08;          /* Data length: 8 bytes */
HCAN_MD9_1 = 0x99;          /* Message contents: 10011001 */
HCAN_MD9_2 = 0x99;          /* Message contents: 10011001 */
HCAN_MD9_3 = 0x99;          /* Message contents: 10011001 */
HCAN_MD9_4 = 0x99;          /* Message contents: 10011001 */
HCAN_MD9_5 = 0x99;          /* Message contents: 10011001 */
HCAN_MD9_6 = 0x99;          /* Message contents: 10011001 */
HCAN_MD9_7 = 0x99;          /* Message contents: 10011001 */
HCAN_MD9_8 = 0x99;          /* Message contents: 10011001 */
/***** Mail Box 10 *****/
HCAN_MC10_5 = 0x80;         /* Set standard format, data frame, and identifier */
HCAN_MC10_6 = 0x88;         /* Set identifier */
HCAN_MC10_1 = 0x08;         /* Data length: 8 bytes */
HCAN_MD10_1 = 0xAA;         /* Message contents: 10101010 */
HCAN_MD10_2 = 0xAA;         /* Message contents: 10101010 */
HCAN_MD10_3 = 0xAA;         /* Message contents: 10101010 */
HCAN_MD10_4 = 0xAA;         /* Message contents: 10101010 */
HCAN_MD10_5 = 0xAA;         /* Message contents: 10101010 */
HCAN_MD10_6 = 0xAA;         /* Message contents: 10101010 */
HCAN_MD10_7 = 0xAA;         /* Message contents: 10101010 */
HCAN_MD10_8 = 0xAA;         /* Message contents: 10101010 */
/***** Mail Box 11 *****/
HCAN_MC11_5 = 0xA0;         /* Set standard format, data frame, and identifier */

```

```

HCAN_MC11_6 = 0xAA;          /* Set identifier */
HCAN_MC11_1 = 0x08;          /* Data length: 8 bytes */
HCAN_MD11_1 = 0xBB;          /* Message contents: 10111011 */
HCAN_MD11_2 = 0xBB;          /* Message contents: 10111011 */
HCAN_MD11_3 = 0xBB;          /* Message contents: 10111011 */
HCAN_MD11_4 = 0xBB;          /* Message contents: 10111011 */
HCAN_MD11_5 = 0xBB;          /* Message contents: 10111011 */
HCAN_MD11_6 = 0xBB;          /* Message contents: 10111011 */
HCAN_MD11_7 = 0xBB;          /* Message contents: 10111011 */
HCAN_MD11_8 = 0xBB;          /* Message contents: 10111011 */

/***** Mail Box 12 *****/
HCAN_MC12_5 = 0xC0;          /* Set standard format, data frame, and identifier */
HCAN_MC12_6 = 0xDD;          /* Set identifier */
HCAN_MC12_1 = 0x08;          /* Data length: 8 bytes */
HCAN_MD12_1 = 0xCC;          /* Message contents: 11001100 */
HCAN_MD12_2 = 0xCC;          /* Message contents: 11001100 */
HCAN_MD12_3 = 0xCC;          /* Message contents: 11001100 */
HCAN_MD12_4 = 0xCC;          /* Message contents: 11001100 */
HCAN_MD12_5 = 0xCC;          /* Message contents: 11001100 */
HCAN_MD12_6 = 0xCC;          /* Message contents: 11001100 */
HCAN_MD12_7 = 0xCC;          /* Message contents: 11001100 */
HCAN_MD12_8 = 0xCC;          /* Message contents: 11001100 */

/***** Mail Box 13 *****/
HCAN_MC13_5 = 0x60;          /* Set standard format, data frame, and identifier */
HCAN_MC13_6 = 0x77;          /* Set identifier */
HCAN_MC13_1 = 0x08;          /* Data length: 8 bytes */
HCAN_MD13_1 = 0xDD;          /* Message contents: 11011101 */
HCAN_MD13_2 = 0xDD;          /* Message contents: 11011101 */
HCAN_MD13_3 = 0xDD;          /* Message contents: 11011101 */
HCAN_MD13_4 = 0xDD;          /* Message contents: 11011101 */
HCAN_MD13_5 = 0xDD;          /* Message contents: 11011101 */
HCAN_MD13_6 = 0xDD;          /* Message contents: 11011101 */
HCAN_MD13_7 = 0xDD;          /* Message contents: 11011101 */
HCAN_MD13_8 = 0xDD;          /* Message contents: 11011101 */

/***** Mail Box 14 *****/
HCAN_MC14_5 = 0x40;          /* Set standard format, data frame, and identifier */
HCAN_MC14_6 = 0x44;          /* Set identifier */
HCAN_MC14_1 = 0x08;          /* Data length: 8 bytes */
HCAN_MD14_1 = 0xEE;          /* Message contents: 11101110 */
HCAN_MD14_2 = 0xEE;          /* Message contents: 11101110 */

```

```

HCAN_MD14_3 = 0xEE;          /* Message contents: 11101110 */
HCAN_MD14_4 = 0xEE;          /* Message contents: 11101110 */
HCAN_MD14_5 = 0xEE;          /* Message contents: 11101110 */
HCAN_MD14_6 = 0xEE;          /* Message contents: 11101110 */
HCAN_MD14_7 = 0xEE;          /* Message contents: 11101110 */
HCAN_MD14_8 = 0xEE;          /* Message contents: 11101110 */

/***** Mail Box 15 *****/

HCAN_MC15_5 = 0xA0;          /* Set standard format, data frame, and identifier */
HCAN_MC15_6 = 0xBB;          /* Set identifier */
HCAN_MC15_1 = 0x08;          /* Data length: 8 bytes */
HCAN_MD15_1 = 0xFF;          /* Message contents: 11111111 */
HCAN_MD15_2 = 0xFF;          /* Message contents: 11111111 */
HCAN_MD15_3 = 0xFF;          /* Message contents: 11111111 */
HCAN_MD15_4 = 0xFF;          /* Message contents: 11111111 */
HCAN_MD15_5 = 0xFF;          /* Message contents: 11111111 */
HCAN_MD15_6 = 0xFF;          /* Message contents: 11111111 */
HCAN_MD15_7 = 0xFF;          /* Message contents: 11111111 */
HCAN_MD15_8 = 0xFF;          /* Message contents: 11111111 */

/* Message transmission */
HCAN_TXPR = 0xFEFF;          /* Set mailboxes 1 to 15 to transmission wait state */
while((HCAN_TXACK & 0xFEFF) != 0xFEFF);
                                /* Wait for completion of transmission */

/* Clear transmission completed flag */
HCAN_TXACK &= 0xFEFF;          /* Clear transmission completed flag */
while(1);

}

```

4.7 Reception Flowcharts

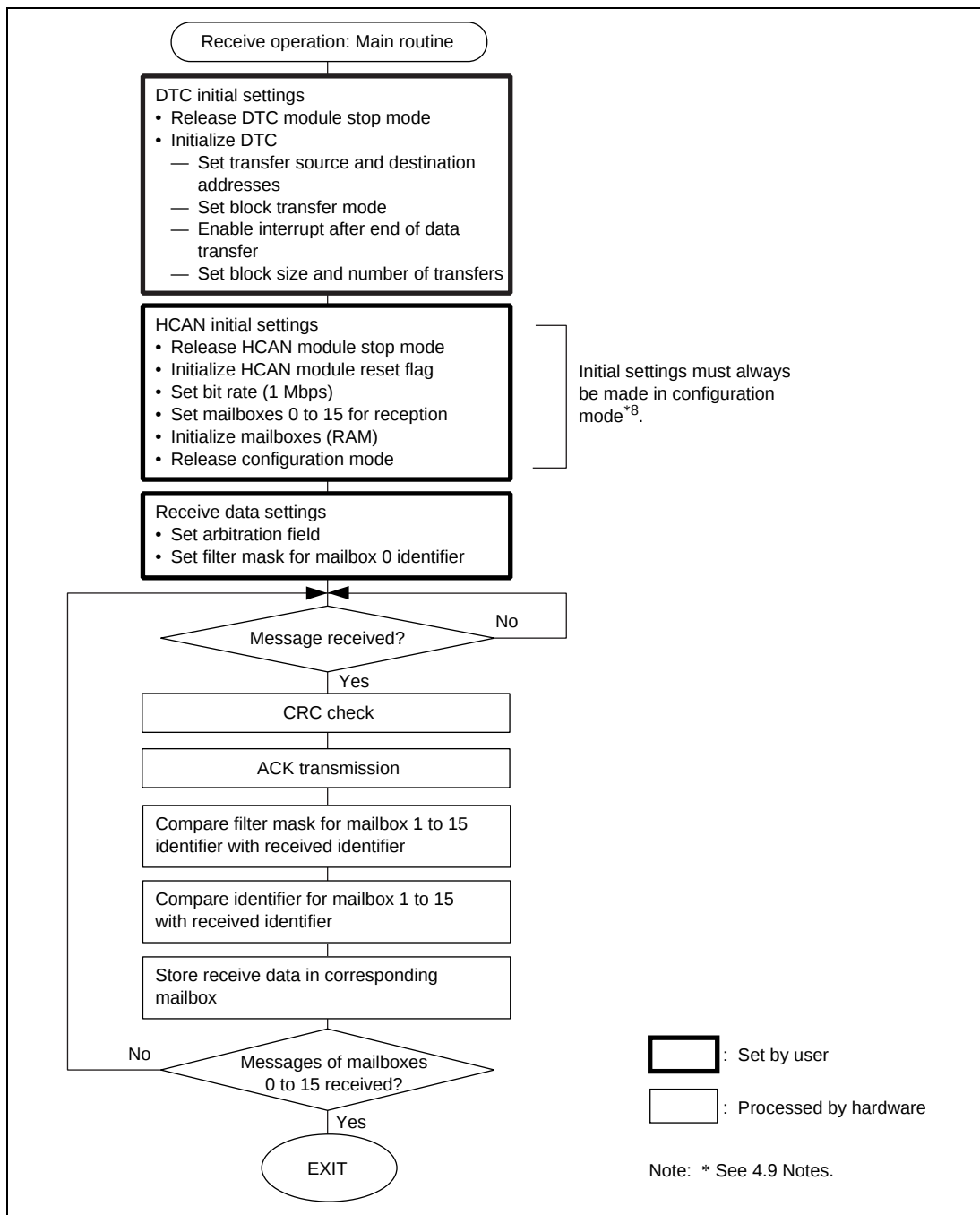


Figure 4.6 Reception Flowchart (1)

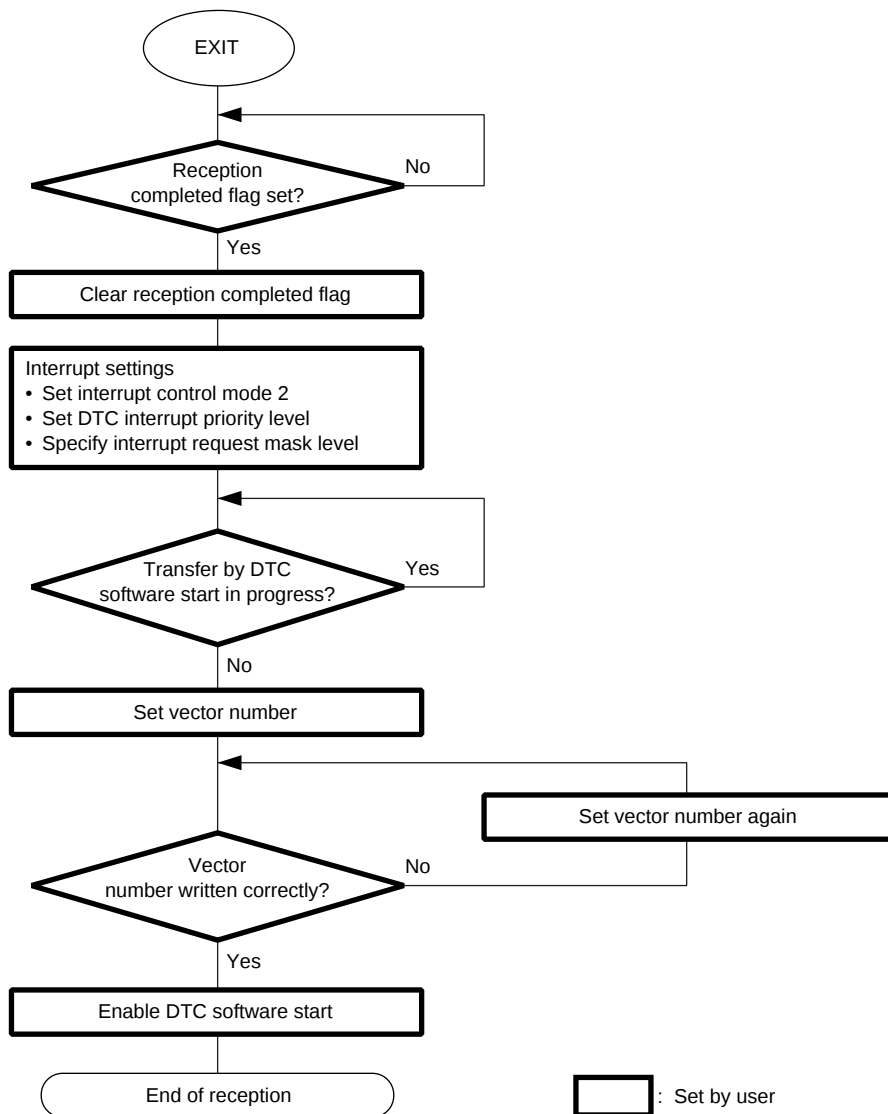


Figure 4.6 Reception Flowchart (2)

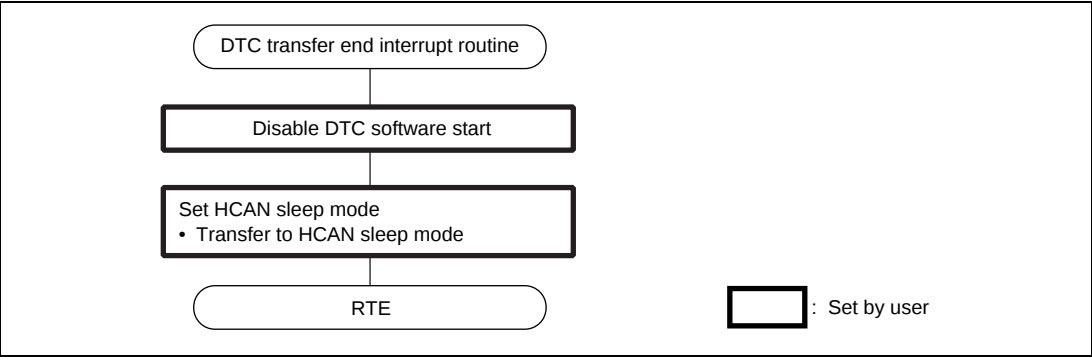


Figure 4.7 DTC Transfer End Interrupt Routine Flowchart

4.8 Reception Program List

```

/*****
/*
                HCAN Reception Program
                */
/*****
#include <stdio.h>                /* Library function header file
#include <machine.h>              /* Library function header file
#include "2623.h"                 /* Peripheral register definition header file
/*****
/*
                Function Protocol Declaration
                */
/*****
void main( void );
/*****
/*
                Definition of Constants
                */
/*****
#define COUNT    (*(unsigned short *)0xFFC000)
volatile struct MB                                /* struct MAILBOX0-15
{
    unsigned char Message_DATA1;                  /* Store receive data 1
    unsigned char Message_DATA2;                  /* Store receive data 2
    unsigned char Message_DATA3;                  /* Store receive data 3
    unsigned char Message_DATA4;                  /* Store receive data 4
    unsigned char Message_DATA5;                  /* Store receive data 5
    unsigned char Message_DATA6;                  /* Store receive data 6
    unsigned char Message_DATA7;                  /* Store receive data 7
    unsigned char Message_DATA8;                  /* Store receive data 8
};
#define MAILBOX0 (*(volatile struct MB *)0xFFC100) /* Mailbox 0 address
#define MAILBOX1 (*(volatile struct MB *)0xFFC108) /* Mailbox 1 address
#define MAILBOX2 (*(volatile struct MB *)0xFFC110) /* Mailbox 2 address
#define MAILBOX3 (*(volatile struct MB *)0xFFC118) /* Mailbox 3 address
#define MAILBOX4 (*(volatile struct MB *)0xFFC120) /* Mailbox 4 address
#define MAILBOX5 (*(volatile struct MB *)0xFFC128) /* Mailbox 5 address
#define MAILBOX6 (*(volatile struct MB *)0xFFC130) /* Mailbox 6 address
#define MAILBOX7 (*(volatile struct MB *)0xFFC138) /* Mailbox 7 address
#define MAILBOX8 (*(volatile struct MB *)0xFFC140) /* Mailbox 8 address
#define MAILBOX9 (*(volatile struct MB *)0xFFC148) /* Mailbox 9 address
#define MAILBOX10 (*(volatile struct MB *)0xFFC150) /* Mailbox 10 address
#define MAILBOX11 (*(volatile struct MB *)0xFFC158) /* Mailbox 11 address
#define MAILBOX12 (*(volatile struct MB *)0xFFC160) /* Mailbox 12 address

```

```

#define MAILBOX13 (*(volatile struct MB *)0xFFC168)    /* Mailbox 13 address */
#define MAILBOX14 (*(volatile struct MB *)0xFFC170)    /* Mailbox 14 address */
#define MAILBOX15 (*(volatile struct MB *)0xFFC178)    /* Mailbox 15 address */
#define SAR        (*(volatile unsigned long *)0xFFEC00) /* Set register information */
#define MRA        (*(volatile unsigned char *)0xFFEC00) /* Set register information */
#define DAR        (*(volatile unsigned long *)0xFFEC04) /* Set register information */
#define MRB        (*(volatile unsigned char *)0xFFEC04) /* Set register information */
#define CRA        (*(volatile unsigned short *)0xFFEC08) /* Set register information */
#define CRB        (*(volatile unsigned short *)0xFFEC0A) /* Set register information */

/*****
/*
Main Routine
*****/

void main(void)
{
/* DTC initial settings */
    MSTPCRA = 0x3F; /* Release DTC module stop mode */
    SAR = (long>(&HCAN_MD0_1); /* Set transfer source address */
    MRA = 0xA8;
        /* Set SAR and DAR set incrementing after transfer, and block transfer
        mode */
    DAR = (long>(&MAILBOX0.Message_DATA1);
        /* Set transfer destination address (on-chip RAM) */
    MRB = 0x00; /* Enable interrupt after end of data transfer by DTC */
    CRA = 0x8080; /* Set block transfer to 128-byte units */
    CRB = 0x0001; /* Set number of block transfers to 1 */
/* HCAN initial settings */
    MSTPCRC = 0xF7; /* Release HCAN module stop mode */
    HCAN_IRR = 0x0100; /* Initialize HCAN module reset flag */
    HCAN_BCR = 0x0025; /* Bit rate: 1 Mbps */
    HCAN_MBCR = 0xFFFF; /* Set mailboxes 0 to 15 for reception */
    for( COUNT = 0; COUNT < 128; COUNT++ )
        /* Initialize mailboxes (RAM) */

        {
            *(char*)&HCAN_MC0_1 + COUNT = 0x00;
        }
    for( COUNT = 0; COUNT < 128; COUNT++ )
        /* Initialize mailboxes (RAM) */

```

```

    {
        *(char*)&HCAN_MD0_1 + COUNT) = 0x00;
    }

    HCAN_MCR &= 0xFE; /* Release configuration mode */
/* Receive data settings*/
/* Mail Box 0 */
    HCAN_MC0_5 = 0xA0; /* Set standard format, data frame, and identifier */
    HCAN_MC0_6 = 0xAA; /* Set identifier */
/* Mail Box 1 */
    HCAN_MC1_5 = 0x00; /* Set standard format, data frame, and identifier */
    HCAN_MC1_6 = 0x11; /* Set identifier */
/* Mail Box 2 */
    HCAN_MC2_5 = 0x20; /* Set standard format, data frame, and identifier */
    HCAN_MC2_6 = 0x22; /* Set identifier */
/* Mail Box 3 */
    HCAN_MC3_5 = 0x20; /* Set standard format, data frame, and identifier */
    HCAN_MC3_6 = 0x33; /* Set identifier */
/* Mail Box 4 */
    HCAN_MC4_5 = 0x40; /* Set standard format, data frame, and identifier */
    HCAN_MC4_6 = 0x44; /* Set identifier */
/* Mail Box 5 */
    HCAN_MC5_5 = 0x40; /* Set standard format, data frame, and identifier */
    HCAN_MC5_6 = 0x55; /* Set identifier */
/* Mail Box 6 */
    HCAN_MC6_5 = 0x60; /* Set standard format, data frame, and identifier */
    HCAN_MC6_6 = 0x66; /* Set identifier */
/* Mail Box 7 */
    HCAN_MC7_5 = 0x60; /* Set standard format, data frame, and identifier */
    HCAN_MC7_6 = 0x77; /* Set identifier */
/* Mail Box 8 */
    HCAN_MC8_5 = 0x80; /* Set standard format, data frame, and identifier */
    HCAN_MC8_6 = 0x88; /* Set identifier */
/* Mail Box 9 */
    HCAN_MC9_5 = 0x80; /* Set standard format, data frame, and identifier */
    HCAN_MC9_6 = 0x99; /* Set identifier */
/* Mail Box 10 */
    HCAN_MC10_5 = 0xA0; /* Set standard format, data frame, and identifier */
    HCAN_MC10_6 = 0xAA; /* Set identifier */
/* Mail Box 11 */
    HCAN_MC11_5 = 0xA0; /* Set standard format, data frame, and identifier */

```

```

        HCAN_MC11_6 = 0xBB;          /* Set identifier */
/* Mail Box 12 */
        HCAN_MC12_5 = 0xC0;          /* Set standard format, data frame, and identifier */
        HCAN_MC12_6 = 0xCC;          /* Set identifier */
/* Mail Box 13 */
        HCAN_MC13_5 = 0xC0;          /* Set standard format, data frame, and identifier */
        HCAN_MC13_6 = 0xDD;          /* Set identifier */
/* Mail Box 14 */
        HCAN_MC14_5 = 0xE0;          /* Set standard format, data frame, and identifier */
        HCAN_MC14_6 = 0xEE;          /* Set identifier */
/* Mail Box 15 */
        HCAN_MC15_5 = 0xE0;          /* Set standard format, data frame, and identifier */
        HCAN_MC15_6 = 0xFF;          /* Set identifier */

        HCAN_LAFMH = 0x0000;          /* Mailbox 0 stores data in case of bit match */

/* Wait for completion of reception */
    while((HCAN_RXPR & 0xFFFF) != 0xFFFF); /* Wait for completion of reception */
    HCAN_RXPR &= 0xFFFF;                  /* Clear reception completed flag */
/* Interrupt settings */
    SYSCR |= 0x20;                        /* Set interrupt control mode 2 */
    INTC.IPRC = 0x07;                    /* Set DTC interrupt priority level to 7 */
    set_imask_exr(0);                    /* Specify interrupt request mask level */
/* DTC software start */
    while(DTC_DTVECR & 0x80);
        /* Confirm whether transfer by DTC software start is in progress */
    DTC_DTVECR |= 0x60;                    /* Set vector number (H'4C0) */
    while((DTC_DTVECR & 0x7F) != 0x60){
        /* Confirm vector number written to DTVECR */
    DTC_DTVECR |= 0x60;                    /* Set vector number (H'4C0) again */
    }
    DTC_DTVECR |= 0x80;                    /* Enable DTC software start */
    while(1);
}
/*****
/*
        DTC Transfer End Interrupt Routine
*****/
#pragma interrupt(SWDTEND)
void SWDTEND(void)
{
    DTC_DTVECR &= 0x7F;                    /* Disable DTC software start */

```

```
/* HCAN sleep mode setting */  
    HCAN_MCR |= 0x20;                /* Transfer to HCAN sleep mode */  
}
```

4.9 Notes

1. **Data frame:** Data to be transferred from the transmission source to the transmission destination.
2. **Arbitration field:** Set unique ID for message and data frame or remote frame.
3. **Control field:** Set the data length to be transmitted, and standard format or extended format.
4. **Data field:** Set message contents (data to be transmitted).
5. **CRC field:** A CRC is generated automatically in the HCAN from all bit data except stuff bits in the data field from SOF, and is used to detect transmit message errors.
The CRC field comprises a 15-bit CRC and a 1-bit delimiter.
The CRC delimiter is always output as a 1 after the CRC.

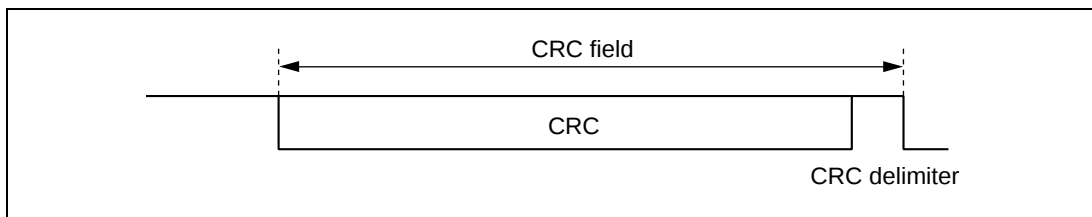


Figure 4.8 CRC Field

About the CRC:

Transmit data polynomial $P(X)$ is multiplied by X^R , then $X^R \cdot P(X)$ is divided by generating polynomial $G(X)$ to give a remainder, $R(X)$. On the side transmitting information, $R(X)$ found from $X^R \cdot P(X)$ is added as check bits, and the result is sent as transmit data $Tx(X)$.

On the side receiving the information, receive data $Rx(X)$ is divided by generating polynomial $G(X)$ to give a remainder. If this remainder is zero, information transmission is regarded as having been completed normally. If the remainder is nonzero, an error is judged to have occurred in the information transmitted.

$$P(X) = \{\text{SOF through data field, excluding stuff bits}\}$$

$$G(X) = X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$$

$$R = 15$$

Note: $G(X)$ is stipulated in the CAN protocol as the polynomial that generates the CRC.

As an example, the procedure is described below for a data frame transmitted using the following settings.

Settings

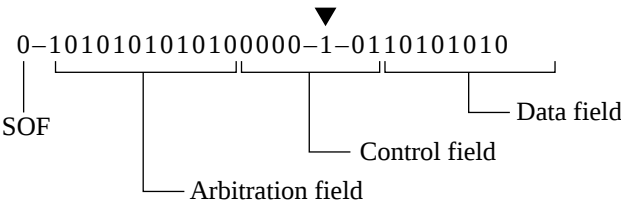
SOF: 0

Arbitration field: 101010101010

Control field: 000001

Data field: 10101010

The bit pattern from SOF through the data field in the transmitted data is as follows (▼ indicates the stuff bit):



Excluding the stuff bit gives the following data subject to CRC computation:

010101010101000000110101010

Thus, $P(X) = X^{25} + X^{23} + X^{21} + X^{19} + X^{17} + X^{15} + X^8 + X^7 + X^5 + X^3 + X^1$

$P(X)$ is multiplied by X^{15} , giving:

$$X^{15} \cdot P(X) = X^{40} + X^{38} + X^{36} + X^{34} + X^{32} + X^{30} + X^{23} + X^{22} + X^{20} + X^{18} + X^{16}$$

and this value is divided by

$$G(X) = X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$$

$$\left(\begin{array}{l} X^{15} \cdot P(X) = X^{40} + X^{38} + X^{36} + X^{34} + X^{32} + X^{30} + X^{23} + X^{22} + X^{20} + X^{18} + X^{16} \\ \quad \downarrow \\ P[40:0] = 1010101010100000011010101000000000000000000000000 \\ G(X) = X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1 \\ \quad \downarrow \\ G[15:0] = 1100010110011001 \end{array} \right)$$

Quotient

1100010110011001

EOR of divisor
and dividend is
taken.

101010101010000001101010100000000000000000	
1100010110011001	
1101111001110010	
1100010110011001	
1101111101011110	
1100010110011001	
1101011000111101	
1100010110011001	
1001110100100010	
1100010110011001	
1011000101110110	
1100010110011001	
1110100111011110	
1100010110011001	
1011000100011100	
1100010110011001	
1110100100001010	
1100010110011001	
1011001001001100	
1100010110011001	
1110111110101010	
1100010110011001	
1010100011001100	
1100010110011001	
1101101010101010	
1100010110011001	
1111100110011000	
1100010110011001	
11110000000010	

← Remainder:
R[14:0]

The following value is obtained from the calculation.

$$R[14:0] = 111100000000010$$

In other words, this is the CRC value, added after the data field to give transmit data $Tx(X)$, which is transmitted.

In practice, a stuff bit ($\blacktriangle$) and CRC delimiter ($\triangle$) are added, so that 11110000010000101 is transmitted in the CRC field.

Next, the calculation is shown which determines whether there is an error in receive data $Rx(X)$.

$Rx(X)$ is the value obtained by adding the remainder given by the previous calculation to $X^{15} \cdot P(X)$. This value is divided by $G(X)$.

$$\left(\begin{array}{r} P[40:0] = 1010101010100000011010101000000000000000 \\ + R[14:0] = 111100000000010 \\ \hline 10101010101000000110101010111100000000010 \end{array} \right)$$

This value is divided by $G[15:0]$.

EOR of divisor and dividend is taken.

Quotient	
	10101010101000000110101010111100000000010
1100010110011001	<u>1100010110011001</u>
	1101111001110010
1100010110011001	<u>1100010110011001</u>
	1101111101011110
1100010110011001	<u>1100010110011001</u>
	1101011000111101
1100010110011001	<u>1100010110011001</u>
	1001110100100010
1100010110011001	<u>1100010110011001</u>
	1011000101110111
1100010110011001	<u>1100010110011001</u>
	1110100111011101
1100010110011001	<u>1100010110011001</u>
	1011000100010011
1100010110011001	<u>1100010110011001</u>
	1110100100010100
1100010110011001	<u>1100010110011001</u>
	1011001000110100
1100010110011001	<u>1100010110011001</u>
	1110111101011010
1100010110011001	<u>1100010110011001</u>
	1010101100001100
1100010110011001	<u>1100010110011001</u>
	1101110100101010
1100010110011001	<u>1100010110011001</u>
	1100010110011001
1100010110011001	<u>1100010110011001</u>
	1100010110011001
	<u>1100010110011001</u>
	0

0 ← Remainder

The remainder from the above calculation is zero, so the transmission is considered to be error-free.

6. **ACK field:** For confirmation of normal reception.
Comprises a 1-bit ACK slot and a 1-bit ACK delimiter.

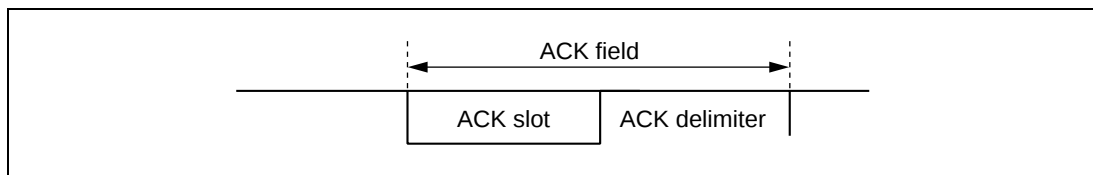


Figure 4.9 ACK Field

The receiving H8S/2623F-ZTAT outputs a high-level ACK slot if it finds an error in the CRC check, and a low-level ACK slot if it finds no error.

7. **Stuff bits:** If there are five consecutive low bits in the data frame, the output is always high for the next bit. Similarly, if there are five consecutive high bits, the output is always low for the next bit.

When stuff bits are output in this way, the bit length of the data frame is increased by the number of stuff bits.

As an example, consider the case where the following settings are made:

Arbitration field: 101010101010

Control field: 000001

The bit pattern in this case is thus 101010101010000001, and so a stuff bit (▼) is output as the second-but-lowest bit (after the five consecutive 0s).

The value transmitted on the CAN bus is therefore 1010101010100000▼101, and the data frame length is increased by the one stuff bit.

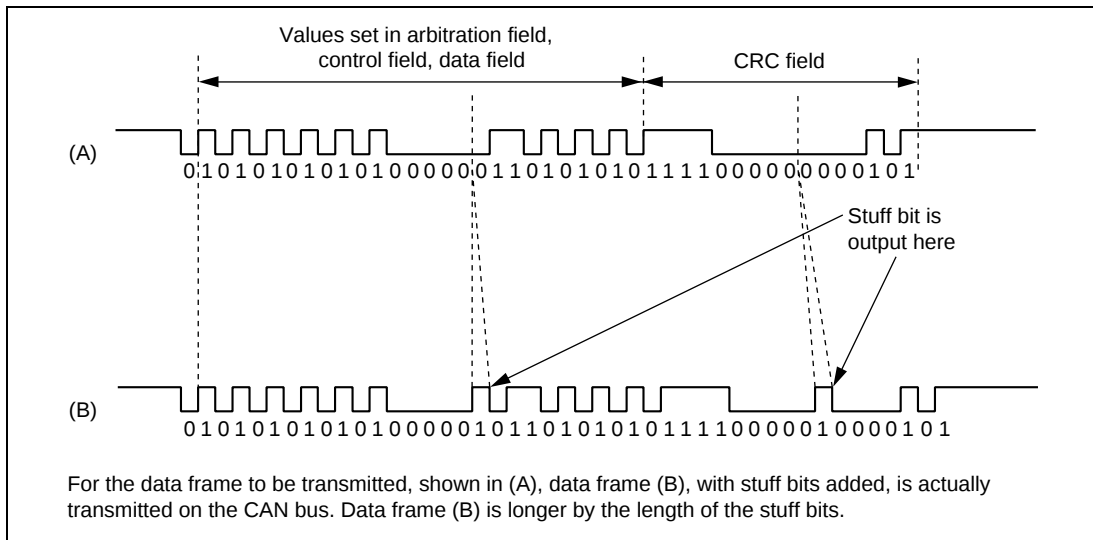


Figure 4.10 Stuff Bits

8. **Configuration mode:** In this mode, the HCAN module is in the reset state. This mode is released by clearing the reset request bit (MCR0) in the master control register (MCR).

Section 5 HCAN Transmission/Reception (Example 5): Remote Frame

MCU: H8S/2623F-ZTAT	Function Used: HCAN
----------------------------	----------------------------

5.1 Specifications

1. Remote frame*¹ transmission/reception (using two H8S/2623F-ZTATs).

Remote frame specifications are shown in figure 5.1.

- a. SOF: Indicates start of remote frame.
- b. Arbitration field*²: Set to 101010101011.
 - RTR = 1: Select remote frame
- c. Control field*³: Set to 00001.
 - IDE = 0: Select standard format
 - R0 = 0: Reserved bit
 - DLC = 0001: Data length setting (Because it is a remote frame, transfer of data from the transmission side (there is no data field*⁴) is not carried out.)
- d. CRC field*⁵: CRC is generated automatically within the HCAN.
- e. ACK field*⁶: 11 is output on the transmitting side, and 01 (in normal operation) on the receiving side. (Mailboxes 1 to 15)
- f. EOF: Indicates the end of a transmit/receive data frame.

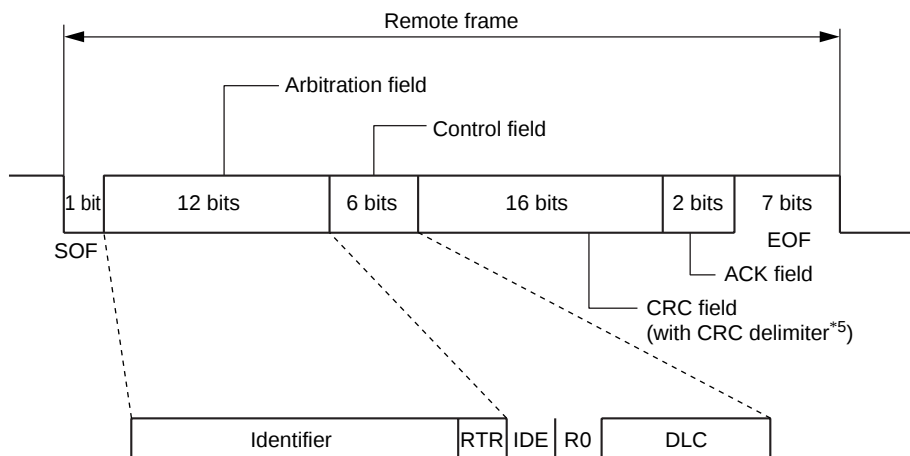
2. A communication speed of 250 kbps (when operating at 20 MHz) is set.

3. Message transmission uses mailbox 1.

4. Message reception uses mailbox 0. The message reception method is to mask the identifier and receive the message in case of a match.

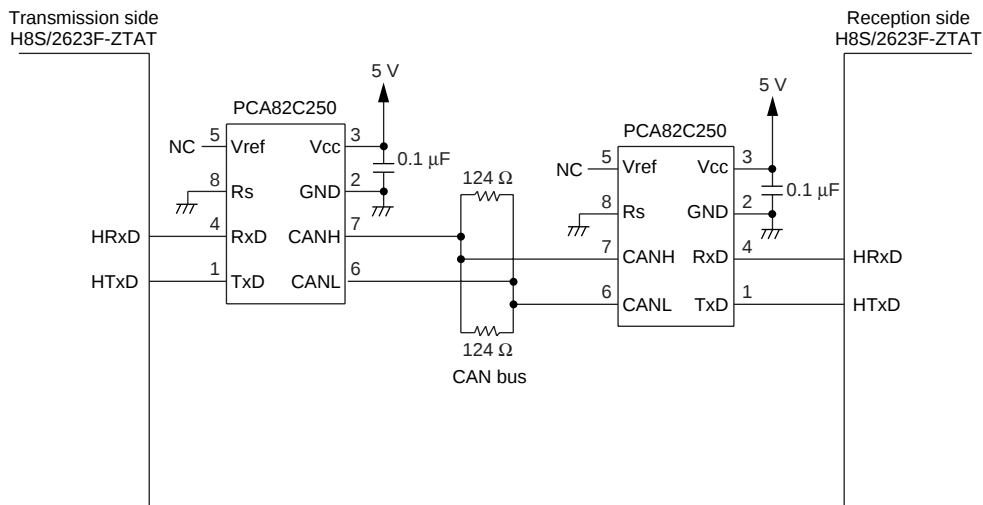
5. Figure 5.2 shows an example of CAN bus connection.

Note: * See 5.9 Notes.



Note: * See 5.9 Notes.

Figure 5.1 Remote Frame Specifications



Note: A bus transceiver IC is necessary to connect an H8S/2623F-ZTAT to the CAN bus. One compatible with the Philips PCA82C50 is recommended.

Figure 5.2 CAN Interface Using H8S/2623F-ZTATs

5.2 Functions

Tables 5.1 and 5.2 show the function allocation of this sample task. This sample task allocates H8S/2623F-ZTAT on-chip HCAN functions as shown in these tables, and carries out HCAN transmission and reception.

Table 5.1 HCAN Function Allocation

HCAN Register		Function
Pins	HTxD	Transmits messages.
	HRxD	Receives messages.
Transmission/ reception registers	IRR	Displays status of each interrupt source.
	BCR	Sets CAN baud rate prescaler and bit timing parameters.
	MBCR	Sets mailbox transmission/reception.
	MCR	Controls CAN interface.
	MC0_1 to MC15_8	Arbitration field and control field settings.
	MD0_1 to MD15_8	Data field settings.
Transmission registers	TXPR	Sets transmission wait state after transmit messages are stored in the mailbox.
	TXACK	Indicates that the transmit message of the corresponding mailbox was transmitted normally.
Reception registers	LAFMH	Sets filter mask for reception mailbox 0 identifier.
	RXPR	Indicates that data has been received normally by the corresponding mailbox.
	RFPR	Indicates that remote frame has been received normally by the corresponding mailbox.

Table 5.2 MSTPCR Function Allocation

MSTPCR Register	Function
MSTPCRC	Controls module stop mode.

5.3 Operation

Operation (Transmission)

Figure 5.3 shows the principle of operation during transmission. HCAN transmission is carried out by H8S/2623F-ZTAT hardware processing and software processing as shown in the figure.

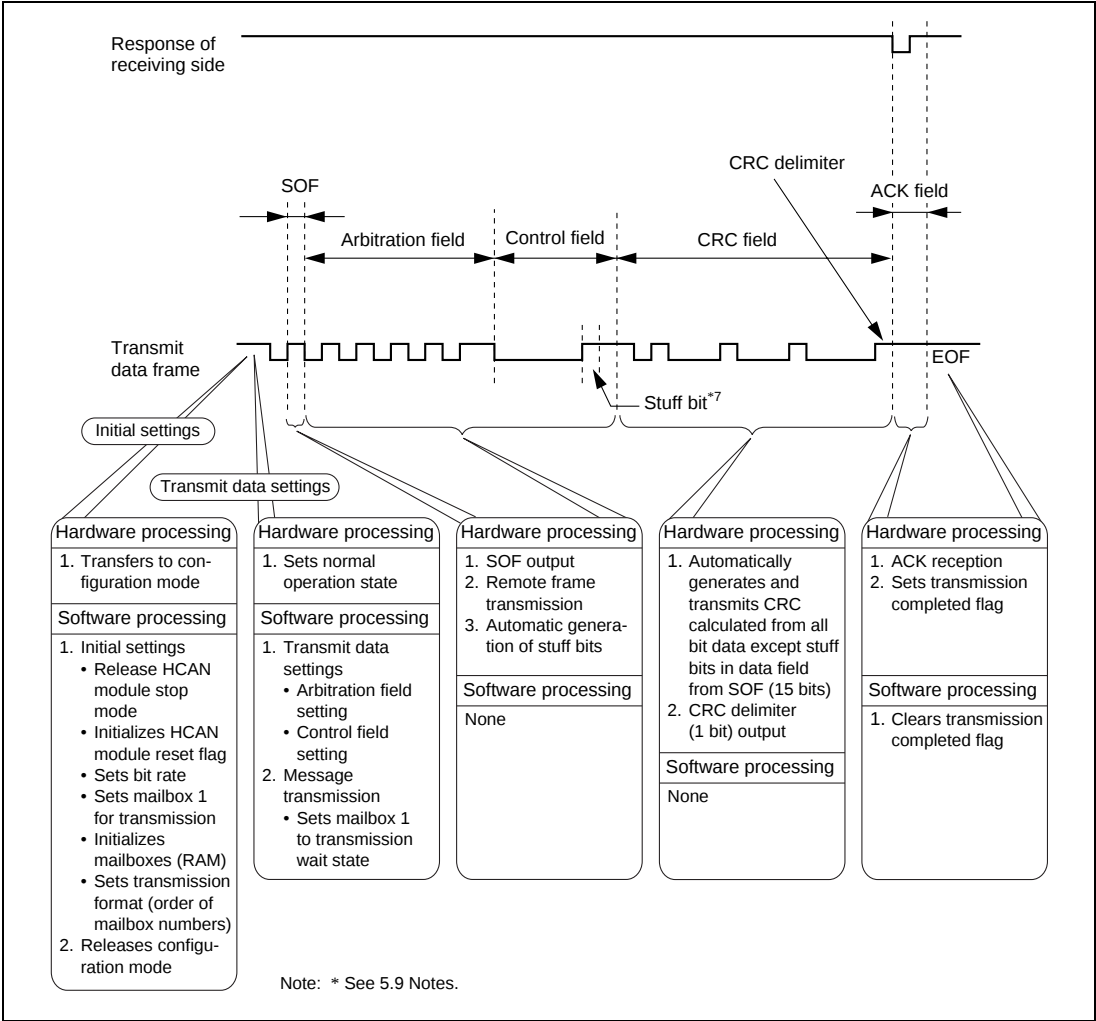


Figure 5.3 Operation During HCAN Transmission

Operation (Reception)

Figure 5.4 shows the principle of operation during transmission. HCAN reception is carried out by H8S/2623F-ZTAT hardware processing and software processing as shown in the figure.

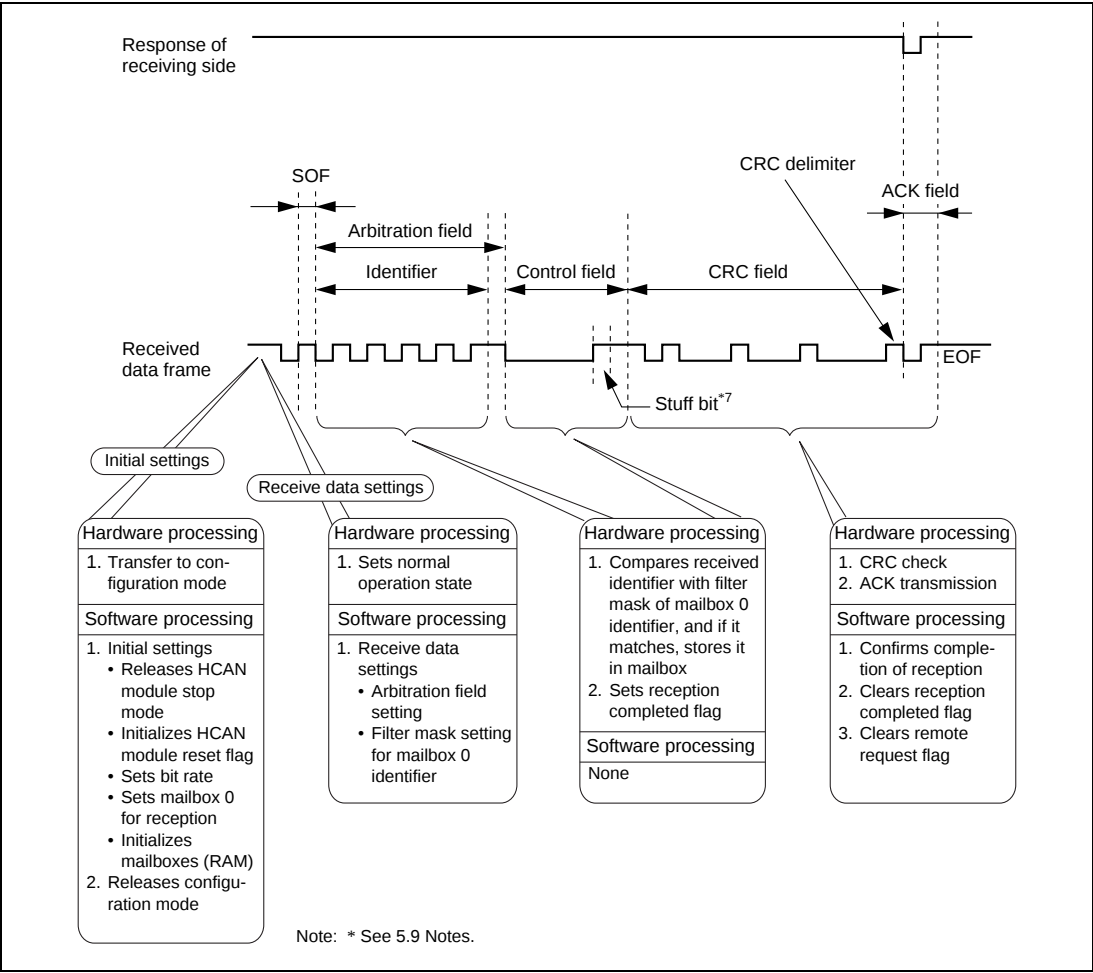


Figure 5.4 Operation During HCAN Reception

5.4 Software

1. Modules

Module Name	Label	Function
Main routine	main	HCAN initial settings and transmission/reception settings.

2. Variables Used

Label	Function	Data Length	Module
COUNT	Initializes HCAN_MCO_1 to MC15_8 and HCAN_MDO_1 to MD15_8.	Unsigned short	Main routine

3. Internal Registers Used

Register Name	Function	Setting	Module
Settings Common to Transmission/Reception			
MSTPCRC	Releases HCAN module stop mode.	0xF7	Main routine
HCAN_IRR	Initializes HCAN module reset flag.	0x0100	
HCAN_BCR	Sets HCAN bit rate to 250 kbps.	0x0334	
Settings for Transmission			
HCAN_MBCR	Sets mailbox 1 for transmission.	0xFDFF	Main routine
HCAN_MCR	Sets transmission in order of mailbox numbers, and clears reset request bit.	0x04	
HCAN_MC1_1	Sets 1-byte data length.	0x01	
HCAN_MC1_5	Selects mailbox 1 remote frame and standard format, and sets identifier.	0xB0	
HCAN_MC1_6	Sets mailbox 1 identifier.	0xAA	
HCAN_TXPR	Sets mailbox 1 to transmission wait state.	0x0200	
HCAN_TXACK	Clears remote frame transmission completed flag.	0x0200	
Settings for Reception			
HCAN_MBCR	Sets mailbox 0 for reception.	0x0100	Main routine
HCAN_MCR	Clears reset request bit.	0xFE	
HCAN_MC0_5	Selects mailbox 0 remote frame and standard format, and sets identifier.	0xB0	
HCAN_MC0_6	Sets mailbox 0 identifier.	0xAA	
HCAN_LAFMH	Sets filter mask for mailbox 0 identifier.	0x0000	
HCAN_RXPR	Clears message reception completed flag.	0x0100	
HCAN_RFPR	Clears remote frame reception completed flag.	0x0100	

5.5 Transmission Flowchart

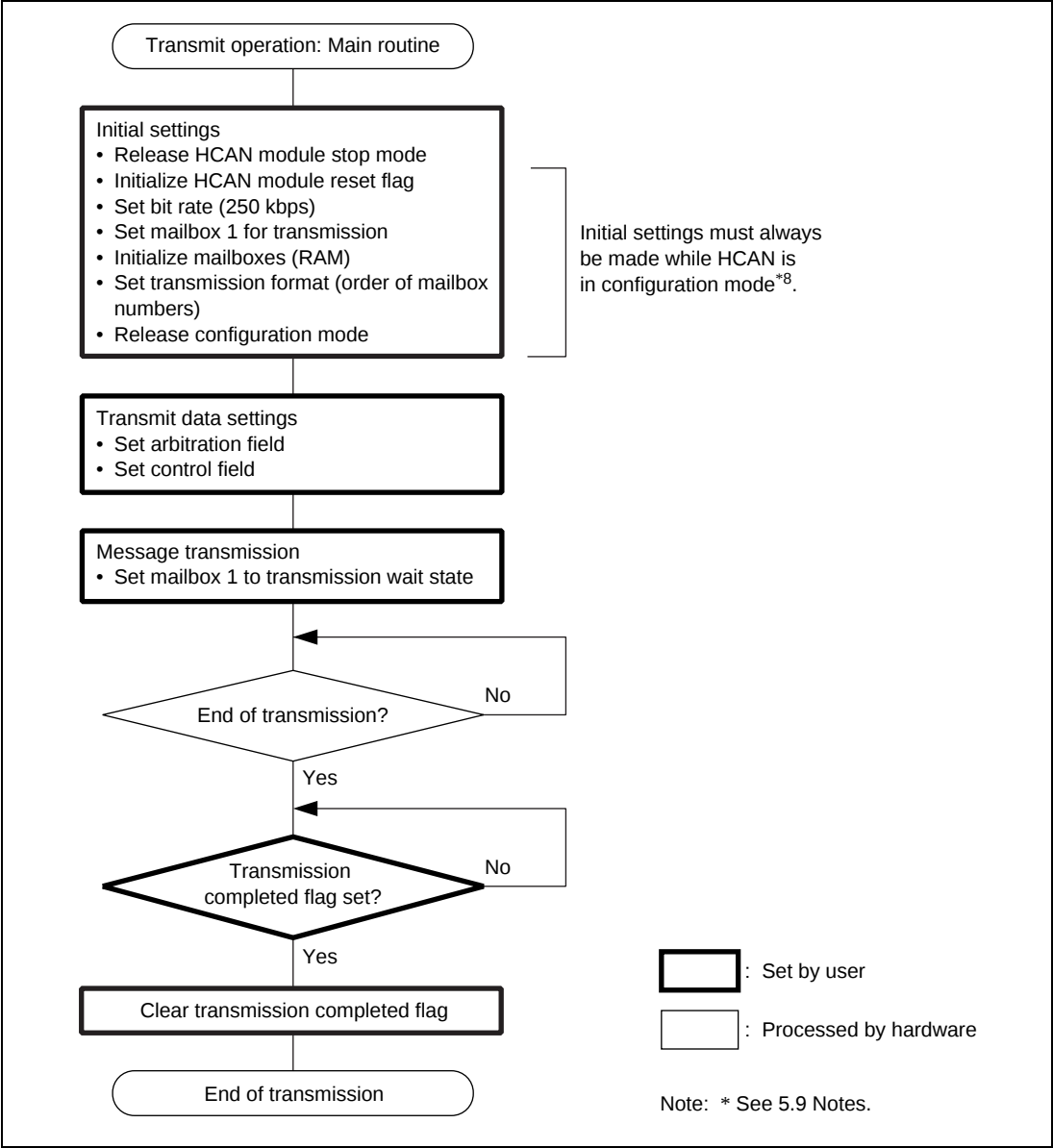


Figure 5.5 Transmission Flowchart

5.6 Transmission Program List

```

/*****
/*
HCAN Transmission Program
*****/

#include <stdio.h>          /* Library function header file */
#include <machine.h>        /* Library function header file */
#include "2623.h"           /* Peripheral register definition header file */
/*****
/*
Function Protocol Declaration
*****/

void main( void );

/*****
/*
Definition of Constants
*****/

#define COUNT (*(unsigned short *)0xFFC000)

/*****
/*
Main Routine
*****/

void main(void)
{
/* Initial Settings */
    MSTPCRC = 0xF7;          /* Release HCAN module stop mode */
    HCAN_IRR = 0x0100;       /* Initialize HCAN module reset flag */
    HCAN_BCR = 0x0334;       /* Bit rate: 250 kbps */
    HCAN_MBCR = 0xFDF;       /* Set mailbox 1 for transmission */
    for( COUNT = 0; COUNT < 128; COUNT++ ) /* Initialize mailbox (RAM) */
    {
        *(char*)&HCAN_MC0_1 + COUNT = 0x00;
    }
    for( COUNT = 0; COUNT < 128; COUNT++ ) /* Initialize mailbox (RAM) */
    {
        *(char*)&HCAN_MD0_1 + COUNT = 0x00;
    }
    HCAN_MCR = 0x04;
    /* Set transmission in order of mailbox numbers, and release configuration
    mode */

/* Transmit data settings */
    HCAN_MC1_5 = 0xB0;
                                /* Set standard format, remote frame, and identifier */
    HCAN_MC1_6 = 0xAA;          /* Set identifier */
}

```

```

    HCAN_MC1_1 = 0x01;                                /* Data length: 1 byte */
/* Message transmission */
    HCAN_TXPR = 0x0200;                                /* Set mailbox 1 to transmission wait state */
    while((HCAN_TXACK & 0x0200) != 0x0200);            /* Wait for completion of transmission */
/* Clear transmission completed flag */
    HCAN_TXACK &= 0x0200;                                /* Clear transmission completed flag */
    while(1);
}

```

5.7 Reception Flowchart

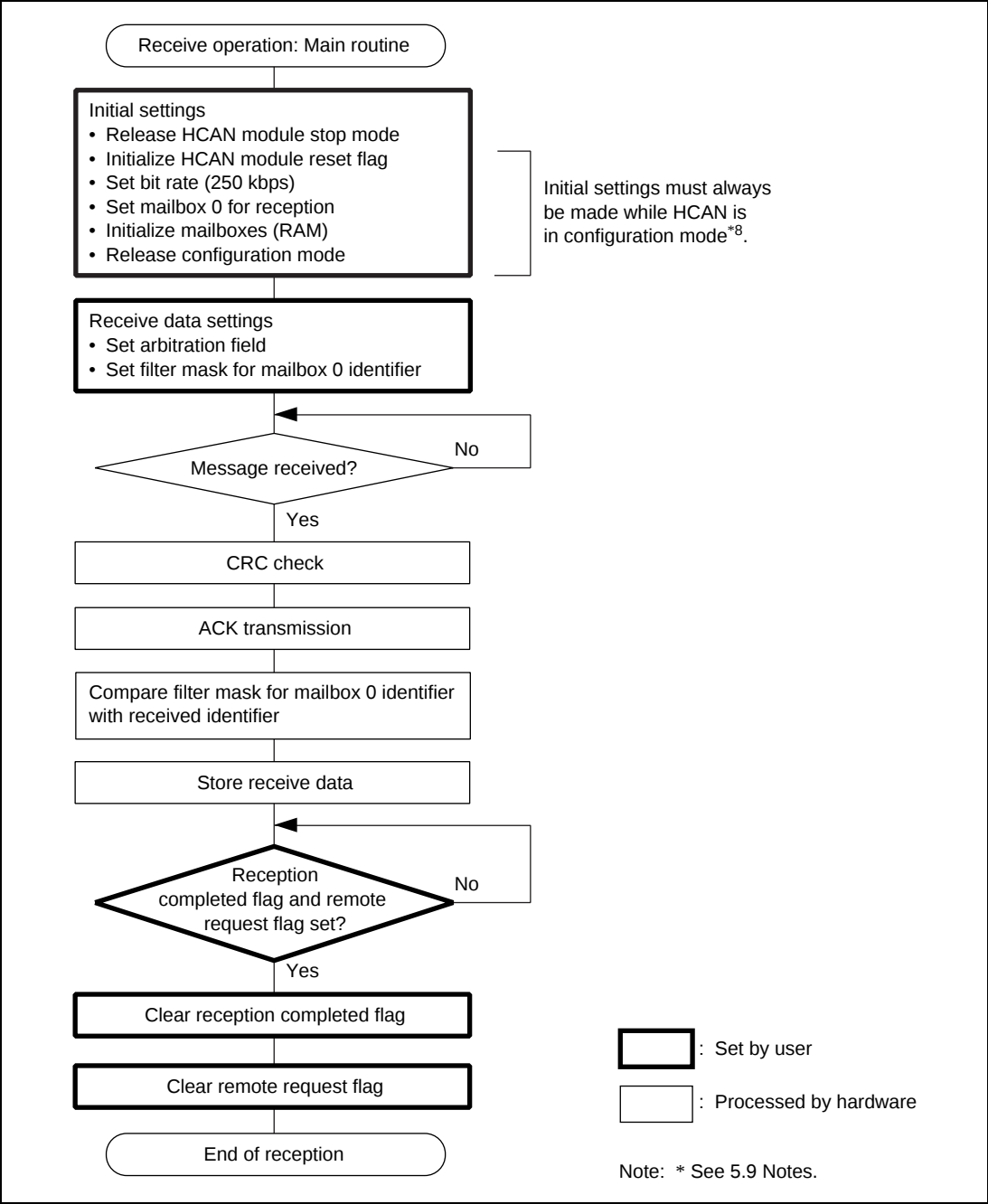


Figure 5.6 Reception Flowchart

5.8 Reception Program List

```

/*****
/*
HCAN Reception Program
*/
*****/
#include <stdio.h>          /* Library function header file
#include <machine.h>        /* Library function header file
#include "2623.h"           /* Peripheral register definition header file
*****/
/*
Function Protocol Declaration
*/
*****/
void main( void );
/*****
/*
Definition of Constants
*/
*****/
#define COUNT (*(unsigned short *)0xFFC000)
/*****
/*
Main Routine
*/
*****/
void main(void)
{
/* Initial settings */
    MSTPCRC = 0xF7;          /* Release HCAN module stop mode
    HCAN_IRR = 0x0100;        /* Initialize HCAN module reset flag
    HCAN_BCR = 0x0334;        /* Bit rate: 250 kbps
    HCAN_MBCR = 0x0100;       /* Set mailbox 0 for reception
    for( COUNT = 0; COUNT < 128; COUNT++ ) /* Initialize mailbox (RAM)
    {
        *(char*)&HCAN_MC0_1 + COUNT) = 0x00;
    }
    for( COUNT = 0; COUNT < 128; COUNT++ ) /* Initialize mailbox (RAM)
    {
        *(char*)&HCAN_MD0_1 + COUNT) = 0x00;
    }
    HCAN_MCR &= 0xFE;        /* Release configuration mode
/* Receive data settings */
    HCAN_MC0_5 = 0xB0;       /* Set standard format, remote frame, and identifier
    HCAN_MC0_6 = 0xAA;        /* Set identifier
    HCAN_LAFMH = 0x0000;      /* Mailbox 0 stores data in case of bit match
/* Reception wait */

```

```

while(((HCAN_RXPR & 0x0100) != 0x0100)
        /* Wait for completion of remote frame reception */
    &&((HCAN_RFPR & 0x0100) != 0x0100));
HCAN_RXPR &= 0x0100;          /* Clear reception completed flag */
HCAN_RFPR &= 0x0100;          /* Clear remote request flag      */
while(1);
}

```

5.9 Notes

1. **Remote frame:** Data frame request is made to the transmission source.
2. **Arbitration field:** Set unique ID for message and data frame or remote frame.
3. **Control field:** Set the data length to be transmitted, and standard format or extended format.
4. **Data field:** Data fields do not exist in remote frames. However, the data length code (DLC) of the control field on the transmission side must have stored in it the data length that should be returned by the data frame.
5. **CRC field:** A CRC is generated automatically in the HCAN from all bit data except stuff bits in the data field from SOF, and is used to detect transmit message errors.
The CRC field comprises a 15-bit CRC and a 1-bit delimiter.
The CRC delimiter is always output as a 1 after the CRC.

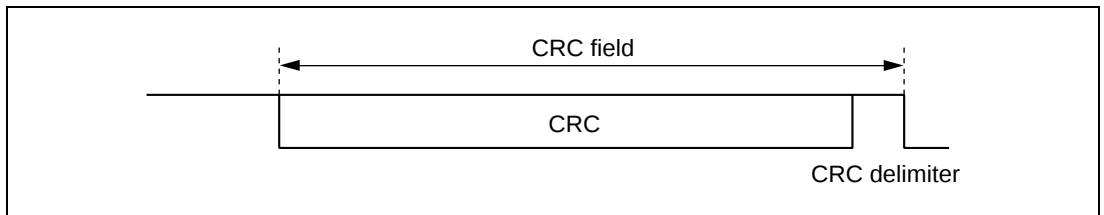


Figure 5.7 CRC Field

6. **ACK field:** For confirmation of normal reception.
Comprises a 1-bit ACK slot and a 1-bit ACK delimiter.

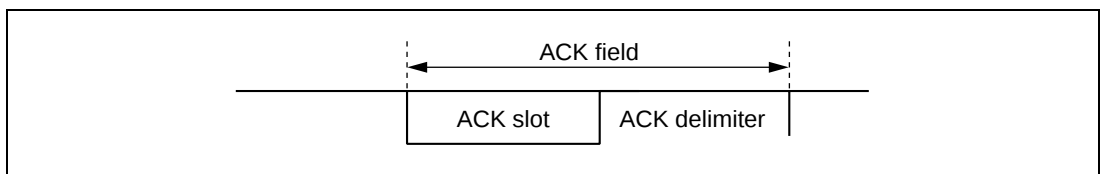


Figure 5.8 ACK Field

The receiving H8S/2623F-ZTAT outputs a high-level ACK slot if it finds an error in the CRC check, and a low-level ACK slot if it finds no error.

7. **Stuff bits:** If there are five consecutive low bits in the data frame, the output is always high for the next bit. Similarly, if there are five consecutive high bits, the output is always low for the next bit.

When stuff bits are output in this way, the bit length of the data frame is increased by the number of stuff bits.

As an example, consider the case where the following settings are made:

Arbitration field: 101010101010

Control field: 000001

The bit pattern in this case is thus 101010101010000001, and so a stuff bit (▼) is output as the second-but-lowest bit (after the five consecutive 0s).

The value transmitted on the CAN bus is therefore 10101010101000001▼01, and the data frame length is increased by the one stuff bit.

8. **Configuration mode:** In this mode, the HCAN module is in the reset state. This mode is released by clearing the reset request bit (MCR0) in the master control register (MCR).

**H8S/2623F-ZTAT™ On-Chip HCAN
(Hitachi Controller Area Network)
Application Note**

Publication Date: 1st Edition, January 2000

Published by: Electronic Devices Sales & Marketing Group
Semiconductor & Integrated Circuits
Hitachi, Ltd.

Edited by: Technical Documentation Group
Hitachi Kodaira Semiconductor Co., Ltd.

Copyright © Hitachi, Ltd., 2000. All rights reserved. Printed in Japan.