To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

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

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

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

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

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

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

H8/300H Tiny Series

ROM Correction

Introduction

ROM correction is a function to fix defective parts or modify specifications of the ROM. This function is implemented by use of the address break function that generates an interrupt when a specific execution address is reached, and external memory such as an EEPROM.

Target Device

H8/300H Tiny Series H8/36014 CPU

Contents

1. Specifications

- Figure 1 illustrates the specification of a ROM correction task.

  ROM correction function
  Defective parts found after mask ROM production waste considerable time and money for reproducing mask ROMs, collecting defective ROMs, and storage of dead stock. H8/3664 incorporates the address break interrupt function that generates an interrupt when the pre-defined address matches the execution address. Use of the address break interrupt function combined with external memory such as EEPROM enables simple replacement of defective parts and modification of specifications of the ROM. This is called the ROM correction function.

  — Activating the program assigns P75 for output to perform the port output processing.
  — If an IRQ1 interrupt occurs during P75 switchover, downloads the correction program from the external EEPROM.
  — Then, uses the address break function to change the execution address to the on-chip RAM that contains the downloaded correction program.
  — The correction program assigns P75 for input and P74 for output to change the port output processing from P75 to P74.
  — After the correction is made, returns to the address set by the correction program.
  — Assigns P75 and P74 for output, changes them to a low level, and then terminates the function.

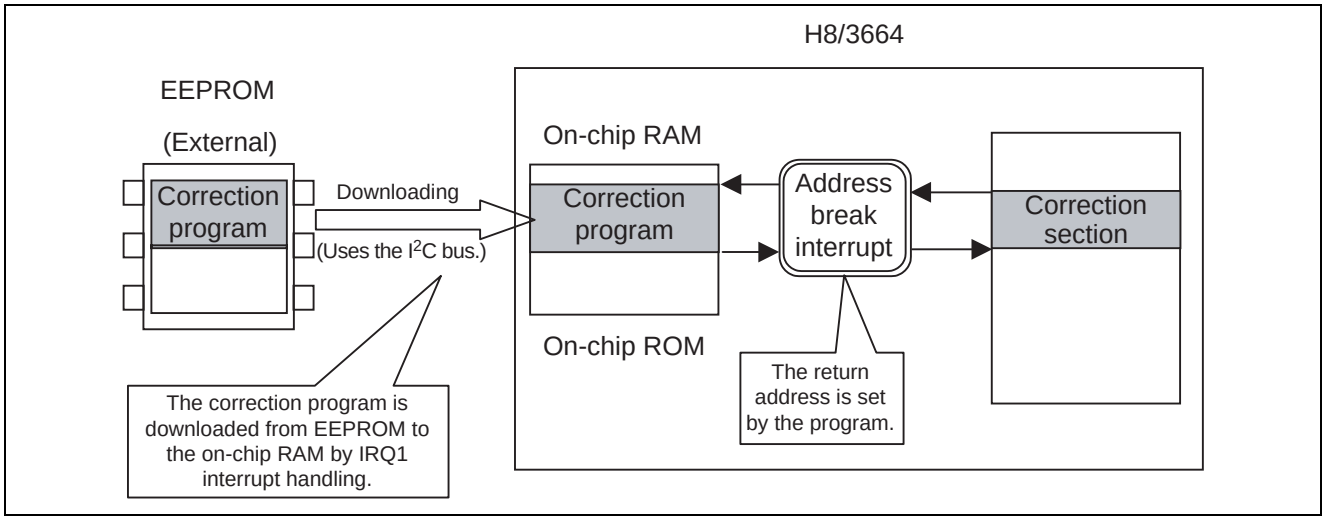


Figure 1 Specification of ROM Correction

2. Description of Functions

2.1 I^2C Bus Interface

- Figure 2 illustrates the block diagram of the I^2C bus interface described in this subsection.
  - — I^2C bus control register (ICCR)
    Consists of the control bits and interrupt request flags of the I^2C bus interface.
  - — I^2C bus mode register (ICMR)
    Sets the transfer format and transfer rate. It can be accessed only when the ICE bit in ICCR is 1.
  - — I^2C bus status register (ICSR)
    Consists of the status flags.
  - — I^2C bus data register (ICDR)
    This is an 8-bit readable/writable register that is used as a transmit data register when transmitting and a receive data register when receiving.

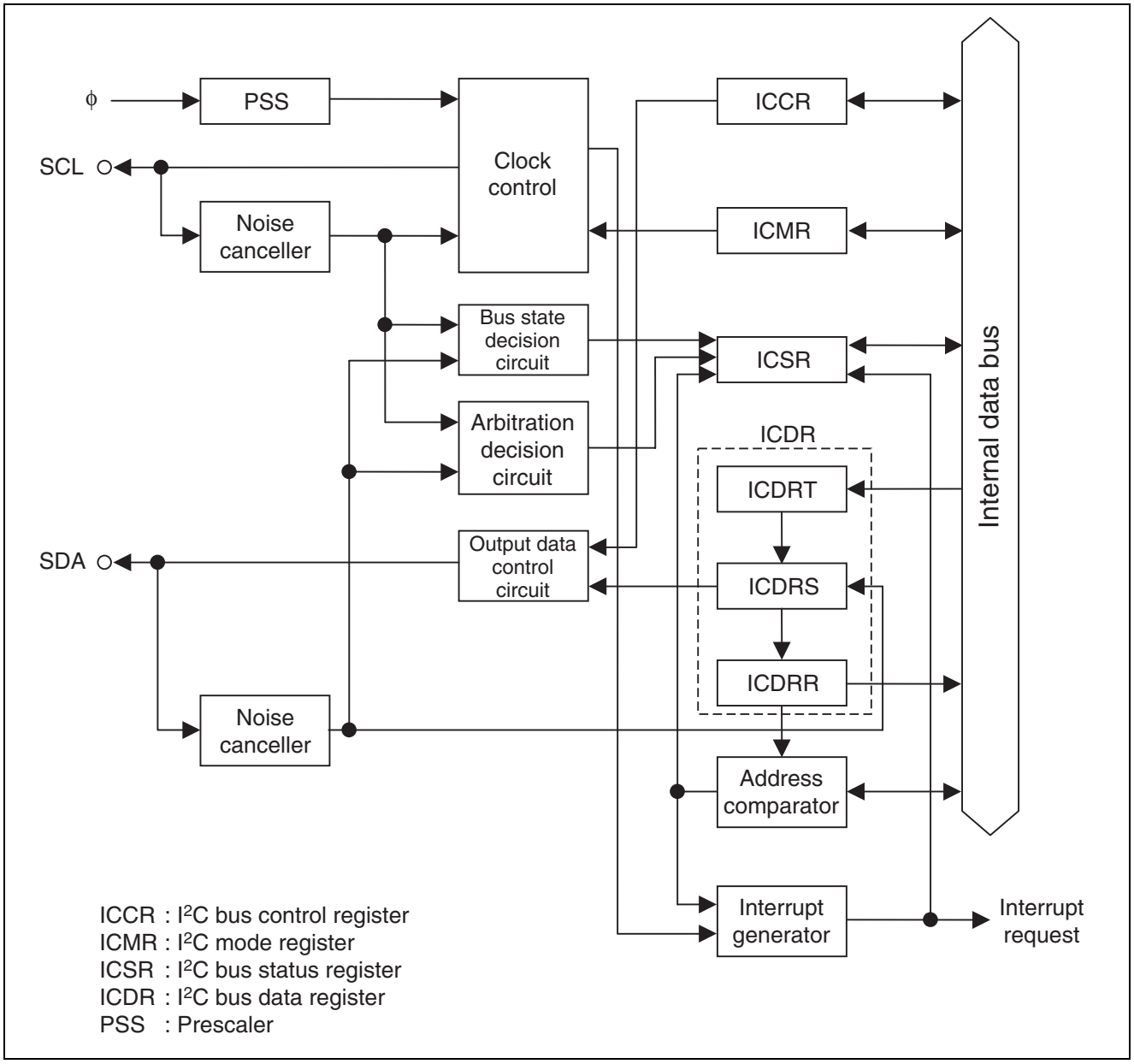


Figure 2 I^2C Bus Interface Block Diagram

- Table 1 lists the pins used by the I^2C bus interface.

Table 1 I^2C Bus Interface Pins

Name	Abbreviation	Input/Output	Function
Serial clock pin	SCL	Input/Output	I^2C serial clock input/output
Serial data pin	SDA	Input/Output	I^2C serial data input/output

2.2 Address Break

- Figure 3 illustrates the block diagram of the address break interrupt handling described in this subsection.

  — Address break control register (ABRKCR)
  Sets address break conditions.
  — Address break status register (ABRKSR)
  Consists of the address break interrupt request flags and their enable bits.
  — Break address register (BAR (BARL, BARH))
  This is a 16-bit readable/writable register that sets the address for generating an address break interrupt. BARH indicates upper eight bits and BARL lower eight bits.

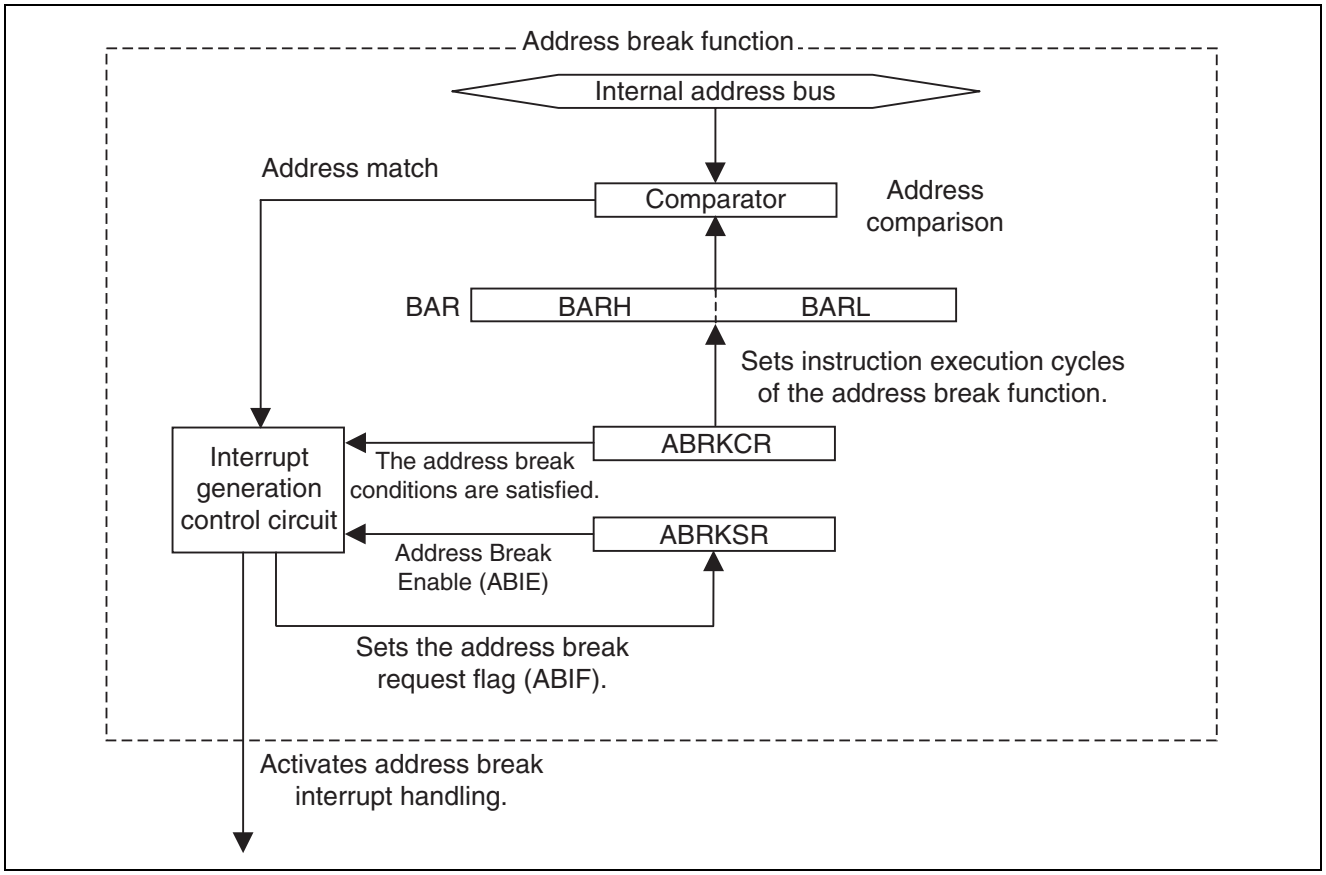


Figure 3 Block Diagram of Address Break Interrupt Handling

3. Description of Operation

- Figure 4 to Figure 7 illustrate the operation of a ROM correction task example described in this subsection.

Activates the H8/3664 program.

1. Generates the IRQ1 interrupt handling and reads the data (shown in Figure 4) from the EEPROM address H'0000.
   - Sets the downloaded break address (H'011E) to the break address register (BAR).
   - Downloads the correction program size (H'52) to H'FC00 in the on-chip RAM.

Set the address break conditions (other than the break address register) after reading the data.

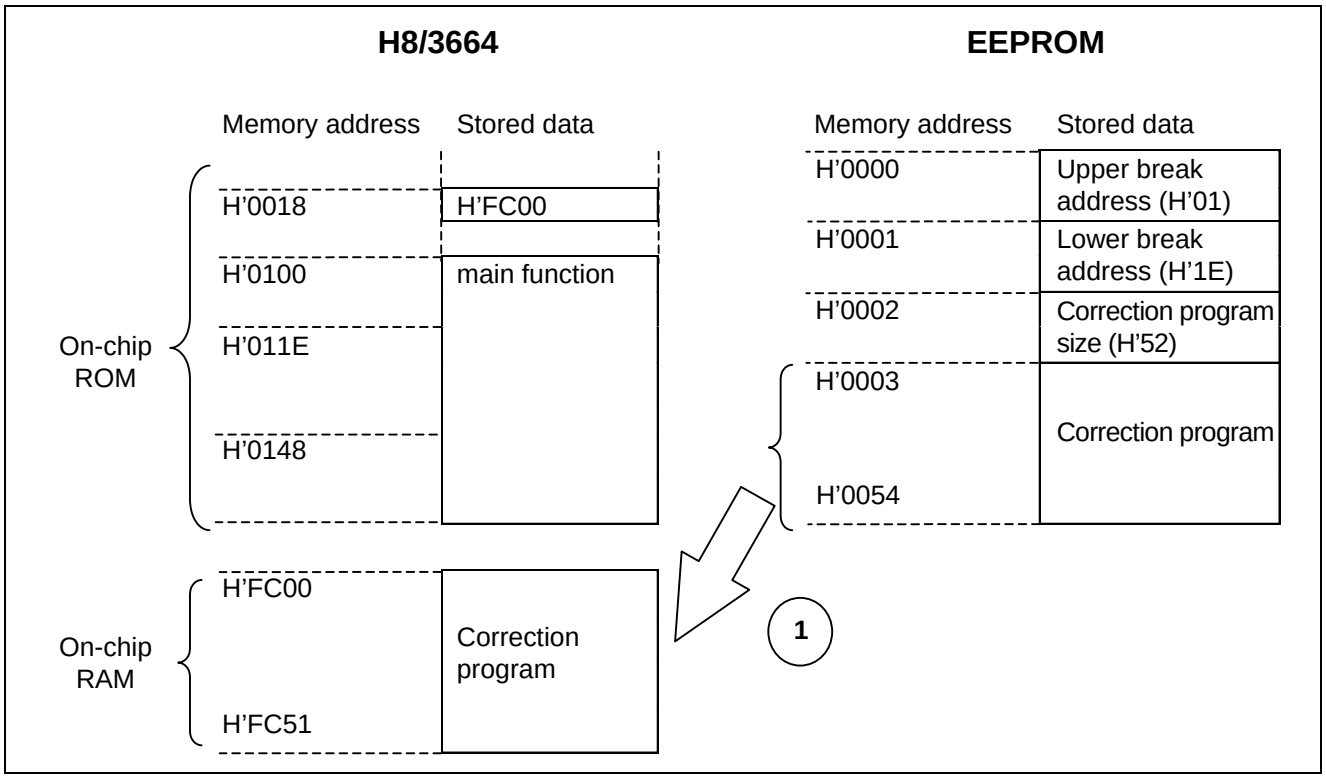


Figure 4 ROM Correction 1

2. When the address break conditions are satisfied (after executing the instruction at the address H'011E), an address break interrupt is generated and the address is changed to the vector address H'0018.

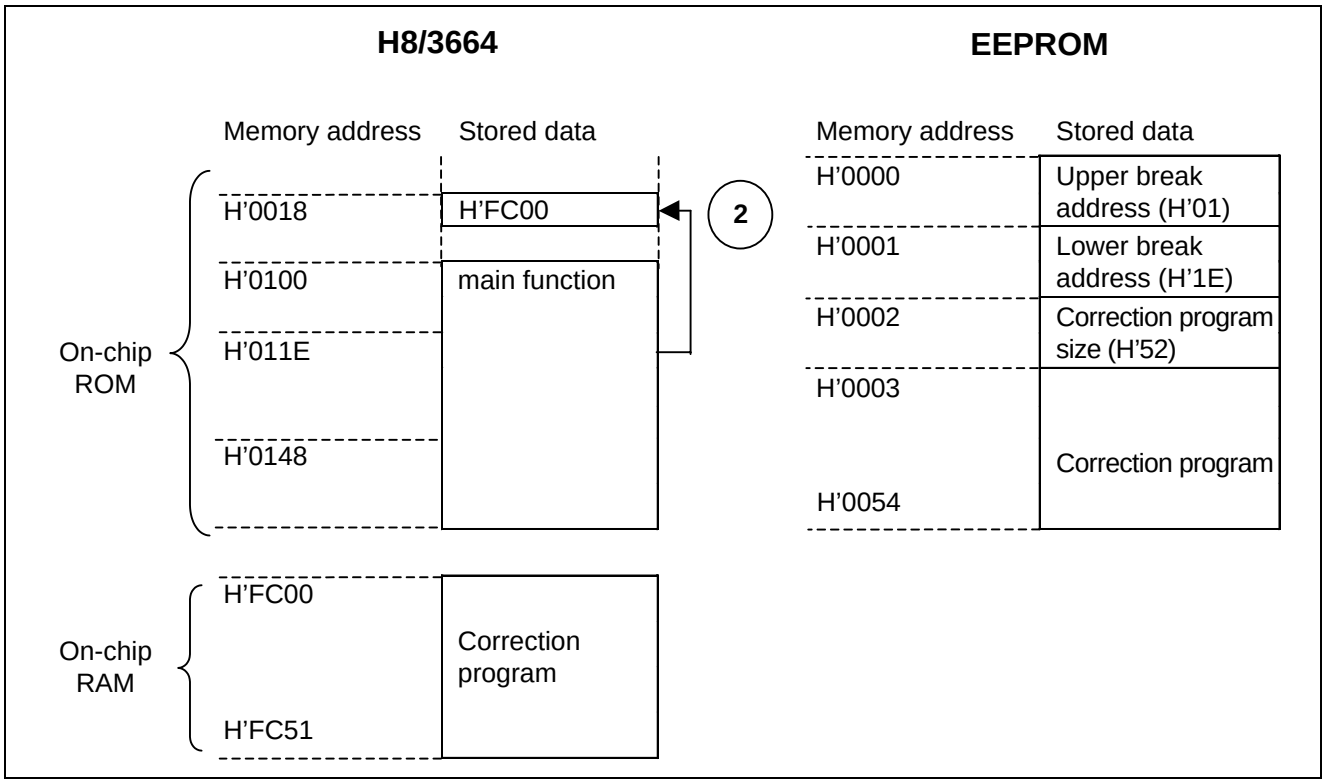


Figure 5 ROM Correction 2

3. Changes to the address (H'FC00) set for the vector address H'0018 and activates the correction program.

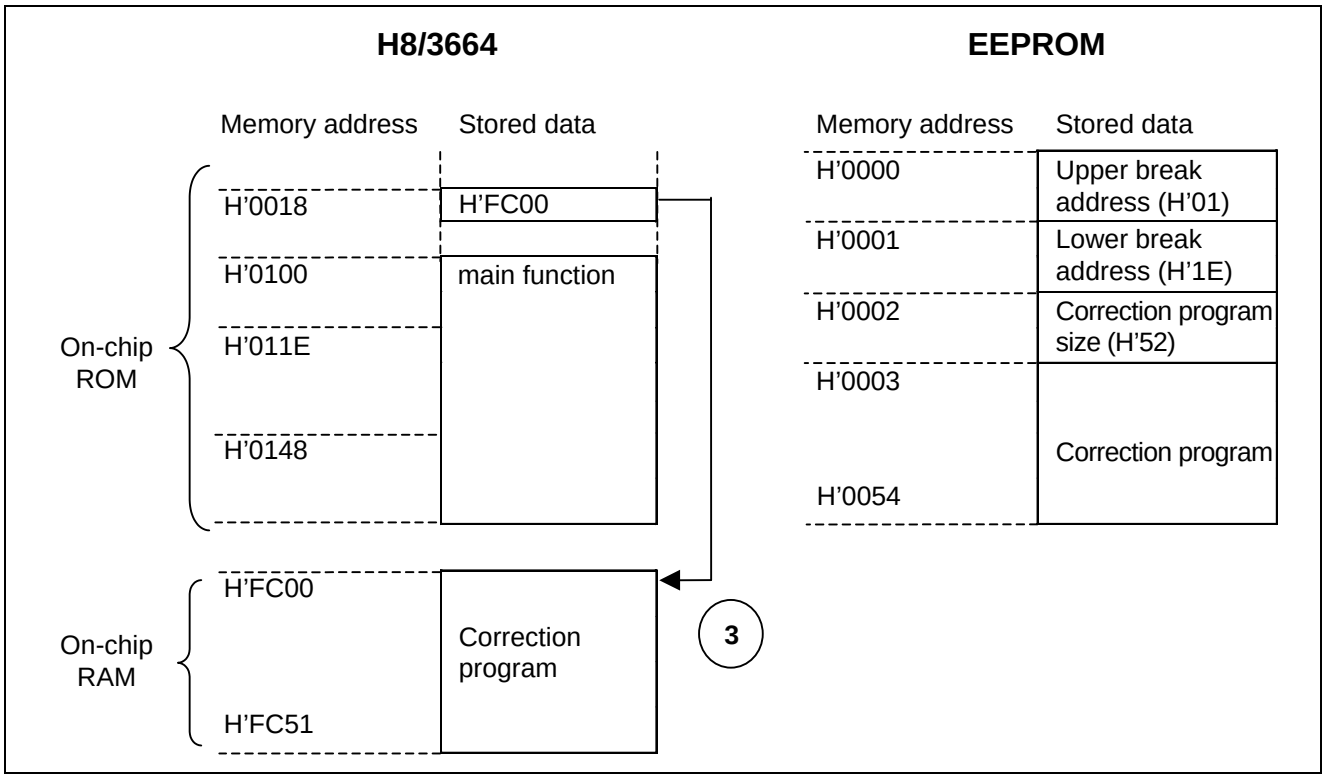


Figure 6 ROM Correction 3

4. Executes the correction program and then returns to the address (H'0148) set by the correction program.

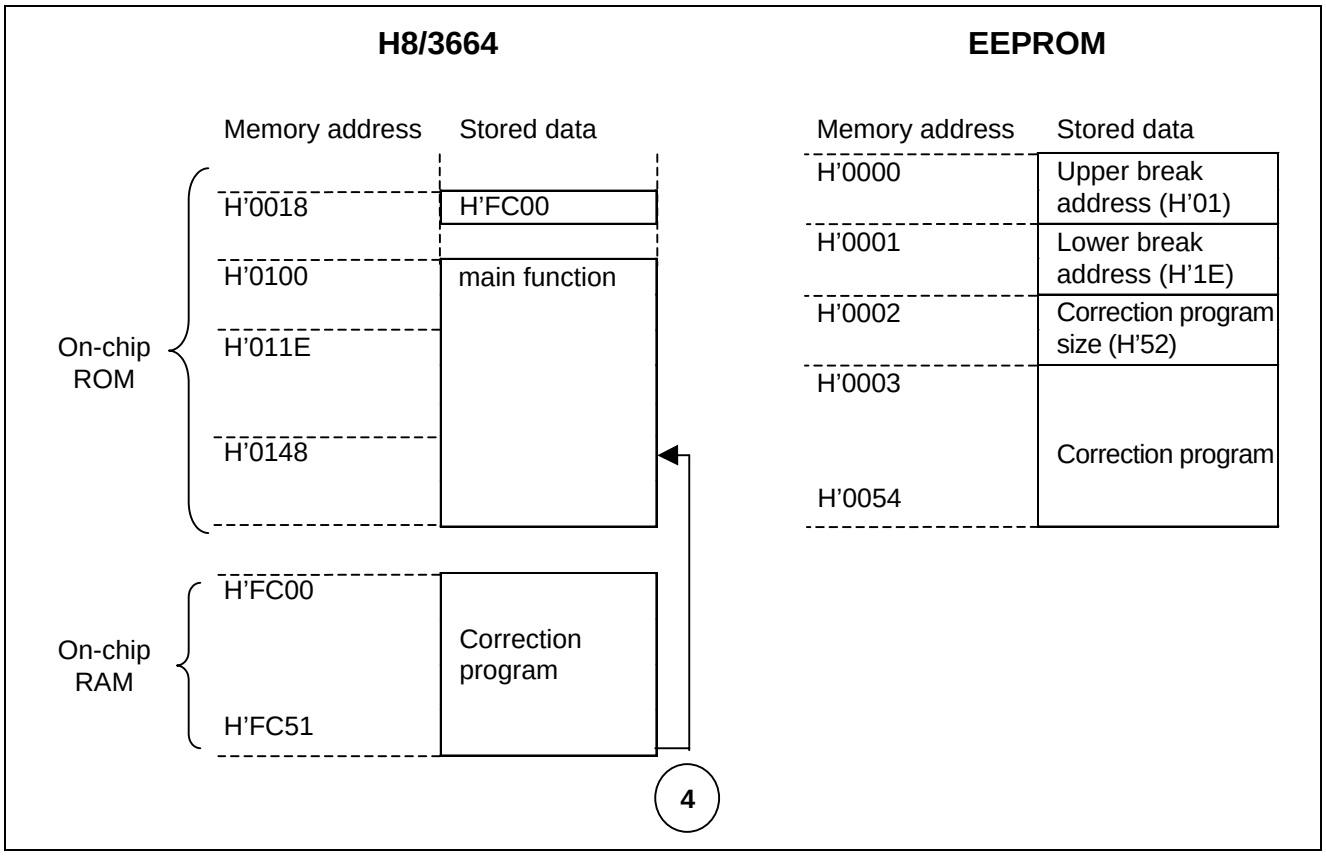


Figure 7 ROM Correction 4

4. Description of Software

4.1 Description of Modules

- Table 2 lists the modules in this task example.

Table 2 Modules

Module	Label	Function
Main routine	main	Sets an IRQ1 interrupt and switches P75 between high and low.
IRQ1 interrupt routine	irq1int	Calls the Read_EEPROM function from the EEPROM and sets address break conditions.
Download setting routine	Read_EEPROM	Calls the Set_Adrs and Recv_data functions.
Download routine	Recv_data	Downloads the correction program from the EEPROM.
Slave address setting routine	Set_Adrs	Sets a slave address.

4.2 Description of Arguments

- Table 3 lists the arguments used in this task example.

Table 3 Arguments

Label	Argument	Description
Read_EEPROM	unsigned short adrs	Start address of the EEPROM for reading data
	unsigned char *rd_data	Read data
	unsigned short *br_ad	Break address pointer
Recv_data	unsigned char *rd_data	Read data
	unsigned short *br_ad	Break address pointer
Set_Adrs	unsigned short adrs	Start address of the EEPROM for reading data

4.3 Description of Return Values

- Table 4 lists the return values used in this task example.

Table 4 Return Values for Each Module

Module	Label	Return Value
Download setting routine	Read_EEPROM	0: Normal termination 1: Abnormal termination
Download routine	Recv_data	0: Normal termination 1: Abnormal termination
Slave address setting routine	Set_Adrs	0: Normal termination 1: Abnormal termination

4.4 Description of Variables

- Table 5 lists the variables used in this task example.

Table 5 Variables

Label	Variables	Description
main	volatile unsigned long i	Wait count
	unsigned char j	P75 high/low switchover count
irq1int	unsigned short adrs	Start address of the EEPROM for reading data
	unsigend short bar	Stores break addresses.
	unsigned short *br_ad	Break address pointer
	unsigned char tmp	Error check
	unsigned char *rd_data	Read data
Read_EEPROM	unsigned char tmp	Error check
Recv_data	unsigned char recv	Reads dummy data and stores the correction program size data.
	unsigend char cnt	Read count of the correction program

4.5 Description of Constants

- Table 6 lists the constants used in this task example.

Table 6 Constants

Constant name	Constant	Description
DEVICE_CODE	H'A0	Device code
SLAVE_ADRS	H'00	Device address code
IIC_DATA_W	H'00	WRITE code
IIC_DATA_R	H'01	READ code
EP_ADRS	H'0000	Start address of the EEPROM for reading data
RAM_AREA	H'FC00	Start address for storing the data read from the EEPROM

4.6 Description of RAM

- In this task example, the start address for storing the correction program downloaded from the EEPROM is H'FC00.

4.7 Description of Internal Register

- This subsection describes the internal registers used in this task example.

— ICCR I²C Bus Control Register (ICCR) Address: H'FFC4

Bit	Bit name	Set value	R/W	Description
7	ICE	1	R/W	I²C Bus Interface Enable When this bit is set to 1, the I²C bus interface module is enabled to send/receive data and drive the bus since it is connected to the SCL and SDA pins. ICMR and ICDR can be accessed. When this bit is cleared, the module is halted and separated from the SCL and SDA pins.
6	IEIC	0	R/W	I²C Bus Interface Interrupt Enable When this bit is 1, interrupt requests are enabled by IRIC.
5	MST	1	R/W	Master/Slave Select
4	TRS	1	R/W	Transmit/Receive Select 00: Slave receive mode 01: Slave transmit mode 10: Master receive mode 11: Master transmit mode
3	ACKE	1	R/W	Acknowledge Bit Judgment Selection 1: If the received acknowledge bit is 1, transfer is interrupted. 0: The value of the received acknowledge bit is ignored, and transfer is performed. The value of the received acknowledge bit is not indicated by the ACKB bit, which is always 0.
2	BBSY	0	R/W	Bus Busy In the master mode, this bit is used to issue the start and stop conditions. Issuing the start conditions: Writes 1 to the BBSY and 0 to the SCP. It is the same for retransmitting the start conditions. The conditions are recognized to be issued when the level of SDA is changed from high to low while SCL = High. After issuing them, this bit is set to 1. Issuing the stop conditions: Writes 0 to the BBSY and 0 to the SCP. The conditions are recognized to be issued when the level of SDA is changed from low to high while SCL = High. After issuing them, this bit is cleared to 0. The MOV instruction is used for issuing the start or stop conditions. The I²C bus interface must be set to the master transmit mode before issuing the start conditions.

Bit	Bit name	Set value	R/W	Description
1	IRIC	0	R/W	I²C Bus Interface Interrupt Request Flag [Setting conditions] (In master mode with I²C bus format) • When a start condition is detected in the bus line state after a start condition is issued • When a wait is inserted between the data and acknowledge bit when WAIT = 1 • When terminating data transfer • When a slave address is received after the device loses arbitration • When 1 is received as the acknowledge bit when the ACKE bit is 1(when the ACKB bit is set to 1) [Clearing condition] • When 0 is written in IRIC after reading IRIC = 1
0	SCP	1	W	Start Condition/Stop Condition Prohibit The SCP bit controls the issue of start/stop conditions in master mode. To issue a start condition, write 1 to BBSY and 0 to SCP. A retransmit start condition is issued in the same way. To issue a stop condition, write 0 to BBSY and 0 to SCP. This bit is always read as 1. If 1 is written, the data is not stored.

— ICSR I²C Bus Status Register (ICSR) Address: H'FFC5

Bit	Bit name	Set value	R/W	Description
0	ACKB	0	R/W	Acknowledge bit In transmit mode, acknowledge data returned from the receive device is loaded. In receive mode, data is received for the transmit device and then the acknowledge data set previously to this bit is transmitted. By reading this bit, the loaded value (returned from the receive device) is read at transmission and the set value is read at reception.

— ICDR I²C Bus Data Register (ICDR) Address: H'FFC6

ICDR is an 8-bit readable/writable register that is used as a transmit data register when transmitting and a receive data register when receiving. ICDR is divided internally into a shift register (ICDRS), receive buffer (ICDRR), and transmit buffer (ICDRT).

Data transfers among the three registers are performed automatically in coordination with changes in the bus state, and affect the status of internal flags such as TDRE and RDRF.

- When TDRE is 1 the transmit buffer is empty and TDRE shows that the next transmit data can be written from the CPU.
- When RDRF is 1, it shows that the valid receive data is stored in the receive buffer.
- If the device is in transmit mode and the next data is in ICDRT (the TDRE flag is 0) following transmission/reception of one frame of data using ICDRS, data is transferred automatically from ICDRT to ICDRS.

- If the device is in receive mode and no previous data remains in ICDRR (the RDRF flag is 0) following transmission/reception of one frame of data using ICDRS, data is transferred automatically from ICDRS to ICDRR.

— ICMR I^2C Bus Mode Register (ICMR) Address: H'FFC7

Bit	Bit name	Set value	R/W	Description
7	MLS	0	R/W	MSB-First/LSB-First Select 0: MSB-first 1: LSB-first Set this bit to 0 when the I^2C bus format is used.
6	WAIT	0	R/W	Wait Insertion Bit This bit is valid only in master mode with the I^2C bus format. When WAIT is set to 1, after the fall of the clock for the final data bit and the IRIC flag is set to 1 in ICCR, a wait state begins (with SCL at the low level). When the IRIC flag is cleared to 0 in ICCR, the wait ends and the acknowledge bit is transferred. If WAIT is cleared to 0, data and acknowledge bits are transferred consecutively with no wait inserted. The IRIC flag in ICCR is set to 1 on completion of the acknowledge bit transfer, regardless of the WAIT setting.
5	CKS2	0	R/W	Serial Clock Select 2 to 0
4	CKS1	0	R/W	This bit is valid only in master mode.
3	CKS0	1	R/W	These bits select the required transfer rate, together with the IICX bit in TSCR.
2	BC2	0	R/W	Bit Counter 2 to 0
1	BC1	0	R/W	These bits specify the number of bits to be transferred next.
0	BC0	0	R/W	With the I^2C bus format, the data is transferred with one additional acknowledge bit. Bit BC2 to BC0 settings should be made during an interval between transfer frames. If bits BC2 to BC0 are set to a value other than 000, the setting should be made while the SCL line is low. The value automatically returns to 000 at the end of data transfer, including the acknowledge bit. I^2C bus formats 000: 9 bits 001: 2 bits 010: 3 bits 011: 4 bits 100: 5 bits 101: 6 bits 110: 7 bits 111: 8 bits

— TSCR Timer Serial Control Register Address: H'FFFC

Bit	Bit name	Set value	R/W	Description
7 to 2	—	All 1	—	Reserved. These bits are always read as 1.
1	IICRST	0	R/W	Resets the control unit except for the I^2C registers. When a hang up occurs due to illegal communication during I^2C operation, setting IICRST to 1 can set a port or reset the I^2C control unit without initializing registers.
0	IICX	0	R/W	Selects the transfer rate in master mode, together with bits CKS2 to CKS0 in ICMR.

Table 7 Transfer Rate for This Task

TSCR	ICMR			Clock	Transfer rate
Bit 0	Bit 5	Bit 4	Bit 3		
IICX	CKS2	CKS1	CKS0		ϕ = 16MHz
0	0	0	1	ϕ /40	400kHz

— ABRKCR Address Break Control Register Address: H'FFC8

Bit	Bit name	Set value	R/W	Description
7	RTINTE	1	R/W	RTE Interrupt Enable When this bit is 0, the interrupt immediately after executing an RTE instruction is masked and then one instruction must be executed. When this bit is 1, the interrupt is not masked.
6	CSEL1	0	R/W	Condition Select 1 and 0
5	CSEL0	0	R/W	These bits set address break conditions. CSEL1 = 0, CSEL0 = 0: Instruction execution cycle
4	ACMP2	0	R/W	Address Compare 2 to 0
3	ACMP1	0	R/W	These bits set comparison conditions between BAR and the internal address bus.
2	ACMP0	0	R/W	ACMP2 = 0, ACMP1 = 0, ACMP0 = 0: Compares 16-bit addresses.
1	DCMP1	0	R/W	Data Compare 1 and 0
0	DCMP0	0	R/W	These bits set the comparison conditions between BDR and the internal data bus. DCMP1 = 0, DCMP0 = 0: No data comparison

— ABRKSR Address Break Status Register Address: H'FFC9

Bit	Bit name	Set value	R/W	Description
7	ABIF	0	R/W	Address Break Interrupt Request Flag ABIF = 1: When the condition set in ABRKCR is satisfied ABIF = 0: Initial value. When 0 is written after ABIF = 1 is read
6	ABIE	1	R/W	Address Break Interrupt Request Enable ABIE = 0: An address break interrupt request is masked. ABIE = 1: An address break interrupt request is enabled.
5 to 0	—	All 1	—	Reserved. These bits are always read as 1.

— BAR Break Address Register Address: H'FFCA
(BARH break address register H Address: H'FFCA)
(BARL break address register L Address: H'FFCB)
Function: Sets the address at which an address break interrupt handling occurs in 16-bit units.

— PMR1 Port Mode Register 1 Address: H'FFE0

Bit	Bit name	Set value	R/W	Description
5	IRQ1	1	R/W	P15/IRQ1 Pin Function Switch 0: General input/output port 1: $\overline{\text{IRQ1}}$ input pin

— IEGR1 Interrupt Edge Select Register 1 Address: H'FFF2

Bit	Bit name	Set value	R/W	Description
1	IEG1	1	R/W	IRQ1 Edge Select 0: Detects the falling edge of the IRQ1 pin input. 1: Detects the rising edge of the IRQ1 pin input.

— IENR1 Interrupt Enable Register Address: H'FFF4

Bit	Bit name	Set value	R/W	Description
1	IEN1	1	R/W	IRQ1 Interrupt Request Enable When this bit is set to 1, the interrupt request of the $\overline{\text{IRQ1}}$ pin is enabled.

— IRR1 Interrupt Flag Register 1 Address: H'FFF6

Bit	Bit name	Set value	R/W	Description
1	IRRI1	0	R/W	IRQ1 Interrupt Request Flag [Setting condition] When the $\overline{\text{IRQ1}}$ pin is designated for interrupt input and the designated signal edge is detected. [Clearing condition] When IRRI1 is cleared by writing 0.

— PCR7 Port Control Register 7 Address: H'FFEA

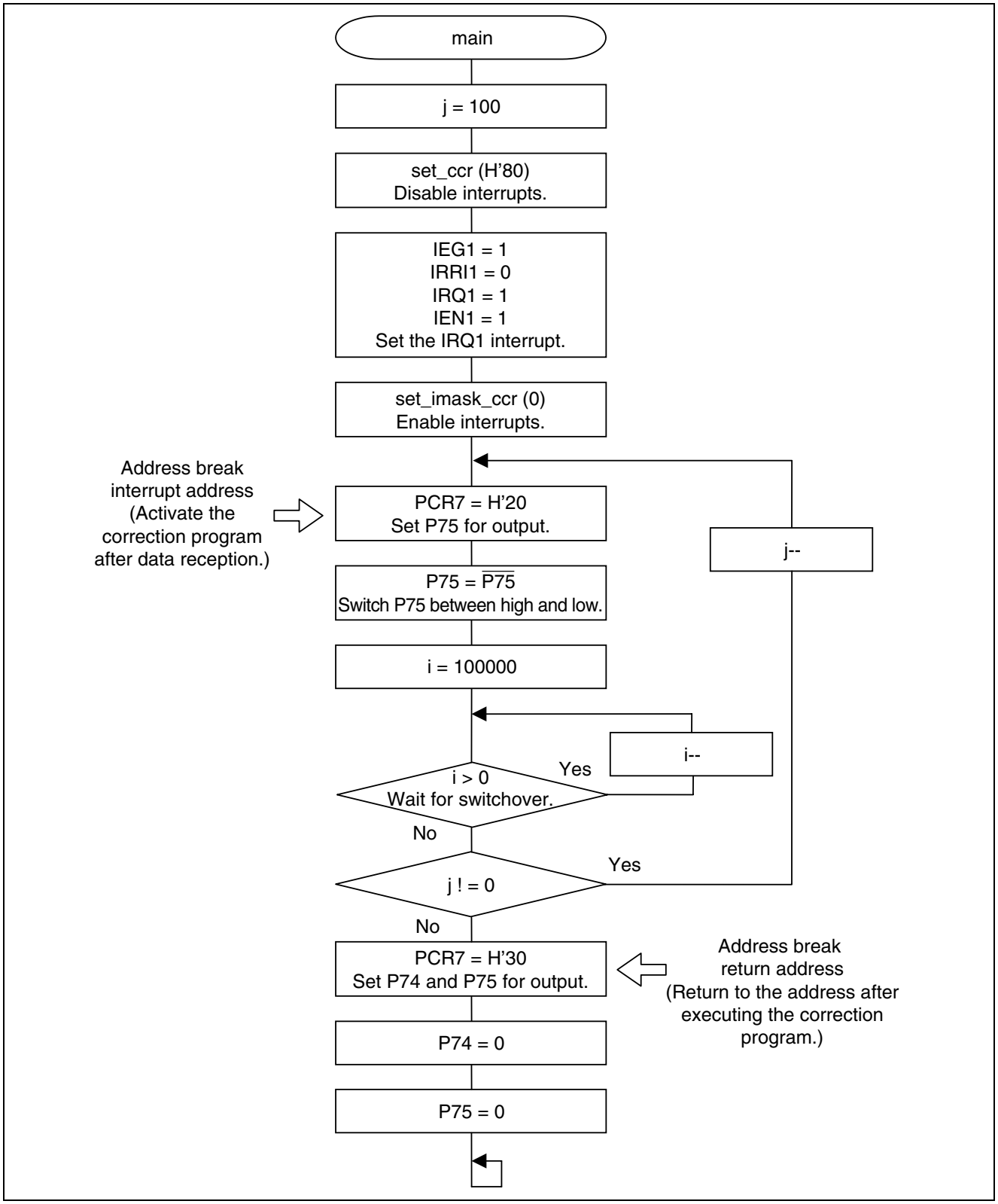
Bit	Bit name	Set value	R/W	Description
5	PCR75	0	W	Setting a PCR7 bit to 1 makes the corresponding pin an output port, while clearing the bit to 0 makes the pin an input port.
4	PCR74	0	W	

— PDR7 Port Data Register 7 Address: H'FFDA

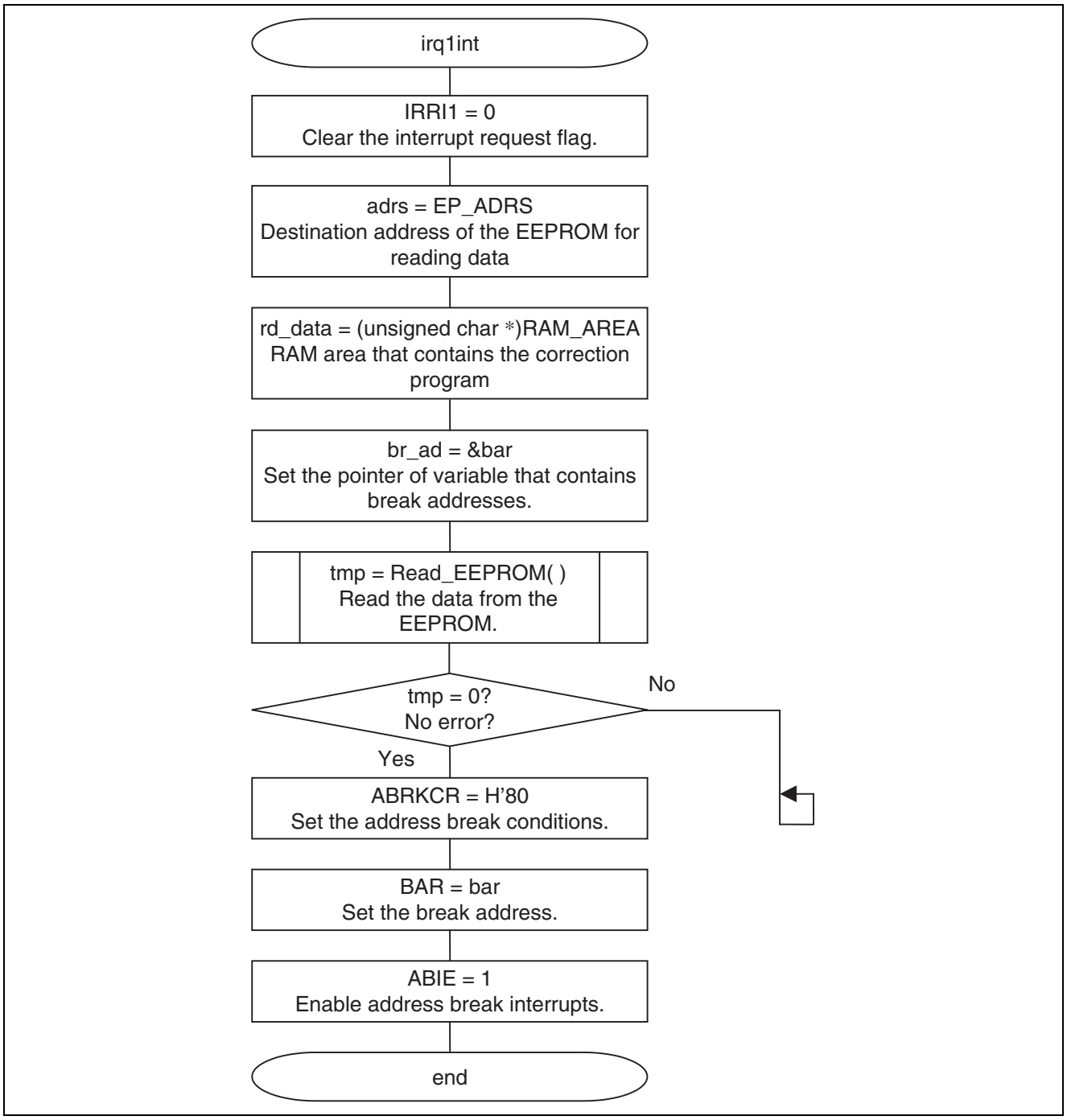
Bit	Bit name	Set value	R/W	Description
5	P75	0	R/W	PDR7 stores output values for general output ports. When PDR7 is read while PCR7 bits are set to 1, the value stored in PDR7 is read. If PDR7 is read while PCR7 bits are cleared to 0, the pin states are read regardless of the value stored in PDR7.
4	P74	0	R/W	

5. Flowchart

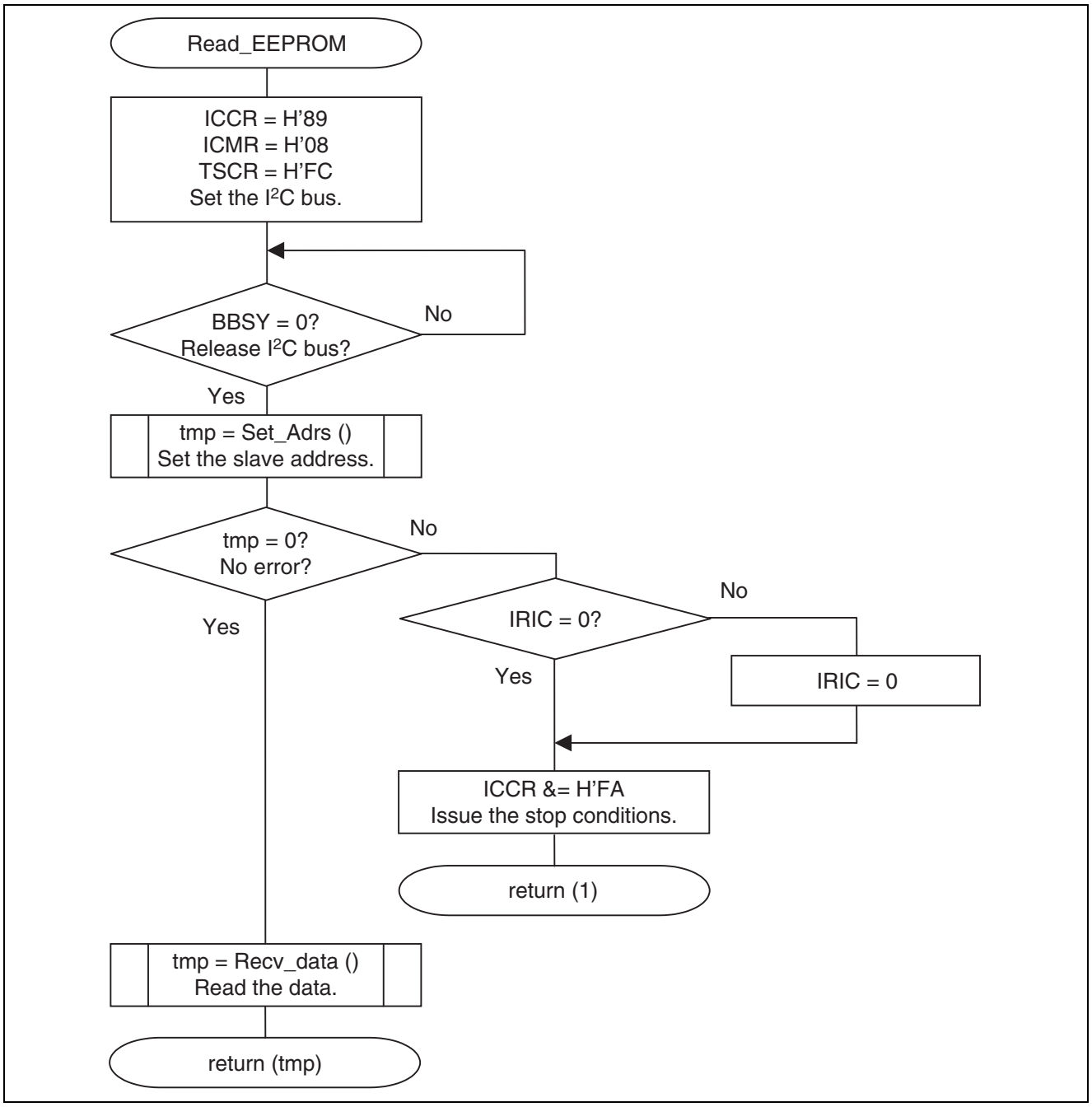
5.1 Main Routine



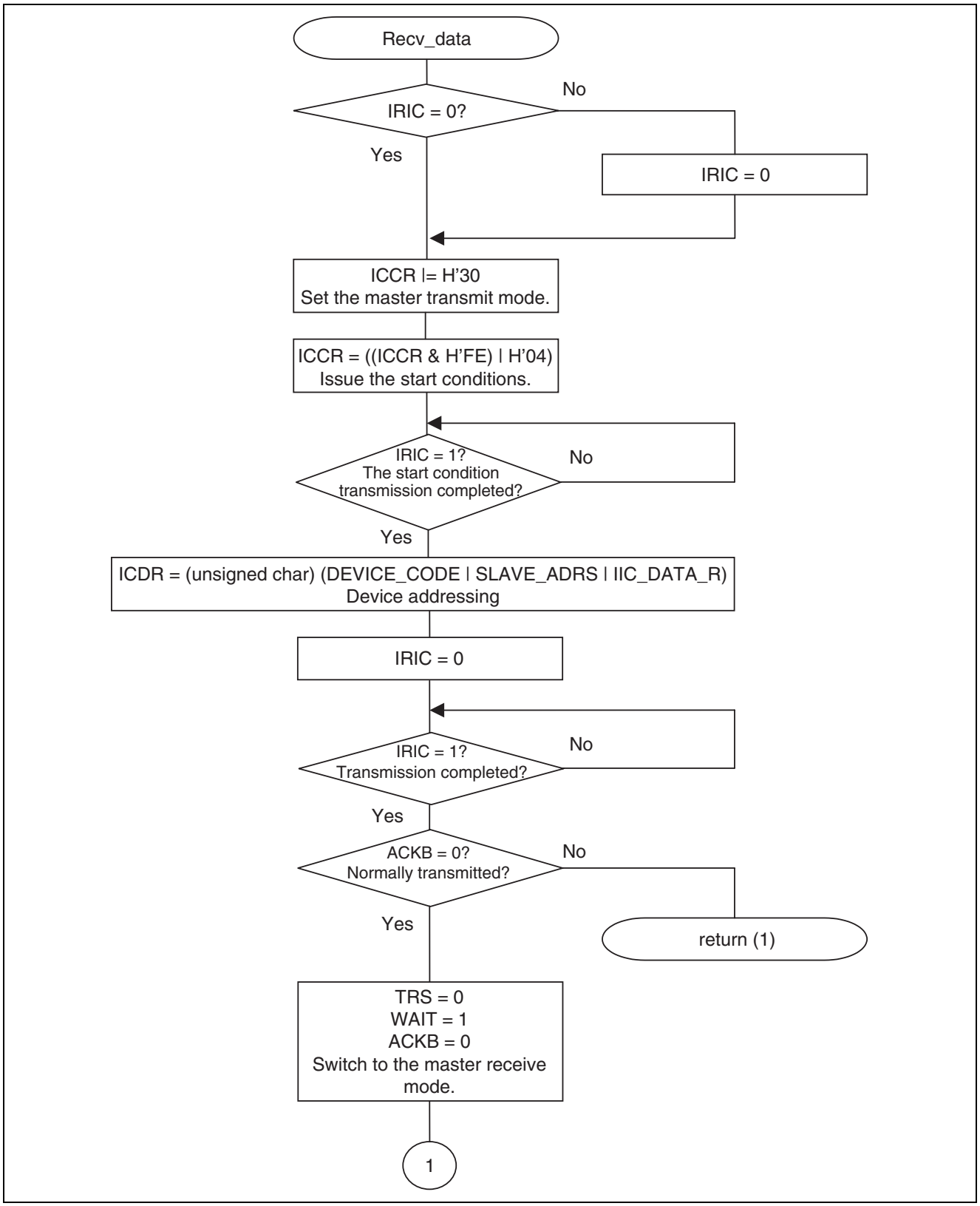
5.2 IRQ1 Interrupt Routine

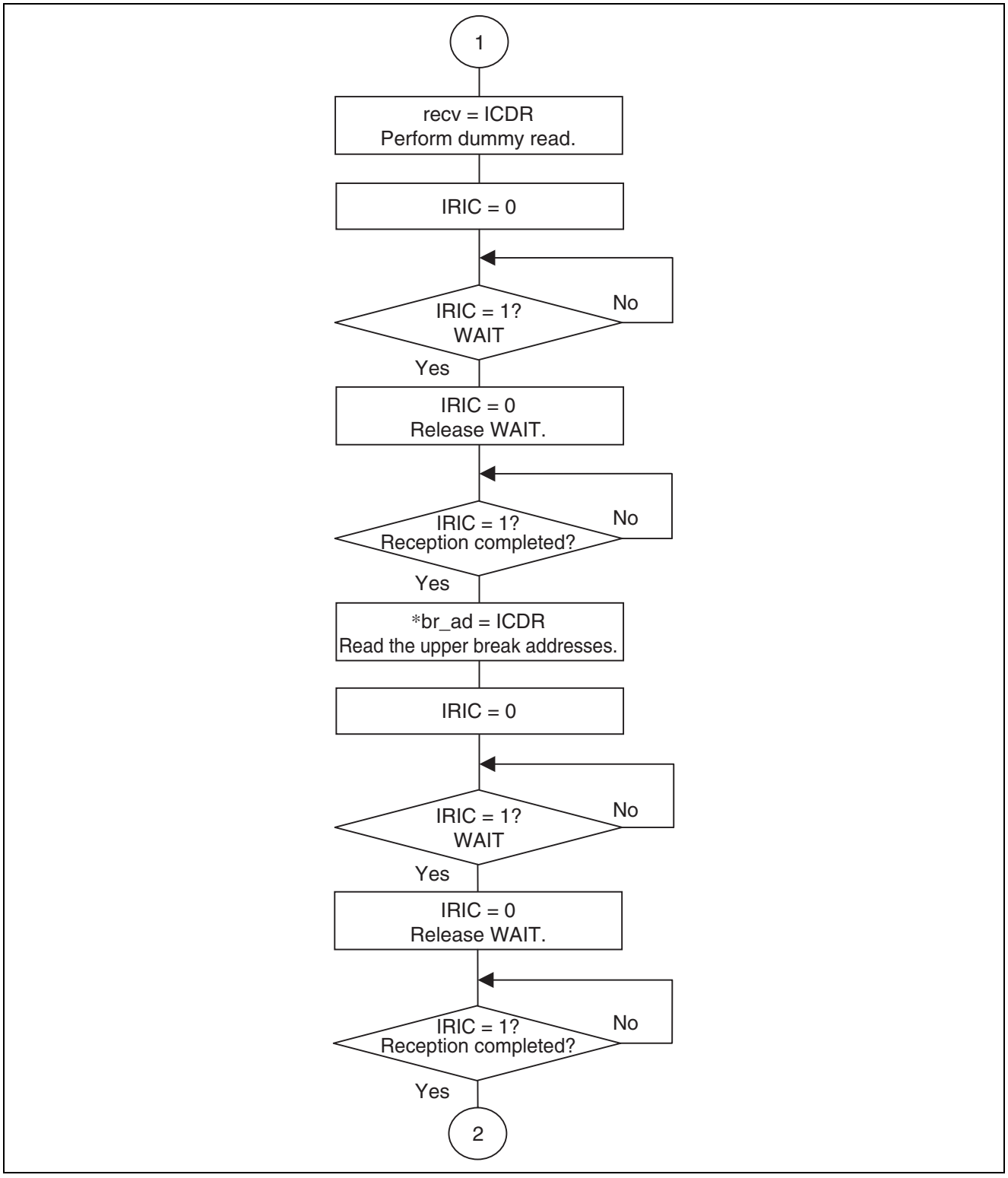


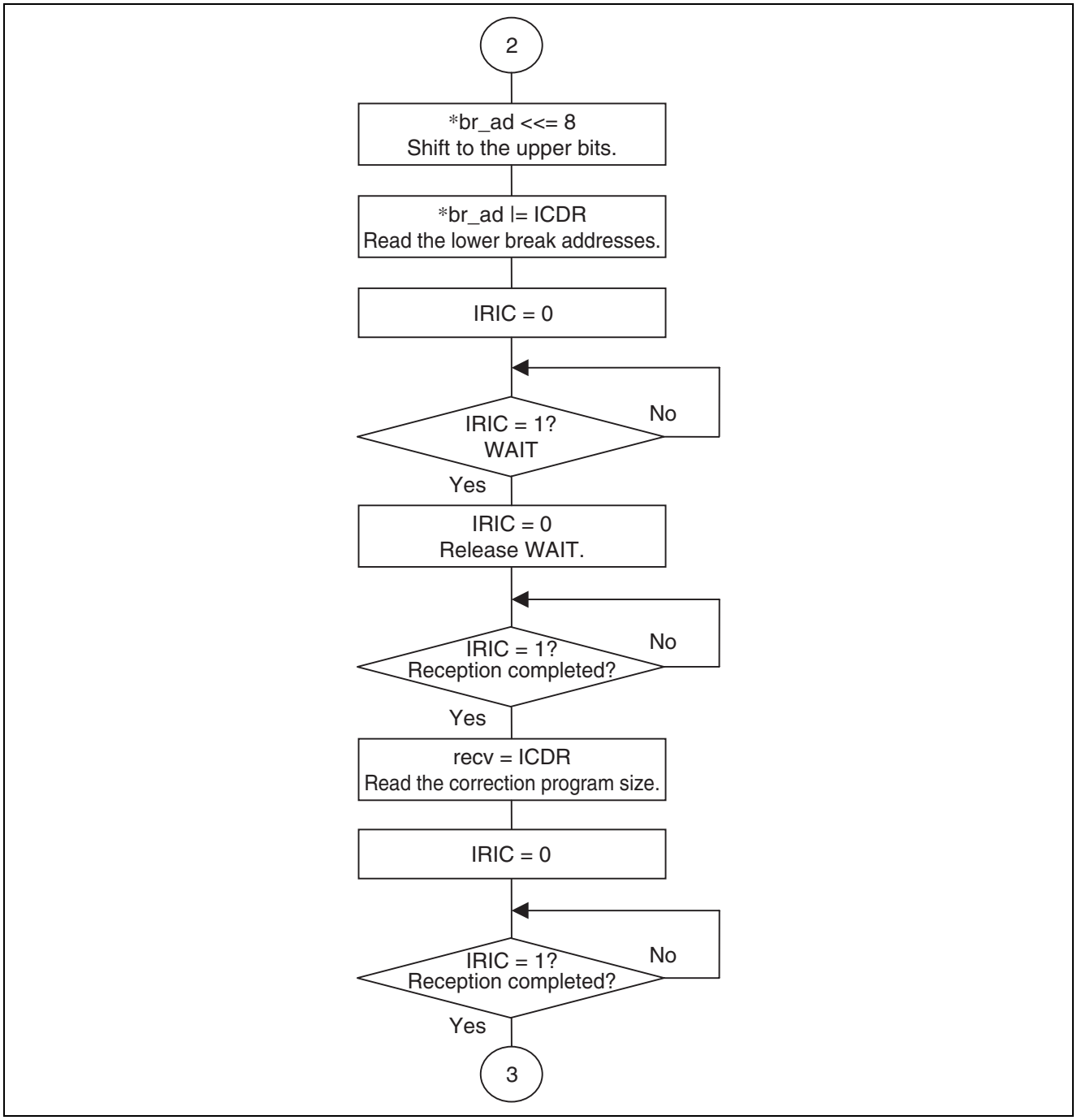
5.3 Download Setting Routine

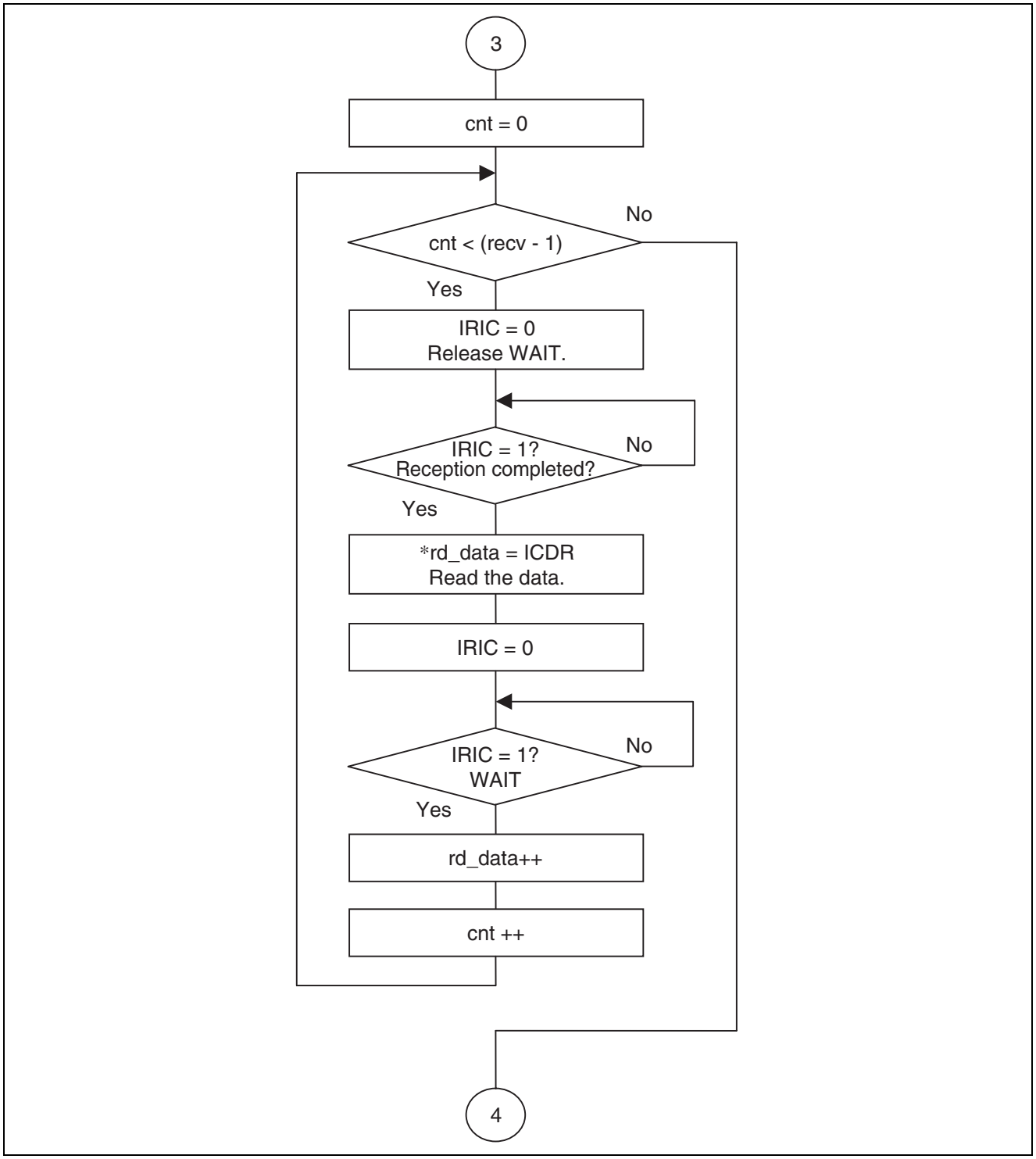


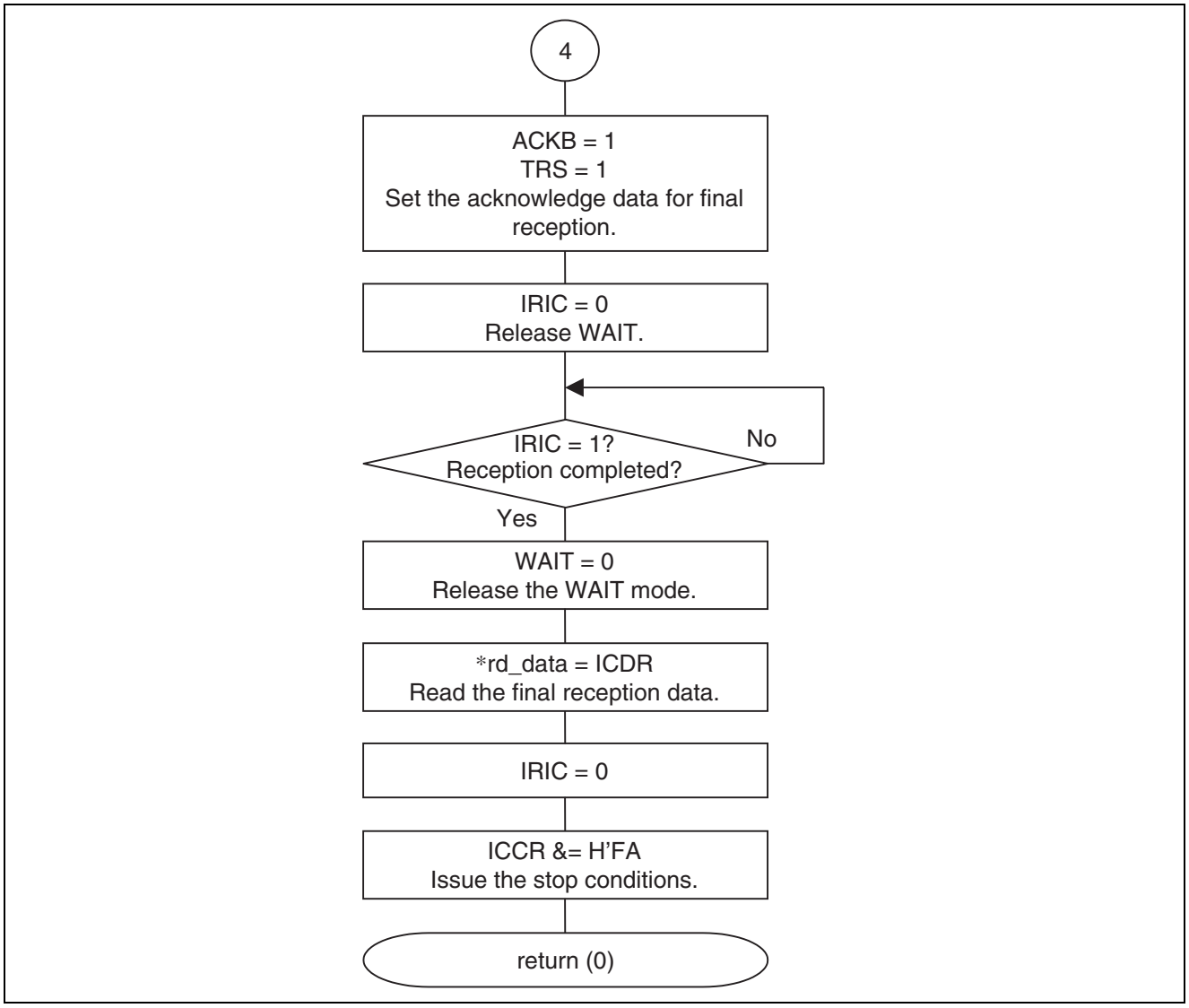
5.4 Download Routine



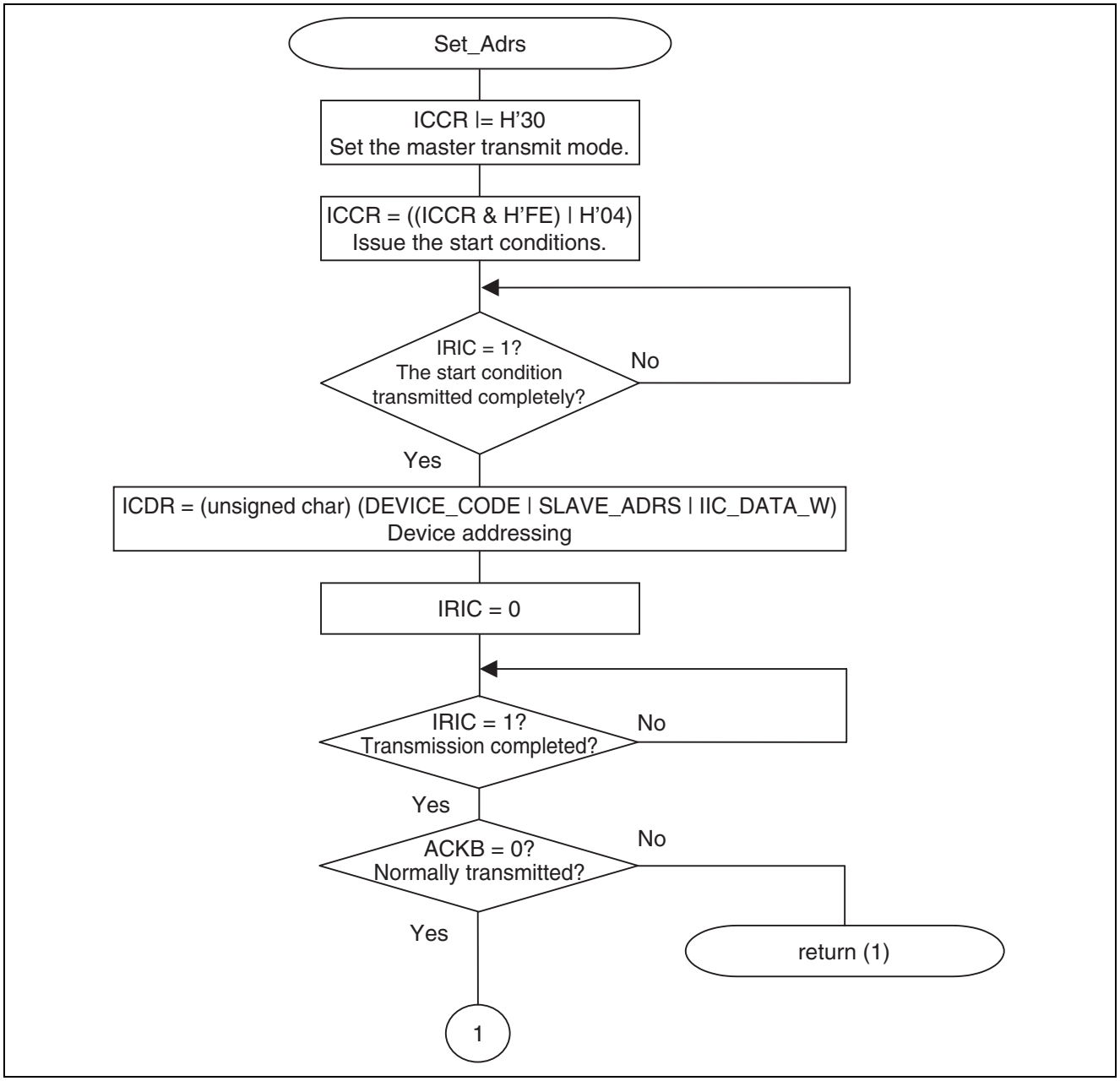

1
recv = ICDR
Perform dummy read.
IRIC = 0
IRIC = 1?
WAIT
No
Yes
IRIC = 0
Release WAIT.
IRIC = 1?
Reception completed?
No
Yes
*br_ad = ICDR
Read the upper break addresses.
IRIC = 0
IRIC = 1?
WAIT
No
Yes
IRIC = 0
Release WAIT.
IRIC = 1?
Reception completed?
No
Yes
2

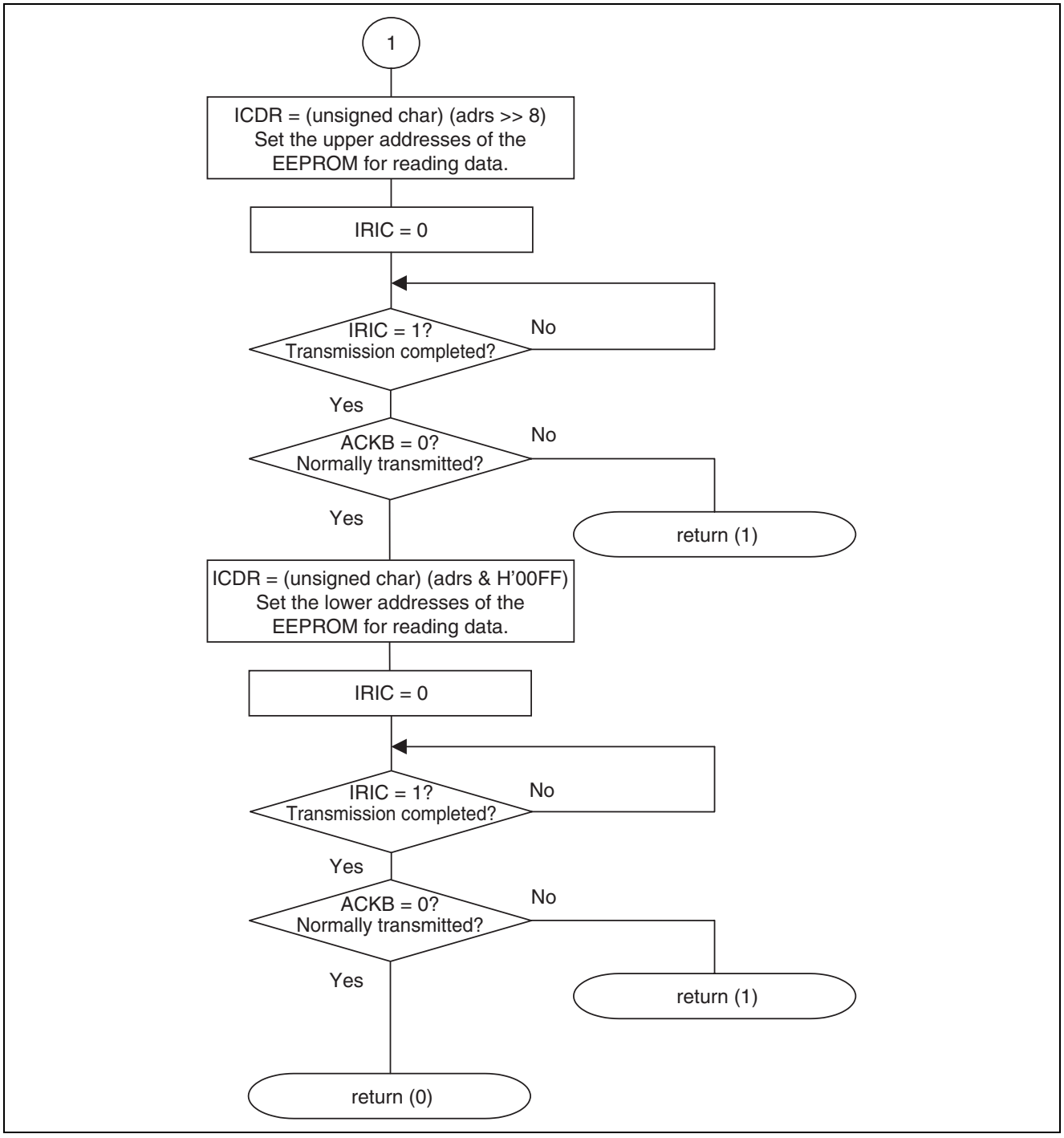

2
*br_ad <<= 8
Shift to the upper bits.
*br_ad |= ICDR
Read the lower break addresses.
IRIC = 0
IRIC = 1?
WAIT
No
Yes
IRIC = 0
Release WAIT.
IRIC = 1?
Reception completed?
No
Yes
recv = ICDR
Read the correction program size.
IRIC = 0
IRIC = 1?
Reception completed?
No
Yes
3


3
cnt = 0
No
cnt < (recv - 1)
Yes
IRIC = 0
Release WAIT.
IRIC = 1?
Reception completed?
No
Yes
*rd_data = ICDR
Read the data.
IRIC = 0
IRIC = 1?
WAIT
No
Yes
rd_data++
cnt ++
4


4
ACKB = 1
TRS = 1
Set the acknowledge data for final reception.
IRIC = 0
Release WAIT.
IRIC = 1?
Reception completed?
No
Yes
WAIT = 0
Release the WAIT mode.
*rd_data = ICDR
Read the final reception data.
IRIC = 0
ICCR &= H'FA
Issue the stop conditions.
return (0)

5.5 Slave Address Setting Routine





- Link address specification

Section name	Address
CV1	H'0000
CV2	H'0018
CV3	H'001E
P	H'0100

6. Program Listing

```
/*****************************************************/
/*                                                   */
/*  H8/300HN Series -H8/3664-                        */
/*  Application Note                                 */
/*                                                   */
/*  'ROM CORRECTION'                                 */
/*                                                   */
/*  External Clock : 16MHz                           */
/*  Internal Clock : 16MHz                           */
/*  Sub Clock      : 32.768kHz                       */
/*                                                   */
/*****************************************************/

#include   <machine.h>

/*****************************************************/
/*  Symbol Definition                                */
/*****************************************************/
struct BIT {
    unsigned char  b7:1;        /* bit7 */
    unsigned char  b6:1;        /* bit6 */
    unsigned char  b5:1;        /* bit5 */
    unsigned char  b4:1;        /* bit4 */
    unsigned char  b3:1;        /* bit3 */
    unsigned char  b2:1;        /* bit2 */
    unsigned char  b1:1;        /* bit1 */
    unsigned char  b0:1;        /* bit0 */
};

#define    ABRKCR     *(volatile unsigned char *)0xFFC8   /* Address Break Control Register   */
#define    ABRKSR_BIT (*(struct BIT *)0xFFC9)             /* Address Break Status Register    */
#define    ABIE       ABRKSR_BIT.b6                       /* Address Break Interrupt Enable   */
#define    BAR        *(volatile unsigned short *)0xFFCA  /* Break Address Register H         */

#define    PCR7       *(volatile unsigned char *)0xFFEA
#define    PCR7_BIT   (*(struct BIT *)0xFFEA)             /* Port Control Register 7          */
#define    PDR7       *(volatile unsigned char *)0xFFDA
#define    PDR7_BIT   (*(struct BIT *)0xFFDA)             /* Port Data Register 7             */
#define    P75        PDR7_BIT.b5                         /* Port Data Register 7 bit5        */
#define    P74        PDR7_BIT.b4                         /* Port Data Register 7 bit4        */

#define    PDR5       *(volatile unsigned char *)0xFFD8
#define    PDR5_BIT   (*(struct BIT *)0xFFD8)             /* Port Data Register 7             */
#define    SCL        PDR5_BIT.b7                         /* IIC serial clock input output    */
#define    SDA        PDR5_BIT.b6                         /* IIC serial data input output     */

#define    ICCR       *(volatile unsigned char *)0xFFC4
#define    ICCR_BIT   (*(struct BIT *)0xFFC4)             /* IIC bus control register         */
#define    TRS        ICCR_BIT.b4                         /* Transmission, reception choice   */
#define    BBSY       ICCR_BIT.b2                         /* Bus busy                         */
#define    IRIC       ICCR_BIT.b1            /* IIC bus interface interrupt, requirement flag */
#define    SCP        ICCR_BIT.b0                     /* start/stop condition prohibition bit */
```

```
#define     ICSR        *(volatile unsigned char *)0xFFC5
#define     ICSR_BIT    (*(struct BIT *)0xFFC5)             /* IIC bus status register          */
#define     ACKB        ICSR_BIT.b0                         /* Acknowledge bit                  */
#define     ICDR        *(volatile unsigned char *)0xFFC6
#define     ICDR_BIT    (*(struct BIT *)0xFFC6)             /* IIC bus data register            */
#define     ICMR        *(volatile unsigned char *)0xFFC7
#define     ICMR_BIT    (*(struct BIT *)0xFFC7)             /* IIC bus mode register            */
#define     WAIT        ICMR_BIT.b6                         /* WAIT insertion bit               */
#define     TSCR        *(volatile unsigned char *)0xFFFC
#define     TSCR_BIT    (*(struct BIT *)0xFFFC)             /* Timer serial control register    */

#define     PMR1    *(volatile unsigned char *)0xFFE0       /* Port Mode Register 1             */
#define     PMR1_BIT    (*(struct BIT *)0xFFE0)             /* Port Mode Register 1             */
#define     IRQ1        PMR1_BIT.b5                         /* P15/IRQ1 Select                  */

#define     IEGR1_BIT   (*(struct BIT *)0xFFF2)             /* Interrupt Edge Select Register 1 */
#define     IEG1        IEGR1_BIT.b1                        /* IEG1 Edge Select                 */
#define     IENR1_BIT   (*(struct BIT *)0xFFF4)             /* Interrupt Enable Register 1      */
#define     IEN1        IENR1_BIT.b1                        /* IEN1 Interrupt Enable            */
#define     IRR1_BIT    (*(struct BIT *)0xFFF6)             /* Interrupt Request Register 1     */
#define     IRRI1       IRR1_BIT.b1                         /* IRRI1 Interrupt Request Register */

#define     DEVICE_CODE     0xA0                            /* EEPROM DEVICE CODE:1010    */
#define     SLAVE_ADRS      0x00                            /* SLAVE ADRS:000             */
#define     IIC_DATA_W      0x00                            /* WRITE_DATA:0               */
#define     IIC_DATA_R      0x01                            /* READ_DATA:1                */

#define     EP_ADRS         0x0000                          /* The first address to read EEPROM
*/
#define     RAM_AREA        0xFC00                          /* Modification program storage address
*/

#pragma interrupt (irq1int)
/*****************************************************/
/*  Function define                                  */
/*****************************************************/
void        main ( void );
unsigned char Read_EEPROM(unsigned short adrs, unsigned char *rd_data, unsigned short *br_ad);
unsigned char Recv_data(unsigned char *rd_data, unsigned short *br_ad);
unsigned char Set_Adrs(unsigned short adrs);
void irq1int(void);

/*****************************************************/
/*  Vector Address                                   */
/*****************************************************/
#pragma section    V1                           /* VECTOR SECTION SET          */
void (*const VEC_TBL1[])(void) = {
    main
};

#pragma section    V2                           /* VECTOR SECTION SET          */
void (*const VEC_TBL2[])(void) = {
    (void *)RAM_AREA                            /* Address Break               */
};
```

```
#pragma section    V3                          /* VECTOR SECTION SET          */
void (*const VEC_TBL3[])(void) = {
    irq1int                                    /* IRQ1 interrupt              */
};

#pragma entry main(sp=0xFF80)
#pragma section                                /* P                           */
/*****************************************************/
/*  Main Program                                     */
/*****************************************************/
void main ( void )
{
    volatile unsigned long i;
    unsigned char j=100;

    set_ccr(0x80);                             /* Initialize CCR/Interrupt Disable       */

    IEG1 = 1;                                  /* Initialize IRQ1 Terminal Input Edge    */
    IRRI1 = 0;                                 /* Initialize IRQ1 Interrupt Request Flag */
    IRQ1 = 1;
    IEN1 = 1;                                  /* IRQ1 Interrupt Enable                  */

    set_imask_ccr(0);

    do{
        PCR7 = 0x20;                           /* P75 output set                         */
        P75 = ~P75;                            /* LED2 ON/OFF                            */
        for(i=100000; i>0; i--);               /* wait                                   */
    }while(j--);

    PCR7 = 0x30;                               /* P74, P75 output set                    */
    P74 = 0;                                   /* LED2 ON                                */
    P75 = 0;                                   /* LED3 ON                                */
    while(1);
}

/*********************************************************/
/*  IRQ1 Interrupt                                       */
/*********************************************************/
void irq1int( void )
{
    unsigned short adrs, bar, *br_ad;
    unsigned char tmp, *rd_data;

    IRRI1 = 0;                                 /* Initialize IRQ1 Interrupt Request Flag      */

    adrs = EP_ADRS;                                  /* initialize EEPROM read address   */
    rd_data = (unsigned char *)RAM_AREA;             /* modification program start address */
    br_ad = &bar;                                    /* Break Address            */
    tmp = Read_EEPROM(adrs, rd_data, br_ad);         /* modification program read  */
    if(tmp != 0)
        while(1);
```

```
    ABRKCR = 0x80;                          /* A setup of Address Break condition  */
    BAR = bar;                              /* A setup of Address Break            */
    ABIE = 1;                               /* A setup of Address Break Enable     */
}

unsigned char Read_EEPROM(unsigned short adrs, unsigned char *rd_data, unsigned short *br_ad)
{
    unsigned char tmp;

    ICCR = 0x89;                            /* ICE=1, ACKE=1, SCP=1                */
    ICMR = 0x08;                            /* CKS0=1                              */
    TSCR = 0xFC;                            /* trace rate 400kHz                   */

    while(BBSY != 0);                       /* Bus Busy?                           */

    tmp = Set_Adrs(adrs);                   /* Address set                         */
    if(tmp != 0){
        if(IRIC == 1)
            IRIC = 0;
        ICCR &= 0xFA                        /* stop condition                      */
        return(1);
    }

    tmp = Recv_data(rd_data, br_ad);        /* modification program data read        */
    return(tmp);
}

unsigned char Recv_data(unsigned char *rd_data, unsigned short *br_ad)
{
    unsigned char recv, cnt=0;

    if(IRIC ==1)
        IRIC = 0;
    ICCR |= 0x30;                           /* master trace mode (MST=1, TRS=1)    */
    ICCR = ((ICCR & 0xFE) | 0x04);          /* start condition                     */
    while(IRIC == 0);

    ICDR = (unsigned char)(DEVICE_CODE | SLAVE_ADRS | IIC_DATA_R);
                                            /* slave address set                   */
    IRIC = 0;
    while(IRIC == 0);                       /* trace OK?                           */

    if(ACKB != 0)                           /* ACK?                                */
        return(1);

    TRS = 0;                                /* Master Receive                      */
    WAIT = 1;                               /* WAIT mode ON                        */
    ACKB = 0;                               /* set ACK = 0                         */

    recv = ICDR;                            /* dummy read                          */
    IRIC = 0;
    while(IRIC == 0);                       /* dummy recv end?                     */

    IRIC = 0;                               /* WAIT OFF                            */
```

```
    while(IRIC == 0);                       /* dummy recv OK?                   */

    *br_ad = ICDR;                          /* Break high Address receive       */
    IRIC = 0;
    while(IRIC == 0);                       /* receive end?                     */

    IRIC = 0;                               /* WAIT OFF                         */
    while(IRIC == 0);                       /* receive OK?                      */

    *br_ad <<= 8;                           /* high bit shift                   */
    *br_ad |= ICDR;                         /* Break low Address receive        */
    IRIC = 0;
    while(IRIC == 0);                       /* receive end?                     */

    IRIC = 0;                               /* WAIT OFF                         */
    while(IRIC == 0);                       /* receive OK?                      */

    recv = ICDR;                            /* modification program size receive   */
    IRIC = 0;
    while(IRIC == 0);                       /* receive end?                     */

    while(cnt < (recv - 1)){                /* (size - 1) LOOP                  */
        IRIC = 0;                           /* WAIT OFF                         */
        while(IRIC == 0);                   /* receive OK?                      */

        *rd_data = ICDR;                    /* modification program receive        */
        IRIC = 0;
        while(IRIC == 0);                   /* receive end?                     */

        rd_data++;
        cnt++;
    }

    ACKB = 1;                               /* ACK=1                            */
    TRS = 1;                                /* trace mode set                   */
    IRIC = 0;                               /* WAIT OFF                         */
    while(IRIC == 0);                       /* receive OK?                      */

    WAIT = 0;                               /* WAIT mode OFF                    */
    *rd_data = ICDR;                        /* Last data receive                */
    IRIC = 0;
    ICCR &= 0xFA;                           /* stop condition                   */

    return(0);
}

unsigned char Set_Adrs(unsigned short adrs)
{
    ICCR |= 0x30;                           /* master trace mode (MST=1, TRS=1)  */
    ICCR = ((ICCR & 0xFE) | 0x04);          /* start condition                  */
    while(IRIC == 0);

    ICDR = (unsigned char)(DEVICE_CODE | SLAVE_ADRS | IIC_DATA_W);
                                            /* slave address set                */
```

```
    IRIC = 0;
    while(IRIC == 0);                       /* trace end?                   */

    if(ACKB != 0){                          /* ACK OK?                      */
        return(1);
    }

    ICDR = (unsigned char)(adrs >> 8);      /* high address set             */
    IRIC = 0;
    while(IRIC == 0)                        /* trace end?                   */

    if(ACKB != 0){                          /* ACK OK?                      */
        return(1);
    }

    ICDR = (unsigned char)(adrs & 0x00FF);  /* low address set              */
    IRIC = 0;
    while(IRIC == 0);                       /* trace end?                   */

    if(ACKB != 0){                          /* ACK OK?                      */
        return(1);
    }

    return(0);
}
```

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Dec.20.03	—	First edition issued

Keep safety first in your circuit designs!

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

Notes regarding these materials

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