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.

M32C/85 Group

Flash Memory Version CPU Rewrite Mode (EW0 Mode) Sample

1. Abstract

This application note presents an example method for using CPU rewrite mode (EW0 mode) in the flash memory version of microcomputers.

2. Introduction

The explanation of this issue is applied to the following condition:
Applicable MCU: M32C/85 Group

This program can also be used when operating other microcomputers within the M16C family, provided they have the same SFR (Special Function Registers) as the M32C/85 microcomputers. However, some functions may have been modified.
Refer to the User's Manual for details. Use functions covered in this Application Note only after careful evaluation.

3. Explanation of Example Usage

Features of EW0 mode:
In EW0 mode, the CPU rewrite program is transferred into the RAM, and by issuing programming and erasing commands from the CPU rewrite program in the RAM, the user ROM and the data areas can be rewritten. Since while in EW0 mode the CPU continues operating even during a programming or erasing operation, peripheral function interrupts can be accepted during a programming or erasing operation providing that the vectors for those interrupts and the interrupt service routines are located in the RAM.

3.1 CPU Rewrite Mode (EW0 Mode) Execution Flow

Figure 1 shows CPU Rewrite Mode (EW0 Mode) Execution Flow.

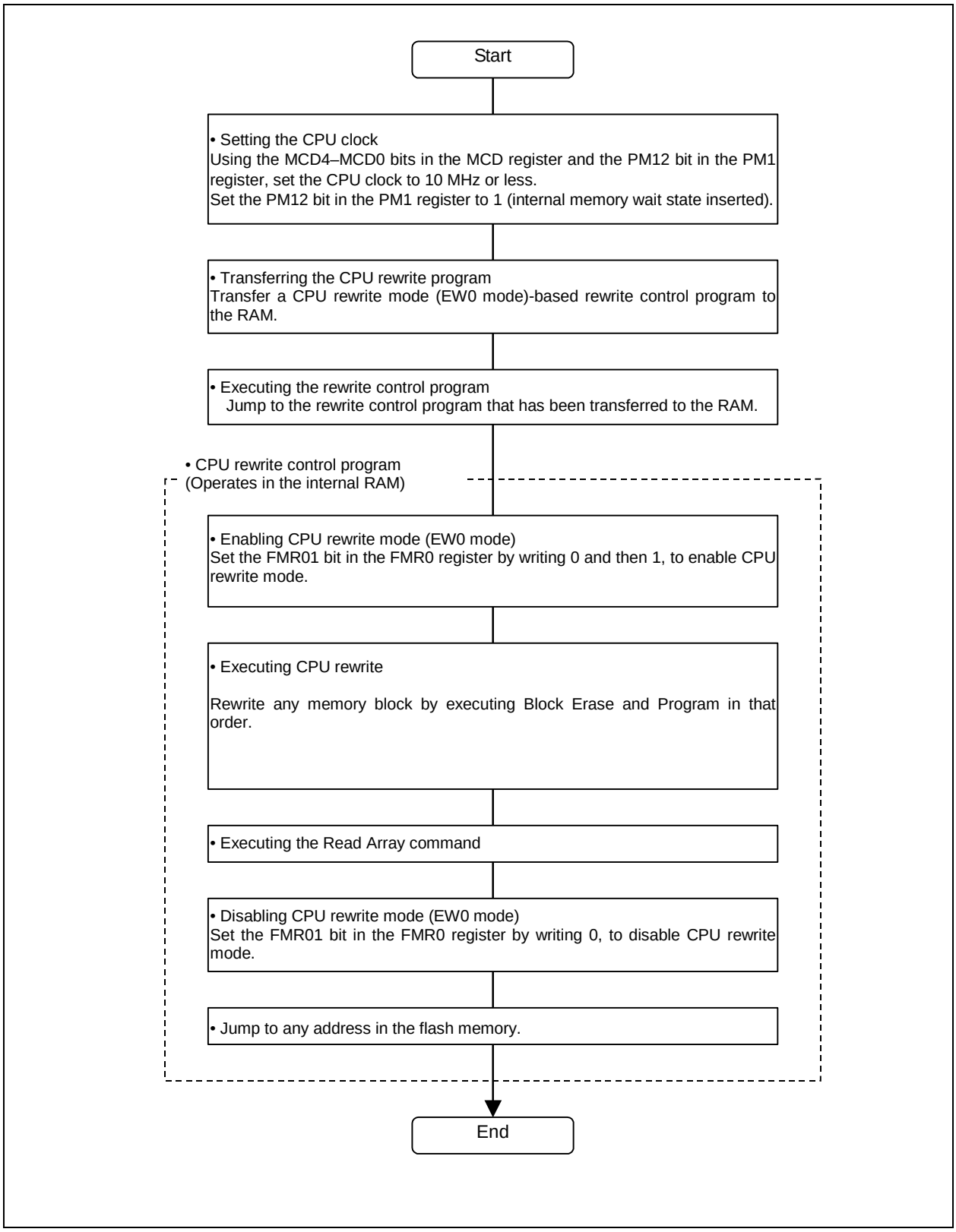
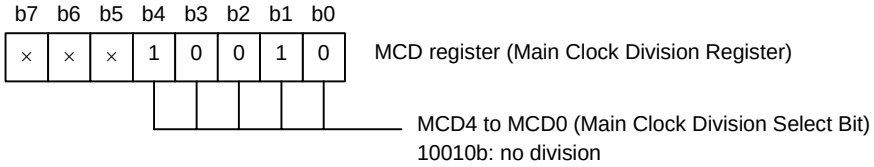


Figure 1. CPU Rewrite Mode (EW0 Mode) Execution Flow

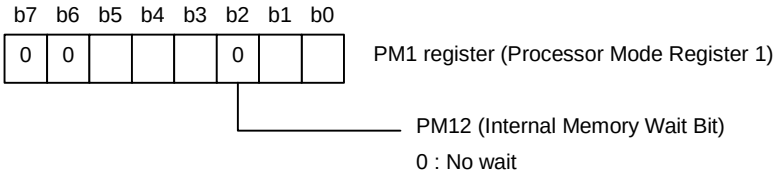
3.2 Set Up Procedure

3.2.1 Setting CPU Clock

(1) Setting the main clock divide-by-N value



(2) Setting internal memory wait states



3.2.2 Transferring the CPU Rewrite Control Program into RAM

The CPU rewrite control program needs to be run in RAM. Here, the following explains an example for transferring the CPU rewrite control program from the ROM area in which it is stored beginning with the address 0FD0000h to an area in RAM.

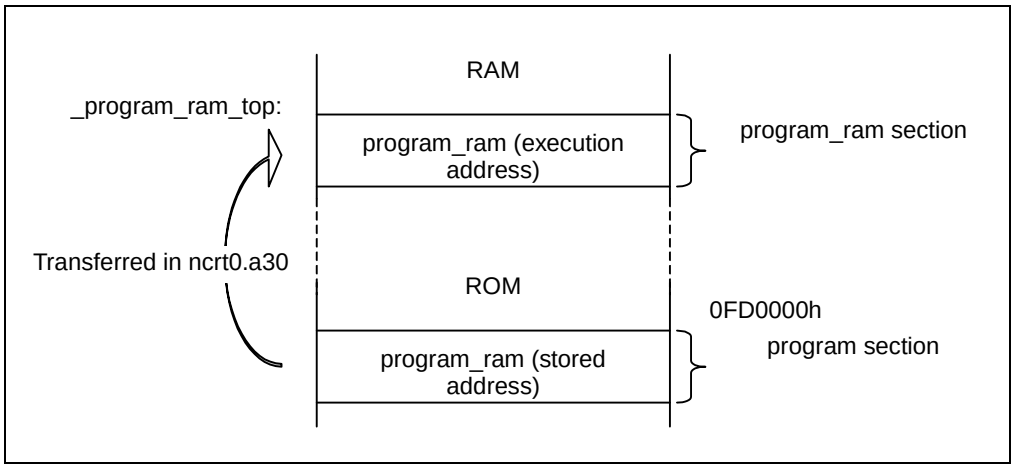


Figure 2. Program Location

(1) Change the Section Name.

Add a section name “program_ram,” and locate the program to be run in RAM in that section. To relocate the program from the program section to the program_ram section, write a process as shown below.

```
void main(void)
{
      /* This program part is located in the program section */
}
```

```
/* The program part following the #pragma SECTION declaration is located in the program_ram section */
#pragma SECTION program program_ram
void low_power(void)
{
      /* This program part is located in the program_ram section */
}
```

(2) Changing sect308.inc

Add the program_ram section to sect308.inc. In the example here, it is located after the heap section. Note also that the program_ram_top label is used when transferring the program.

```
;------------------------------------------------------------
;       heap section
;------------------------------------------------------------
.if __HEAP__ != 1
      .section  heap,DATA
heap_top:
      .blkb     HEAPSIZE
.endif
```

```
;------------------------------------------------------------
; RAM program area
;------------------------------------------------------------
      .section  program_ram,ALIGN
_program_ram_top:
      .glb      _program_ram_top
```

Add here.

(3) Transferring the CPU rewrite control program

Add a process for transferring the program into RAM in the startup routine (ncrt0.a30).

```
;====================================================================
; Initialize standard I/O
;--------------------------------------------------------------------
.if __STANDARD_IO__ != 1
      .glb      _init
      .call     _init,G
      jsr.a     _init
.endif
```

```
;====================================================
; Program Ram initialize
;   _from_addr is defined by as308 option "-D_from_addr=0fd000h"
;------------------------------------------------------------
     BCOPY    _from_addr,_program_ram_top,program_ram
;
```

Add here.

```
;====================================
; Call main() function
;------------------------------------------------------------------
        ldc     #0h,fb    ; for debuger

        .glb    _main
        jsr.a   _main
```

(4) Specifying the Program Storage Location

To run the program transferred into RAM, it is necessary to specify in the linker (ln308) that the program storage address (in ROM) and execution address (in RAM) be located separately.

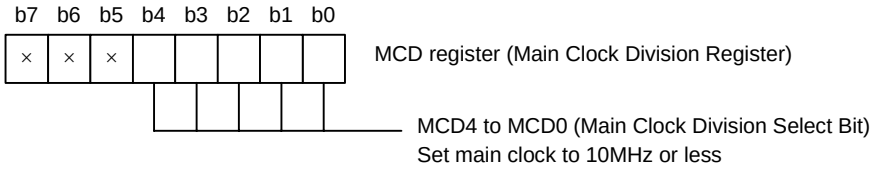
```
ln308 –LOC program_ram=0FD0000
```

In the above option, the program_ram section is stored beginning with the address 0FD0000h.

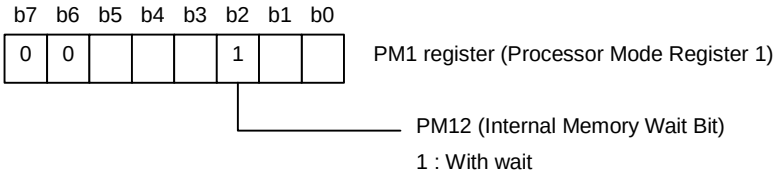
3.2.3 Processing in the CPU Rewrite Control Program

(1) Set the CPU clock to 10MHz or less.

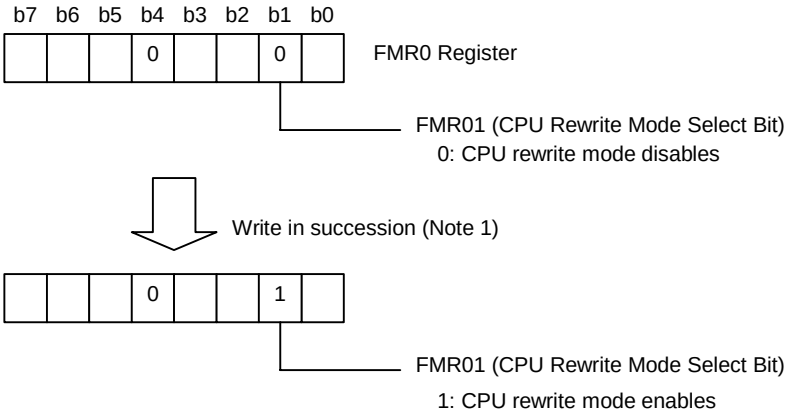
Setting the main clock divide-by-N value



Setting internal memory wait states



(2) Enable the CPU rewrite mode.

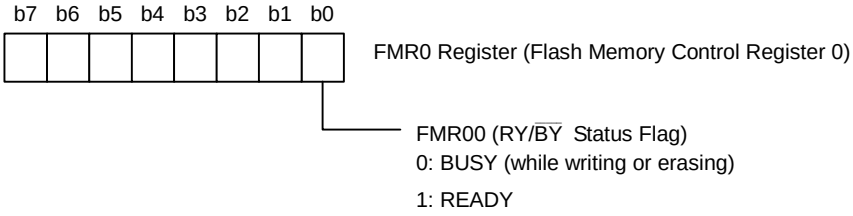


Note: To set the FMR01 bit to “1”, write “0” in 8-bit unit and then “1” to the FMR01 bit in succession. Make sure no interrupts or DMA transfers occur before the CPU writes “1” after writing “0”.
Make sure writes to the FMR01 bit is performed in other than the internal flash memory.
Also make sure this write operation is performed while the $\overline{\text{NMI}}$ pin is in the high state.

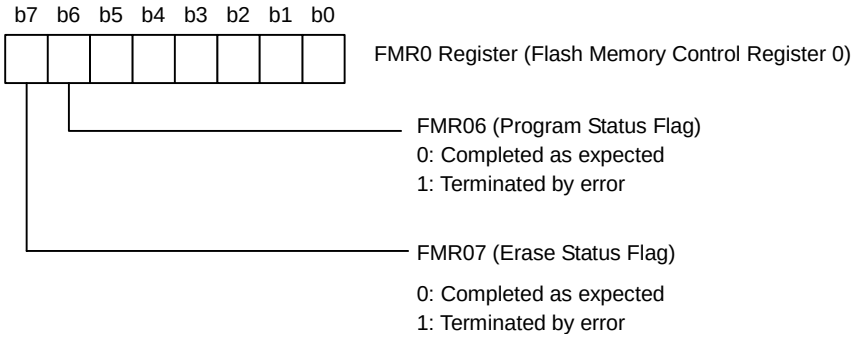
(3) Block erase processing

- Executing the Block Erase command
  Write “0020h” and then “00d0h” in succession to the most significant address of the memory block to be block-erased.

- Waiting for Block Erase to complete
  Wait until the FMR00 bit in the FMR0 register is set to “1” (ready).



- Status check
  Check the FMR06 and FMR07 bits in the FMR0 register to see if an erase error has occurred. If an error is found to have occurred during the erase operation, write “0050h” (Clear Status command) to the address to which the Block Erase command was written, to stop CPU rewrite processing.



(4) Programming process
Program the entire area of the relevant memory block one word at a time, by following the procedure described below.

- Executing the Program command
  Write “0040h” (Program command) and then the program data to the address to be programmed.

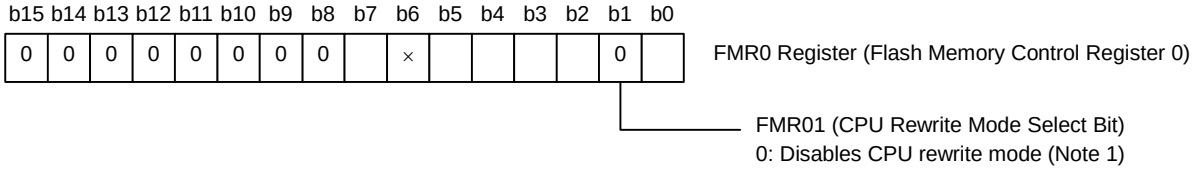
- Waiting for Program to complete
  Wait until the FMR00 bit in the FMR0 register is set to “1” (ready).

- Status check
  Check the FMR06 and FMR07 bits in the FMR0 register to see if a programming error has occurred. If an error is found to have occurred during the programming operation, write “0050h” (Clear Status command) to the address to which the Program command was written, to stop CPU rewrite processing.

(5) Disable CPU rewrite mode.

- Executing the Read Array command
  Write “00FFh” (Read Array command) to the most significant address of the relevant memory block.

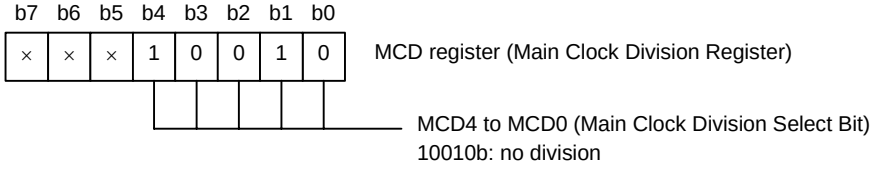
- Disable CPU rewrite mode



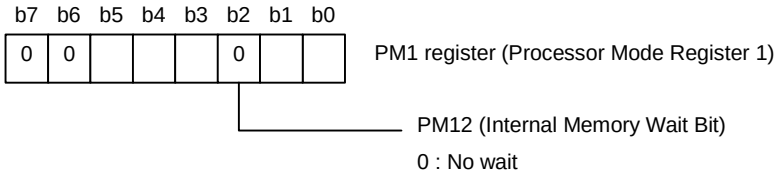
Note 1: To change a FMR01 bit setting from “1” to “0”, enter read array mode to write to addresses 0057h in 16-bit unit. Write “00h” into 8 high-order bits.

(6) Restore the CPU clock to the original one.

- Setting the main clock divide-by-N value



- Setting internal memory wait states



(7) Return to the program in the flash memory.

3.3 Precautions in CPU Rewrite Mode (EW0 Mode)

The following describes the precautions to be observed when using CPU rewrite mode (EW0 mode). (Please consult the manual to get the latest information.)

(1) Operating Speed

Set the MCD register to CPU clock frequency of 10 MHz or less before entering CPU rewrite mode (EW mode 0 or EW mode 1). Also, set the PM12 bit in the PM1 register to "1" (wait state).

(2) Prohibited Instructions

The following instructions cannot be used in EW mode 0 because the CPU tries to read data in the flash memory: the UND instruction, INTO instruction, JMPS instruction, JSRS instruction, and BRK instruction.

(3) Interrupts (EW Mode 0)

- To use interrupts having vectors in a relocatable vector table, the vectors must be relocated to the RAM area.
- The $\overline{\text{NMI}}$ and watchdog timer interrupts are available since the FMR0 and FMR1 registers are forcibly reset when either interrupt occurs. Allocate the jump addresses for each interrupt routine to the fixed vector table. Flash memory rewrite operation is aborted when the $\overline{\text{NMI}}$ or watchdog timer interrupt occurs. Execute the rewrite program again after exiting the interrupt routine.
- The address match interrupt is not available since the CPU tries to read data in the flash memory.

(4) Interrupts (EW Mode 1)

- Do not acknowledge any interrupts with vectors in the relocatable vector table or address match interrupt during the auto program or auto erase period.
- Do not use the watchdog timer interrupt.
- The $\overline{\text{NMI}}$ interrupt is available since the FMR0 and FMR1 registers are forcibly reset when either interrupt occurs. Allocate the jump address for the interrupt routine to the fixed vector table. Flash memory rewrite operation is aborted when the $\overline{\text{NMI}}$ interrupt occurs. Execute the rewrite program again after exiting the interrupt routine.

(5) How to Access

To set the FMR01, FMR02 or FMR11 bit to "1", set to "1" in 8-bit units immediately after setting to "0". Do not generate an interrupt or a DMA transfer between the instruction to set the bit to "0" and the instruction to set the bit to "1". Set the bit while a high-level ("H") signal is applied to the $\overline{\text{NMI}}$ pin. To change the FMR01 bit from "1" to "0", enter read array mode first, and write into address 0057h in 16-bit units. Eight high-order bits must be set to "00h".

(6) Rewriting in the User ROM Area (EW Mode 0)

If the supply voltage drops while rewriting the block where the rewrite control program is stored, the flash memory cannot be rewritten because the rewrite control program is not rewritten as expected. If this error occurs, rewrite the user ROM area while in standard serial I/O mode or parallel I/O mode.

(7) Rewriting in the User ROM Area (EW Mode 1)

Do not rewrite the block where the rewrite control program is stored.

(8) DMA Transfer

In EW mode 1, do not generate a DMA transfer while the FMR00 bit in the FMR0 register is set to “0” (busy-programming or erasing).

(9) Writing Command and Data

Write commands and data to even addresses in the user ROM area.

(10) Wait Mode

When entering wait mode, set the FMR01 bit to "0" (CPU rewrite mode disabled) before executing the WAIT instruction.

(11) Stop Mode

When entering stop mode, the following settings are required:

- Set the FMR01 bit to "0" (CPU rewrite mode disabled). Disable a DMA transfer before setting the CM10 bit to “1” (stop mode).
- Execute the instruction to set the CM10 bit to "1" (stop mode) and then the JMP.B instruction.

e.g.,

```
    BSET 0, CM1; Stop mode
    JMP.B L1
L1:
    Program after exiting stop mode
```

(12) Low-Power Consumption Mode and On-Chip Oscillator Low-Power Consumption Mode

If the CM05 bit is set to "1" (main clock stopped), do not execute the following commands:

- Program
- Block erase
- Erase all unlocked blocks
- Lock bit program
- Read lock bit status

4. Sample Programming Code

The following shows an example program for using CPU rewrite mode (EW0 mode) to back up the internal RAM (addresses 1800h–3FFFh) to the data block (addresses F80000h–F8FFFFh) as triggered by an $\overline{\text{INT0}}$ interrupt request generated. In this example program, block 12 is block-erased and then programmed (to save the RAM area).

Memory map:

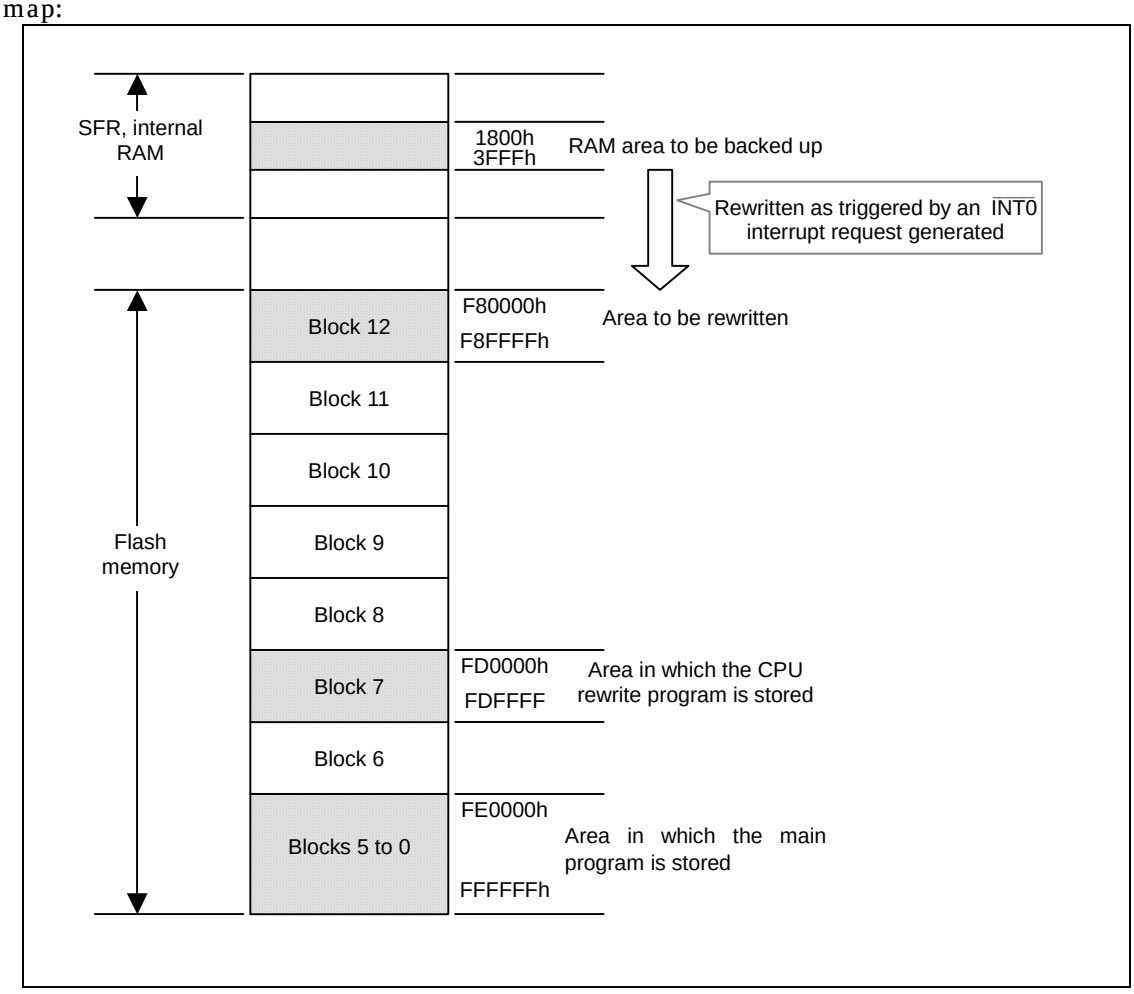


Figure 3. Memory Map

Operating Conditions:

(1) VCC1=VCC2=5V

(2) XIN=8MHz, PLL= Multiply-by-4, f1=32MHz

(During CPU rewrite mode, the device must be operated with 10 MHz or less by setting the relevant bits in the MCD register and the PM12 bit in the PM1 register.)

4.1 Processing Flow

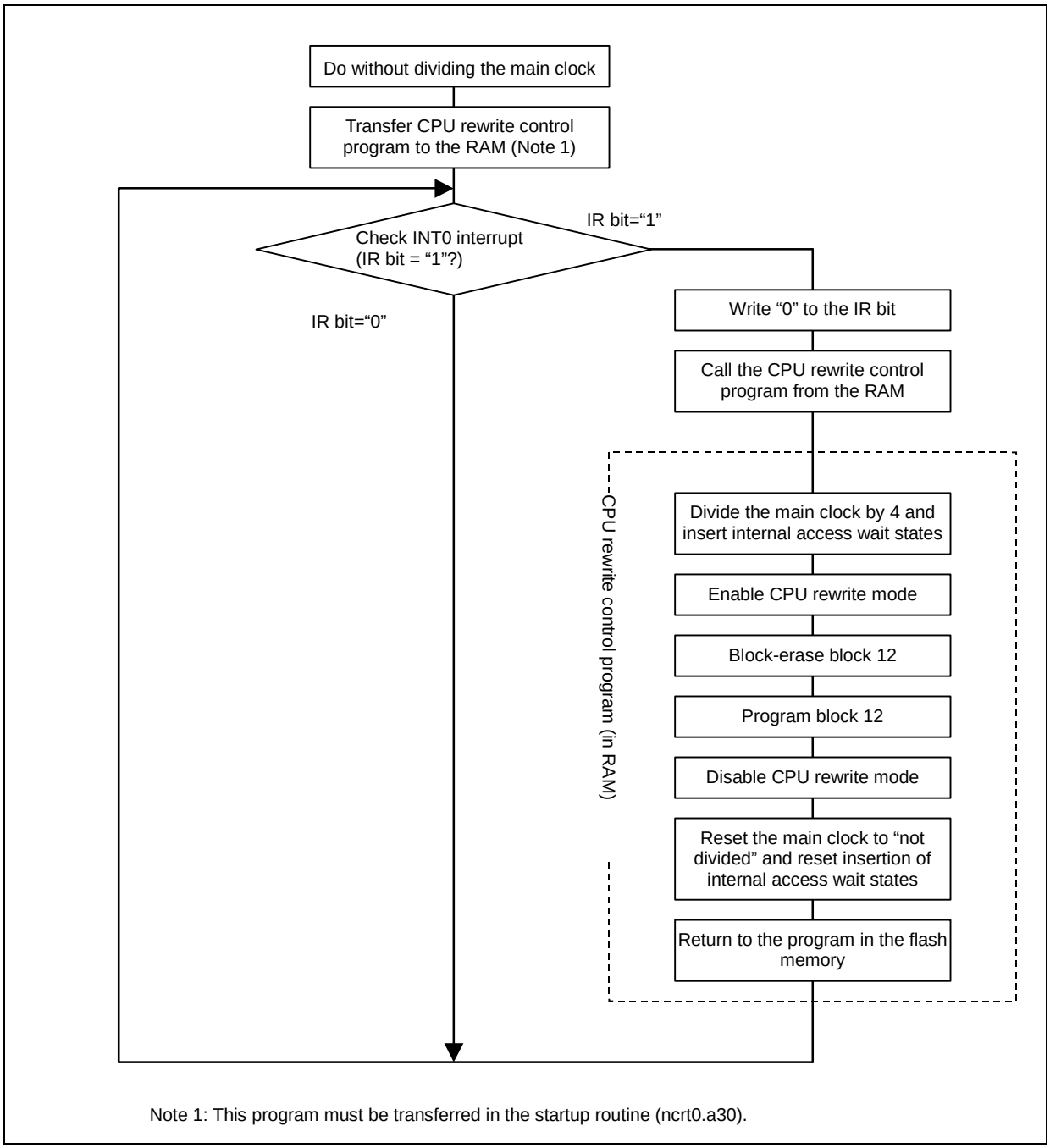


Figure 4. Sample Program Processing Flow

4.2 Program source

(1) Program (rjj05b0575_src.c)

```
/************************************************************************/
/*                                                                      */
/*  M32C/85 Group Program Collection                                    */
/*                                                                      */
/*  FILE NAME : rjj05b0575_src.c                                        */
/*  CPU       : This program is the execution sample                    */
/*              in CPU rewriting mode.                                  */
/*  HISTORY   : 2004.09.15 Ver 1.00                                     */
/*                                                                      */
/*  Copyright (C) 2004. Renesas Technology Corp.                        */
/*  Copyright (C) 2004. Renesas Solutions Corp.                         */
/*  All right reserved.                                                 */
/*                                                                      */
/************************************************************************/
/*************************************/
/*   include file                    */
/*************************************/
#include "sfr32c8586.h"              //  Special Function Register Header File

/*************************************/
/*   SFR declaration                 */
/*************************************/
#pragma ADDRESS    plc0_w     0026H   // PLL control register 0 & 1
unsigned short     plc0_w;
#pragma ADDRESS    fmr0_w     0057H   // Flash memory control register 0 (short)
unsigned short     fmr0_w;

/*************************************/
/*   Function declaration            */
/*************************************/
int    ew0_mode_program(void);        // CPU rewrite control routine.
int    full_chk(void);               // full status check routine.

/*************************************/
/*   global variable declaration     */
/*************************************/
unsigned short volatile    *wp;       // Erase/Write address pointer
unsigned short volatile    *ramp;     // Ram save address.

/*************************************/
/*   symbol declaration              */
/*************************************/
#define    OK     0
#define    NG     1

/*************************************/
/*   main function                   */
/*************************************/
void main(void)
{
   short   pll_wait;                // PLL wait counter

   // Set up a CPU operation clock.
   prcr = 0x03;                     // protect disabled.
   plc0_w = 0x0254;                 // PLL clock = Main clock x4.
   plc0   = 0xd4;                   // Start PLL.
                                   // It waits until a PLL clock is stabilized.
   for (pll_wait=0;pll_wait<4500;pll_wait++);
   cm17 = 1;                        // Main clock = PLL clock.
   mcd = 0x12;                      // Set main-clock no division mode.

   pm12 = 0;                        // Reset a internal memory wait bit.
   prcr = 0;                        // Protect enabled.

   int0ic = 0;                      // Set INT0IC register.
```

```
                                //  <ILVL2-0> : interrupt disabled
                                //  <POL>     : falling edge
                                //  <LVS>     : Edge sense

    asm("   fclr I");
    while(1)
    {
                                // Waiting for an INT0 interruption demand.
        if(ir_int0ic == 1)
        {
            int0ic     = 0;          // An INT0 interruption demand is cleared.

            if (ew0_mode_program()==NG) // CPU rewriting program execution.
            {                       // CPU rewrite error!
                p0  = 0x55;
                pd0 = 0xff;           // error display.
                while(1);
            }
        }
    }

}

#pragma SECTION program program_ram
/*************************************/
/*   CPU rewrite(EW0 mode) program   */
/*************************************/
int ew0_mode_program(void)
{
    unsigned short  fmr0_tmp;         // fmr0 register access temporary

    // A CPU operation clock is set to CPU rewriting modes.
    prcr = 0x03;                     // protect disabled.
    mcd  = 0x04;                     // main-clock divid-by-4 mode.
                                //  main-clock = 8MHz(XIN(32MHz))
    pm12 = 1;                       // Set a internal memory wait bit.
    prcr = 0;                       // Protect enabled.

    fmr01 = 0;                      // CPU rewrite mode enabled.
    fmr01 = 1;

    wp = (unsigned short *)0xf80000;
    *wp = 0x20;                      // Erase Command(1st bus cycle) write
    *wp = 0xd0;                      // Erase Command(2nd bus cycle) write
    while (fmr00 == 0);               // Wait for FMR00(RY/BY status) bit on
    if (full_chk() != 0) {            // Full-status check error
        *wp = 0xff;                  // Read-Array Command Write.
        fmr0_tmp = fmr0_w & 0x00fd;    // fmr01 clear.
        fmr0_w = fmr0_tmp;            // CPU rewrite mode disabled.
        fmr0_w &= 0x00fd;
                                // return to main routine
        return(NG);
    }

                                // Block-12 programed.
    wp = (unsigned short *)0xf80000;
    for (ramp=(unsigned short *)0x01800; ramp<(unsigned short *)0x04000;ramp++) {
        *wp = 0x40;                  // Program Command(1st bus cycle) weite
        *wp = *ramp;                 // Program Command(2nd bus cycle(DATA)) write.
        while (fmr00 == 0);           // Wait for FMR00(RY/BY status) bit on
        if (full_chk() != 0) {        // Full-status check error
            *wp = 0xff;              // Read-Array Command Write.
            fmr0_tmp = fmr0_w & 0x00fd; // fmr01 clear.
            fmr0_w = fmr0_tmp;         // CPU rewrite mode disabled.
                                // return to main routine;
            return(NG);
        }
        wp++;                       // Program address count up.
    }
```

```
    wp = (unsigned short *)0xf80000;
    *wp = 0xff;                       // Read-Array Command Write.
    fmr0_tmp = fmr0_w & 0x00fd;         // fmr01 clear.
    fmr0_w = fmr0_tmp;                 // CPU rewrite mode disabled.

    // Set up a CPU operation clock.
    prcr = 0x03;                      // protect disabled.
    mcd  = 0x12;                      // main-clock no-divid mode.
    pm12 = 0;                         // Reset a internal memory wait bit.
    prcr = 0;                         // Protect enabled.

                                    // Restore the General-purpose register, address-register.
    return(OK);                       // Return to the INT0 interrupt wait.

}

/************************************/
/*  Full-status check routine        */
/*                                */
/*     return value : 0=normal      */
/*                  1=error       */
/************************************/
int full_chk(void)
{
    if ((fmr0 & 0xc0) == 0xC0) {
                                    // Command sequence error
        *wp = 0x50;                   // Clear status register.
        return(NG);                   // Error return.

    } else if ((fmr0 & 0x80) != 0) {
                                    // Erase error
        *wp = 0x50;                   // Clear status register.
        return(NG);                   // Error return.

    } else if ((fmr0 & 0x40) != 0) {
                                    // Program error
        *wp = 0x50;                   // Clear status register.
        return(NG);                   // Error return.

    }

    return(OK);                       // Normal return.

}
```

(2) Start Up File (ncrt0.a30)

```
;***********************************************************************
;
;    C COMPILER for M16C/80
; COPYRIGHT(C) 1999(2000-2002) RENESAS TECHNOLOGY CORPORATION
; ALL RIGHTS RESERVED AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED
;
;
;    ncrt0.a30 : NC308 startup program
;
;    This program is applicable when using the basic I/O library
;
;    $Id: ncrt0.a30,v 1.18 2003/03/27 10:49:07 simomura Exp $
;
;***********************************************************************

;------------------------------------------------------------------------
; HEEP SIZE definition
;------------------------------------------------------------------------
HEAPSIZE .equ     300h

;------------------------------------------------------------------------
; STACK SIZE definition
;------------------------------------------------------------------------
STACKSIZE         .equ     300h

;------------------------------------------------------------------------
; INTERRUPT STACK SIZE definition
;------------------------------------------------------------------------
ISTACKSIZE        .equ     300h

;------------------------------------------------------------------------
; INTERRUPT VECTOR ADDRESS  definition
;------------------------------------------------------------------------
VECTOR_ADR        .equ     0fffd00h

;-----------------------------------------------------------------
; special page definition
;-----------------------------------------------------------------
;      macro define for special page
;
;Format:
;        SPECIAL  number
;

SPECIAL  .macro   NUM
         .org     0FFFFFEH-(NUM*2)
         .glb     __SPECIAL_@NUM
         .word    __SPECIAL_@NUM  & 0FFFFH
.endm
;------------------------------------------------------------------------
; Section allocation
;------------------------------------------------------------------------
         .list OFF
         .include sect308.inc
         .list ON

;------------------------------------------------------------------------
; SBDATA area definition
;------------------------------------------------------------------------
         .glb     __SB__
__SB__   .equ     data_SE_top

;======================================================================
; Initialize Macro declaration
;------------------------------------------------------------------------
;
; when copy less 64K byte
BZERO    .macro   TOP_ ,SECT_
         mov.b    #00H, R0L
```

```
        mov.l    #TOP_, A1
        mov.w    #sizeof SECT_ , R3
        sstr.b
        .endm

BCOPY .macro     FROM_,TO_,SECT_
        mov.l    #FROM_ ,A0
        mov.l    #TO_ ,A1
        mov.w    #sizeof SECT_ , R3
        smovf.b
        .endm

; when copy over 64K byte
;BZEROL  .macro   TOP_,SECT_
;        push.w   #sizeof SECT_ >> 16
;        push.w   #sizeof SECT_  & 0ffffh
;        pusha    TOP_
;        .stk     8
;
;        .glb     _bzero
;        .call    _bzero,G
;        jsr.a    _bzero
;        .endm
;
;
;BCOPYL  .macro   FROM_ ,TO_ ,SECT_
;        push.w   #sizeof SECT_  >> 16
;        push.w   #sizeof SECT_  & 0ffffh
;        pusha    TO_
;        pusha    FROM_
;        .stk     12
;
;        .glb     _bcopy
;        .call    _bcopy,G
;        jsr.a    _bcopy
;        .endm
;


;======================================================================
; Interrupt section start
;----------------------------------------------------------------------
        .insf    start,S,0
        .glb     start
        .section interrupt
start:
;----------------------------------------------------------------------
; after reset,this program will start
;----------------------------------------------------------------------
        ldc      #istack_top,      isp      ;set istack pointer
        mov.b    #02h,0ah
        mov.b    #00h,04h          ;set processer mode
        mov.b    #00h,0ah
        ldc      #0080h,  flg
        ldc      #stack_top,       sp       ;set stack pointer
        ldc      #data_SE_top,     sb       ;set sb register

        fset     b                          ;switch to bank 1
        ldc      #data_SE_top,     sb       ;set sb register
        fclr     b                          ;switch to bank 0

        ldc      #VECTOR_ADR,intb

;======================================================================
; NEAR area initialize.
;----------------------------------------------------------------------
; bss zero clear
;----------------------------------------------------------------------
        BZERO    bss_SE_top,bss_SE
        BZERO    bss_SO_top,bss_SO
```

```
        BZERO   bss_NE_top,bss_NE
        BZERO   bss_NO_top,bss_NO

        mov.b   #00H, R0L
        mov.l   #400h, A1
        mov.w   #(4400h - 400h) , R3
        sstr.b

;------------------------------------------------------------------------
; initialize data section
;------------------------------------------------------------------------
        BCOPY   data_SEI_top,data_SE_top,data_SE
        BCOPY   data_SOI_top,data_SO_top,data_SO
        BCOPY   data_NEI_top,data_NE_top,data_NE
        BCOPY   data_NOI_top,data_NO_top,data_NO

;========================================================================
; FAR area initialize.
;------------------------------------------------------------------------
; bss zero clear
;------------------------------------------------------------------------
;       BZERO   bss_SE_top,bss_SE
;       BZERO   bss_SO_top,bss_SO
;       BZERO   bss_6E_top,bss_6E
;       BZERO   bss_6O_top,bss_6O
        BZERO   bss_FE_top,bss_FE
        BZERO   bss_FO_top,bss_FO

;------------------------------------------------------------------------
; Copy edata_E(O) section from edata_EI(OI) section
;------------------------------------------------------------------------
;       BCOPY   data_SEI_top,data_SE_top,data_SE
;       BCOPY   data_SOI_top,data_SO_top,data_SO
;       BCOPY   data_6EI_top,data_6E_top,data_6E
;       BCOPY   data_6OI_top,data_6O_top,data_6O
        BCOPY   data_FEI_top,data_FE_top,data_FE
        BCOPY   data_FOI_top,data_FO_top,data_FO

        ldc     #stack_top,sp

;       .stk    -??     ; Validate this when use BZEROL,BCOPYL

;========================================================================
; heap area initialize
;------------------------------------------------------------------------
.if __HEAP__ != 1
        .glb    __mbase
        .glb    __mnext
        .glb    __msize
        mov.l   #(heap_top&0FFFFFFH), __mbase
        mov.l   #(heap_top&0FFFFFFH), __mnext
        mov.l   #(HEAPSIZE&0FFFFFFH), __msize
.endif
;========================================================================
; Initialize standard I/O
;------------------------------------------------------------------------
.if __STANDARD_IO__ != 1
        .glb    _init
        .call   _init,G
        jsr.a   _init
.endif

;========================================================================
; Program Ram initialize
;  _from_addr is defined by as308 option "-D_from_addr=0fd000h"
;------------------------------------------------------------------------
   BCOPY   _from_addr,_program_ram_top,program_ram
;

;========================================================================
```

```
; Call main() function
;--------------------------------------------------------------------
        ldc     #0h,fb   ; for debuger

        .glb    _main
        jsr.a   _main

;====================================================================
; exit() function
;--------------------------------------------------------------------
        .glb    _exit
        .glb    $exit
_exit:                      ; End program
$exit:
        jmp     _exit
        .einsf

;====================================================================
; dummy interrupt function
;--------------------------------------------------------------------
dummy_int:
        reit
        .end
;*****************************************************************************
;
;    C COMPILER for M16C/80
; COPYRIGHT(C) 1999(2000-2002) RENESAS TECHNOLOGY CORPORATION
; ALL RIGHTS RESERVED AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED
;
;
;*****************************************************************************
```

(3) Section define File (sect308.inc)

```
;******************************************************************************
;
;    C Compiler for M16C/80
; COPYRIGHT(C) 1999(2000-2002) RENESAS TECHNOLOGY CORPORATION
; ALL RIGHTS RESERVED AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED
;
;
;    Written by T.Aoyama
;
;    sect30.inc    : section definition
;    This program is applicable when using the basic I/O library
;
;    $Id: sect308.inc,v 1.13 2003/03/27 10:49:07 simomura Exp $
;
;******************************************************************************
;------------------------------------------------------------
;
;        Arrangement of section
;
;------------------------------------------------------------
; Near RAM data area
;------------------------------------------------------------
; SBDATA area
        .section data_SE,DATA
        .org    400H
data_SE_top:

        .section bss_SE,DATA,ALIGN
bss_SE_top:

        .section data_SO,DATA
data_SO_top:

        .section bss_SO,DATA
bss_SO_top:

; near RAM area
        .section data_NE,DATA,ALIGN
data_NE_top:

        .section bss_NE,DATA,ALIGN
bss_NE_top:

        .section data_NO,DATA
data_NO_top:

        .section bss_NO,DATA
bss_NO_top:

;------------------------------------------------------------
; Stack area
;------------------------------------------------------------
        .section stack,DATA,ALIGN
        .blkb   STACKSIZE
        .align
stack_top:

        .blkb   ISTACKSIZE
        .align
istack_top:


;------------------------------------------------------------
;        heap section
;------------------------------------------------------------
.if __HEAP__ != 1
        .section heap,DATA
heap_top:
        .blkb   HEAPSIZE
```

```
.endif

;------------------------------------------------------------------
; RAM program area
;------------------------------------------------------------------
         .section program_ram,ALIGN
_program_ram_top:
         .glb     _program_ram_top

;------------------------------------------------------------------
; Near ROM data area
;------------------------------------------------------------------
         .section rom_NE,ROMDATA,ALIGN
rom_NE_top:

         .section rom_NO,ROMDATA
rom_NO_top:

;------------------------------------------------------------------
; Far RAM data area
;------------------------------------------------------------------
; SBDATA area for #pragma SB16DATA
;        .section data_SE,DATA
;        .org             10000H
;data_SE_top:
;
;        .section bss_SE,DATA,ALIGN
;bss_SE_top:
;
;        .section data_SO,DATA
;data_SO_top:
;
;        .section bss_SO,DATA
;bss_SO_top:
;
;        .section data_6E,DATA,ALIGN
;data_6E_top:
;
;        .section bss_6E,DATA,ALIGN
;bss_6E_top:
;
;        .section data_6O,DATA
;data_6O_top:
;
;        .section bss_6O,DATA
;bss_6O_top:
;
         .section data_FE,DATA
         .org             20000H
data_FE_top:

         .section bss_FE,DATA,ALIGN
bss_FE_top:

         .section data_FO,DATA
data_FO_top:

         .section bss_FO,DATA
bss_FO_top:


;------------------------------------------------------------------
; Far ROM data area
;------------------------------------------------------------------
         .section rom_FE,ROMDATA
         .org             0FE0000H
rom_FE_top:

         .section rom_FO,ROMDATA
rom_FO_top:
```

```
;-----------------------------------------------------------------
; Initial data of 'data' section
;-----------------------------------------------------------------
         .section data_SEI,ROMDATA
data_SEI_top:

         .section data_SOI,ROMDATA
data_SOI_top:

;        .section data_6EI,ROMDATA
;data_6EI_top:
;
;        .section data_6OI,ROMDATA
;data_6OI_top:

         .section data_NEI,ROMDATA
data_NEI_top:

         .section data_NOI,ROMDATA
data_NOI_top:

         .section data_FEI,ROMDATA
data_FEI_top:

         .section data_FOI,ROMDATA
data_FOI_top:

;-----------------------------------------------------------------
; code area
;-----------------------------------------------------------------
         .section interrupt,ALIGN

         .section program,ALIGN

         .section program_S
         .org              0FF0000H

;-----------------------------------------------------------------
; variable vector section
;-----------------------------------------------------------------
         .section vector,ROMDATA            ; variable vector table
         .org     VECTOR_ADR

         .lword   dummy_int                 ; BRK (software int 0)
         .lword   dummy_int                 ;
         .lword   dummy_int                 ;
         .lword   dummy_int                 ;
         .lword   dummy_int                 ;
         .lword   dummy_int                 ;
         .lword   dummy_int                 ;
         .lword   dummy_int                 ;
         .lword   dummy_int                 ; DMA0 (software int 8)
         .lword   dummy_int                 ; DMA1 (software int 9)
         .lword   dummy_int                 ; DMA2 (software int 10)
         .lword   dummy_int                 ; DMA3 (software int 11)
         .lword   dummy_int                 ; TIMER A0 (software int 12)
         .lword   dummy_int                 ; TIMER A1 (software int 13)
         .lword   dummy_int                 ; TIMER A2 (software int 14)
         .lword   dummy_int                 ; TIMER A3 (software int 15)
         .lword   dummy_int                 ; TIMER A4 (software int 16)
         .lword   dummy_int                 ; uart0 trance (software int 17)
         .lword   dummy_int                 ; uart0 receive (software int 18)
         .lword   dummy_int                 ; uart1 trance (software int 19)
         .lword   dummy_int                 ; uart1 receive (software int 20)
         .lword   dummy_int                 ; TIMER B0 (software int 21)
         .lword   dummy_int                 ; TIMER B1 (software int 22)
         .lword   dummy_int                 ; TIMER B2 (software int 23)
         .lword   dummy_int                 ; TIMER B3 (software int 24)
         .lword   dummy_int                 ; TIMER B4 (software int 25)
```

```
        .lword   dummy_int                  ; INT5 (software int 26)
        .lword   dummy_int                  ; INT4 (software int 27)
        .lword   dummy_int                  ; INT3 (software int 28)
        .lword   dummy_int                  ; INT2 (software int 29)
        .lword   dummy_int                  ; INT1 (software int 30)
        .lword   dummy_int                  ; INT0 (software int 31)
        .lword   dummy_int                  ; TIMER B5 (software int 32)
        .lword   dummy_int                  ; uart2 trance/NACK (software int 33)
        .lword   dummy_int                  ; uart2 receive/ACK (software int 34)
        .lword   dummy_int                  ; uart3 trance/NACK (software int 35)
        .lword   dummy_int                  ; uart3 receive/ACK (software int 36)
        .lword   dummy_int                  ; uart4 trance/NACK (software int 37)
        .lword   dummy_int                  ; uart4 receive/ACK (software int 38)
        .lword   dummy_int                  ; uart2 bus collision (software int 39)
        .lword   dummy_int                  ; uart3 bus collision (software int 40)
        .lword   dummy_int                  ; uart4 bus collision (software int 41)
        .lword   dummy_int                  ; A-D Convert (software int 42)
        .lword   dummy_int                  ; input key (software int 43)
        .lword   dummy_int                  ; software int 44
        .lword   dummy_int                  ; software int 45
        .lword   dummy_int                  ; software int 46
        .lword   dummy_int                  ; software int 47
        .lword   dummy_int                  ; software int 48
        .lword   dummy_int                  ; software int 49
        .lword   dummy_int                  ; software int 50
        .lword   dummy_int                  ; software int 51
        .lword   dummy_int                  ; software int 52
        .lword   dummy_int                  ; software int 53
        .lword   dummy_int                  ; software int 54
        .lword   dummy_int                  ; software int 55
        .lword   dummy_int                  ; software int 56
        .lword   dummy_int                  ; software int 57
        .lword   dummy_int                  ; software int 58
        .lword   dummy_int                  ; software int 59
        .lword   dummy_int                  ; software int 60
        .lword   dummy_int                  ; software int 61
        .lword   dummy_int                  ; software int 62
        .lword   dummy_int                  ; software int 63

;=====================================================================
; fixed vector section
;---------------------------------------------------------------------
        .section fvector,ROMDATA                    ; fixed vector table
;=====================================================================
; special page defination
;---------------------------------------------------------------------
;       macro is defined in ncrt0.a30
;       Format: SPECIAL number
;
;---------------------------------------------------------------------
;       SPECIAL 255
;       SPECIAL 254
;       SPECIAL 253
;       SPECIAL 252
;       SPECIAL 251
;       SPECIAL 250
;       SPECIAL 249
;       SPECIAL 248
;       SPECIAL 247
;       SPECIAL 246
;       SPECIAL 245
;       SPECIAL 244
;       SPECIAL 243
;       SPECIAL 242
;       SPECIAL 241
;       SPECIAL 240
;       SPECIAL 239
;       SPECIAL 238
;       SPECIAL 237
;       SPECIAL 236
```

```
;       SPECIAL 235
;       SPECIAL 234
;       SPECIAL 233
;       SPECIAL 232
;       SPECIAL 231
;       SPECIAL 230
;       SPECIAL 229
;       SPECIAL 228
;       SPECIAL 227
;       SPECIAL 226
;       SPECIAL 225
;       SPECIAL 224
;       SPECIAL 223
;       SPECIAL 222
;       SPECIAL 221
;       SPECIAL 220
;       SPECIAL 219
;       SPECIAL 218
;       SPECIAL 217
;       SPECIAL 216
;       SPECIAL 215
;       SPECIAL 214
;       SPECIAL 213
;       SPECIAL 212
;       SPECIAL 211
;       SPECIAL 210
;       SPECIAL 209
;       SPECIAL 208
;       SPECIAL 207
;       SPECIAL 206
;       SPECIAL 205
;       SPECIAL 204
;       SPECIAL 203
;       SPECIAL 202
;       SPECIAL 201
;       SPECIAL 200
;       SPECIAL 199
;       SPECIAL 198
;       SPECIAL 197
;       SPECIAL 196
;       SPECIAL 195
;       SPECIAL 194
;       SPECIAL 193
;       SPECIAL 192
;       SPECIAL 191
;       SPECIAL 190
;       SPECIAL 189
;       SPECIAL 188
;       SPECIAL 187
;       SPECIAL 186
;       SPECIAL 185
;       SPECIAL 184
;       SPECIAL 183
;       SPECIAL 182
;       SPECIAL 181
;       SPECIAL 180
;       SPECIAL 179
;       SPECIAL 178
;       SPECIAL 177
;       SPECIAL 176
;       SPECIAL 175
;       SPECIAL 174
;       SPECIAL 173
;       SPECIAL 172
;       SPECIAL 171
;       SPECIAL 170
;       SPECIAL 169
;       SPECIAL 168
;       SPECIAL 167
;       SPECIAL 166
```

```
;       SPECIAL 165
;       SPECIAL 164
;       SPECIAL 163
;       SPECIAL 162
;       SPECIAL 161
;       SPECIAL 160
;       SPECIAL 159
;       SPECIAL 158
;       SPECIAL 157
;       SPECIAL 156
;       SPECIAL 155
;       SPECIAL 154
;       SPECIAL 153
;       SPECIAL 152
;       SPECIAL 151
;       SPECIAL 150
;       SPECIAL 149
;       SPECIAL 148
;       SPECIAL 147
;       SPECIAL 146
;       SPECIAL 145
;       SPECIAL 144
;       SPECIAL 143
;       SPECIAL 142
;       SPECIAL 141
;       SPECIAL 140
;       SPECIAL 139
;       SPECIAL 138
;       SPECIAL 137
;       SPECIAL 136
;       SPECIAL 135
;       SPECIAL 134
;       SPECIAL 133
;       SPECIAL 132
;       SPECIAL 131
;       SPECIAL 130
;       SPECIAL 129
;       SPECIAL 128
;       SPECIAL 127
;       SPECIAL 126
;       SPECIAL 125
;       SPECIAL 124
;       SPECIAL 123
;       SPECIAL 122
;       SPECIAL 121
;       SPECIAL 120
;       SPECIAL 119
;       SPECIAL 118
;       SPECIAL 117
;       SPECIAL 116
;       SPECIAL 115
;       SPECIAL 114
;       SPECIAL 113
;       SPECIAL 112
;       SPECIAL 111
;       SPECIAL 110
;       SPECIAL 109
;       SPECIAL 108
;       SPECIAL 107
;       SPECIAL 106
;       SPECIAL 105
;       SPECIAL 104
;       SPECIAL 103
;       SPECIAL 102
;       SPECIAL 101
;       SPECIAL 100
;       SPECIAL 99
;       SPECIAL 98
;       SPECIAL 97
;       SPECIAL 96
```

```
;       SPECIAL 95
;       SPECIAL 94
;       SPECIAL 93
;       SPECIAL 92
;       SPECIAL 91
;       SPECIAL 90
;       SPECIAL 89
;       SPECIAL 88
;       SPECIAL 87
;       SPECIAL 86
;       SPECIAL 85
;       SPECIAL 84
;       SPECIAL 83
;       SPECIAL 82
;       SPECIAL 81
;       SPECIAL 80
;       SPECIAL 79
;       SPECIAL 78
;       SPECIAL 77
;       SPECIAL 76
;       SPECIAL 75
;       SPECIAL 74
;       SPECIAL 73
;       SPECIAL 72
;       SPECIAL 71
;       SPECIAL 70
;       SPECIAL 69
;       SPECIAL 68
;       SPECIAL 67
;       SPECIAL 66
;       SPECIAL 65
;       SPECIAL 64
;       SPECIAL 63
;       SPECIAL 62
;       SPECIAL 61
;       SPECIAL 60
;       SPECIAL 59
;       SPECIAL 58
;       SPECIAL 57
;       SPECIAL 56
;       SPECIAL 55
;       SPECIAL 54
;       SPECIAL 53
;       SPECIAL 52
;       SPECIAL 51
;       SPECIAL 50
;       SPECIAL 49
;       SPECIAL 48
;       SPECIAL 47
;       SPECIAL 46
;       SPECIAL 45
;       SPECIAL 44
;       SPECIAL 43
;       SPECIAL 42
;       SPECIAL 41
;       SPECIAL 40
;       SPECIAL 39
;       SPECIAL 38
;       SPECIAL 37
;       SPECIAL 36
;       SPECIAL 35
;       SPECIAL 34
;       SPECIAL 33
;       SPECIAL 32
;       SPECIAL 31
;       SPECIAL 30
;       SPECIAL 29
;       SPECIAL 28
;       SPECIAL 27
;       SPECIAL 26
```

```
;       SPECIAL 25
;       SPECIAL 24
;       SPECIAL 23
;       SPECIAL 22
;       SPECIAL 21
;       SPECIAL 20
;       SPECIAL 19
;       SPECIAL 18
;
;===============================================================
; fixed vector section
;---------------------------------------------------------------
        .org    0FFFFDCh
UDI:
        .lword  dummy_int
OVER_FLOW:
        .lword  dummy_int
BRKI:
        .lword  dummy_int
ADDRESS_MATCH:
        .lword  dummy_int
SINGLE_STEP:
        .lword  dummy_int
WDT:
        .lword  dummy_int
DBC:
        .lword  dummy_int
NMI:
        .lword  dummy_int
RESET:
        .lword  start
;
;******************************************************************************
;
;    C Compiler for M16C/80
; COPYRIGHT(C) 1999(2000-2002) RENESAS TECHNOLOGY CORPORATION
; ALL RIGHTS RESERVED AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED
;
;
;******************************************************************************
```

5. Reference

Renesas Technology Corporation Home Page
http://www.renesas.com/

E-mail Support
E-mail: csc@renesas.com

Hardware Manual
M32C/85 Group Hardware Manual
(Use the latest version on the home page: http://www.renesas.com)

REVISION HISTORY

Rev.	Date	Description	
		Page	Summary
1.00	2005.03.25	-	First edition issued
1.01	2005.04.25	4	3.2.2 title modified

Keep safety first in your circuit designs!

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

Notes regarding these materials

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