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.

M16C/62P Group

Example Application for Reduced Power Consumption in the Flash Memory Version

1. Abstract

This document describes the procedure and example usage for reducing the power consumption in the flash memory version of microcomputer

2. Introduction

The explanation of this issue is applied to the following condition:
Applicable MCU: M16C/62P Group (flash memory version)

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 M16C/62P 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

3.1 Implementation Flow for Reduced Power Consumption

Figure 1 shows flowchart for implementing reduced power consumption.

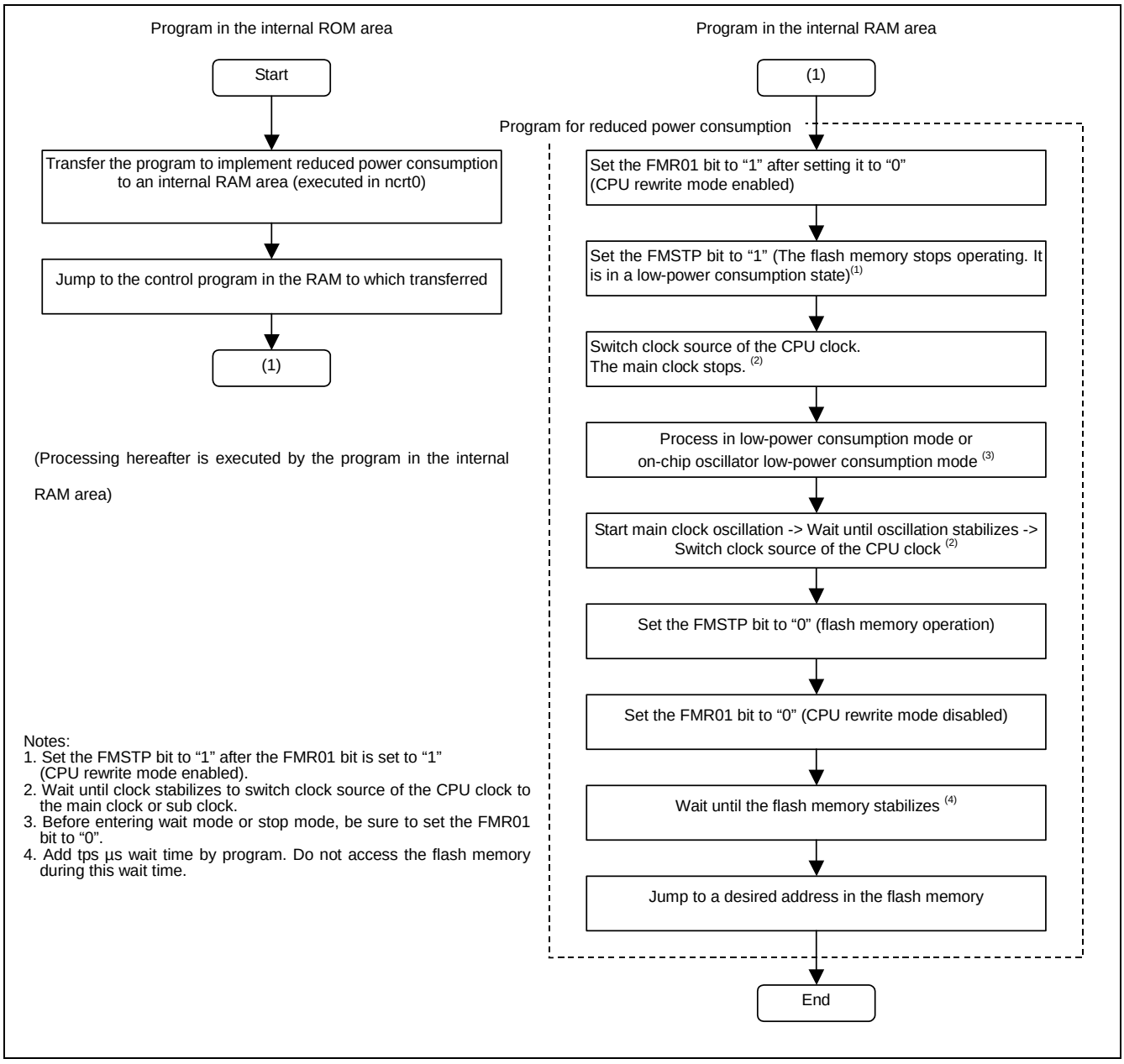


Figure 1. Flowchart for Implementing Reduced Power Consumption

3.2 Set Up Procedure

3.2.1 Transferring the Reduced-power Consumption Program into RAM

The program to implement reduced power consumption needs to be run in RAM. Here, the following explains an example for transferring the reduced-power consumption program from the ROM area in which it is stored beginning with the address 0F1000h to an area in RAM.

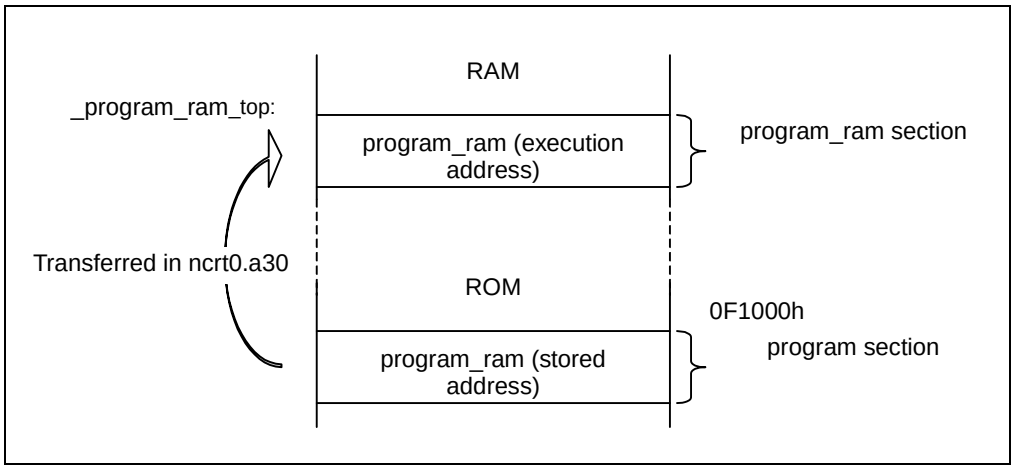


Figure 2. Program Location

(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 sect30.inc

Add the program_ram section to sect30.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
;------------------------------------------------------------
      .section  heap,DATA
heap_top:
      .blkb     HEAPSIZE
```

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

Add here.

(3) Transferring the 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 as30 option "-D_from_addr=0f1000h"
;---------------------------------------------------------------
     N_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

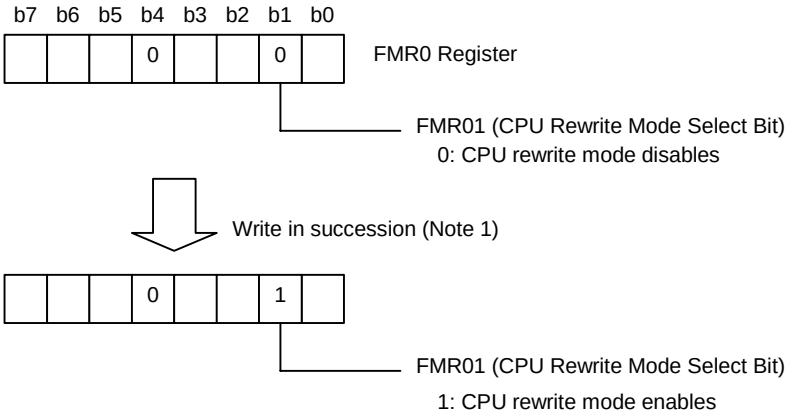
To run the program transferred into RAM, it is necessary to specify in the linker (ln30) that the program storage address (in ROM) and execution address (in RAM) be located separately.

```
ln30 –LOC program_ram=0F1000
```

In the above option, the program_ram section is stored beginning with the address 0F1000h.

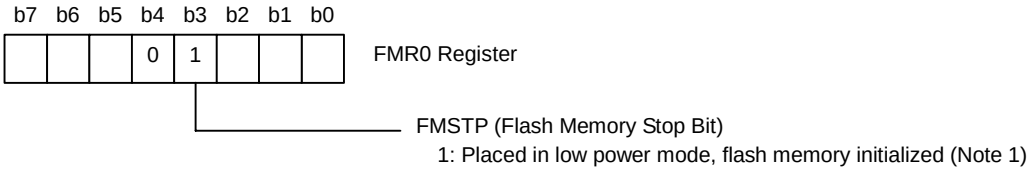
3.2.2 Processing in the Reduced-power Consumption Program

(1) Enable the CPU rewrite mode.



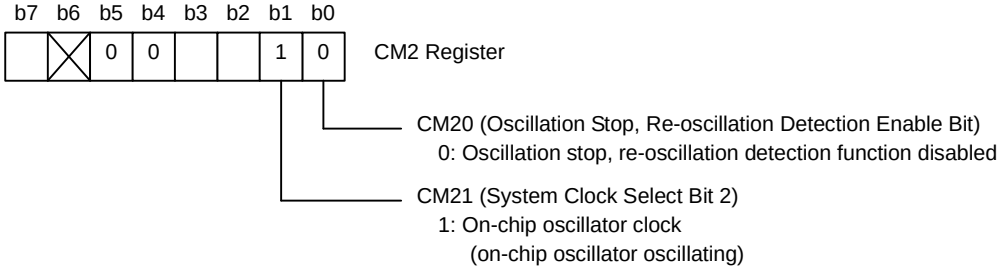
Note: To set the FMR01 bit to “1”, write “0” and then “1” to the FMR01 bit in succession. Make sure no interrupts or DMA transfers occur before the CPU writes “1” after writing “0”.
Make sure writes to the FMR01 bit is performed in other than the internal flash memory.
Also make sure this write operation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

(2) Turn the flash memory off.

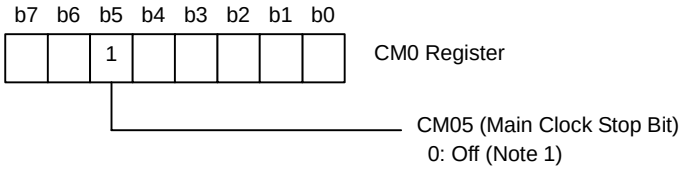


Note 1: Write to this bit from a program in other than the flash memory.
Effective when the FMR01 bit = 1 (CPU rewrite mode). If the FMR01 bit = 0, although the FMR03 bit can be set to “1” by writing “1” in a program, the flash memory is neither placed in low power mode nor initialized.

(3) Change the CPU clock source to the on-chip oscillator clock. (When using the on-chip oscillator)



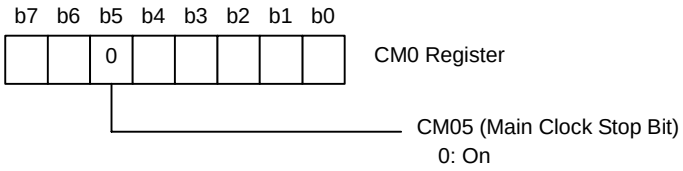
(4) Stop the main clock.



Note 1: This bit is provided to stop the main clock when the low power dissipation mode or on-chip oscillator low power dissipation mode is selected. This bit cannot be used for detection as to whether the main clock stopped or not. To stop the main clock, set bits in the following order.
(a) Set the CM07 bit to "1" (Sub-clock select) or the CM21 bit of CM2 register to "1" (On-chip oscillator select) with the sub-clock stably oscillating.
(b) Set the CM20 bit of CM2 register to "0" (Oscillation stop, re-oscillation detection function disabled).
(c) Set the CM05 bit to "1" (Stop).

(5) Execute the user program that runs during reduced power consumption with the on-chip oscillator clock.

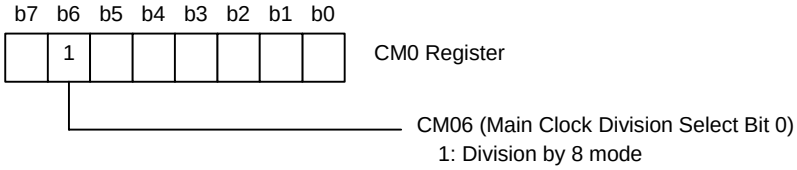
(6) Let the main clock start oscillating.



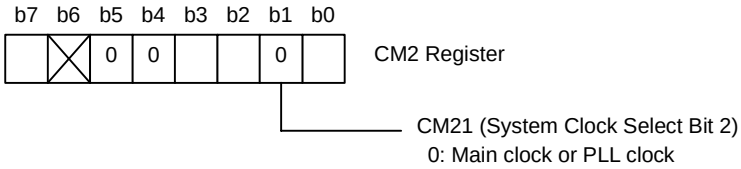
(7) Wait about 1 ms (Note 1) until the main clock oscillation stabilizes.

Note 1: In this document, the main clock is assumed to have an oscillation stabilization time of 1 ms. Since the oscillation stabilization time varies with each type of resonator, check the resonator used in your system to find the time to be waited until it stabilizes.

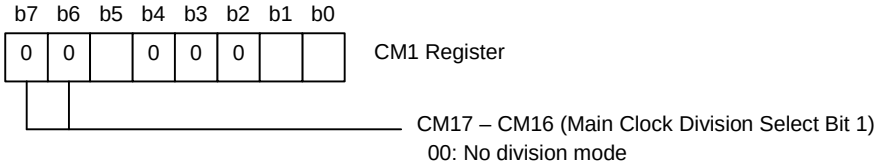
(8) Set the main clock divide ratio to divide-by-8 mode.

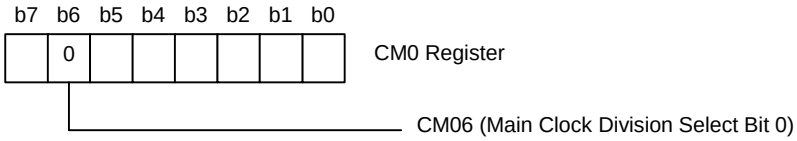


(9) Change the CPU clock source to the main clock.

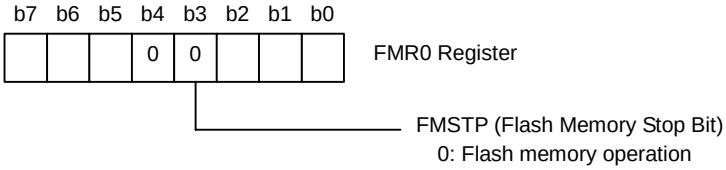


(10) Set the main clock divide ratio to not-divided mode.



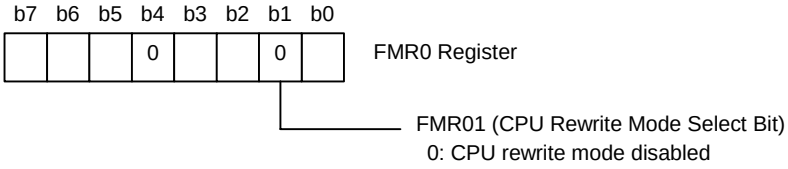


(11) Turn the flash memory on.



Note 1: Write to this bit from a program in other than the flash memory.
Effective when the FMR01 bit = 1 (CPU rewrite mode). If the FMR01 bit = 0, although the FMR03 bit can be set to “1” by writing “1” in a program, the flash memory is neither placed in low power mode nor initialized.

(12) Disable the CPU rewrite mode.



(13) Wait until the flash memory circuit stabilizes.
Wait a finite time equal to the flash memory circuit stabilization time (tPS).

(14) Jump to any address in the flash memory.

4. Sample Programming Code

The following shows an example program for entering and exiting reduced-power consumption mode clocked by on-chip oscillator repeatedly as triggered by an $\overline{INT0}$ interrupt request generated.

Operating Conditions:
(1) VCC1=VCC2=5V
(2)XIN=16MHz

Amount of memory used:
The reduced-power consumption program transferred into RAM requires 115 bytes of memory when it is compiled without optimization options.

4.1 Processing Flow

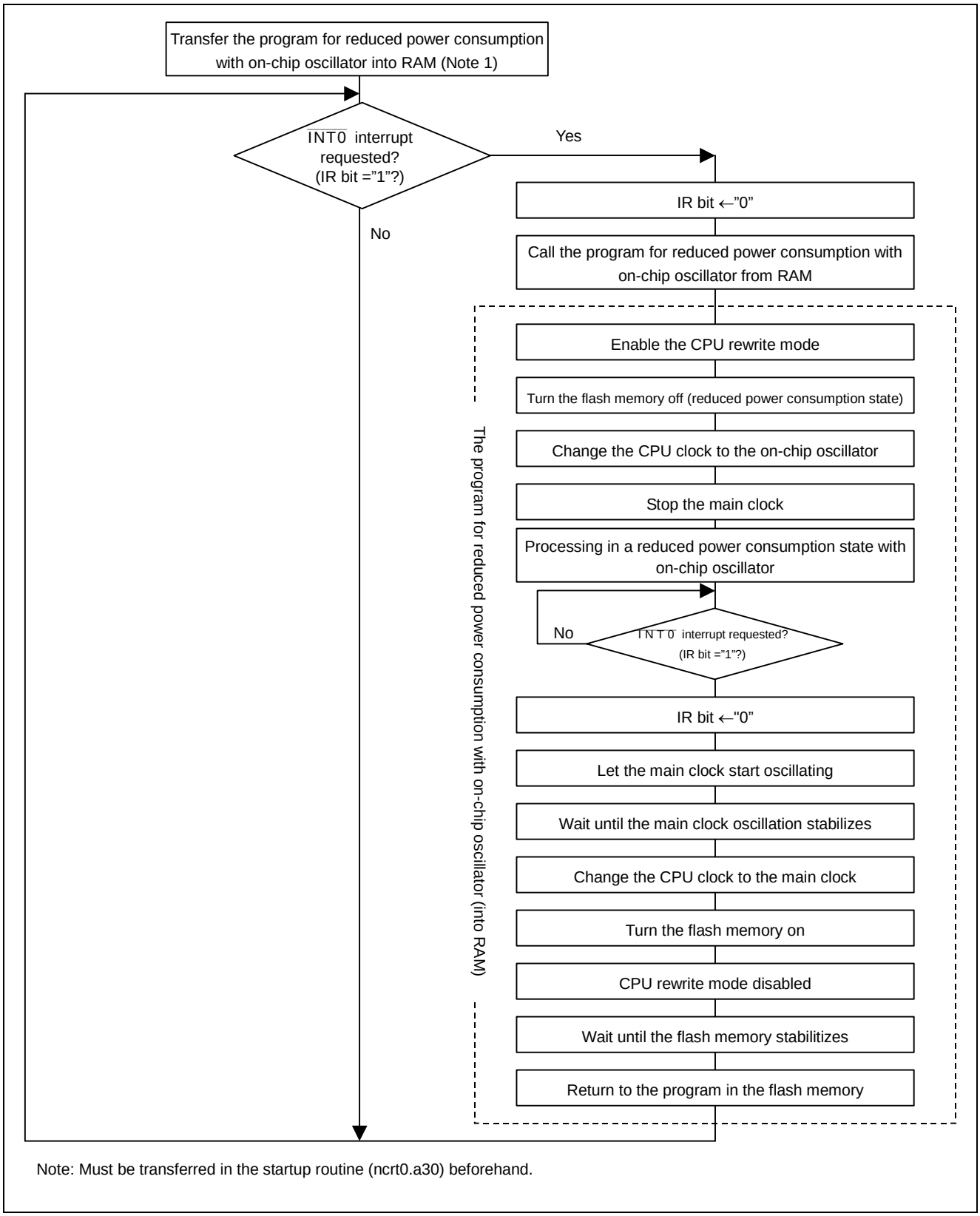


Figure 3. Sample Program Processing Flow

4.2 Program source

(1) Program (rjj05b0642_src.c)

```
/***********************************************************************/
/*                                                                     */
/*  M16C/62P Group Program Collection                                  */
/*                                                                     */
/*  FILE NAME : rjj05b0642_src.c                                       */
/*  CPU      : This program is the example of application of the       */
/*             reduction in on-chip oscillator power consumption.      */
/*                                                                     */
/*  HISTORY   : 2004.11.01 Ver 1.00                                    */
/*                                                                     */
/*  Copyright (C) 2004. Renesas Technology Corp.                       */
/*  Copyright (C) 2004. Renesas Solutions Corp.                        */
/*  All right reserved.                                                */
/*                                                                     */
/***********************************************************************/
/*************************************/
/*   include file                    */
/*************************************/
#include "sfr62p.h"              //  Special Function Register Header File

/*************************************/
/*   Function declaration            */
/*************************************/
void    low_power(void);          // Low power consumption-ized program.

/*************************************/
/*   symbol declaration              */
/*************************************/
#define    WAIT_MAIN_STB   65     // Main clock oscillation stable waiting time.
                              //  1ms (at on-chip-oscillator-clock(1MHz) no-division mode)
#define    WAIT_tPS       16     // Flash memory circuit stable waiting time.
                              //  15us (at main-clock(16MHz) no-division mode)

/*************************************/
/*   main function                   */
/*************************************/
void main(void)
{

    int0ic = 0;                 // Set INT0 interrupt proprity level
    ifsr = 0;                   // INT0 select a falling edge.

    asm("   fclr I");

    while(1)
    {
                               // Waiting for an INT0 interruption demand.
        while(!ir_int0ic);
        int0ic     = 0;          // An INT0 interruption demand is cleared.
        low_power();             // Low power consumption-ized program execution.

    }

}

#pragma SECTION program program_ram
/*****************************************/
/*   Low power consumption-ized program   */
/*****************************************/
void low_power(void)
{
```

```
    unsigned short  loop;          // wait counter.

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

    fmstp = 1;                    // Stop flash memory.

    prc0 = 1;                     // Protect disabled.

    cm21 = 1;                     // An on-tip oscillator is oscillated
                                 // and it is made a CPU clock.

    cm05 = 1;                     // Stop main-clock.

    prc0 = 0;                     // Protect enabled.

                                 // Waiting for an INT0 interruption demand.
    while(!ir_int0ic);
    int0ic     = 0;               // An INT0 interruption demand is cleared.

    prc0 = 1;                     // Protect disabled.

    cm05 = 0;                     // Main-clock oscillation.

                                 // Waiting for main clock oscillation stability.
    for (loop=0;loop<WAIT_MAIN_STB;loop++) {
       asm("");                   // The asm Function suppresses optimization.
    };

    cm06 = 1;                     // Set main-clock divid-by-8 mode.

    cm21 = 0;                     // A main clock is used as a CPU clock.

    cm1 = cm1 & 0x3f;              // Set "0" to cm17 and cm16.
    cm06 = 0;                     // Set main-clock no-divid mode.

    prc0 = 0;                     // Protect enabled.

    fmstp = 0;                    // A flash memory is operated.

    fmr01 = 0;                    // CPU rewrite mode disabled.

                                 // Waiting for flash memory stability.
    for (loop=0;loop<WAIT_tPS;loop++) {
       asm("");                   // The asm Function suppresses optimization.
    };

    return;

}
```

(2) Start Up File (ncrt0.a30)

```
;****************************************************************************** ;
; C COMPILER for R8C/Tiny, M16C/60,30,20,10
; COPYRIGHT(C) 1999(2000-2004) RENESAS TECHNOLOGY CORPORATION
; AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED
;
;
; ncrt0.a30 : NC30 startup program
;
; This program is applicable when using the basic I/O library
;
; $Id: ncrt0.a30,v 1.27 2004/03/08 04:13:56 muranaka Exp $
;
;******************************************************************************

;------------------------------------------------------------------------
; HEEP SIZE definition
;------------------------------------------------------------------------
.if    __HEAP__ == 1
HEAPSIZE       .equ    0H
.else

.if    __HEAPSIZE__ == 0
HEAPSIZE       .equ    300H
.else
HEAPSIZE       .equ    __HEAPSIZE__
.endif

.endif

;------------------------------------------------------------------------
; STACK SIZE definition
;------------------------------------------------------------------------
.if    __USTACKSIZE__ == 0
.if    __R8C__ != 1
STACKSIZE      .equ    300h
.else
STACKSIZE      .equ    80h
.endif
.else
STACKSIZE      .equ    __USTACKSIZE__
.endif

;------------------------------------------------------------------------
; INTERRUPT STACK SIZE definition
;------------------------------------------------------------------------
.if    __ISTACKSIZE__ == 0
.if    __R8C__ != 1
ISTACKSIZE     .equ    300h
.else
ISTACKSIZE     .equ    80h
.endif
.else
ISTACKSIZE     .equ    __ISTACKSIZE__
.endif

;------------------------------------------------------------------------
; INTERRUPT VECTOR ADDRESS  definition
;------------------------------------------------------------------------
.if    __R8C__ != 1
VECTOR_ADR     .equ    0ffd00h
SVECTOR_ADR    .equ    0ffe00h
.else
VECTOR_ADR     .equ    0fedch
.endif

;------------------------------------------------------------------------
```

```
; Section allocation
;------------------------------------------------------------------------
       .list OFF
       .include sect30.inc
       .list ON

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

;========================================================================
; Initialize Macro declaration
;------------------------------------------------------------------------
N_BZERO        .macro  TOP_ ,SECT_
       mov.b   #00H, R0L
       mov.w   #(TOP_ & 0FFFFH), A1
       mov.w   #sizeof SECT_ , R3
       sstr.b
       .endm

N_BCOPY .macro FROM_,TO_,SECT_
       mov.w   #(FROM_ & 0FFFFH),A0
       mov.b   #(FROM_ >>16),R1H
       mov.w   #TO_ ,A1
       mov.w   #sizeof SECT_ , R3
       smovf.b
       .endm

BZERO  .macro  TOP_,SECT_
       push.w  #sizeof SECT_ >> 16
       push.w  #sizeof SECT_  & 0ffffh
       pusha   TOP_ >>16
       pusha   TOP_ & 0ffffh
       .stk    8
       .glb    _bzero
       .call   _bzero,G
       jsr.a   _bzero
       .endm

BCOPY  .macro  FROM_ ,TO_ ,SECT_
       push.w  #sizeof SECT_  >> 16
       push.w  #sizeof SECT_  & 0ffffh
       pusha   TO_ >>16
       pusha   TO_  & 0ffffh
       pusha   FROM_ >>16
       pusha   FROM_  & 0ffffh
       .stk    12
       .glb    _bcopy
       .call   _bcopy,G
       jsr.a   _bcopy
       .endm

.if    __R8C__ != 1
;
; for M16C/60,30,20,10 series
;
;              .glb    __BankSelect
;__BankSelect  .equ    0BH

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

```
SPECIAL        .macro  NUM
       .org    0FFFFEH-(NUM*2)
       .glb    __SPECIAL_@NUM
       .word   __SPECIAL_@NUM  & 0FFFFH
.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   #08h,06h
       mov.b   #20h,07h                ;set mein clock no-division mode
       mov.b   #00h,0ah
       ldc     #0080h, flg
       ldc     #stack_top,     sp      ;set stack pointer
       ldc     #data_SE_top,   sb      ;set sb register
       ldintb  #VECTOR_ADR

;=====================================================================
; NEAR area initialize.
;---------------------------------------------------------------------
; bss zero clear
;---------------------------------------------------------------------
       N_BZERO bss_SE_top,bss_SE
       N_BZERO bss_SO_top,bss_SO
       N_BZERO bss_NE_top,bss_NE
       N_BZERO bss_NO_top,bss_NO

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

;=====================================================================
; FAR area initialize.
;---------------------------------------------------------------------
; bss zero clear
;---------------------------------------------------------------------
       BZERO   bss_FE_top,bss_FE
       BZERO   bss_FO_top,bss_FO

;---------------------------------------------------------------------
; Copy edata_E(O) section from edata_EI(OI) section
;---------------------------------------------------------------------
       BCOPY   data_FEI_top,data_FE_top,data_FE
       BCOPY   data_FOI_top,data_FO_top,data_FO

       ldc     #stack_top,sp
       .stk    -40

;=====================================================================
; heap area initialize
;---------------------------------------------------------------------
.if __HEAP__ != 1
       .glb    __mbase
       .glb    __mnext
       .glb    __msize
```

```
        mov.w   #(heap_top&0FFFFH), __mbase
        mov.w   #(heap_top>>16), __mbase+2
        mov.w   #(heap_top&0FFFFH), __mnext
        mov.w   #(heap_top>>16), __mnext+2
        mov.w   #(HEAPSIZE&0FFFFH), __msize
        mov.w   #(HEAPSIZE>>16), __msize+2
.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 as30 option "-D_from_addr=0fa000h"
;--------------------------------------------------------------------
   N_BCOPY _from_addr,_program_ram_top,program_ram
;

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

        .glb    _main
        jsr.a   _main

.else  ; __R8C__

;--------------------------------------------------------------------
; for R8C/Tiny
;--------------------------------------------------------------------

;====================================================================
; 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
        ldintb  #VECTOR_ADR

;====================================================================
; NEAR area initialize.
;--------------------------------------------------------------------
; bss zero clear
;--------------------------------------------------------------------
        N_BZERO bss_SE_top,bss_SE
        N_BZERO bss_SO_top,bss_SO
        N_BZERO bss_NE_top,bss_NE
        N_BZERO bss_NO_top,bss_NO

;--------------------------------------------------------------------
; initialize data section
```

```
;-----------------------------------------------------------------------
       N_BCOPY data_SEI_top,data_SE_top,data_SE
       N_BCOPY data_SOI_top,data_SO_top,data_SO
       N_BCOPY data_NEI_top,data_NE_top,data_NE
       N_BCOPY data_NOI_top,data_NO_top,data_NO

;=======================================================================
; FAR area initialize.
;-----------------------------------------------------------------------
; bss zero clear
;-----------------------------------------------------------------------
;      BZERO   bss_FE_top,bss_FE
;      BZERO   bss_FO_top,bss_FO

;-----------------------------------------------------------------------
; Copy edata_E(O) section from edata_EI(OI) section
;-----------------------------------------------------------------------
;      BCOPY   data_FEI_top,data_FE_top,data_FE
;      BCOPY   data_FOI_top,data_FO_top,data_FO

       ldc     #stack_top,sp
;      .stk    -40

;=======================================================================
; heap area initialize
;-----------------------------------------------------------------------
.if __HEAP__ != 1
       .glb    __mbase
       .glb    __mnext
       .glb    __msize
       mov.w   #(heap_top&0FFFFH), __mbase
       mov.w   #(heap_top&0FFFFH), __mnext
       mov.w   #(HEAPSIZE&0FFFFH), __msize
.endif

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

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

       .glb    _main
       jsr.a   _main

.endif ; __R8C__

;=======================================================================
; exit() function
;-----------------------------------------------------------------------
       .glb    _exit
       .glb    $exit
_exit:                  ; End program
$exit:
       jmp     _exit
       .einsf
;=======================================================================
; dummy interrupt function
;-----------------------------------------------------------------------
       .glb    dummy_int
dummy_int:
       reit
```

```
        .end
;************************************************************************
;
; C COMPILER for R8C/Tiny, M16C/60,30,20,10
; COPYRIGHT(C) 1999(2000-2004) RENESAS TECHNOLOGY CORPORATION
; AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED
;
;************************************************************************
```

(3) Section define File (sect30.inc)

```
;*****************************************************************************
;
; C Compiler for R8C/Tiny, M16C/60,30,20,10
; COPYRIGHT(C) 1999(2000-2002) RENESAS TECHNOLOGY CORPORATION
; 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: sect30.inc,v 1.22 2004/02/16 05:17:14 muranaka Exp $
;
;*****************************************************************************

.if   __R8C__ != 1
;
;     for M16C/60,30,20,10
;
;------------------------------------------------------------
;
;     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
      .blkb   STACKSIZE
stack_top:

      .blkb   ISTACKSIZE
istack_top:

;------------------------------------------------------------
;     heap section
```

```
;------------------------------------------------------------------
      .section        heap,DATA
heap_top:
      .blkb   HEAPSIZE

;------------------------------------------------------------------
; 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
;------------------------------------------------------------------
      .section        data_FE,DATA
      .org            10000H
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            0F0000H
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_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:
```

```
;-----------------------------------------------------------------
; Switch Table Section
;-----------------------------------------------------------------
      .section      switch_table,ROMDATA
switch_table_top:


;-----------------------------------------------------------------
; code area
;-----------------------------------------------------------------

      .section       program

      .section       interrupt
      ;.org   ;must be set internal ROM area
      .section       program_S

.if   __MVT__==0
;-----------------------------------------------------------------
; variable vector section
;-----------------------------------------------------------------
      .section       vector,ROMDATA  ; variable vector table
      .org    VECTOR_ADR

.if   M60TYPE == 1
      .lword  dummy_int              ; vector 0 (BRK)
      .lword  dummy_int              ; vector 1
      .lword  dummy_int              ; vector 2
      .lword  dummy_int              ; vector 3
      .lword  dummy_int              ; vector 4
      .lword  dummy_int              ; vector 5
      .lword  dummy_int              ; vector 6
      .lword  dummy_int              ; vector 7
      .lword  dummy_int              ; vector 8
      .lword  dummy_int              ; vector 9
      .lword  dummy_int              ; vector 10
      .lword  dummy_int              ; DMA0 (for user) (vector 11)
      .lword  dummy_int              ; DMA1 2 (for user) (vector 12)
      .lword  dummy_int              ; input key (for user) (vector 13)
      .lword  dummy_int              ; AD Convert (for user) (vector 14)
      .lword  dummy_int              ; vector 15
      .lword  dummy_int              ; vector 16
      .lword  dummy_int              ; uart0 trance (for user) (vector 17)
      .lword  dummy_int              ; uart0 receive (for user) (vector 18)
      .lword  dummy_int              ; uart1 trance (for user) (vector 19)
      .lword  dummy_int              ; uart1 receive (for user) (vector 20)
      .lword  dummy_int              ; TIMER A0 (for user) (vector 21)
      .lword  dummy_int              ; TIMER A1 (for user) (vector 22)
      .lword  dummy_int              ; TIMER A2 (for user) (vector 23)
      .lword  dummy_int              ; TIMER A3 (for user) (vector 24)
      .lword  dummy_int              ; TIMER A4 (for user) (vector 25)
      .lword  dummy_int              ; TIMER B0 (for user) (vector 26)
      .lword  dummy_int              ; TIMER B1 (for user) (vector 27)
      .lword  dummy_int              ; TIMER B2 (for user) (vector 28)
      .lword  dummy_int              ; INT0 (for user) (vector 29)
      .lword  dummy_int              ; INT1 (for user) (vector 30)
      .lword  dummy_int              ; INT2 (for user) (vector 31)
.else
      .lword  dummy_int              ; BRK   (vector 0)
      .lword  dummy_int              ;       (vector 1)
      .lword  dummy_int              ;       (vector 2)
      .lword  dummy_int              ;       (vector 3)
      .lword  dummy_int              ; int3(for user)(vector 4)
      .lword  dummy_int              ; timerB5(for user)(vector 5)
      .lword  dummy_int              ; timerB4(for user)(vector 6)
      .lword  dummy_int              ; timerB3(for user)(vector 7)
      .lword  dummy_int              ; si/o4 /int5(for user)(vector 8)
      .lword  dummy_int              ; si/o3 /int4(for user)(vector 9)
      .lword  dummy_int              ; Bus collision detection(for user)(v10)
```

```
        .lword  dummy_int               ; DMA0(for user)(vector 11)
        .lword  dummy_int               ; DMA1(for user)(vector 12)
        .lword  dummy_int               ; Key input interrupt(for user)(vect 13)
        .lword  dummy_int               ; A-D(for user)(vector 14)
        .lword  dummy_int               ; uart2 transmit(for user)(vector 15)
        .lword  dummy_int               ; uart2 receive(for user)(vector 16)
        .lword  dummy_int               ; uart0 transmit(for user)(vector 17)
        .lword  dummy_int               ; uart0 receive(for user)(vector 18)
        .lword  dummy_int               ; uart1 transmit(for user)(vector 19)
        .lword  dummy_int               ; uart1 receive(for user)(vector 20)
        .lword  dummy_int               ; timer A0(for user)(vector 21)
        .lword  dummy_int               ; timer A1(for user)(vector 22)
        .lword  dummy_int               ; timer A2(for user)(vector 23)
        .lword  dummy_int               ; timer A3(for user)(vector 24)
        .lword  dummy_int               ; timer A4(for user)(vector 25)
        .lword  dummy_int               ; timer B0(for user)(vector 26)
        .lword  dummy_int               ; timer B1(for user)(vector 27)
        .lword  dummy_int               ; timer B2(for user)(vector 28)
        .lword  dummy_int               ; int0 (for user)(vector 29)
        .lword  dummy_int               ; int1 (for user)(vector 30)
        .lword  dummy_int               ; int2 (for user)(vector 31)
.endif
        .lword  dummy_int               ; vector 32 (for user or MR30)
        .lword  dummy_int               ; vector 33 (for user or MR30)
        .lword  dummy_int               ; vector 34 (for user or MR30)
        .lword  dummy_int               ; vector 35 (for user or MR30)
        .lword  dummy_int               ; vector 36 (for user or MR30)
        .lword  dummy_int               ; vector 37 (for user or MR30)
        .lword  dummy_int               ; vector 38 (for user or MR30)
        .lword  dummy_int               ; vector 39 (for user or MR30)
        .lword  dummy_int               ; vector 40 (for user or MR30)
        .lword  dummy_int               ; vector 41 (for user or MR30)
        .lword  dummy_int               ; vector 42 (for user or MR30)
        .lword  dummy_int               ; vector 43 (for user or MR30)
        .lword  dummy_int               ; vector 44 (for user or MR30)
        .lword  dummy_int               ; vector 45 (for user or MR30)
        .lword  dummy_int               ; vector 46 (for user or MR30)
        .lword  dummy_int               ; vector 47 (for user or MR30)
        .lword  dummy_int               ; vector 48
        .lword  dummy_int               ; vector 49
        .lword  dummy_int               ; vector 50
        .lword  dummy_int               ; vector 51
        .lword  dummy_int               ; vector 52
        .lword  dummy_int               ; vector 53
        .lword  dummy_int               ; vector 54
        .lword  dummy_int               ; vector 55
        .lword  dummy_int               ; vector 56
        .lword  dummy_int               ; vector 57
        .lword  dummy_int               ; vector 58
        .lword  dummy_int               ; vector 59
        .lword  dummy_int               ; vector 60
        .lword  dummy_int               ; vector 61
        .lword  dummy_int               ; vector 62
        .lword  dummy_int               ; vector 63
.else ; __MVT__

        .section        __NC_rvector,ROMDATA
        .org            VECTOR_ADR

.endif ; __MVT__


.if   __MST__ == 0
;=============================================================
; fixed vector section
;-------------------------------------------------------------
        .section        svector,ROMDATA         ; specialpage vector table
        .org            SVECTOR_ADR
;=============================================================
```

```
; 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
.else
      .section        __NC_svector,ROMDATA
      .org            SVECTOR_ADR
.endif ; __MST
;
;=============================================================
; fixed vector section
;-------------------------------------------------------------
      .section        fvector,ROMDATA
;     .org    0fffdcH
;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
      .org    0ffffcH
RESET:
      .lword  start

.else ; __R8C__
```

```
;
; for R8C/Tiny
;
;------------------------------------------------------------
;
;     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
stack_top:

      .blkb   ISTACKSIZE
istack_top:

;------------------------------------------------------------
;     heap section
;------------------------------------------------------------
      .section        heap,DATA
heap_top:
      .blkb   HEAPSIZE

;------------------------------------------------------------
; Near ROM data area
;------------------------------------------------------------
      .section        rom_NE,ROMDATA
      .org            0e000H
rom_NE_top:

      .section        rom_NO,ROMDATA
rom_NO_top:

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

```
      .section        data_SOI,ROMDATA
data_SOI_top:

      .section        data_NEI,ROMDATA,ALIGN
data_NEI_top:

      .section        data_NOI,ROMDATA
data_NOI_top:

;------------------------------------------------------------------
; Switch Table Section
;------------------------------------------------------------------
      .section       switch_table,ROMDATA
switch_table_top:

;------------------------------------------------------------------
; code area
;------------------------------------------------------------------

      .section        program,CODE,ALIGN

      .section        interrupt,CODE,ALIGN

.if   __MVT__ == 0
;------------------------------------------------------------------
; variable vector section
;------------------------------------------------------------------
      .section        vector,ROMDATA  ; variable vector table
      .org    VECTOR_ADR

      .lword  dummy_int               ; vector 0
      .lword  dummy_int               ; vector 1
      .lword  dummy_int               ; vector 2
      .lword  dummy_int               ; vector 3
      .lword  dummy_int               ; vector 4
      .lword  dummy_int               ; vector 5
      .lword  dummy_int               ; vector 6
      .lword  dummy_int               ; vector 7
      .lword  dummy_int               ; vector 8
      .lword  dummy_int               ; vector 9
      .lword  dummy_int               ; vector 10
      .lword  dummy_int               ; vector 11
      .lword  dummy_int               ; vector 12
      .lword  dummy_int               ; vector 13
      .lword  dummy_int               ; vector 14
      .lword  dummy_int               ; vector 15
      .lword  dummy_int               ; vector 16
      .lword  dummy_int               ; vector 17
      .lword  dummy_int               ; vector 18
      .lword  dummy_int               ; vector 19
      .lword  dummy_int               ; vector 20
      .lword  dummy_int               ; vector 21
      .lword  dummy_int               ; vector 22
      .lword  dummy_int               ; vector 23
      .lword  dummy_int               ; vector 24
      .lword  dummy_int               ; vector 25
      .lword  dummy_int               ; vector 26
      .lword  dummy_int               ; vector 27
      .lword  dummy_int               ; vector 28
      .lword  dummy_int               ; vector 29
      .lword  dummy_int               ; vector 30
      .lword  dummy_int               ; vector 31
      .lword  dummy_int               ; vector 32
      .lword  dummy_int               ; vector 33
      .lword  dummy_int               ; vector 34
      .lword  dummy_int               ; vector 35
      .lword  dummy_int               ; vector 36
      .lword  dummy_int               ; vector 37
```

```
        .lword  dummy_int               ; vector 38
        .lword  dummy_int               ; vector 39
        .lword  dummy_int               ; vector 40
        .lword  dummy_int               ; vector 41
        .lword  dummy_int               ; vector 42
        .lword  dummy_int               ; vector 43
        .lword  dummy_int               ; vector 44
        .lword  dummy_int               ; vector 45
        .lword  dummy_int               ; vector 46
        .lword  dummy_int               ; vector 47
        .lword  dummy_int               ; vector 48
        .lword  dummy_int               ; vector 49
        .lword  dummy_int               ; vector 50
        .lword  dummy_int               ; vector 51
        .lword  dummy_int               ; vector 52
        .lword  dummy_int               ; vector 53
        .lword  dummy_int               ; vector 54
        .lword  dummy_int               ; vector 55
        .lword  dummy_int               ; vector 56
        .lword  dummy_int               ; vector 57
        .lword  dummy_int               ; vector 58
        .lword  dummy_int               ; vector 59
        .lword  dummy_int               ; vector 60
        .lword  dummy_int               ; vector 61
        .lword  dummy_int               ; vector 62
        .lword  dummy_int               ; vector 63
.else ; __MVT__
        .section        __NC_rvector,ROMDATA
        .org            VECTOR_ADR
.endif

;=====================================================================
; fixed vector section
;---------------------------------------------------------------------
        .section        fvector,ROMDATA         ; fixed vector table
;       .org    0ffdcH
;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
        .org    0fffcH
RESET:
        .lword  start

.endif; __R8C

;---------------------------------------------------------------------
; far ROM data area
;---------------------------------------------------------------------
;
;       .section        rom_FE,ROMDATA
;       .org            10000H
;
;       .section        rom_FO,ROMDATA
;
;       .section        data_FEI,ROMDATA,ALIGN
;data_FEI_top:
```

```
;
;     .section        data_FOI,ROMDATA
;data_FOI_top:
;
;***********************************************************************
;
;     C Compiler for R8C/Tiny, M16C/60,30,20,10
; COPYRIGHT(C) 1999(2000-2002) RENESAS TECHNOLOGY CORPORATION
; AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED
;
;***********************************************************************
```

5. Reference

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

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

Hardware Manual
M16C/62P Group Hardware Manual Rev.2.30
(Use the latest version on the home page: http://www.renesas.com)

6. REVISION HISTORY

Rev.	Date	Description	
		Page	Summary
1.00	2005.01.14	-	First edition issued

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.