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.

GRADE A

MESC TECHNICAL NEWS No.M16C-22-9904

MESC TECHNICAL NEWS
'No.M16C-20-9903' replace

MESC TECHNICAL NEWS 'No.M16C-20-9903' has an error, so we will correct.
Please replace old Technical News 'No.M16C-20-9903' to corrected Technical News 'M16C/62 (Flash Memory Version) Precautions for Power Consumption'.

[Correction contents]

Example 2 of countermeasures was flash memory 5 V version exclusive use. Therefore we added example 3 as flash memory 3 V version afresh.

[Attached]

Corrected Technical News 'No.M16C-22-9904'
'M16C/62 (Flash Memory Version) Precautions for Power Consumption' ..... 12 page

MESC TECHNICAL NEWS No.M16C-22-9904

M16C/62 (Flash Memory Version)
Precautions for Power Consumption

1. Related devices

Flash memory 5V version : M30624FGFP, M30624FGGP, M30625FGGP
Flash memory 3V version : M30624FGLFP, M30624FGLGP, M30625FGLGP

2. Precautions

[Low-speed mode]

Because the internal memory control method of the flash memory version differs from that of the mask ROM version, the device's power consumption does not fully drop even in low-speed mode (operating with XCIN with XIN turned off). Current consumption (typ.) during low-speed mode is listed below.

Flash memory 5V version at VCC=5V: Approx. 8mA standard (Mask ROM/One-time PROM version : 90 µA standard)
Flash memory 5V version at VCC=3V : Approx. 5mA standard (Mask ROM/One-time PROM version:40µA standard)
Flash memory 3V version at VCC=3V: Approx. 700µA standard (Mask ROM/One-time PROM version:40µA standard)

[High-speed mode, Middle-speed mode]

Because the internal memory control method of the flash memory version differs from that of the mask ROM version, the device's power consumption does not fully drop even when the CPU clock operating frequency is reduced.

3. Countermeasures

Example 1 : Using stop mode, wait mode (Flash memory 5 V version, 3 V version commonness)
Example 2 : Using a bit to reduce power consumption (Flash memory 5 V version only)
Example 3 : Using a program is executed with built-in RAM (Flash memory 3 V version only)

Example 1 : Using stop mode, wait mode (common)

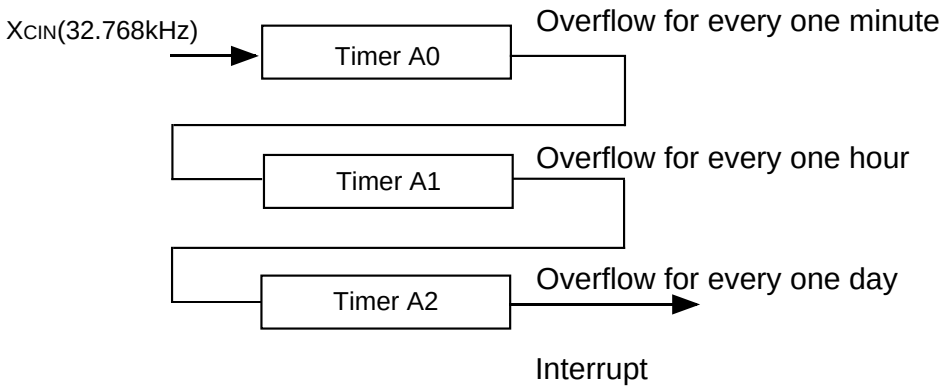
By entering stop or wait mode, it is possible to accomplish the low power consumption comparable to that of the mask ROM version without special settings on the flash memory control register. Please consider use of stop or wait mode. Especially when counting clock in wait mode, it is possible to count hours and even a day from XCIN.

(1) Content

- Timer A0 counts for up to 1 minute using XCIN as the count source.
- Timer A1 counts for up to 1 hour using timer A0 as the count source.
- Timer A2 counts for up to 1 day using timer A1 as the count source.

A setup example for each timer necessary to accomplish this is shown below. The interrupt is generated upon overflow of timer A2. Before executing the WAIT instruction, enable the interrupt for timer A2.

(2) Connection of timer



(3) Setting register

TA0MR (timer A0 mode register : Address 0396_{16})	=	0C0H
Timer A0 (timer A0 : Addresses 0387_{16}, 0386_{16})	=	0EFFFH
TA1MR (timer A1 mode register : Address 0397_{16})	=	001H
Timer A1 (timer A1 : Addresses 0389_{16}, 0388_{16})	=	0003BH
TA2MR (timer A2 mode register : Address 0398_{16})	=	001H
Timer A2 (timer A2 : Addresses $038B_{16}$, $038A_{16}$)	=	00017H
TRGSR (Trigger select register : Address 0383_{16})	=	00AH

Example 2. Using a bit to reduce power consumption(5V version only)

A new bit to reduce power consumption has been incorporated into the flash memory control register (Figure 1). Although setting this bit to "1" helps to reduce the device's power consumption, programs cannot be read from the internal flash memory. Make sure the operation to set this bit to "1" and other operations to be performed while this bit remains "1" are executed in areas outside flash memory.

Flash memory control register 2

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0		0	0

Symbol: FMCR2 | Address: $03B6_{16}$ | When reset: $XXXXX0XX_2$

Bit symbol	Bit name	Function	R	W
Reserved bit		Must always be set to "0"	—	○
FMCR22	Flash memory power supply-OFF bit(Note)	0: Flash memory power supply is connected 1: Flash memory power supply-off	○	○
Reserved bit		Must always be set to "0"	—	○

Note : For this bit to be set to "1", the user needs to write a "0" and then a "1" to it in succession. When it is not this procedure, it is not enacted in "1". This is necessary to ensure that no interrupt or DMA transfer will be executed during the interval. During parallel I/O mode,programming, erase or read of flash memory is not controlled by this bit,only by external pins.

Figure 1. Flash memory control register 2

A flow chart and a sample program for setting/resetting this bit are shown below.

(1) Program configuration

The sample program consists of the following:

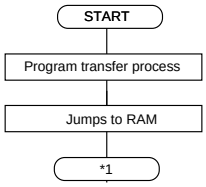
- **Program to realize low power consumption**
  This program sets the flash memory control register, switches over the clock, and executes the WAIT instruction. When a timer A0 interrupt occurs, it reexecutes the WAIT instruction; when an external interrupt occurs, the program exists from low-speed mode (flow chart 2).
- **Interrupt program to return from low-speed mode (INT0 interrupt)**
  When an external interrupt occurs, data is set in RAM.
- **Interrupt program to count clock (TA0 interrupt)**
  This program counts clock.
- **Program to transfer programs to RAM**
  This program transfers the above programs to RAM (flow chart 1).

(2) Precautions

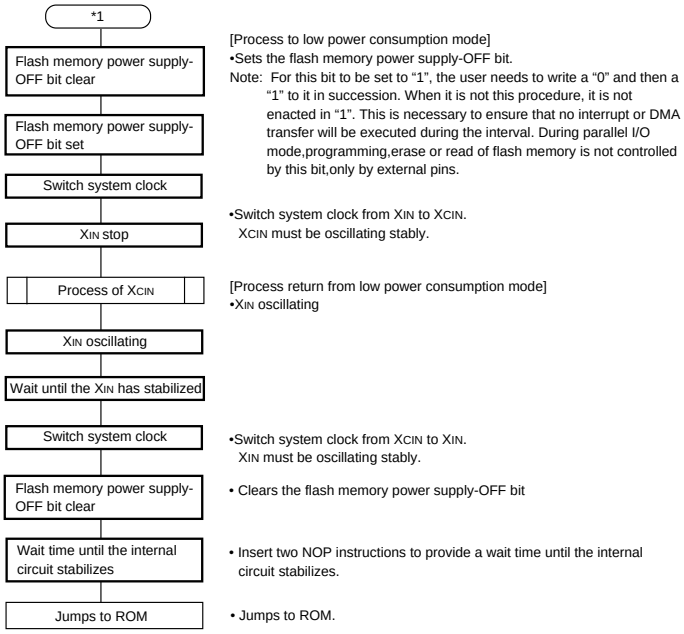
1. Before the interrupts can be used, the interrupt programs and interrupt vector table must also be transferred to RAM. In this case, be sure to transfer the interrupt programs and interrupt vector table before you modify the interrupt table register (INTB). Also, make sure the interrupts are disabled while you are modifying the interrupt table register (INTB).
2. The sample program only sets the flash memory control register 2 and switches over the clock. Other settings necessary to enter low-power consumption mode (e.g., port setting, interrupt enable/disable, Vref not connect) need to be performed by users according to each application system.
3. When installing the sample program, fully evaluate it on your system.

(3) Flowchart

A flow chart on ROM
(flow chart 1)



A flow chart on RAM
(flow chart 2)



(4) Sample program

```
;////////////////////////////////////////////////////////////////////////////
        RAM program transfer process
;////////////////////////////////////////////////////////////////////////////
PROG__MOVE:
        mov.w       #PROG_TOP&0FFFFh,A0
        mov.b       #PROG_TOP>>16,R1H
        mov.w       #Vram_P,A1
        mov.w       #(PROG_END-PROG_TOP),R3
        smovf.b
;////////////////////////////////////////////////////////////////////////////
        Interrupt program transfer process
;////////////////////////////////////////////////////////////////////////////
VECT_PROG__MOVE:
        mov.w       #VECT_PRG_TOP&0FFFFh,A0
        mov.b       #VECT_PRG_TOP>>16,R1H
        mov.w       #Vram_Vect,A1
        mov.w       #(VECT_PRG_END-VECT_PRG_TOP),R3
        smovf.b
```

```
;////////////////////////////////////////////////////////////////////////
;       Variable vector transfer process
;////////////////////////////////////////////////////////////////////////
VECT_TBL__MOVE:
        mov.w       #VECT_TBL_TOP&0FFFFh,A0
        mov.b       #VECT_TBL_TOP>>16,R1H
        mov.w       #Vram_VectT,A1
        mov.w       #(VECT_TBL_END-VECT_TBL_TOP),R3
        smovf.b
        fclr        i
        ldintb      #Vram_VectT             ; intb change
        nop
        nop
        fset        i
        jmp.a Vram_P                        ; ram program jump
;=====================================================================
;       RAM transfer program
;=====================================================================
PROG_TOP:
        mov.b       #00h,3B6h
        bset        2,3B6h                  ; Flash memory supply voltage off
        bset        PRC0
        bset        CM02
        bset        CM07                    ;CPU CLOCK = SUB
        bset        CM05                    ;Xin STOP
        mov.b       #11111111b,P1
?:
        wait                                ; program wait
        nop
        cmp.b       #1,INT_FLG
        jne         ?-
        mov.b       #0,INT_FLG
        bset        PRC0
        bclr        CM05                    ;Xin START
        bclr        CM07                    ;CPU CLOCK = MAIN
        mov.b       #00h,3B6h               ; Flash memory supply voltage on
        nop
        nop
        jmp.a       ROM_TEST_PRG_TOP
PROG_END:
```

```
;////////////////////////////////////////////////////////////////////////
;       Interrupt
;////////////////////////////////////////////////////////////////////////
VECT_PRG_TOP:
Dummy:
        REIT
;=======================================================================
;               Timer A0 interrupt (count of clock)
;=======================================================================
TA0i:
        mov.b       #1,ONE_SEC_FLG
        add.b       #1,SEC
        cmp.b       #60,SEC
        jne         CHK_end
        mov.b       #0,SEC
        add.b       #1,MIN
MIN_chk:
        cmp.b       #60,MIN
        jne         CHK_end
        mov.b       #0,MIN
        add.b       #1,HUR
CHK_end:
        REIT
;=======================================================================
;               External interrupt (return signal from low power consumption mode)
;=======================================================================
INT0i:
        mov.b       #0,INT_FLG
        btst        P8_2
        jne         INT0_end
        mov.b       #1,INT_FLG
INT0_end:
        REIT
VECT_PRG_END:
```

```
;==========================================================================
;       INT TABLE Area for RAM
;==========================================================================
VECT_TBL_TOP:
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+TA0i- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+INT0i- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
 VECT_TBL_END:
```

*Label in program

Vram_P	:Start address of RAM in which the execution program is stored.
Vram_Vect	:Start address of RAM in which the interrupt program is stored.
Vram_VectT	:Start address of RAM in which the interrupt vector table is stored.

Example 3. Using a program is executed with built-in RAM(3V version only)

(1) Program configuration

The sample program consists of the following:

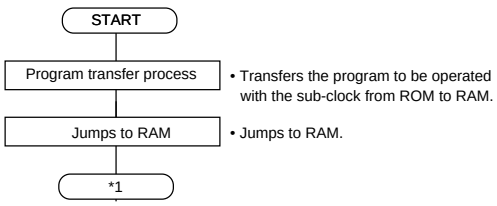
- **Program to realize low power consumption**
  This program switches over the clock, and executes the WAIT instruction. When a timer A0 interrupt occurs, it reexecutes the WAIT instruction; when an external interrupt occurs, the program exists from low-speed mode (flow chart 2).
- **Interrupt program to return from low-speed mode (INT0 interrupt)**
  When an external interrupt occurs, data is set in RAM.
- **Interrupt program to count clock (TA0 interrupt)**
  This program counts clock.
- **Program to transfer programs to RAM**
  This program transfers the above programs to RAM (flow chart 1).

(2) Precautions

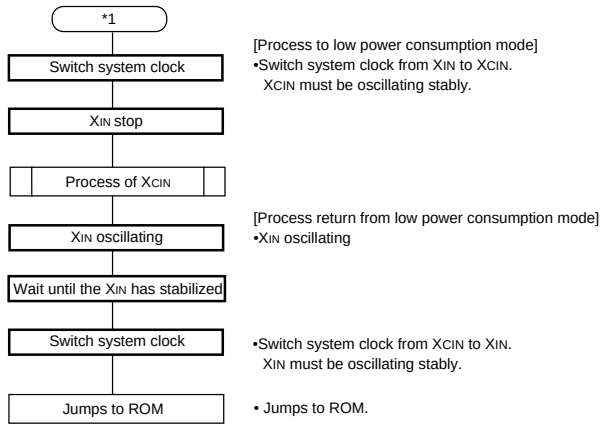
1. Before the interrupts can be used, the interrupt programs and interrupt vector table must also be transferred to RAM. In this case, be sure to transfer the interrupt programs and interrupt vector table before you modify the interrupt table register (INTB). Also, make sure the interrupts are disabled while you are modifying the interrupt table register (INTB).
2. The sample program only switches over the clock. Other settings necessary to enter low-power consumption mode (e.g., port setting, interrupt enable/disable, Vref not connect) need to be performed by users according to each application system.
3. When installing the sample program, fully evaluate it on your system.

(3) Flowchart

A flow chart on ROM
(flow chart 1)



A flow chart on RAM
(flow chart 2)



(4) Sample program

```
;////////////////////////////////////////////////////////////////////////////
        RAM program transfer process
;////////////////////////////////////////////////////////////////////////////
PROG__MOVE:
        mov.w       #PROG_TOP&0FFFFh,A0
        mov.b       #PROG_TOP>>16,R1H
        mov.w       #Vram_P,A1
        mov.w       #(PROG_END-PROG_TOP),R3
        smovf.b
;////////////////////////////////////////////////////////////////////////////
        Interrupt program transfer process
;////////////////////////////////////////////////////////////////////////////
VECT_PROG__MOVE:
        mov.w       #VECT_PRG_TOP&0FFFFh,A0
        mov.b       #VECT_PRG_TOP>>16,R1H
        mov.w       #Vram_Vect,A1
        mov.w       #(VECT_PRG_END-VECT_PRG_TOP),R3
        smovf.b
;////////////////////////////////////////////////////////////////////////////
;       Variable vector transfer process
;////////////////////////////////////////////////////////////////////////////
VECT_TBL__MOVE:
        mov.w       #VECT_TBL_TOP&0FFFFh,A0
        mov.b       #VECT_TBL_TOP>>16,R1H
        mov.w       #Vram_VectT,A1
        mov.w       #(VECT_TBL_END-VECT_TBL_TOP),R3
        smovf.b
        fclr        i
        ldintb      #Vram_VectT             ; intb change
        nop
        nop
        fset        i
        jmp.a       Vram_P                  ; ram program jump
```

```
;=====================================================================
;       RAM transfer program
;=====================================================================
PROG_TOP:
        bset        PRC0
        bset        CM02
        bset        CM07                ;CPU CLOCK = SUB
        bset        CM05                ;Xin STOP
        mov.b       #11111111b,P1
?:
        wait                            ; program wait
        nop
        cmp.b       #1,INT_FLG
        jne         ?-
        mov.b       #0,INT_FLG
        bset        PRC0
        bclr        CM05                ;Xin START
        bclr        CM07                ;CPU CLOCK = MAIN
        jmp.a       ROM_TEST_PRG_TOP
PROG_END:
```

```
;////////////////////////////////////////////////////////////////////////
;       Interrupt
;////////////////////////////////////////////////////////////////////////
VECT_PRG_TOP:
Dummy:
        REIT
;=====================================================================
;               Timer A0 interrupt (count of clock)
;=====================================================================
TA0i:
        mov.b       #1,ONE_SEC_FLG
        add.b       #1,SEC
        cmp.b       #60,SEC
        jne         CHK_end
        mov.b       #0,SEC
        add.b       #1,MIN
MIN_chk:
        cmp.b       #60,MIN
        jne         CHK_end
        mov.b       #0,MIN
        add.b       #1,HUR
CHK_end:
        REIT
;=====================================================================
;               External interrupt (return signal from low power consumption mode)
;=====================================================================
INT0i:
        mov.b       #0,INT_FLG
        btst        P8_2
        jne         INT0_end
        mov.b       #1,INT_FLG
INT0_end:
        REIT
VECT_PRG_END:
```

```
;======================================================================
;       INT TABLE Area for RAM
;======================================================================
VECT_TBL_TOP:
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+TA0i- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+INT0i- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
        .LWORD    (Vram_Vect+Dummy- VECT_PRG_TOP)
 VECT_TBL_END:
```

*Label in program

Vram_P	:Start address of RAM in which the execution program is stored.
Vram_Vect	:Start address of RAM in which the interrupt program is stored.
Vram_VectT	:Start address of RAM in which the interrupt vector table is stored.