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/Tiny Series

Operation of Watchdog Timer

1. Abstract

This application note describes how to use watchdog timer of the M16C/Tiny series microcomputers.

2. Introduction

The explanation of this issue is applied to the following condition:

- MCU: M16C/26A Group
  M16C/28 Group
  M16C/29 Group

This program can be operated under the condition of M16C family products with the same SFR (Special Function Register) as 26A, 28, 29 group products. Because some functions may be modified of the M16C family products, see the user's manual. When using the functions shown in this application note, evaluate them carefully for an operation.

3. Specifications

3.1 Operation

(1) Writing to the watchdog timer start register initializes the watchdog timer to “7FFFh” and causes it to start a down count.
(2) With the watchdog timer’s counting in progress, writing to the watchdog timer start register again initializes the watchdog timer to “7FFFh” and causes it to resume counting.
(3) Either executing the WAIT instruction or going to the stopped state causes the watchdog timer to stop counting and to hold the current value of counter. The watchdog timer resumes counting after returning from the execution of the WAIT instruction or from the stopped state.
(4) If the watchdog timer underflows, it is initialized to “7FFFh” and continues counting. Simultaneously, a watchdog timer interrupt occurs.

Notes:

- The watchdog timer and the prescaler both are inactive after reset, so that the watchdog timer is activated to start counting by writing to the WDTS register. Write the WDTS register with shorter cycle than the watchdog timer cycle in order not to generate any watchdog timer interrupt. Set the WDTS register also in the beginning of the watchdog timer interrupt routine.
- If the watchdog timer function select bit in Processor Mode Register 1 (PM1 register) is set to 1 (watchdog timer reset), the pins, CPU, and SFR are initialized when a watchdog timer interrupt occurs, and the program is executed from the address indicated by the reset vector.

Figure 1 shows the operation timing of watchdog.

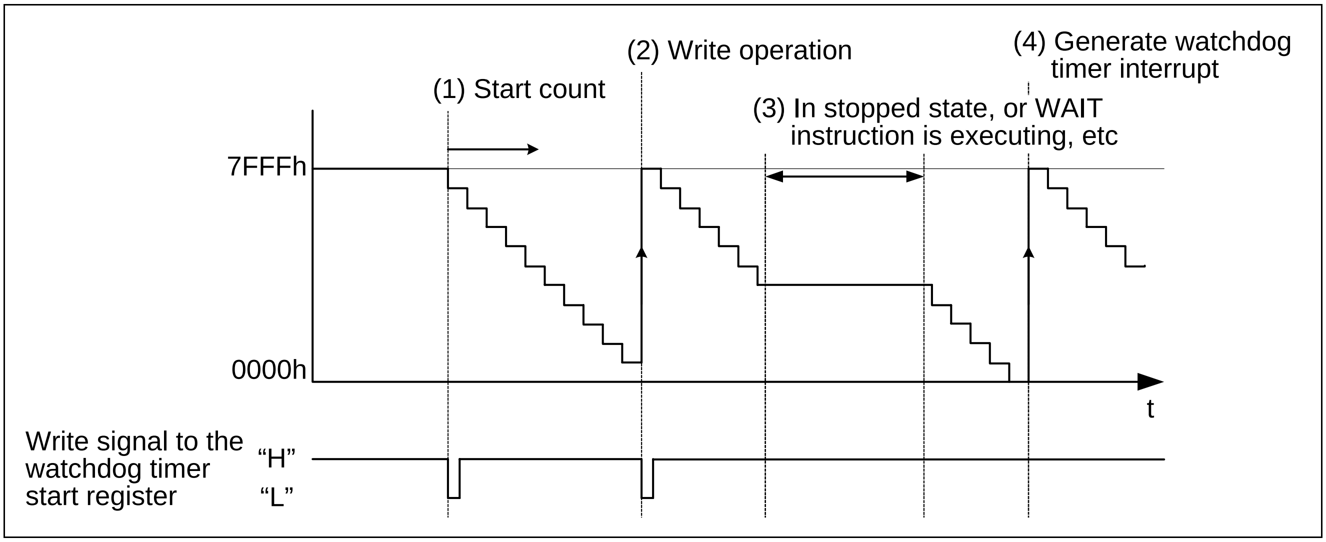
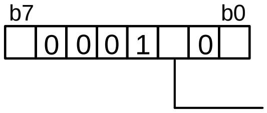

(2) Write operation
(4) Generate watchdog timer interrupt
(1) Start count
(3) In stopped state, or WAIT instruction is executing, etc
7FFFh
0000h
t
Write signal to the watchdog timer start register
“H”
“L”

Figure 1. Operation Timing of Watchdog

3.2 Register setting

The following procedure in this application note is based on M16C/29 group products, other M16C/Tiny series's setup procedure please refer to the hardware user's manual.

(1) Setting watchdog timer function select bit (Note 1)

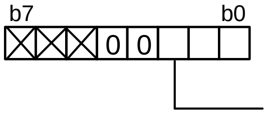


Processor mode register 1 [Address 0005h] PM1

Watchdog timer function select bit
0 : Watchdog timer interrupt
1 : Watchdog timer reset (Note 2)

Note 1: Write to this register after setting the PRC1 bit in the PRCR register to "1" (write enable).
Note 2: PM12 bit is set to "1" by writing "1" in a program. (Writing "0" has no effect.)

(2) Setting WDT count source protective bit

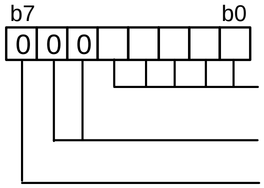


Processor mode register 2 [Address 001Eh] PM2

WDT count source protective bit (Note 1, 2)
0 : Select CPU clock
1 : Select On-chip oscillator clock

Note 1: Once this bit is set to "1", it cannot be cleared to "0" in a program.
Note 2: Setting PM22 bit to "1" results in the following condition:
- The on-chip oscillator clock becomes the WDT count source.
- The CM10 bit of CM1 register is disabled against write. (writing "1" has no effect, nor is stop mode entered.)
- The WDT does not stop when in wait mode.

(3) Setting watchdog timer control register



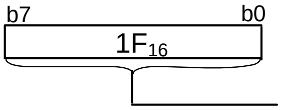
Watchdog timer control register [Address 000Fh] WDC

High-order bit of watchdog timer

Must set to "0"

Prescaler select bit
0 : Divided by 16
1 : Divided by 128

(4) Setting watchdog timer start register



Watchdog timer start register [Address 000Eh] WDTS

The watchdog timer is initialized and starts counting after a write instruction to this register. The watchdog timer value is always initialized to "7FFFh" regardless of whatever value is written.

4. The example of reference program

Figure 2 shows the sample circuit of reference program.

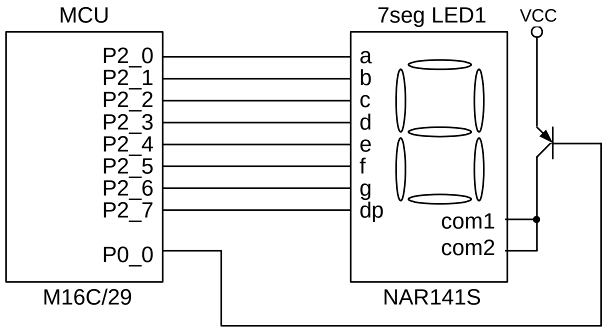


Figure 2. Sample Circuit of Reference Program

4.1 Using the watchdog timer interrupt program

While this program writes to the watchdog timer start register, it increases the indication of port P2. When the output of port P2 reaches “9”, the program stops writing to the watchdog timer start register to stop updating the indication of port P2.

When a watchdog timer interrupt occurs, the program writes to the watchdog timer start register while it decreases the indication of port P2 in a watchdog timer interrupt service routine. When the output of port P2 reaches “0”, the program stops updating the indication of port P2.

```
/********************************************************/
/*                                                      */
/* M16C/Tiny Series Program Collection                  */
/*                                                      */
/* FILE NAME    : rec05b0006-0102_int.c                 */
/* CPU          : M16C/29 Group                         */
/* FUNCTION     : Operation of Watchdog Timer           */
/* HISTORY      : 2006.04.13 Ver 1.02                   */
/*                                                      */
/* Copyright (C) 2006. Renesas Technology Corp.         */
/* All right reserved.                                  */
/********************************************************/

/********************************************************/
/*    Include File                                      */
/********************************************************/
#include "sfr29.h"       // Special function register header file

/********************************************************/
/*    Function Definition                               */
/********************************************************/
void init_mcu(void);     // SFR initialize
void wait_10ms(void);    // Main clock oscillation stable wait routine
#pragma INTERRUPT wdt_int
```

```
/*******************************************************/
/*    Define Label                                     */
/*******************************************************/
#define PRODUCT_TYPE 0   // 28,29 group: 0   26A group: 1
#define PIN_TYPE 0       // 80 pin: 0        64 pin: 1 (28,29 group)
                         // 48 pin: 0        42 pin: 1 (26A group)

/*******************************************************/
/*    Define Const                                     */
/*******************************************************/
/* port2_0 - port2_7 data : 0  1  2  3  4  5  6  7  8  9 */
unsigned char count_data[10] =
{0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xD8,0x80,0x90};

unsigned int i,j;

/*******************************************************/
/*    Main Program                                     */
/*******************************************************/
void main(void)
{
   init_mcu();               // SFR initialize

   j = 0;
   i = 0;

   wdc = 0;                  // Setting watchdog timer control register
                             // Prescaler select bit is set to 0 (0: divided by 16)

   wdts = 1;                 // Setting watchdog timer start register

   while (1)
   {
      while (ir_ta0ic == 0); // 1ms?

      ta0ic = 0x00;          // Timer A0 interrupt level: 0
      i++;

      if ( i == 500 )
      {
         i = 0;
         if ( j <= 9 )
         {
            p2 = count_data[j];
         }
         j++;
      }
      if ( j <= 9 )
      {
         wdts = 1;           // Setting watchdog timer start register
      }
   }
}
```

```
/*******************************************************/
/*    Watchdog Timer Interrupt Routine                 */
/*******************************************************/
void wdt_int()
{
   wdts = 1;                    // Set the WDTS register in the beginning of the
                                // watchdog timer interrupt routine.

   while (1)
   {
      wdts = 1;                 // For long-time loop, set WDTS register again to
                                // avoid the underflow of watchdog timer.

      while (ir_ta0ic == 0);    // 1ms?

      ta0ic = 0x00;             // Timer A0 interrupt level: 0
      i++;

      if ( i == 500 )
      {
         i = 0;
         if ( j != 0 )
         {
            j--;
            if ( j <= 9 )
            {
               p2 = count_data[j];
            }
         }
      }
   }
}

/*******************************************************/
/*    MCU Initialize                                   */
/*******************************************************/
void init_mcu(void)
{
   prcr = 0x03;      // Protect register off

                     // Set processor mode registers
   pm0 = 0x00;       // Single-chip mode
   pm1 = 0x08;       // No expansion, No wait

   wait_10ms();      // Waiting for main clock oscillation stable

                     // Set system clock control registers
   cm2 = 0x00;       // System clock select Main clock or PLL clock
   cm1 = 0x20;       // Xin-Xout High, Main clock is no division
   cm0 = 0x08;       // Xcin-Xcout drive capacity select bit (1: high)

   pclkr = 0x03;     // TimerA, B clock select bit (1: f1)
```

```
                     // WDT clock setting
   pm22 = 0;         // Set WDT count source protective bit
                     // <PM22>  : select CPU clock
                     // 0: select CPU clock

                     // Set WDT function select bit
   pm12 = 0;         // <PM12>  :0: WDT interrupt;  1: WDT reset
                     // (writing a 1 by program, writing a 0 has no effect)

   prcr = 0x00;      // Protect register on

   #if PRODUCT_TYPE      // Product selection: 26A group
      ifsr2a = 1;        // Interrupt request cause select register2 IFSR2A
                         // <IFSR20>   : Reserved bit (Must be set to "1")
      prcr = 0x04;       // Protect register off
      #if PIN_TYPE       // Port setting
         pacr = 0x01;    // 42pin type
      #else
         pacr = 0x04;    // 48pin type
      #endif
      prcr = 0x00;       // Protect register on
   #else                 // Product selection: 28,29 group
      ifsr2a = 0;        // Interrupt request cause select register2 IFSR2A
                         // <IFSR20>   : Reserved bit (Must be set to "0")
      prcr = 0x04;       // Protect register off
      #if PIN_TYPE       // Port setting
         pacr = 0x02;    // 64pin type
      #else
         pacr = 0x03;    // 80pin type
      #endif
      prcr = 0x00;       // Protect register on
   #endif

   p0 = 0x02;        // Select LED1
   p2 = 0;           // Port2 output
   pd0 = 0xff;       // Port direction0: output mode
   pd2 = 0xff;       // Port direction2: output mode

                     // Timer A0 setup
   ta0mr = 0x40;     // Selection of timer mode
                     // Pulse output function select bit (0: pulse is not output)
                     // Gate function select bit (00: gate function not available)
                     // Count source (01: f8)
   ta0 = 2500-1;     // Setting counter value (1msec @20MHz, f8)
   ta0ic = 0x00;     // Setting interrupt priority levels in timer A0
   ta0s = 1;         // Timer A0 count start
}

/*******************************************************/
/*    Main Clock Oscillation Stable Wait 10ms Routine  */
/*******************************************************/
void wait_10ms(void)
```

```
{
   ta0mr = 0x00;     // Set Timer A0 mode register (Timer mode, count source: f1)

   ta0 = 20000-1;    // Setting counter value (10msec @4MHz/2, f1)

   ta0ic = 0x00;     // Clear interrupt request bit

   tabsr = 0x01;     // Timer A0 start counting

   while (ir_ta0ic == 0){    }

   ir_ta0ic = 0;     // Clear interrupt request bit

   tabsr = 0x00;     // Timer A0 stops counting
}
```

In order for this program to run properly, the watchdog timer interrupt vector needs to point to the service routines for the interrupt. The interrupt vector table information is included in the startup file "sect30.inc". Insert the function label " _wdt_int " into the interrupt vector table locations as shown below.

```
;/******************************************************************
; C Compiler for R8C/Tiny, M16C/60, 30, 20, 10
; Copyright(C) 1999(2000-2004). Renesas Technology Corp.
; and Renesas Solutions Corp., 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 1.23.4.6 2004/10/29 14:06:39 simomura Exp $
;
;*******************************************************************/

;===================================================================
; fixed vector section
;-------------------------------------------------------------------
   .section fvector,ROMDATA
   .org  0fffdcH
   .glb _wdt_int
;UDI:
;  .lword   dummy_int
;OVER_FLOW:
;  .lword   dummy_int
;BRKI:
;  .lword   dummy_int
;ADDRESS_MATCH:
;  .lword   dummy_int
;SINGLE_STEP:
;  .lword   dummy_int
   .org  0ffff0H
WDT:
   .lword   _wdt_int
;DBC:
;  .lword   dummy_int
;NMI:
;  .lword   dummy_int
   .org  0ffffcH
RESET:
   .lword   start
;*******************************************************************
```

4.2 Using the watchdog timer interrupt to reset program

When the watchdog timer underflows after reaching terminal count, the program is reset and writes to the watchdog timer start register while it increases the indication of port P2. When the output of port P2 reaches “9”, the program stops updating the indication of port P2. Because the program stops writing to the watchdog timer start register, the program will be reset soon.

```
/*******************************************************/
/*                                                     */
/* M16C/Tiny Series Program Collection                 */
/*                                                     */
/* FILE NAME   : rec05b0006-0102_rst.c                 */
/* CPU         : M16C/29 Group                         */
/* FUNCTION    : Operation of Watchdog Timer           */
/* HISTORY     : 2006.04.13 Ver 1.02                   */
/*                                                     */
/* Copyright (C) 2006. Renesas Technology Corp.        */
/* All right reserved.                                 */
/*                                                     */
/*******************************************************/

/*******************************************************/
/*    Include File                                     */
/*******************************************************/
#include "sfr29.h"       // Special function register header file

/*******************************************************/
/*    Function Definition                              */
/*******************************************************/
void init_mcu(void);     // SFR initialize
void wait_10ms(void);    // Main clock oscillation stable wait routine

/*******************************************************/
/*    Define Label                                     */
/*******************************************************/
#define PRODUCT_TYPE 0   // 28,29 group: 0   26A group: 1
#define PIN_TYPE 0       // 80 pin: 0        64 pin: 1 (28,29 group)
                         // 48 pin: 0        42 pin: 1 (26A group)

/*******************************************************/
/*    Define Const                                     */
/*******************************************************/
/* port2_0 - port2_7 data : 0  1  2  3  4  5  6  7  8  9 */
unsigned char count_data[10] =
{0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xD8,0x80,0x90};

unsigned int i,j;

/*******************************************************/
/*    Main Program                                     */
/*******************************************************/
void main(void)
{
   init_mcu();                // SFR initialize

   j = 0;
   i = 0;

   wdc = 0;                   // Setting watchdog timer control register
                              // Prescaler select bit is set to 0 (0: divided by 16)

/* wdt reset   */
   wdts = 1;                  // Setting watchdog timer start register

   while (1)
   {
      while (ir_ta0ic == 0); // 1ms?

      ta0ic = 0x00;           // Timer A0 interrupt level: 0
      i++;

      if ( i == 500 )
      {
         i = 0;
         if ( j <= 9 )
         {
            p2 = count_data[j];
         }
         j++;
      }
      if ( j <= 9 )
      {
         wdts = 1;            // Setting watchdog timer start register
      }
   }
}

/*******************************************************/
/*    MCU Initialize                                   */
/*******************************************************/
void init_mcu(void)
{
   prcr = 0x03;      // Protect register off

                     // Set processor mode registers
   pm0 = 0x00;       // Single-chip mode
   pm1 = 0x08;       // No expansion, No wait

   wait_10ms();      // Waiting for main clock oscillation stable

                     // Set system clock control registers
   cm2 = 0x00;       // System clock select Main clock or PLL clock
   cm1 = 0x20;       // Xin-Xout High, Main clock is no division
   cm0 = 0x08;       // Xcin-Xcout drive capacity select bit (1: high)

   pclkr = 0x03;     // TimerA, B clock select bit (1: f1)
```

```
                     // WDT clock setting
   pm22 = 0;         // Set WDT count source protective bit
                     // <PM22>  : select CPU clock
                     // 0: select CPU clock

                     // Set WDT function select bit
   pm12 = 1;         // <PM12>  :0: WDT interrupt;  1: WDT reset
                     // (writing a 1 by program, writing a 0 has no effect)

   prcr = 0x00;      // Protect register on

   #if PRODUCT_TYPE      // Product selection: 26A group
      ifsr2a = 1;        // Interrupt request cause select register2 IFSR2A
                         // <IFSR20>   : Reserved bit (Must be set to "1")
      prcr = 0x04;       // Protect register off
      #if PIN_TYPE       // Port setting
         pacr = 0x01;    // 42pin type
      #else
         pacr = 0x04;    // 48pin type
      #endif
      prcr = 0x00;       // Protect register on
   #else                 // Product selection: 28,29 group
      ifsr2a = 0;        // Interrupt request cause select register2 IFSR2A
                         // <IFSR20>   : Reserved bit (Must be set to "0")
      prcr = 0x04;       // Protect register off
      #if PIN_TYPE       // Port setting
         pacr = 0x02;    // 64pin type
      #else
         pacr = 0x03;    // 80pin type
      #endif
      prcr = 0x00;       // Protect register on
   #endif

   p0 = 0x02;        // Select LED1
   p2 = 0;           // Port2 output
   pd0 = 0xff;       // Port direction0: output mode
   pd2 = 0xff;       // Port direction2: output mode

                     // Timer A0 setup
   ta0mr = 0x40;     // Selection of timer mode
                     // Pulse output function select bit (0: pulse is not output)
                     // Gate function select bit (00: gate function not available)
                     // Count source (01: f8)
   ta0 = 2500-1;     // Setting counter value (1msec @20MHz, f8)
   ta0ic = 0x00;     // Setting interrupt priority levels in timer A0
   ta0s = 1;         // Timer A0 count start
}

/*******************************************************/
/*    Main Clock Oscillation Stable Wait 10ms Routine   */
/*******************************************************/
void wait_10ms(void)
```

```
{
   ta0mr = 0x00;     // Set Timer A0 mode register (Timer mode, count source: f1)

   ta0 = 20000-1;    // Setting counter value (10msec @4MHz/2, f1)

   ta0ic = 0x00;     // Clear interrupt request bit

   tabsr = 0x01;     // Timer A0 start counting

   while (ir_ta0ic == 0){    }

   ir_ta0ic = 0;     // Clear interrupt request bit

   tabsr = 0x00;     // Timer A0 stops counting
}
```

5. Reference

Renesas web-site
http://www.renesas.com/

Inquiry
http://www.renesas.com/inquiry
csc@renesas.com

Hardware manual
M16C/26A Group (M16C/26A, M16C/26T) Hardware Manual Rev.1.00
M16C/28 Group Hardware Manual Rev.1.01
M16C/28 Group (T-ver./V-ver.) Hardware Manual Rev.1.00
M16C/29 Group Hardware Manual Rev.1.00
(Use the latest version on the web-site: http://www.renesas.com)

Technical update/Technical news
(Use the latest version on the web-site: http://www.renesas.com)

Revision

Rev.	Issue data	Description	
		Page	Summary
1.00	Dec.23.05	-	First edition issued
1.01	Jan.25.06	-	Sample program modified: define PRODUCT_TYPE
1.02	Apr.14.06	-	Modified function “wait_10ms” in sample program

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.