To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

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

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

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

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

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

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

H8/38076F

An Example of a Simple Clock

Introduction

The Renesas H8/38076F is a new microcontroller of the Super Low Power family, introducing the 16-bit core H8/300H in a series of devices up to now distinguished to have 8-bit core and low power consumption for battery powered application.
This application note provides example C code to initialise and configure the RTC and the LCD Controller found on the H8/38076F. The code is written for the Renesas MCS C compiler and was developed using the High-Performance Embedded Workshop (HEW). The code was debugged using the E7 emulator with the debugger integrated in the HEW.

Contents

Preface

The H8/38076F is suitable for a wide range of applications where performance and low power consumption are the key requirements. Based around the 16-Bit H8/300H CPU core, the H8/38076F features 32 Kbytes of single supply FLASH memory, a host of integrated peripherals and can be clocked at a maximum of 10MHz. In the picture 1 is shown the block diagram.

This microcontroller, like all the SLP devices, supports power down modes such as Subactive mode, Sleep mode, Subsleep and watch mode, that are used in the application here described.

The E7 used for debugging is a low-cost emulation tool for the SLP and H8/300H Tiny families, and the HEW used is the version 3.

The objective of this application note is to produce example C code, which can be used to demonstrate how to initialise and manage the RTC, the LCD Controller and a couple of low power modalities.

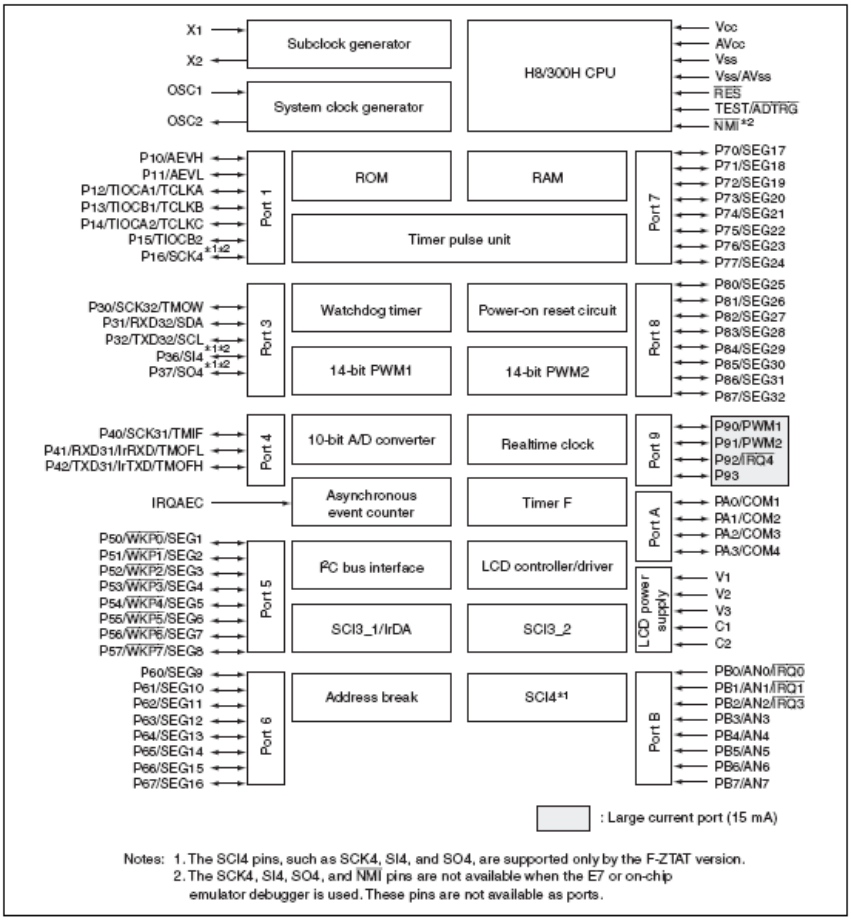

Figure 1 – H8/38076F block diagram

Description of Hardware

1. Description of RTC

The realtime clock (RTC) is a timer used to count time ranging from a second to a week. It is possible to program it to be used as free funning counter or as clock.

The main features are:

- counts seconds, minutes, hours, and day-of-week, with start/stop and reset function
- possibility to read/write the counters with BCD codes
- Interrupts can be generated ranging from 0.25 seconds to a week

The RTC has the following registers.

- Second data register/free running counter data register (RSECDR)
- Minute data register (RMINDR)
- Hour data register (RHRDR)
- Day-of-week data register (RWKDR)
- RTC control register 1 (RTCCR1)
- RTC control register 2 (RTCCR2)
- Clock source select register (RTCCSR)
- RTC Interrupt flag register (RTCFLG)

Figure 2 shows the procedure for the initial setting of the RTC. To change the setting of the RTC also follow this procedure.

When the second, minute, hour, or day-of-week data is set, check the BSY bit. When the BSY bit is cleared to 0, clear the RUN bit in RTCCR1 to 0 to stop the RTC operation.

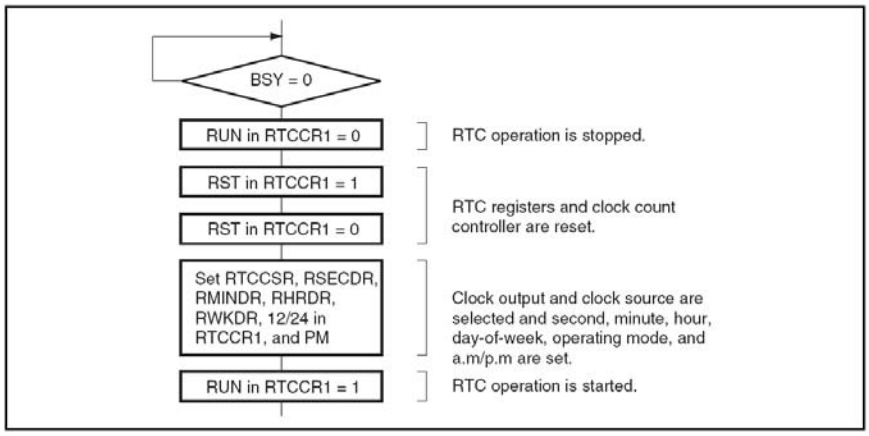

Figure 2 – Initial setting procedure

All the registers containing the time datum have a bit named BSY that is asserted during the update of the register and is reset after the completion of the update. For reading procedure is necessary to check the BSY bit before read in order to have the right data.

The routine read_time_from_rtc() detailed later shows an example.

2. Description of LCD Controller

This LSI has an on-chip peripheral able to directly drive an LCD panel. In this circuit are present the segment-type LCD control circuit, the LCD driver and the power supply circuit.

The RAM used for LCD is on chip and is 16 bytes long. All the segments not used for the display can be selected as I/O port, grouped by 4. The refresh frequency is selectable in a range of 11 available.

For this application has been used a 32x4 display, described in the next paragraph, but only 24x4 segments are connected and driven since in the EDK38076 the first 8 of these segments are connected to LEDs (non used for the application but in any case available). The initialisation sequence of the peripheral is the following:

```
SYSTEM.CKSTPR2.BIT.LDCKSTP = 1;        // accendo periferica
LCD.LPCR.BYTE = 0xea;
LCD.LCR.BYTE = 0xf2;
LCD.LCR2.BIT.LCDAB = 0;       / waveform A
LCD.LCR2.BIT.CHG = 1;
```

where:

- LDCKSTP switch on the peripheral.
- LPCR selects the duty cycle (DTS + CMX -> ¼ duty) and the pin functions (SEG1 to SEG8 used as I/O ports, SEG9 to SEG32 as segments driver).

DTS1	DTS0	CMX	-	SGS3	SGS2	SGS1	SGS0
1	1	1	0	1	0	1	0

- LCR controls LCD drive power supply (PSW 1 -> LCD drive power supply is turned on and ACT 1 -> LCD controller/driver operates) and display data, and selects the frame frequency (based on subclock/4).

-	PSW	ACT	DISP	CKS3	CKS2	CKS1	CKS0
1	1	1	1	0	0	1	0

- LCR2 controls switching between the A (selected) and B waveform, selection of the step-up clock for the 3-V constant-voltage circuit, connection with the LCD power-supply split resistor (selected), and turning on or off 3-V constant-voltage power supply.

3. Description of display

To realise this application has been used the 3DK38076 connected to an external LCD, detailed in figure 3. The LCD is a Varitronix VIM-828-DP (documentation can be downloaded from www.varitronix.com) with 8 character (14 Segment Starburst) where only six are used.

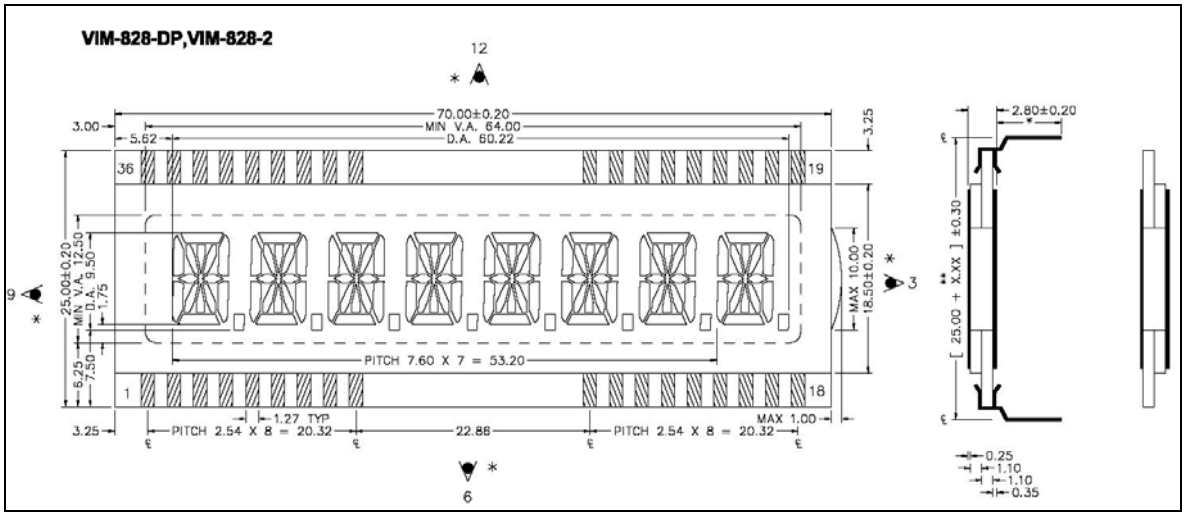

Fig.3 - LCD

The LCD is directly connected to the microcontroller, in particular the segments from SEG8 to SEG32 were used. This choice was due to the 3DK connections (on the first 8 segments are connected LEDs). Since the LCD used is a 32 segments x 4 commons, the first two characters are not connected. In any case, only six numbers are necessary, because the time is expressed in the notation:

hh.mm.ss

The LCD RAM is mapped as shown in the following table, and the figure 4 shows as the segments of single character are internally connected.

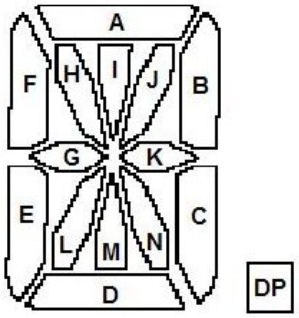

Figure 4 – digit configuration

LCD RAM address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F740	N.U.	N.U.	N.U.	N.U.	N.U.	N.U.	N.U.	N.U.
F741	N.U.	N.U.	N.U.	N.U.	N.U.	N.U.	N.U.	N.U.
F742	N.U.	N.U.	N.U.	N.U.	N.U.	N.U.	N.U.	N.U.
F743	N.U.	N.U.	N.U.	N.U.	N.U.	N.U.	N.U.	N.U.
F744	1M	1L	1G	1H	1DP	1C	1B	1A
F745	1N	1K	1J	1I	1D	1E	1F	-
F746	2M	2L	2G	2H	2DP	2C	2B	2A
F747	2N	2K	2J	2I	2D	2E	2F	-
F748	3M	3L	3G	3H	3DP	3C	3B	3A
F749	3N	3K	3J	3I	3D	3E	3F	-
F74A	4M	4L	4G	4H	4DP	4C	4B	4A
F74B	4N	4K	4J	4I	4D	4E	4F	-
F74C	5M	5L	5G	5H	5DP	5C	5B	5A
F74D	5N	5K	5J	5I	5D	5E	5F	-
F74E	6M	6L	6G	6H	6DP	6C	6B	6A
F74F	6N	6K	6J	6I	6D	6E	6F	-
	COM4	COM3	COM2	COM1	COM4	COM3	COM2	COM1

Description of the application

The schematics of the application are shown in figure 6.

After the initialisation of the internal registers and internal peripherals, the microcontroller is driven in watch mode in order to have very low power consumption.

The RTC is running linked with the subclock, as the LCD Controller.

Two different interrupt sources are enabled: the RTC (half second and second alternatively) and the IRQ4.

The interrupt of RTC is used to refresh the value of time on the display: during the initialization phase is enabled the 0.5 second interrupt. The ISR blinks the points between hours, minutes and seconds, disables this interrupt and enables the 1 second interrupt. This interrupt set a flag used by the main to read the time from RTC and update the display. After this the micon is driven in watch mode another time.

So the application is for most of the time in watch mode, and the power consumption (measured with a multimeter) is varying from 1.8 to 2.2 microamps without display and from 2.9 up to 5 microamps with display segments driven (the figure is strictly dependent by number of segments ON).

Occurrence of interrupts drive the microcontroller from watch to subactive mode, and the power consumption rises dramatically, up to 17 microamps.

The timing of the application is reported in figure 5:

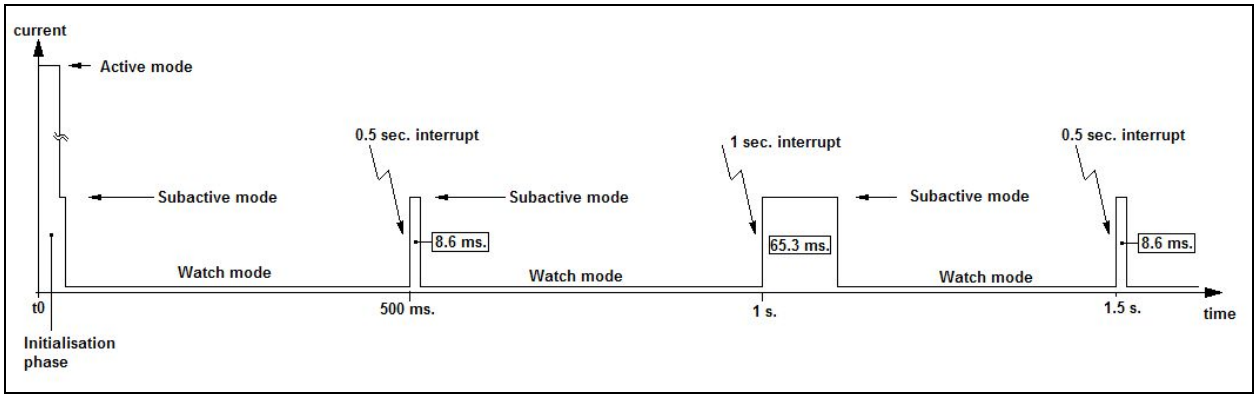

Figure 5 - Timing

As said above, there is another interrupt enabled that is able to switch the micon in subactive mode. In order to permit the programming of time, a switch connected to IRQ4 pin is available to select this working phase.

In this phase the microcontroller runs in subactive mode and on the display is shown the present time (without seconds) and the letter „M“ or „H“ indicating if the figure modifiable (pressing the button connected to IRQAEC) is minutes or hours.

The first one is used to refresh the value of time on the display, and the second one is used to select the programming mode, in order to set up the new time for the RTC.

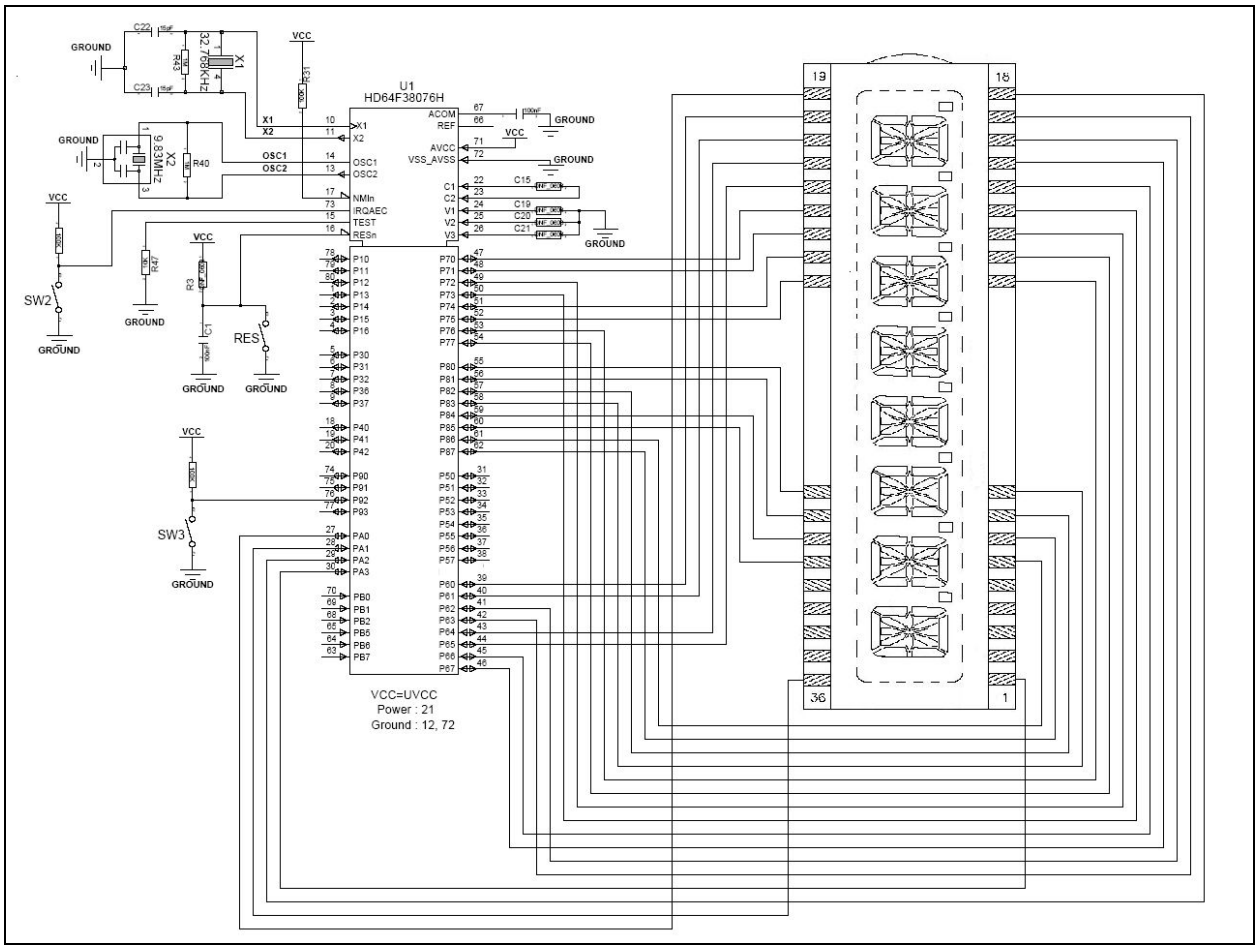

Fig.6 - Schematics

Appendix A: Application Code

File RTC_38076.c

```
/************************************************************************/
/*                                                                      */
/*  FILE        :RTC_38076.c                                            */
/*  DATE        :Tue, Nov 02, 2004                                      */
/*  DESCRIPTION :Main Program                                           */
/*  CPU TYPE    :H8/Other                                               */
/*                                                                      */
/*  This file is generated by Renesas Project Generator (Ver.3.1).      */
/*                                                                      */
/************************************************************************/

#include    <machine.h>
#include   <string.h>
#include   <stdio.h>
#include   "iodefine.h"
#include   "dati.h"

unsigned char figures[11][2] =
{
{0x07, 0x0e},{0x06, 0x00},{0x23, 0x4c},{0x07, 0x48},{0x26, 0x42},
{0x25, 0x4a},{0x25, 0x4e},{0x07, 0x00},{0x27, 0x4e},{0x27, 0x4a},{0x0, 0x0}
};

unsigned int  count_s, count_m, count_h, count_d;
unsigned char seconds;
unsigned char minute;
unsigned char hours;
unsigned char current_day;
unsigned char program_flag,plus_flag;
unsigned char flag_second,prog_minute,prog_hours;

unsigned char ram_lcd[12];

/************************************************************************
Function Name:   watch_mode
Description:  change the operating mode of CPU from the running one to
        watch setting register and then transition by means of
        sleep instruction
Parameters:   N/A
Return value:    N/A
************************************************************************/
void watch_mode (void)
{
   SYSTEM.SYSCR1.BIT.LSON = 1;
   SYSTEM.SYSCR2.BIT.MSON = 1;
   SYSTEM.SYSCR1.BIT.SSBY = 1;
   SYSTEM.SYSCR2.BIT.DTON = 0;
   SYSTEM.SYSCR1.BIT.TMA3 = 1;
   sleep();
}

/************************************************************************
```

```
Function Name:   subactive_mode
Description:  change the operating mode of CPU from the running one to
        subactive setting register for direct transition and
        then transition by means of sleep instruction
Parameters:   N/A
Return value:    N/A
*********************************************************************/
void subactive_mode (void)
{
   SYSTEM.SYSCR1.BIT.LSON = 1;
   SYSTEM.SYSCR1.BIT.SSBY = 1;
   SYSTEM.SYSCR2.BIT.DTON = 1;
   SYSTEM.SYSCR1.BIT.TMA3 = 1;
   sleep();
}

/*********************************************************************
Function Name:   active_mode
Description:  change the operating mode of CPU from the running one to
        full speed setting register for direct transition and
        then transition by means of sleep instruction
Parameters:   N/A
Return value:    N/A
*********************************************************************/
void active_mode (void)
{
   SYSTEM.SYSCR1.BIT.LSON = 0;
   SYSTEM.SYSCR2.BIT.MSON = 0;
   SYSTEM.SYSCR1.BIT.SSBY = 1;
   SYSTEM.SYSCR2.BIT.DTON = 1;
   SYSTEM.SYSCR1.BIT.TMA3 = 1;
   sleep();
}

/*********************************************************************
Function Name:   main
Description:  main program. After initialization the micon is driven in
        subactive, then there is sequentially a check if the RTC
        interrupt related to „second“ is arrived (in this case the
        LCD is refreshed), a check if a programming phase is
        Requested and at the end the micon is driven in watch mode.
Parameters:   N/A
Return value:    N/A
*********************************************************************/
void main(void)
{
unsigned char i,y;

   initialize();
   flag_second = 0;
   subactive_mode();

   while(1)
   {
    if (flag_second == 1)
    {
```

```
        write_lcd();
        LCD_RAM.MEM.BYTE.DIG5 |= 0x08;
        LCD_RAM.MEM.BYTE.DIG9 |= 0x08;

        flag_second = 0;
     }
     if (program_flag == TRUE)
     {
        RTC.RTCCR1.BIT.RUN = 0;        // stop rtc
        INT.IENR1.BIT.IENEC2 = 1;      // enable interrupt aec
        program_time();
        write_time_to_rtc();
        program_flag = FALSE;          // disable interrupt aec
        INT.IENR1.BIT.IENEC2 = 0;
        RTC.RTCCR1.BIT.RUN = 1;        // restart rtc
     }
     else
        watch_mode();
   }
}
```

File init_peripherals.c

```
/************************************************************************/
/*                                                                      */
/*  FILE        :init_peripherals.c                                     */
/*  DATE        :Tue, Nov 02, 2004                                      */
/*  DESCRIPTION :Peripherals setup                                      */
/*  CPU TYPE    :H8/Other                                               */
/*                                                                      */
/*  This file is generated by Renesas Project Generator (Ver.3.1).      */
/*                                                                      */
/************************************************************************/

#include   "iodefine.h"
#include   <machine.h>
#include   "dati.h"

/************************************************************************
Function Name:   PowerDown_Init
Description:  initialisation of system registers in order to use various
        operating mode and switch off of all the peripherals
Parameters:   N/A
Return value:    N/A
************************************************************************/
void PowerDown_Init(void)
{
   SYSTEM.CKSTPR1.BYTE = 0x8a;
   SYSTEM.CKSTPR2.BYTE = 0x88;

   INT.IENR2.BIT.IENDT = 1;  //enable direct transition interrupts

   SYSTEM.SYSCR2.BIT.NESEL = 0;    //noise elimination bit
   SYSTEM.SYSCR1.BIT.STS = 1;
   SYSTEM.SYSCR1.BIT.MA = 2;
   SYSTEM.SYSCR2.BIT.SA = 2;

   SYSTEM.SYSCR1.BIT.SSBY = 0;
   SYSTEM.SYSCR1.BIT.LSON = 0;
   SYSTEM.SYSCR2.BIT.DTON = 1;  // enable direct transition
   SYSTEM.SYSCR2.BIT.MSON = 0;

}

/************************************************************************
Function Name:   lcd_init
Description:  initialisation of LCD controller on-chip
Parameters:   N/A
Return value:    N/A
************************************************************************/
void lcd_init (void)
{
unsigned char i,y;

   SYSTEM.CKSTPR2.BIT.LDCKSTP = 1; // switch on peripheral
   LCD.LPCR.BYTE = 0xea;           // 1/4 duty, SEG9 to SEG32 enabled
   LCD.LCR.BYTE = 0xf2;            // LCD supply on, fiw/8
```

```
    LCD.LCR2.BIT.LCDAB = 0;         // waveform A
    LCD.LCR2.BIT.CHG = 1;

    // clear lcd ram and segments of display
    for (i=0;i<12;i++)
    {
      ram_lcd[i] = 0;
      LCD_RAM.MEM.DIGIT[i] = 0;
    }

    // write 00.00.00 on display
    y = 0;
    for (i=0; i<6; i++)
    {
      LCD_RAM.MEM.DIGIT[y++] = figures[ram_lcd[i]][0];
      LCD_RAM.MEM.DIGIT[y++] = figures[ram_lcd[i]][1];
    }
    LCD_RAM.MEM.BYTE.DIG5 |= 0x08;
    LCD_RAM.MEM.BYTE.DIG9 |= 0x08;
}

/************************************************************************
Function Name:   init_rtc
Description:  initialisation of real time clock on-chip
Parameters:   N/A
Return value:    N/A
************************************************************************/
void init_rtc (void)
{
    SYSTEM.CKSTPR1.BIT.RTCCKSTP = 1;
    while ((RTC.RSECDR.BYTE | 0x80) == 0);
    RTC.RTCCR1.BIT.RUN = 0;
    RTC.RTCCR1.BIT.RST = 1;
    RTC.RTCCR1.BIT.RST = 0;
    RTC.RTCCR1.BIT.B12_24 = 1;
    RTC.RTCCSR.BIT.SUB32K = 1;
    RTC.RTCCSR.BIT.RCS3 = 1;
    RTC.RTCCR2.BYTE = 0;
    PORT.PMR3.BYTE = 0xff;
    INT.IENR1.BIT.IENRTC = 1;          // interrupt enabled
    RTC.RTCCR2.BIT.B05SEIE = 1;
    RTC.RTCCR1.BIT.RUN = 1;         // start rtc
}


/************************************************************************
Function Name:   init_io
Description:  all the I/O pins are initialised
Parameters:   N/A
Return value:    N/A
************************************************************************/
void init_io (void)
{

    PORT.PMR1.BYTE = 0xfc;    // double function pins selected as I/O
    PORT.PCR1.BYTE = 0xfd;    // output pins selected pin11 in input
```

```
    PORT.PUCR1.BYTE = 0x80;      // pull-up disabled
    PORT.PDR1.BYTE = 0xfd;    // high level set

    PORT.PMR3.BYTE = 0xfe;    // double function pins selected as I/O
    PORT.PCR3.BYTE = 0xff;    // output pins selected
    PORT.PUCR3.BYTE = 0x3e;      // pull-up disabled
    PORT.PDR3.BYTE = 0xff;    // high level set

    PORT.PMR4.BYTE = 0xf8;    // double function pins selected as I/O
    PORT.PCR4.BYTE = 0xff;    // output pins selected
    PORT.PDR4.BYTE = 0xff;    // high level set

    PORT.PMR5.BYTE = 0x00;    // double function pins selected as I/O
    PORT.PCR5.BYTE = 0xff     // output pins selected
    PORT.PUCR5.BYTE = 0;      // pull-up disabled
    PORT.PDR5.BYTE = 0xff;    // high level set

    PORT.PMR9.BYTE = 0xf4;    // p92 selected as IRQ4
    PORT.PCR9.BYTE = 0xfb;    // output pins selected
    PORT.PDR9.BYTE = 0xff;    // high level set

    INT.IENR1.BIT.IEN4 = 1;      // enable irq4 interrupt
    INT.IEGR.BIT.IEG4 = 0;    // falling edge selected

    PORT.PMRB.BYTE = 0xe8;    // all pin selected as input pin
}

/*****************************************************************************
Function Name:   init_variable
Description:  initialization of the system registers, of the global
        variable and call of routine for initialize peripherals
Parameters:   N/A
Return value:    N/A
*****************************************************************************/
void initialize(void)
{
    PowerDown_Init();
    lcd_init();
    init_io();
    init_rtc();
}
```

File utility.c

```
/*****************************************************************/
/*                                                               */
/*  FILE        :utility.c                                       */
/*  DATE        :Tue, Nov 02, 2004                               */
/*  DESCRIPTION :utility routines                                */
/*  CPU TYPE    :H8/Other                                        */
/*                                                               */
/*  This file is generated by Renesas Project Generator (Ver.3.1).     */
/*                                                               */
/*****************************************************************/

#include   "iodefine.h"
#include    <machine.h>
#include   "dati.h"

/*****************************************************************
Function Name:   hextobcd
Description:  routine to convert hexadecimal data in BCD format as
        requested by RTC registers
Parameters:   datum hexadecimal to be converted in BCD
Return value:    datum converted in BCD format
*****************************************************************/
unsigned char hextobcd(unsigned char datohex)
{
unsigned char datobcd,dummy;

   datobcd = datohex % 10;
   dummy = datohex / 10;
   dummy = dummy % 10;
   datobcd |= (dummy << 4);
   return datobcd;
}

/*****************************************************************
Function Name:   bcdtohex
Description:  routine to convert BCD data in hexadecimal format used
        for computation and visualisation on LCD
Parameters:   datum BCD to be converted in hexadecimal
Return value:    datum converted in hexadecimal format
*****************************************************************/
unsigned char bcdtohex(unsigned char datobcd)
{
unsigned char datohex,dummy;

   datohex = datobcd & 0xf0;
   datohex >>= 4;
   datohex *= 10;
   dummy = datobcd & 0x0f;
   datohex += dummy;
   return datohex;
}

/*****************************************************************
Function Name:   read_time_from_rtc
Description:  routine used to read time from RTC.
```

```
Parameters:   N/A
Return value:    N/A
*********************************************************************************/
void read_time_from_rtc(void)
{
   while (RTC.RSECDR.BIT.BSY == 1);
   seconds = bcdtohex(RTC.RSECDR.BYTE & 0x7f);
   while (RTC.RMINDR.BIT.BSY == 1);
   minute = bcdtohex(RTC.RMINDR.BYTE & 0x7f);
   while (RTC.RHRDR.BIT.BSY == 1);
   hours = bcdtohex(RTC.RHRDR.BYTE & 0x3f);
   while (RTC.RWKDR.BIT.BSY == 1);
   current_day = (RTC.RWKDR.BYTE & 0x7);
}

/*********************************************************************************
Function Name:   write_time_to_rtc
Description:  routine used to modify time in RTC.
Parameters:   N/A
Return value:    N/A
*********************************************************************************/
void write_time_to_rtc(void)
{
unsigned char prova;

   RTC.RTCCR1.BIT.RUN = 0;           // stop RTC operation
   RTC.RSECDR.BYTE = 0;
   RTC.RMINDR.BYTE = hextobcd(minute);
   RTC.RHRDR.BYTE = hextobcd(hours);
   RTC.RWKDR.BYTE = current_day;
   RTC.RTCCR1.BIT.RUN = 1;           // start RTC operation
}

/*********************************************************************************
Function Name:   write_lcd
Description:  read data from RTC and update time on the LCD
Parameters:   N/A
Return value:    N/A
*********************************************************************************/
void write_lcd(void)
{
unsigned char i,y;

   read_time_from_rtc();

   ram_lcd[1] = seconds/10;
   ram_lcd[0] = seconds - (ram_lcd[1] * 10);
   ram_lcd[3] = minute/10;
   ram_lcd[2] = minute - (ram_lcd[3] * 10);
   ram_lcd[5] = hours/10;
   ram_lcd[4] = hours - (ram_lcd[5]*10);

   y = 0;
   for (i=0; i<6; i++)
   {
    LCD_RAM.MEM.DIGIT[y++] = figures[ram_lcd[i]][0];
```

```
     LCD_RAM.MEM.DIGIT[y++] = figures[ram_lcd[i]][1];
    }
}

/***********************************************************************
Function Name:   write_program
Description:  called during the programming phase, refresh the LCD with
        the increasing value of minutes or hours
Parameters:   N/A
Return value:    N/A
***********************************************************************/
void write_program(void)
{
unsigned char i,y;

    ram_lcd[1] = minute/10;
    ram_lcd[0] = minute - (ram_lcd[1] * 10);
    ram_lcd[3] = hours/10;
    ram_lcd[2] = hours - (ram_lcd[3]*10);

    y = 0;
    for (i=0; i<4; i++)
    {
     LCD_RAM.MEM.DIGIT[y++] = figures[ram_lcd[i]][0];
     LCD_RAM.MEM.DIGIT[y++] = figures[ram_lcd[i]][1];
    }

    if (prog_minute == FALSE)
    {
     LCD_RAM.MEM.DIGIT[y++] = 0x00;
     LCD_RAM.MEM.DIGIT[y++] = 0x00;
     LCD_RAM.MEM.DIGIT[y++] = 0x16;
     LCD_RAM.MEM.DIGIT[y++] = 0x26;
    }
    else
    {
     LCD_RAM.MEM.DIGIT[y++] = 0x00;
     LCD_RAM.MEM.DIGIT[y++] = 0x00;
     LCD_RAM.MEM.DIGIT[y++] = 0x26;
     LCD_RAM.MEM.DIGIT[y++] = 0x46;
    }

}

/***********************************************************************
Function Name:   program_time
Description:  handle of the programming phase, read the present time
        from the RTC and update the minutes or the hours if the
        flag associated to the increment is true.
Parameters:   N/A
Return value:    N/A
***********************************************************************/
void program_time(void)
{
    read_time_from_rtc();
```

```
    while (prog_minute == FALSE)
    {
     if (plus_flag == TRUE)
     {
        minute++;
        if (minute > 59)
           minute = 0;
        plus_flag = FALSE;
     }
     write_program();
    }
    while (prog_hours == FALSE)
    {
     if (plus_flag == TRUE)
     {
        hours++;
        if (hours > 23)
           hours = 0;
        plus_flag = FALSE;
     }
     write_program();
    }
    prog_minute = FALSE;
    prog_hours = FALSE;
}
```

File intprg.c

```
/************************************************************************/
/*                                                                      */
/*  FILE        :intprg.c                                               */
/*  DATE        :Tue, Nov 02, 2004                                      */
/*  DESCRIPTION :Interrupt Program file                                 */
/*  CPU TYPE    :H8/Other                                               */
/*                                                                      */
/*  This file is generated by Renesas Project Generator (Ver.3.1).      */
/*                                                                      */
/************************************************************************/

#include <machine.h>
#include "iodefine.h"
#include "dati.h"

#pragma section IntPRG

//  vector 8 IRQAEC
/************************************************************************
Function Name:   INT_IRQAEC
Description:  connected to the switch PLUS (activated only during the
        program phase) set a flag handled by routine program_time.
Parameters:   N/A
Return value:    N/A
************************************************************************/
__interrupt(vect=8) void INT_IRQAEC(void)
{
   INT.IRR1.BIT.IRREC2 &= 0;
   plus_flag = TRUE;
}

//  vector 10 IRQ4
/************************************************************************
Function Name:   INT_IRQ4
Description:  connected to the switch PROGRAMMING (request in sequence
        the program phase for minutes, for hours and then the exit
        from the procedure).
Parameters:   N/A
Return value:    N/A
************************************************************************/
__interrupt(vect=10) void INT_IRQ4(void)
{
   INT.IRR1.BIT.IRRI4 &= 0;
   if (program_flag == FALSE)
     program_flag = TRUE;
   else
   {
     if (prog_minute == FALSE)
        prog_minute = TRUE;
     else
        prog_hours = TRUE;
   }
}

//  vector 20 RTC 0.5 sec
```

```
/***********************************************************************
Function Name:   INT_RTC_HALF_SECOND
Description:  interrupt activated each half second that modifies the LCD
        dots (and enables the „second“ interrupt
Parameters:   N/A
Return value:    N/A
***********************************************************************/
__interrupt(vect=20) void INT_RTC_HALF_SECOND(void)
{
   RTC.RTCFLG.BIT.B05SEIFG &= 0;
   RTC.RTCCR2.BIT.B05SEIE = 0;
   RTC.RTCCR2.BIT.B1SEIE = 1;

   LCD_RAM.MEM.BYTE.DIG5 &= 0xf7;
   LCD_RAM.MEM.BYTE.DIG9 &= 0xf7;
}
//  vector 21 RTC 1 sec
/***********************************************************************
Function Name:   INT_RTC_SECOND
Description:  interrupt activated each second that modifies the LCD dots
        and set a flag used by main routine.
Parameters:   N/A
Return value:    N/A
***********************************************************************/
__interrupt(vect=21) void INT_RTC_SECOND(void)
{
   RTC.RTCFLG.BIT.SEIFG &= 0;
   RTC.RTCCR2.BIT.B05SEIE = 1;
   RTC.RTCCR2.BIT.B1SEIE = 0;
   flag_second = 1;
}

//  vector 43 Direct Transition
/***********************************************************************
Function Name:   INT_Direct_Transition
Description:  Interrupt for direct transition.
Parameters:   N/A
Return value:    N/A
***********************************************************************/
__interrupt(vect=43) void INT_Direct_Transition(void)
{
   INT.IRR2.BIT.IRRDT &= 0;
}
```

File dati.h

```
/************************************************************************/
/*                                                                      */
/*  FILE        :dati.h                                                 */
/*  DATE        :Tue, Nov 02, 2004                                      */
/*  DESCRIPTION :Main Program                                           */
/*  CPU TYPE    :H8/Other                                               */
/*                                                                      */
/*  This file is generated by Renesas Project Generator (Ver.3.1).      */
/*                                                                      */
/************************************************************************/

void lcd_init (void);
void init_peripherals (void);

void subactive_mode(void);
void change_speed(unsigned char next_mode);

extern unsigned char   figures[11][2];

extern unsigned char   program_flag,plus_flag,select_uart;
extern unsigned char   flag_second,prog_minute,prog_hours;
extern unsigned char   seconds;
extern unsigned char   minute;
extern unsigned char   hours;
extern unsigned char   current_day;
extern unsigned char   receive_byte;

extern unsigned char   ram_lcd[12];

extern unsigned int count_s, count_m, count_h, count_d;

struct mem_lcd {
     union {
        unsigned char DIGIT[12];
        struct {
           unsigned char DIG1:8;
           unsigned char DIG2:8;
           unsigned char DIG3:8;
           unsigned char DIG4:8;
           unsigned char DIG5:8;
           unsigned char DIG6:8;
           unsigned char DIG7:8;
           unsigned char DIG8:8;
           unsigned char DIG9:8;
           unsigned char DIG10:8;
           unsigned char DIG11:8;
           unsigned char DIG12:8;
        }BYTE;
     }MEM;
};

#define LCD_RAM   (* (volatile struct mem_lcd *) (0xF374))

#define FALSE 0
#define TRUE      !FALSE
```

Website and Support

Renesas Technology Website
http://www.renesas.com/

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

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, 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 products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life

   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas 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 applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.