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.

RENESAS

H8/38602

ADC in Low Power Application

Introduction

The H8/38602 device is a high performance 16-bit microcontroller from Renesas. It has ten peripherals, a power on reset circuit, 13 I/O pins, 9 I/P pins and eight power modes. These make it ideal for low power applications such as metering.
This application note details an example of how to use the H8/38602 device in a low power mode, wake the device into a higher power active mode, perform some A/D conversions and return to the low power mode, watch mode. For development of the code, HEW4 (version 4.00.02.008), the Renesas H8, H8/300 Standard toolchain version 6.1.0.0, the MB-H838602 board and an E8 debugger was used.

Contents

Development Environment

The following figure, figure 1, shows the environment used to develop the code.

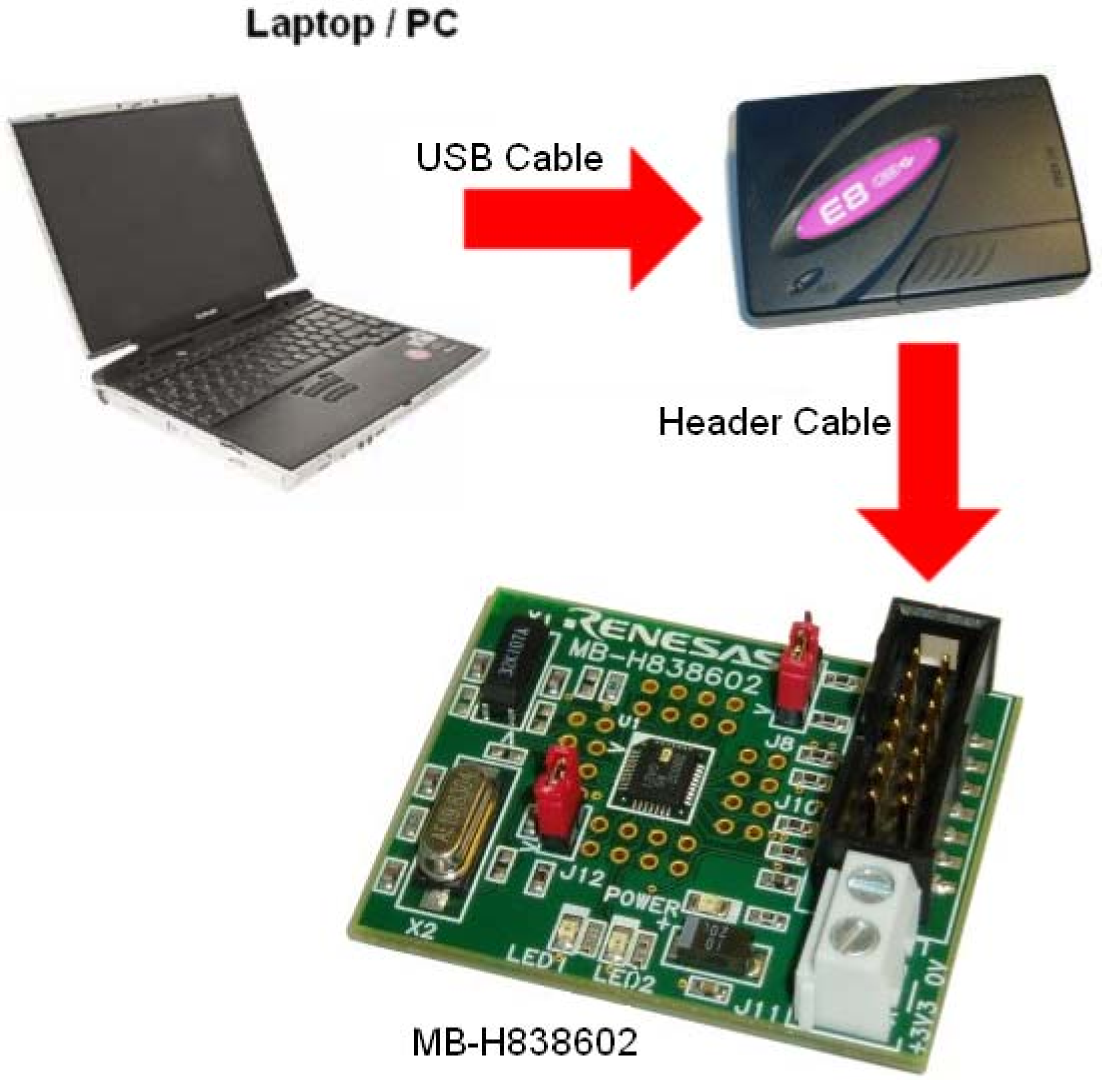

Figure 1: H8/38602 Development Environment

A laptop or PC is required to run HEW4 (Renesas' IDE) and the E8 driver. The USB port of the E8 may then be plugged into a PC USB port using a USB cable. A header cable may then be connected between the E8 and the MB-H838602 board.

External Program Operation – Release operation

The program when run on the MB-H838602 will enter watch mode for around eight seconds, wake up into subactive mode where it performs 6 A/D conversions, then enters watch mode once more. Figure 2 shows the program flow of the code.

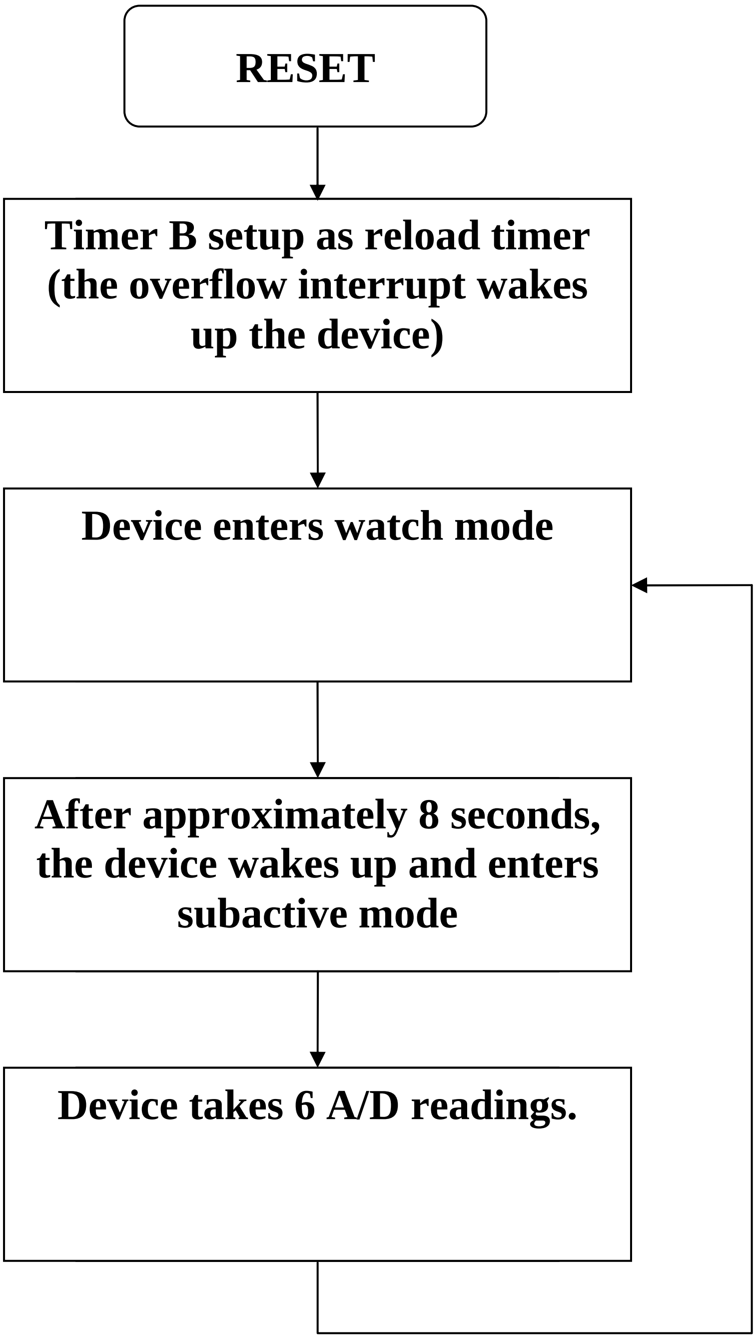

Figure 2: Program flow for Release Build

External Program Operation – Debug operation

The program when being debugged via the E8 on the MB-H838602 will enter subactive mode permanently. The code will perform 6 A/D conversions and then flash LED1 five times. Figure 3 shows the program flow of the code.

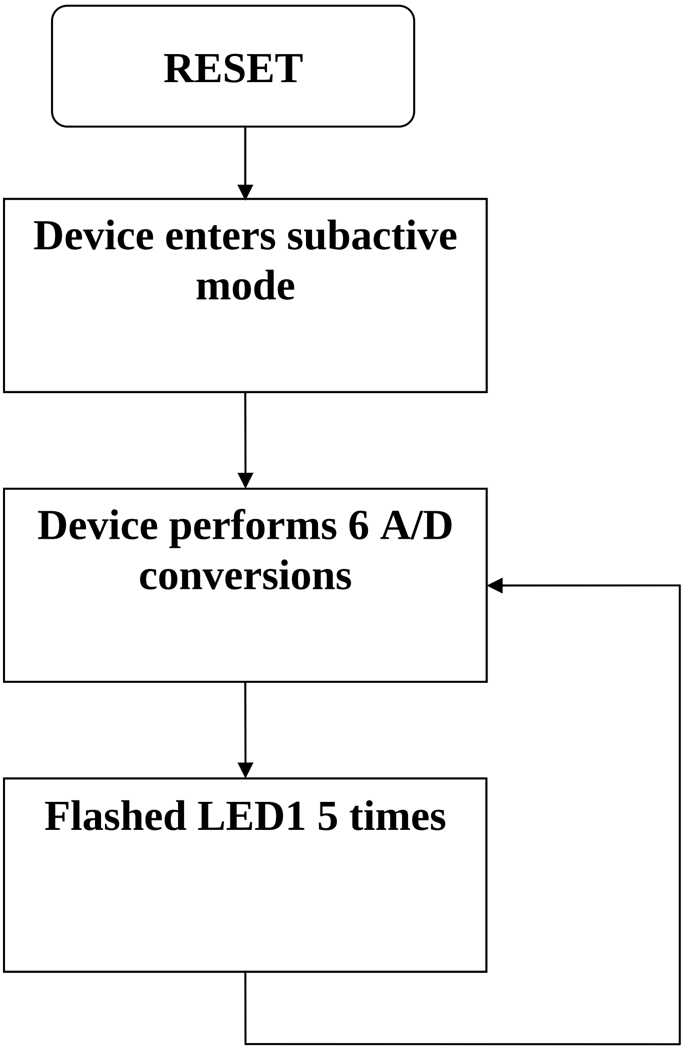

Figure 3: Program Flow for Debug configuration

Since it is not possible to perform current measurement accurately with the E8 debugger header plugged onto the MB-H838602 board, there is little to be gained by entering watch mode when debugging the code. Since current consumption is not to be measured in debug mode, it is possible to flash an LED in the debug configuration to show the device is in subactive mode and running code.

Application Workspace and code

This section outlines the code used to drive the watchdog timer.

1. Workspace -Files

Figure 4 shows the workspace view window

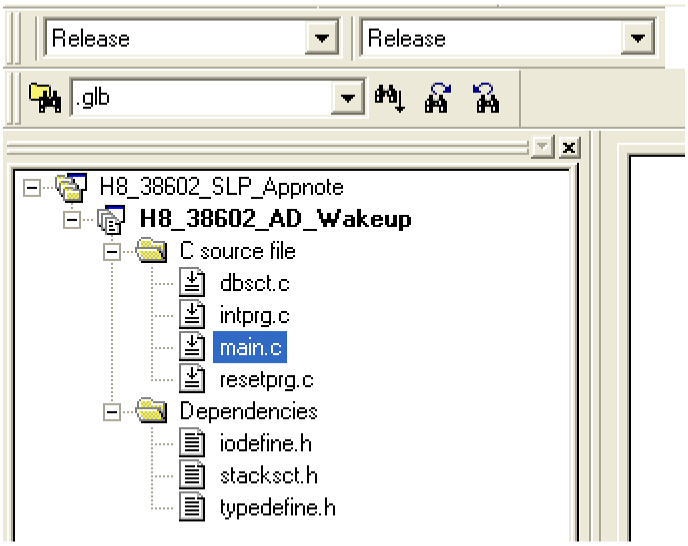

Figure 4: Workspace for the H8/38602 code.

There are four source files in the workspace. These are “resetprg.c”, “intprg.c”, “dbsct.c” and “main.c”.

dbsct.c

This C file initialises the various data sections.

intprg.c

This file lists the interrupt vector table.

resetprg.c

Contains the code called by the reset vector.

main.c

The main function is contained in this file.

2. Workspace – Sessions and Build Configurations

There are two session files in the workspace, these being "Release" and "Debug". There are also two build configurations; Release and Debug. Please use the Release build configuration with the Release session and the Debug build configuration in the Debug session.

Debug

This session is for debugging the code using the E8. Please ensure the build configuration is also set to Debug when using this session. This is because the session file uses placeholders to locate the download file (.abs) and this is relative to the current build configuration selected. The wrong ".abs" file will be located if the build configuration is set to Release. When the connect icon is pressed, please select the first option "Download emulator firmware" in the window that appears. The device on the board is a H8/38602 which is connected to a crystal of 9.8304 MHz. Once a connection is established between the device and the E8, the ".abs" file may then be downloaded. Once this is complete the device may be reset, run, stepped etc. via the debugging icons.

Release

This session is for downloading the finished monitor code into the device via the E8. In the Release session, when the connection dialogue appears, please select the third option "Writing Flash Memory". Once the device has connected to the E8 successfully, the ".mot" file which appears in the download section of the workspace window may then be downloaded to the device. Once a checksum is returned, disconnect the E8 and reset to device to start the code running on the device.

Note: There are differences between the Release and Debug code. The debug code does not use timer B or watch mode, and instead enters subactive mode and remains there. This is to make it easier to stop, run, step the code etc. as the CPU remains active. It also flashes an LED (LED1) after the A/D conversions are complete.

3. Workspace – Memory Sections

The sections of memory used in the program may be found by going to Build -> H8S, H8/300 Standard Toolchain… and then clicking on the Link/Library tab. Select "Section" in the category tab and the window shown in figure 5 will appear.

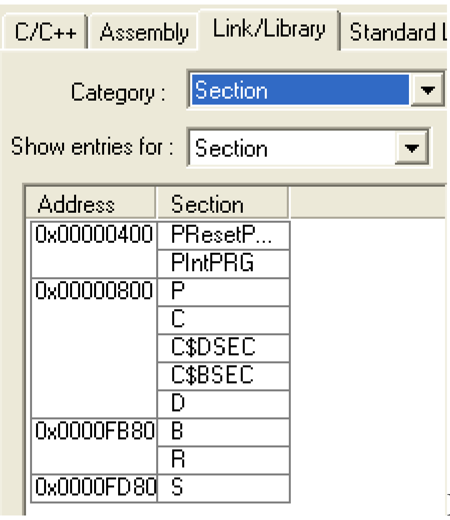

Figure 5: Memory Sections

The sections prefixed with a "P" are code sections, those prefixed "C" are constant data sections and "D" prefixed sections indicate initialised data. These sections therefore need to be placed in ROM either because it is program code or because they contain dedicated values. The "B" section is for uninitialised data. It is important that the D section has a mirror (the R section) in RAM as initialised variables are subject to change within the code. The D section in ROM holds their initial values whilst the R section in RAM contains their current values as the program runs.

4. Program Code

This section provides code segments which are then described in detail

The dbsct.c and intprg.c files setup the memory sections of the code and provide the interrupt vector table.

ResetPRG.c

```
// Include machine header file for CCR access and header to define stack size
#include <machine.h>
#include <_h_c_lib.h>
#include "typedefine.h"
#include "stacksct.h"

// Prototype for function main
extern void main(void);
// Prototype for power on reset
void PowerON_Reset(void);

#ifdef __cplusplus
extern "C" {
#endif
extern void HardwareSetup(void);
#ifdef __cplusplus
}
#endif

// Declare a section called ResetPRG
#pragma section ResetPRG

// power on reset vector declaration
__entry(vect=0) void PowerON_Reset(void)
{
    // Set interrupt mask bit in CCR
       set_imask_ccr((_UBYTE)1);
      // Set up the memory sections
      _INITSCT();
      // Unmask the I bit in CCR
      set_imask_ccr((_UBYTE)0);
      // Enter function main
      main();

      sleep();
}
```

The ResetPRG.c file contains the section ResetPRG which is located in the ROM part of the device as it consists of code. This code is called by the reset vector. Firstly the I bit (Interrupt mask bit) in the CCR (Condition Code Register) is set. The INITSCT function is then run which sets up the memory sections of the device. The Interrupt mask bit is then unmasked to allow interrupts to be handled. The main function (user code) may then be entered.

Main.c

```
void main(void)
{
    #ifdef Debug
    unsigned int i,j = 0;
    #endif

    // Declare variable for determing
    // A/D channel to be sampled
    unsigned char CHANNEL;

    // WDT Disable
    WDT.TCSRWD1.BYTE = 0x9e;
    WDT.TCSRWD1.BYTE = 0xa2;
    WDT.TCSRWD1.BYTE = 0x8e;

    #ifdef Debug
    // Enables only S3CKSTP, TB1, ADCKSTP, FLASH
    CKSTPR1.BYTE = 0x56;
    #endif

    #ifdef Release
    // Enables only A/D, Timer B, Flash
    CKSTPR1.BYTE = 0x16;
    #endif

    // Switch off all other modules
    CKSTPR2.BYTE = 0x00;

    // All outputs ON
    // Prevents leakage I
    IO.PCR1 = 0xff;
    IO.PDR1.BYTE = 0xff;
    IO.PCR3 = 0xff;
    IO.PDR3.BYTE = 0xff;
    IO.PCR8 = 0xff;
    IO.PDR8.BYTE = 0xff;
    IO.PCR9 = 0xff;
    IO.PDR9.BYTE = 0xff;
```

For the debug build, two variables are defined which control the flashing of LED1. The variable CHANNEL determines the channel to be sampled for A/D conversion. The Watchdog timer is then disabled. The clock stop registers are set to enable only the modules that are used for each build. The output pins of the device are then set to outputs to prevent leakage current.

```
// Use subactive clock thi/w
    SYSCR2.BIT.SA = 3;
    // Use active mode clock thiosc/64
    SYSCR1.BIT.MA = 3;

    // Enable timer B interrupt
    IENR2.BIT.IENTB1 = 1;

    // Switch timer B off
    TB1.TMB1.BIT.STR = 0;
    // Set as reload timer
    TB1.TMB1.BIT.RLD = 1;
    // Set clock speed as slowest
    TB1.TMB1.BIT.CKS = 6;

    // Set Reload reg to 1 for long interval
    // between timer B interrupts
    TB1.TLB1 = 0x01;
    // Clear A/D Flag for safety
    IRR2.BIT.IRRAD = 0;


    //--------------------------------------------------------------------------
-
    // Select channel and fi/2 clock mode
    AD.AMR.BIT.CKS = 3;
    // Ladder resistance halted
    // when A/D idle
    AD.ADSR.BIT.LADS = 1;
    // Disable external trigger of A/D
    AD.AMR.BIT.TRGE = 0;
```

The SYSCR (System control regs) are then set up so that in subactive mode, the clock used is Φw (32 KHz) and in active mode, the clock used is $\Phi/64$. The timer B interrupt is then enabled, and timer B is set up as a reload timer with the slowest clock speed of $\Phi w/1024$. A low value is placed in the timer B load register as this gives a long period between timer B interrupts. The AD flag is then cleared just for safety. The A/D peripheral is then configured to use $\Phi w/2$ clock and to disable the external trigger function.

```
    // Enter infinite loop
    while(1)
    {
        // For standalone, go into watch mode
        #ifdef Release
        /* Set appropriate bits in SYSCR regs for watch mode */
        Set_watch_mode();
        // Start the timer
        TB1.TMB1.BIT.STR = 1;
        // Set clock in subactive to thiw
        SYSCR2.BIT.SA = 3;
        // Enable timer B interrupt
        IENR2.BIT.IENTB1 = 1;
        // Go to sleep
        // Timer B interrupt will wake device up
        sleep();
        #endif

        // If debugging stay in subactive
        #ifdef Debug
        SYSCR1.BIT.SSBY = 1;
        SYSCR1.BIT.LSON = 1;
        SYSCR1.BIT.TMA3 = 1;
        SYSCR2.BIT.DTON = 1;
        sleep();
        #endif

        // Disable the timer interrupt
        IENR2.BIT.IENTB1 = 0;

        // now in subactive
        // perform 6 AD channel conversions
        for(CHANNEL=4; CHANNEL < (MAX_AD_CHANNELS + 4); CHANNEL++)
        {   // Set the channel required for conversion
            // in the register
            AD.AMR.BIT.CH = CHANNEL;
            // Start the conversion
            AD.ADSR.BIT.ADSF = 1;
            // Wait until the conversion has ended
        while(AD.ADSR.BIT.ADSF == 1);
        }
```

The code then enters an infinite loop, which for release code, puts the device in watch mode after timer B module and interrupt is enabled. After approximately* 8 seconds, the device will wake up out of watch mode and into subactive mode. For the debug code, the device is put straight into subactive mode. The timer B interrupt is then disabled as it is no longer required and six A/D conversions (one for each channel) are performed.

*An approximate value is given as the crystal used has some inaccuracy.

```
        // If debugging using E8, flash LED1
        #ifdef Debug
            // Set port 8 to output
            IO.PCR8 = 0xFF;
            // and for 5 times
            for(j = 0; j < 5; j++)
            {   // make a delay
                for(i = 0; i < 100; i++)
                {   // Switch LED1 ON
                    IO.PDR8.BYTE = 0x00;
                }
                // make another delay
                for(i = 0; i < 100; i++)
                {   // Switch LED1 OFF
                    IO.PDR8.BYTE = 0xFF;
                }
            }
        #endif
    }
}
```

The last part of the main function flashes LED1 for the debug code.

The following code fragment is contained in the main.c file.

```
// Function to set SYSCR bits
// for entry to watch mode
void Set_watch_mode(void)
{
    // Set LSON and MSON to enter
    // subactive on exit from watch
    SYSCR1.BIT.LSON = 1;
    SYSCR2.BIT.MSON = 0;
    SYSCR1.BIT.SSBY = 1;
    SYSCR1.BIT.TMA3 = 1;
    SYSCR2.BIT.DTON = 0;
    // Set stabalisation time
    SYSCR1.BIT.STS  = 5;
}

// ISR for timer B
__interrupt(vect=33) void INT_TIMERB1(void)
{
 // Clear the timer B flag
 IRR2.BIT.IRRTB1 = 0;
}
```

The function Set_watch_mode sets the appropriate bits in the system control regs (SYSCR's) to allow entry into watch mode.

The ISR for the timer B module is also shown in the previous code; the ISR just clears the timer B flag.

Current Measurement

The MB-H838602 board was programmed in the Release configuration using the release session. The device was then disconnected from the E8 both within HEW and physically. A Fluke 87 IV current meter was then placed across J12, and a Thurlby K2 PL320 power supply used. The following results were obtained.

Voltage	Actual Current Measured
3.3V	~4.15uA min /10uA max
2.7V	~2.48uA min / 7uA max

Please note: Resistors R9 and R10 were removed from the MB-H838602 board.

Conclusion

Code was written to place the H8/38602 device into watch mode then wake up into subactive mode after approximately* eight seconds, take some A/D readings and then go back into watch mode again. The current consumption of the H8/38602 device was ~2.48 uA min and ~10 uA max.

* Approximately due to crystal inaccuracies.

Website and Support

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

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

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.