

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.

Using SIM IO for Print(f) with the Renesas Compilers

1.0 Introduction

This application note is intended as a guide to using the printf Standard C Library Function of the Renesas H8/SH Compilers. Once able to use this function the user may wish to make use of some of the other standard IO functions of the compiler. However these functions are not discussed in this application note.

2.0 Standard I/O in an Embedded System

When writing applications using a native compiler (such as application to run on a PC) it is likely that the target system will have some form of standard input and output device. On a PC system the Standard Input device is the keyboard and the standard output device is the VDU or Monitor.

An embedded system does not generally have a standard input or output device. It is the responsibility of the embedded systems developer to decide how to implement input and output to the embedded system and to write software routines to facilitate this input and output. These routines must be able to read a byte of data in from an external device and output a byte of data to an external device. In the examples in this application note, a serial port is used to for both input and output. Some simple examples of software routines to input and output a byte are given. Please note that these examples are based on code from the EVB tutorials, they are basic and do not include any code to implement error detection or correction.

2.1 Inputting and Outputting Bytes of Data

It is considered standard practice to write two functions to input and output data. One function to output a byte of data and one to input a byte of data. Typically, these functions are named putchar and getchar. The putchar routine will be passed a byte of data as a parameter and will output this byte of data to the standard output device. The getchar routine will read a byte of data from the standard input device and return it as a parameter. These functions can be written in C code or Assembly code.

2.2 The SIM IO Feature of HEW with the Renesas Simulators

The Renesas High Performance Workshop (HEW) has the Simulator software included in the install. This can provide a simulated I/O feature. This is an extra window within HEW, which can be used as a Standard I/O device from the simulator. The window will display text in a similar manner to a terminal emulator program. To use this feature the HEW project generator can be setup to create source files to support IO streams. This includes low level functions called charget and charput to send and receive bytes of data to and from the SIM I/O window of HDI. If these functions are rewritten to send and receive data to a different device then the standard IO functions such as printf can be used to output data from a real embedded system, instead of a software simulation of an H8 or SH device.

3.0 Using the HEW Project Generator to Support Standard I/O

The HEW project generator can create source files to support standard I/O. Some extra options must be selected within the HEW project generator to create these files. A run down of the HEW Project Generator Steps is shown below

3.1 Running the HEW Project Generator

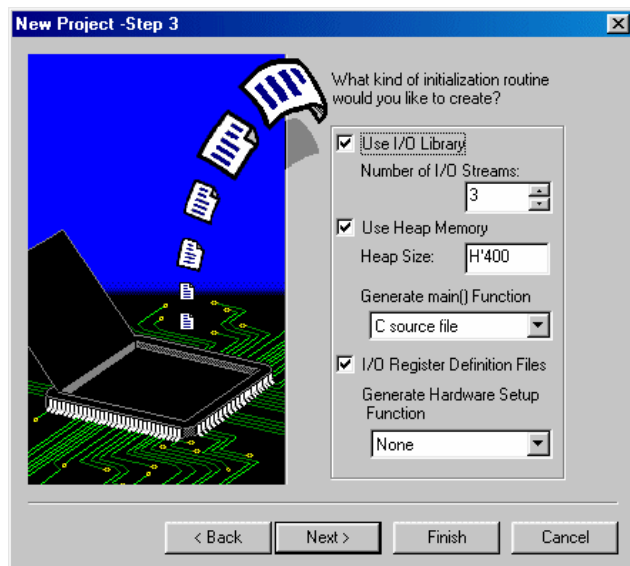
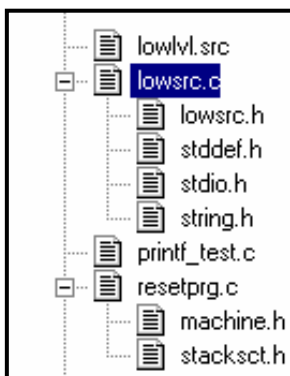


Figure 1 : Step 3 of the HEW Project Generator

The HEW project generator is run with all the usual options selected up to Step 3. Here the option to **Use I/O Library** can be selected. The number of I/O Streams has been left at three in this example. When this options is selected two extra source files are added to the project by the project generator. These are the C file `lowsrc.c` and the assembly file `lowlev.src`. The `lowsrc.c` C file contains many functions used by the Standard I/O Library functions. The `lowlev.src` file contains the `charput` and `charget` functions used to send and receive data to the SIM I/O window of HEW. If these functions are replaced with different code written either in C or assembly to input and output bytes of data from a different source the Standard I/O Library functions can be used in a real embedded system. The `lowsrc.c` file also `#includes` some of the



Standard I/O Library function header files and `lowsrc.h`. These are shown as dependencies in the HEW Workspace window. The `lowlev.src` assembly file has no dependencies.

The project generator will also change the `resetprg.c` file. The `PowerON_Reset()` function will now also call the `_INIT_IOLIB()` and `_CLOSEALL()` functions used to initialise the I/O Streams before calling `main()` and then close the I/O Streams after `main()` has returned. If these functions from the `lowsrc.c` file are not called the Standard I/O Library functions will not work correctly.

Figure 2 : Extra Files in the Workspace window of HEW

Listing from `resetprg.c`

```
void PowerON_Reset(void)
{
    set_imask_ccr(1);
    _INITSCT();
    _INIT_IOLIB(); // Use SIM I/O

    // errno=0; // Remove the comment when you use errno
    // srand(1); // Remove the comment when you use rand()
    // _slptr=NULL; // Remove the comment when you use strtok()
    // HardwareSetup(); // Remove the comment when you use
    // Hardware Setup

    main();
    _CLOSEALL(); // Use SIM I/O
    sleep();
}
```

4.0 Adding User Written charput and charget Functions

The charget and charput functions in the lowlvl.src file from the project generator can be replaced by user written functions (in either C or assembly). The user functions can be written to input and output bytes from a real peripheral in the embedded system. In the example used here a SCI serial port is used, but any available peripheral capable of inputting and outputting bytes of data could be used.

4.1 Example of Adding User Written charput and charget Functions

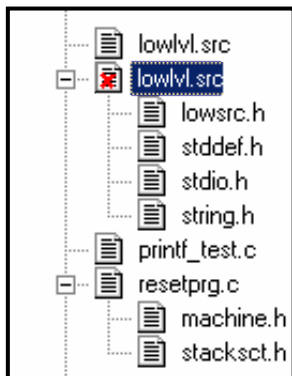


Figure 3 : lowlvl.src is excluded from the build

The first step taken here is to exclude the file lowlvl.src from the build. This is done by right clicking on the file in the HEW workspace window and selecting Exclude Build. The file could also be removed from the project altogether. A new file called putchar.c is written and added to the project. This file contains the user written charput and charget functions, which will input and output a byte of data using a serial port in the device. The functions in this example are adapted from one of the EVB tutorials. In this case serial port SCI1 is used. This code can be adapted to use any free serial port. The HardwareSetup function is written to initialise SCI1 for a baud rate of 19200 with a clock frequency of 18.432MHz. A PutString function is also added, this makes it easy to output a string to the serial port without using the printf standard library function. A listing of the two files is shown below.

Listing of HardwareSetup(), file hwsetup.c

```

/* Define constants for peripheral register values */
#define ORER_FER_PER 0x38      /* Error bits of SCI */
#define TE_RE 0x30           /* TX and RX enable bits of SCI */
#define BAUD38400 0x0E       /* BRR Setting for 38400 baud at 18.432MHz */
#define BAUD19200 0x1D       /* BRR Setting for 19200 baud at 18.432MHz */

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

void HardwareSetup(void);

void HardwareSetup(void)
{
    unsigned short c;

    MSTPCR.BYTE.L &= 0x1F; /* enable all SCI */
    SCI1.SCMR.BYTE = 0xF2; /* TX data LSB first-no invert-normal SCI mode */
    SCI1.SCR.BYTE &= 0x00; /* disable RX and TX-disable all SCI interrupts -
                           internal clock */

    SCI1.SMR.BYTE = 0x00; /* async mode-8 data bits-no parity-1 stop bit */
    // SCI1.BRR = BAUD38400; /* 38400 baud */
    SCI1.BRR = BAUD19200; /* 19200 baud */

    for(c=0;c<30000L;c++); /* one bit delay */

    SCI1.SCR.BYTE |= TE_RE; /* enable TX and RX */
    //do not set or clear interrupt mask when using monitor
    set_imask_ccr(0); /* Enable interrupts */
}

```

Listing for charput(), charget() and PutStr(), file putchar.c

```
/* Define constants for peripheral register values */
#define ORER_FER_PER 0x38      /* Error bits of SCI */
#define TE_RE 0x30           /* TX and RX enable bits of SCI */
#define BAUD38400 0x0E       /* BRR Setting for 38400 baud at 18.432MHz */
#define BAUD19200 0x1D       /* BRR Setting for 19200 baud at 18.432MHz */

#include "iodefine.h"

char charget(void);
void charput(char);
void PutStr(char *);

char charget(void)
{
    char InputChar;
    while (SCI1.SSR.BIT.RDRF == 0)
    {
        /* ignore errors */
        if ((SCI1.SSR.BYTE & ORER_FER_PER) != 0x00)
            SCI1.SSR.BYTE &= ~ORER_FER_PER;
    }
    InputChar = SCI1.RDR;
    SCI1.SSR.BIT.RDRF = 0;
    return (InputChar);
}

void charput(char OutputChar)
{
    while ((SCI1.SSR.BIT.TDRE) == 0);
    SCI1.TDR = OutputChar;
    SCI1.SSR.BIT.TDRE = 0;
}

void PutStr(char *str)
{
    while (*str != '\0')
        charput(*str++);
}

```

Now these user written charput() and charget() routines will be called to input and output characters as bytes of data. The user written PutStr() function takes a character string as its parameter (the string is declared as an array of chars, this is treated as a char * pointer). It then passes each char in the string to charput() to output the whole string to the serial port a byte at a time. Of course a simple function like this is not able to handle formatted strings in the way that printf() and other standard I/O routines can. Library I/O routines such as printf() and scanf() will also use the user written charput() and charget() functions. This means that printf() can now be used to output formatted data from the serial port of the device. If a terminal emulator, such as Windows HyperTerminal, is setup to connect to the serial port of the device the chars being sent from the serial port will be displayed for the user to see. In a real application a charput() routine could be written to output data to a LCD or LED display device for example.

5.0 Calling the printf() I/O Library Routine

Once the charput() and charget() routines have been successfully created the method for calling the printf() routine is similar to any other c compiler. The library routines that support using formatted I/O are quite large. These routines are used to convert different data types to ASCII text. They also require a large stack. The Renesas compiler allows the user to disable the use of the floating point formatters. If it is not necessary to handle floating point number formatting the program size can be kept to a minimum by not including the code for the floating point formatters. The floating point formatters can be disabled by including the file no_float.h prior to including the standard header file stdio.h.

The full formatters (with floating point formatters enabled) will add approximately 22kbytes to the code size of the project. If the no_float.h header is included and floating point formatters are disabled then the formatters will add approximately 8kbytes to the code size, around 14kbytes less than the full formatters. This is code size over and above the user written code and other libraries that are used. A call to printf can require around H'160 bytes of stack space if no float formatters are used. If float formatters are used this can increase to over H'200 bytes.

5.1 Example of calling printf()

In this example the no_float.h header file is firstly included, followed by the standard header stdio.h. The user written charput() function is called directly a number of times to output a series of individual characters. Then the user written PutStr() function is called to output a string from the serial port. Then a number of calls to printf() are used to output a menu to HyperTerminal. Within two of these calls an integer value is converted to ASCII by the formatter. To read in an integer value the library routine scanf() is used. This reads in a value entered using HyperTerminal and converts it from ASCII to an integer value. If you build this small example program you should find that entering 1 from HyperTerminal gives the same result as entering 01 (you will have to press return once you have pressed the numeric keys). This is because the formatter is converting the '1' character or the "01" string to an integer value of 1.

Listing for calls to printf(), file main.c

```
#include <no_float.h>
#include <stdio.h>

void main(void);
extern void PutStr(char *);
extern void charput(char);

void main(void)
{
    short InputValue;
    static const char string[] = {"Hello from PutStr \n\r"};
    charput('H'); charput('E'); charput('L'); charput('L'); charput('O');
    charput('\n'); charput('\r');
    PutStr((char *)string);
    while(1)
    {
        printf("\n\r");
        printf("Menu ----- \n\r");
        printf("Key to Press ----- \n\r");
        printf("%d) for exclamation mark -- \n\r", (unsigned short)1 );
        printf("%d) for question mark ----- \n\r", (unsigned short)2 );
        printf("\n\r");

        scanf("%d", &InputValue);

        if(InputValue == 1)
            printf("!!!!!!!!!!!!!!!!!!!!!! \n\r");
        else if(InputValue == 2)
            printf("???????????????????? \n\r");
        else
            printf("Invalid selection -- \n\r");
    }
}
```

Renesas Technology Europe.

The example code in this Application note was written to run on the EVB2357. It should be relatively simple to edit the code to run on other Renesas microcontrollers though. Although the formatters of the `stdio.h` library provide a quick and easy way to output formatted data, for many embedded applications the code size involved may be prohibitive. It may be more efficient for the user to write specific formatting routines to handle only the exact data that they need to output for their application.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corp. 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 Corp. or a third party.

2. Renesas Technology Corp. 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 represent information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. 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 Corp. 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 Corp. by various means, including the Renesas Technology Corp. 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 Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.

5. Renesas Technology Corp. 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 Corp. or an authorized Renesas Technology Corp. 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 Corp. 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 Corp. for further details on these materials or the products contained therein.