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

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.

H8/300L

Writing a printf function to LCD and a serial port (Bprintf)

Introduction

This application note demonstrated how to write a printf function. Either calling from the library function or custom coded. It also shows the ways to direct its output messages. i.e. either to the SIM IO window, LCD or serial port.

Unlike the PC, which has a standard input (keyboard) and output (console) device, developers have to define the input source and output destination for embedded system.

One such example is the printf function. Embedded developers have to define the output destination for this function. Commonly used outputs are LCD panel, and hyper terminal of PC (through the device serial port).

In this application note, five HEW 2.1 project files are provided, to demonstrate five different scenarios targeting at the SLP H8/38024F. The demonstration is done on the SLP CPU Board (or ALE300L) and the application board.

1. Simulated I/O
2. Printf to LCD panel (using standard library function)
3. Printf to Serial Port (using standard library function)
4. Basic Printf to LCD panel (using custom written code)
5. Basic Printf to Serial Port (using custom written code)

Target Device

H8/300L Super Low Power (SLP) series – H8/38024

Contents

1. Usage of Printf function

Printf is a commonly used function by embedded developer, to provide information to the outside world.

Two main usages are:

i. Providing a user interface (e.g. Blood pressure reading on the LCD panel of a blood pressure product).

ii. Provide a mean for debugging (e.g. sending of raw ADC reading to the PC hyper terminal through the serial port)

Note:

The following examples are build on HEW2.1 (H8 TINY / Super Low Power Tool chain). Previous version of HEW will not be able to access the project files. A simple mean is to create a new project based on user HEW version, and replace the essential C code and header file into the newly generated project directory.

2. Simulated I/O

Developers can start work on the HEW simulator, before the hardware is ready. In order to provide a more efficient debugging environment, Simulated I/O was introduced in the HEW simulator. Simulated I/O provides a mean for the developers to output their results (debugging information) to a debug (SIM IO) window within HEW. i.e. the printf function will direct its messages to be displayed in this window.

The setup of SIM I/O function is created by the HEW project generator. [For this example, the project is built on H8S, H8/300 standard Tool chain. The free H8 TINY / Super Low Power Tool chain do not have the simulator function.

A quick and simple guide to observe the effect of the simulated I/O is listed as follow

i. At step 3 of the project generation, select the option **Use I/O Library,** and **Use heap memory** and leave the number of I/O streams to be **3.**

ii. This will generate/ modify the following files

   a. Lowsrc.c (contain the low level standard I/O function)

   b. Lowlev.src (contain the charget and charput destination – to the PC SIM IO window)

   c. Resetprg.c (addition of the calling functions, _INIT_IOLIB() and _CLOSEALL() to initialized the standard I/O)

   d. Sbrk.c (allocation of heap memory)

iii. Add a printf() in the main routine (remember to add - #include <stdio.h>)

iv. Change the default debug session from ”Debug” to “SimDebug_H8-300L”

v. **Build** the project (F7)

vi. Goto Option/**Debug Setting** window, and set the session to “SimSessionH8-300L” and the following will be automatically set.

a. Setup the target as “H8/300L Simulator”

b. Setup the default Debug Format as “Elf/Dwarf2”

c. Add the download modules as the compiled file in step v.

vii. Change the session to “SimSession H8-300L”

viii. Goto Option/ simulator/ **Simulator Memory Resources**

Add memory available in the memory map as read/write. The allocated resources will be display in the system memory resources. [Automatically set by HEW]

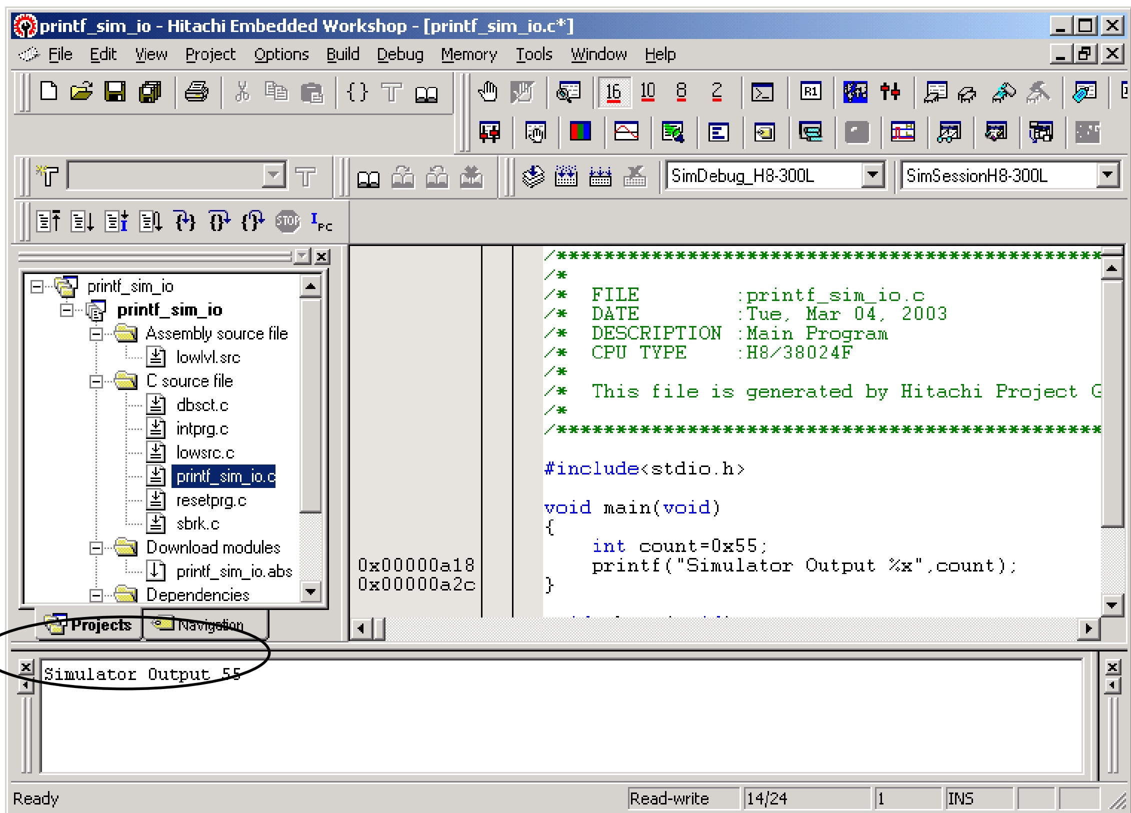
ix. Goto Option/ simulator/ **Simulator System** -> to enable system call address [Automatically set by HEW]

x. Goto Debug/**Download Modules** -> to load the files [Previously set in Debug setting]

xi. Goto View/ **Simulated IO**

xii. Goto Debug/**Reset Go**

xiii. Observe the message in the simulated I/O window



A detailed steps-by steps guide is provided in the HEW on-line user manual.

The following is the HEW-generated reset routine.

```
__entry(vect=0) void PowerON_Reset(void)
{
    set_imask_ccr(1);
   _INITSCT();
// _CALL_INIT();     // Remove the comment when you use global class object
   _INIT_IOLIB();    // Use SIM I/O
// errno=0;          // Remove the comment when you use errno
// srand(1);         // Remove the comment when you use rand()
// _s1ptr=NULL;      // Remove the comment when you use strtok()
// HardwareSetup();  // Remove the comment when you use Hardware Setup
   set_imask_ccr(0);

   main();

   _CLOSEALL();      // Use SIM I/O
// _CALL_END();      // Remove the comment when you use global class object
   sleep();
}
```

The following is HEW-generated Lowlev.src, to send character to SIM IO

```
         .EXPORT    _charput
         .EXPORT    _charget
SIM_IO:  .EQU       H'0000
         .SECTION   P,CODE,ALIGN=2
;------------------------------------------------------------------------
;  _charput:
;------------------------------------------------------------------------
_charput:
         MOV.B      R0L,@IO_BUF
         MOV.W      #H'0102,R0
         MOV.W      #IO_BUF,R1
         MOV.W      R1,@PARM
         MOV.W      #PARM,R1
         JSR        @SIM_IO
         RTS
;------------------------------------------------------------------------
;  _charget:
;------------------------------------------------------------------------
_charget:
         MOV.W      #H'0101,R0
         MOV.W      #IO_BUF,R1
         MOV.W      R1,@PARM
         MOV.W      #PARM,R1
         JSR        @SIM_IO
         MOV.B      @IO_BUF,R0L
         RTS
;------------------------------------------------------------------------
;  I/O Buffer
;------------------------------------------------------------------------
         .SECTION   B,DATA,ALIGN=2
PARM:    .RES.W     1
IO_BUF:  .RES.B     1
         .END
```

3. Modification of Charput and Charget functions

From the above examples, we can observe that the HEW function generator has generated all the necessary functions for the printf(), to output its message to the SIM IO window. If programmers have the intention to output the messages to other means, such as serial port and LCD, the charput function in lowsrc.src file can be modified. Programmers may write these two functions in C and placed in another .c file.

The following shows two examples of output mean:

i. Serial port

ii. LCD

3.1 Printf to Serial port

3.1.1 Coding Description

This demonstration makes use of the Serial Port 3 (SCI-3) in the SLP application board.

The main routine will initialize the general IO and the SCI-3. Next the printf function will output a series of characters through the charput function. The charput function will send the data out through the serial port.

The “no_float.h” must be declared before “stdio.h“, this is to reduce the printf function code size (if programmers are not going use the floating point formatter).

```
#include <no_float.h>
#include <stdio.h>
#include "iodefine.h"
#include <machine.h>

static const char string[] = {"\n\n\rCan putstr too!"};

void main(void)
{
   int count=0;

   init_io();
   init_sci();

   printf("\n\n\n\rDemonstration of printf function");
   printf("\n\rCounting = ");

   for (count=0;count<10;count++)
      printf(" %d",count);

   PutStr((char *)string);
}
```

The following elaborate the three functions that main function called;

i. void init_io(void); //general initialization routine

```
void init_io(void)
{
   P_IO.PCR3.BYTE = 0x00;    //P37..P31 : inputs
   P_IO.PUCR3.BYTE = 0x00;   //Turn off the MOS pull-up

   //PMR3 : |AEVL|AEVH|---|---|---|TMOFH|TMOFL|---|
   P_IO.PMR3.BYTE = 0x00;

   P_IO.PCR4.BYTE = 0xF8;    //P40 is connected to keypad 0

   //PMR2 : |---|---|POF1|---|---|---|---|IRQ0| : |1|1|0|1|1|0|0|1|
   P_IO.PMR2.BYTE = 0xD9;

   //PMR9 : |---|---|---|---|PIOFF|---|PWM2|PWM1|
   P_IO.PMR9.BYTE = 0xF0;

   //PMRB : |---|---|---|---|IRQ1|---|---|---|
   P_IO.PMRB.BYTE = 0xF7;

#if ALE300L_38024||ALE300L_3802
   init_sci();
#endif

   //IEGR : |---|---|---|---|---|---|IEG1|IEG0| : |1|1|1|0|0|0|0|0|
   P_SYSCR.IEGR.BYTE = 0xE0;

   //IENR1 : |IENTA|---|IENWP|---|---|IENEC2|IEN1|IEN0| : |0|0|0|0|0|
   P_SYSCR.IENR1.BYTE = 0x01;
   P_SYSCR.IENR2.BYTE = 0x10;

   set_imask_ccr(0);
}
```

ii. void init_sci(void); // initialization of SCI-3 to 2400bps, 8 bit, 1 stop bit, no parity

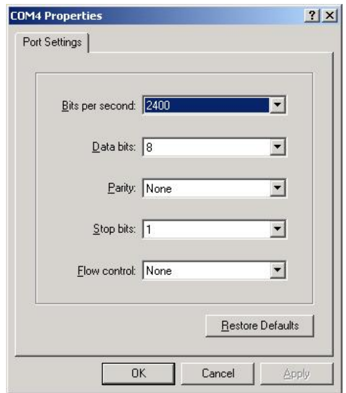
```
void init_sci(void)
{  unsigned char  temp = 0;

   //SCR3 : |TIE|RIE|TE|RE|MPIE|TEIE|CKE1|CKE0|
   //asynchronous mode, internal clock source, SCK32 functions as I/O port
   P_SCI3.SCR3.BYTE = 0x30;

   //SMR : |COM|CHR|PE|PM|STOP|MP|CKS1|CKS0| : |0|0|0|0|0|0|0|0|
   P_SCI3.SMR.BYTE = 0x00;

   //Bit rate = 19200 bps, n = 0, N = 64 // MODIFY TO 2400bps
   P_SCI3.BRR = 64;

   //SPCR : |---|---|SPC32|---|SCINV3|SCINV2|---|---| : |1|1|1|0|0|0|0|0|
   P_SCI3.SPCR.BYTE = 0xE0;

   //SSR  : |TDRE|RDRF|OER|FER|PER|TEND|MPBR|MPBT|
   P_SCI3.SSR.BYTE = 0x84;                //Initialise upon reset to 0x84
}
```

iii. void charput(char outputchar) // send data out to serial port 3

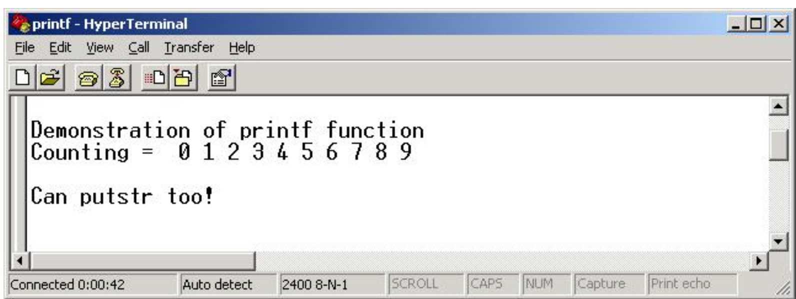
```
void charput(char OutputChar)              //Serial Port
{
   while ((P_SCI3.SSR.BIT.TDRE) == 0);

   P_SCI3.TDR = OutputChar;
   P_SCI3.SSR.BIT.TDRE = 0;
}
```

3.1.2 Hardware Setup

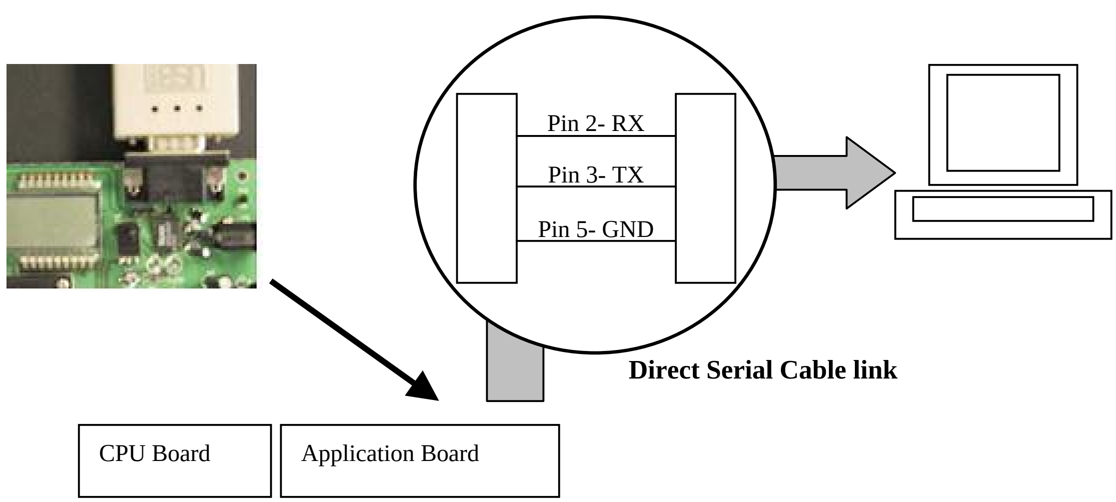
The PC window hyper terminal can be set up to view this message. A serial cable (direct pin to pin type) must be used to link the application board to the PC serial port. A snapshot of the hyper terminal setup is as shown.



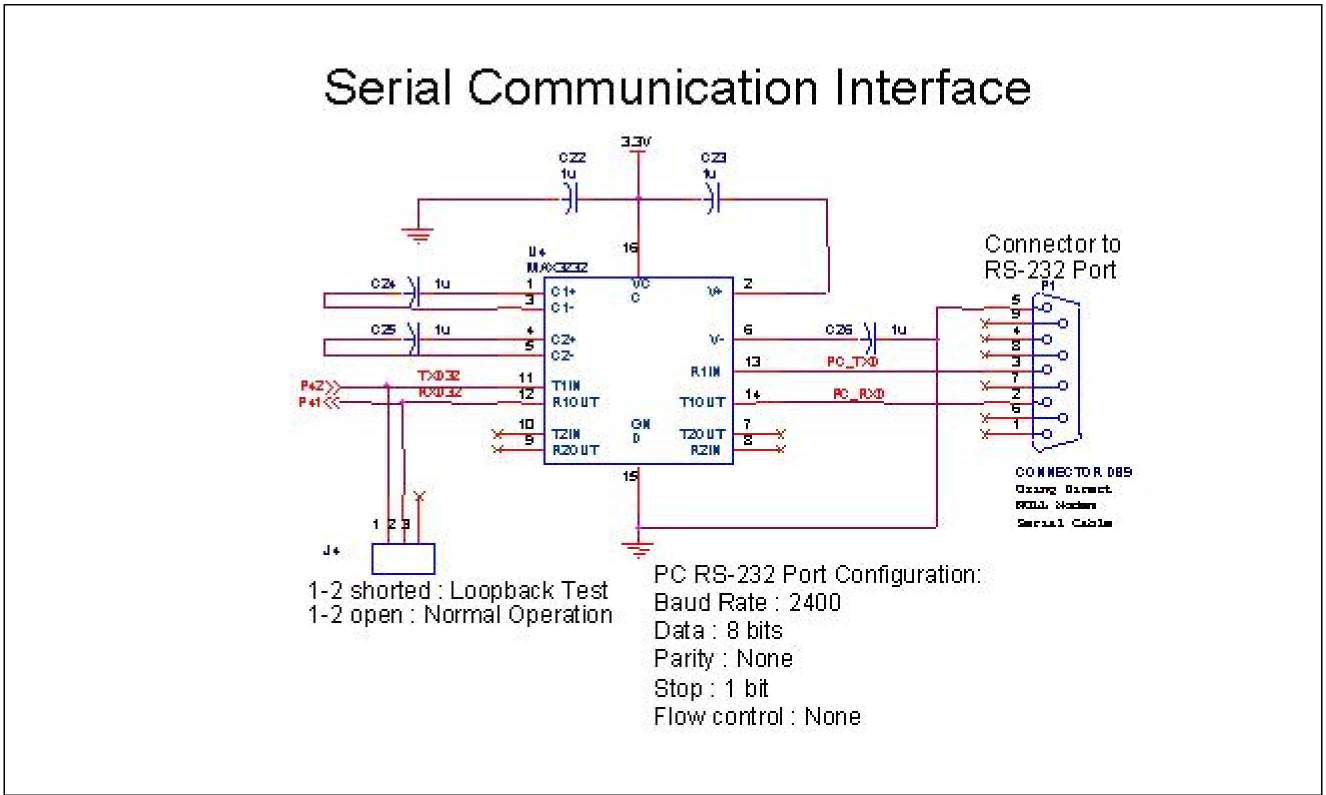
Another snap shot of the output message.



The physical setup:



A RS232 driver, which is built on the application board, is required to interface the MCU to the PC.



3.2 Printf to LCD

3.2.1 Coding Description

This demonstration makes use of the LCD in the SLP application board.

After the main routine initialized the general I/O and LCD, the standard library Printf function can be called. At this time, the charput function directs its output to the LCD (1 row x 7 characters). The global variable, *position*, determines the position of the character to be displayed on the LCD.

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

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

unsigned int position=7;

void main(void)
{
  char temp1 ='e';
  int  temp2 =15; //0xF

  init_io();
  init_lcd();

  printf("HI %c %x", (BYTE)temp1, (DWORD)temp2);
 }
```

The sample code contains four main functions;

i. void init_io(void); //general initialization routine (identical as section 3.1)

ii. void init_lcd(void) // initialization of LCD

```
void init_lcd(void)
{
   unsigned char  temp_a;
   unsigned char  *dest;

   //clear LCD RAM
   dest = (unsigned char *)0xF740;
   for (temp_a = 0 ; temp_a < 16 ; temp_a++)
   {
      *dest++ = 0;
   }

   //LPCR : |DTS1|DTS0|CMX|---|SGS3|SGS2|SGS1|SGS0| : |1|1|0|0|0|1|1|0|
   //|DTS1|DTS0| = |1|1| : 1/4 duty
   //|CMX | = 0
   //Bit 4 is reserved; only 0 can be written to this bit
   //|SGS3|SGS2|SGS1|SGS0| = |1|0|0|0| : Use SEG1 to SEG32
   P_LCD.LPCR.BYTE = 0xC8;   //1/4 duty cycle

   //LCR : |---|PSW|ACT|DISP|CKS3|CKS2|CKS1|CKS0| : |1|1|1|1|1|1|1|1|
   //Bit 7 is reserved; always read as 1 and cannot be modified
   //PSW = 1 : LCD drive power supply on
   //ACT = 1 : LCD controller/driver operates
   //DISP = 1 : LCD RAM data is displayed
   P_LCD.LCR.BYTE = 0xFF;    //display is faint

   //LCR2 : |LCDAB|---|---|---|---|---|---|---|
   //LCDAB : 0 : drive using A waveform
   //Bits 6 and 5 are reserved; always read as 1 and cannot be modified
   //Bits 4 to 0 are reserved; only 0 can be written to these bits
   P_LCD.LCR2.BYTE = 0x60;
}
```

iii. void Display_number(…) // display the 'number' in the LCD, which can only displays 7 characters

```
void display_number(unsigned char digit, unsigned char number, unsigned char
decimal_point)
{
   unsigned short *dest;

   switch(digit)
   {
      case 0:     dest = (unsigned short *)0xF740;
               break;
      case 1:     dest = (unsigned short *)0xF742;
               break;
      case 2:     dest = (unsigned short *)0xF744;
               break;
      case 3:     dest = (unsigned short *)0xF746;
               break;
      case 4:     dest = (unsigned short *)0xF748;
               break;
      case 5:     dest = (unsigned short *)0xF74A;
               break;
      case 6:     dest = (unsigned short *)0xF74C;
               break;
      case 7:     dest = (unsigned short *)0xF74E;
               break;
      default:    break;
   }

   if (decimal_point)
      *dest = (unsigned short)(lcd_number_data[number] | 0x0800);
   else
      *dest = lcd_number_data[number];
}
```

iv. void charput(char outputchar) // call display_number() to display characters.

```
void charput(char OutputChar)
{
    display_number(position, OutputChar, 0);
    position--;
}
```

v. Printf_lcd.h // look up table to convert character to be display in LCD

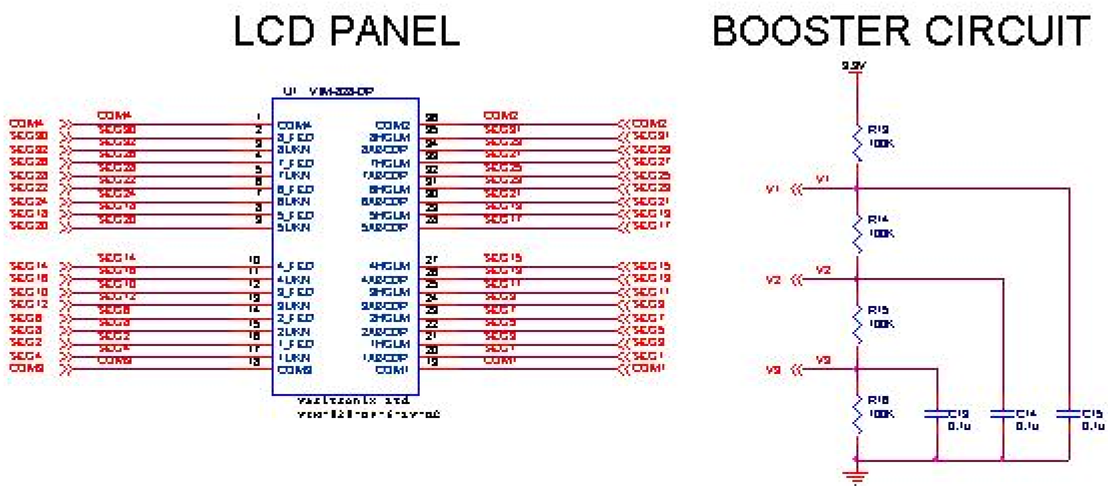
```
//for 1/4 duty cycle
const unsigned short lcd_number_data[128]
= {                   //ASCII
                      0x0039, //  00.   'NUL'     :Display
                      0x0039, //  01.   'SOH'     :Display
                      0x0039, //  02.   'STX'     :Display
                      0x0039, //  03.   'ETX'     :Display
                      0x0039, //  04.   'EOT'     :Display
                      0x0039, //  05.   'ENQ'     :Display
                      0x0039, //  06.   'ACK'     :Display
                      0x0039, //  07.   'BEL'     :Display
                      0x0039, //  08.   'BS'      :Display
                      0x0039, //  09.   'HT'      :Display
                      0x0039, //  0A.   'LF'      :Display
                      0x0039, //  0B    'VT'      :Display
                      0x0039, //  0C.   'FF'      :Display
                      0x0039, //  0D.   'CR'      :Display
                      0x0039, //  0E.   'SO'      :Display
                      0x0039, //  0F.   'SI'      :Display
                      0x0039, //  10.   'DLE'     :Display
                      0x0039, //  11.   'DC1'     :Display
                      0x0039, //  12.   'DC2'     :Display
                      0x0039, //  13.   'DC3'     :Display
                      0x0039, //  14.   'DC4'     :Display
                      0x0039, //  15.   'NAK'     :Display
                      0x0039, //  16.   'SYN'     :Display
                      0x0039, //  17.   'ETB'     :Display
                      0x0039, //  18.   'CAN'     :Display
                      0x0039, //  19.   'EM'      :Display
                      0x0039, //  1A.   'SUB'     :Display
                      0x0039, //  1B.   'ESC'     :Display
                      0x0039, //  1C.   'FS'      :Display
                      0x0039, //  1D.   'GS'      :Display
                      0x0039, //  1E.   'RS'      :Display
                      0x0039, //  1F.   'US'      :Display
                      0x0000, //  20.   'SP'      :Space, all segment off
                      0x0039, //  21.   '!'       :Display
                      0x2200, //  22.   '"'       :"
                      0x0039, //  23.   '#'       :Display
                      0xA55A, //  24.   '$'       :$
                      0x0039, //  25.   '%'       :Display
                      0x0039, //  26.   '&'       :Display
                      0x0010, //  27.   '''       :'
                      0x0039, //  28.   '('       :Display
                      0x0039, //  29.   ')'       :Display
                      0x00E7, //  2A.   '*'       :*
                      0x005A, //  2B.   '+'       :+
```

```
0x0039, //  2C.   ','       :Display
0x0042, //  2D.   '-'       :-
0x0800, //  2E.   '.'       :.
0x0024, //  2F.   '/'       :/
0xE724, //  30.   '0'       :0
0x0600, //  31.   '1'       :1
0xC342, //  32.   '2'       :2
0x8742, //  33.   '3'       :3
0x2642, //  34.   '4'       :4
0xA542, //  35.   '5'       :5
0xE542, //  36.   '6'       :6
0x0700, //  37.   '7'       :7
0xE742, //  38.   '8'       :8
0x2742, //  39.   '9'       :9
0x0039, //  3A.   ':'       :Display
0x0039, //  3B.   ';'       :Display
0x00A0, //  3C.   '<'       :<
0x8100, //  3D.   '='       :=
0x0005, //  3E.   '>'       :>
0x0039, //  3F.   '?'       :Display
0x0039, //  40.   '@'       :Display
0x6742, //  41.   'A'       :A
0xE442, //  42    'B'       :b
0xE100, //  43.   'C'       :C
0xC642, //  44.   'D'       :D
0xE142, //  45.   'E'       :E
0x6142, //  46.   'F'       :F
0xE540, //  47.   'G'       :G
0x6642, //  48.   'H'       :H
0x8118, //  49.   'I'       :I
0xC600, //  4A.   'J'       :J
0x60A2, //  4B.   'K'       :K
0xE000, //  4C.   'L'       :L
0x6621, //  4D.   'M'       :M
0x6681, //  4E.   'N'       :N
0xE700, //  4F.   'O'       :O
0x6342, //  50.   'P'       :P
0xE780, //  51.   'Q'       :Q
0x63C2, //  52.   'R'       :R
0xA542, //  53.   'S'       :S
0x0118, //  54.   'T'       :T
0xE600, //  55.   'U'       :U
0x0681, //  56.   'V'       :V
0x6684, //  57.   'W'       :W
0x00A5, //  58.   'X'       :x
0x0029, //  59.   'Y'       :Y
0x8124, //  5A.   'Z'       :Z
0xE100, //  5B.   '['       :[
0x0081, /*  5C.   '\'       :\       */
0x8700, //  5D    ']'       :]
0x0084, //  5E.   '^'       :^
0x0000,  //  5F.   ' '       :
0x0001, //  60.   '`'       :`
0xC742, //  61.   'a'       :a
```

```
                    0xE442, //  62     'b'        :b
                    0xC042, //  63.    'c'        :c
                    0xC642, //  64.    'd'        :d
                    0xE142, //  65.    'e'        :E
                    0x6142, //  66.    'f'        :F
                    0xE540, //  67.    'g'        :G
                    0x6442, //  68.    'h'        :h
                    0x8118, //  69.    'i'        :I
                    0xC600, //  6A.    'j'        :J
                    0x60A2, //  6B.    'k'        :K
                    0xE000, //  6C.    'l'        :L
                    0x444A, //  6D.    'm'        :m
                    0x4442, //  6E.    'n'        :n
                    0xC442, //  6F.    'o'        :o
                    0x6342, //  70.    'p'        :P
                    0xE780, //  71.    'q'        :Q
                    0x63C2, //  72.    'r'        :R
                    0xA542, //  73.    's'        :S
                    0x0118, //  74.    't'        :T
                    0xE600, //  75.    'u'        :U
                    0x0681, //  76.    'v'        :V
                    0x6684, //  77.    'w'        :W
                    0x00A5, //  78.    'x'        :x
                    0x0025, //  79.    'y'        :Y
                    0x8124, //  7A.    'z'        :Z
                    0x0039, //  7B.    '{'        :Display
                    0x0018, //  7C.    '|'        :|
                    0x0039, //  7D.    '}'        :Display
                    0x0039, //  7E.    '~'        :Display
                    0x0039, //  7F.    'DEL'      :Display
};
```

3.2.2 Hardware setup

The LCD glass is directly connected to the SLP MCU.



A snap shot of the application board's LCD display .

```
char k='e';
int j =15;

printf("Hi %c%x",(BYTE)k,(DWORD)j);
```

4. Disadvantages of printf() library function

The printf() is a easy to use but a very complex library function. (refers to the H8 compiler user manual for the detail function description) When used in SLP MCU, the function has the following disadvantages

4.1 ROM size

The printf() functions takes up a huge ROM space, as it has to deal with IO stream, heap and lots of arguments. (refer to section 6 for the detail comparison)

Note: SLP H8/38024 has a maximum of 32K Bytes ROM

4.2 RAM size

As printf() has to use the heap memory, a size of H’80 Bytes are reserved purely for printf (If the programmers are not going use the heap for other purposes).

Note: SLP H8/38024 has a maximum of 1K Bytes RAM.

5. Custom written Printf function – Bprintf()

The solution is to custom write a function. In this application note, a Basic Printf (Bprintf) function is generated. Programmers may like to further customize it to their application need.

As compared to the implementation stated in session 2-3, this Bprintf function does not need

i. Lowsrc.c

ii. Lowlev.src

iii. _INIT_IOLIB() and _CLOSEALL() functions call in Resetprg.c

iv. sbrk.c

In another word, it does not require the IO stream and heap memory. It is just a custom written function.

5.1 Function Usage

The Prototype

```
BYTE Bprintf(const char *fmt, BYTE arg1, DWORD arg2);
```

Generally, this function can be called just like normal printf function except that it is limited by

i. The no of display characters (20)

ii. There are only two arguments

iii. The first argument is a BYTE, whereas the second is a DWORD.

iv. The supported arguments are: %x, %c, & %u.

Example of usage:

i. Bprintf("Hello world",0,0);

ii. Bprntf("\n\rGet data = %x from address %x", (BYTE)data, (DWORD)address);

5.2 Functions Description

Unlike the library printf(), the Bprintf() is a simplify version of printf function. It is restricted by

i. MAXCHARS, which determine the string size. This is also the major components to determine the depth of stack used.

ii. The three cases statement, which restrict the use of argument to only %x, %c & %u

iii. The variable, num, which determine the number of arguments to two.

The Bprintf() function will called the charput function to output the message to the destination.

```
#define MAXCHARS       20
#define LEN 9

BYTE Bprintf(const char *fmt, BYTE arg1, DWORD arg2);
void itoab(char **buf, DWORD i, unsigned int base);


BYTE Bprintf(const char *fmt, BYTE arg1, DWORD arg2)
{
   DWORD u;
   BYTE num=0;               // argument index
   BYTE index=0;             // string index
   char buf[MAXCHARS];
   char *buf_ptr;

   buf_ptr = buf;

   // Rearranging output strings
   while (*fmt && index<(MAXCHARS-1))
   {  if (*fmt != '%')
            *buf_ptr++ = *fmt++;         // store string into buf
      else
      {   switch (*++fmt)                // if %, check what type
         {
              case 'x':                  // %x, hexadecimal unsigned number
               if (num == 0)
                  u = (DWORD)arg1;
               else if (num == 1)
                     u = (DWORD)arg2;
                  else
                     break;              //ignore > 2 arg
                  num++;
               *buf_ptr++ = '0';
               *buf_ptr++ = 'x';
                     b_itoab((char **)&buf_ptr, u, (unsigned int)16);
                     break;
```

```
               case 'u':                // %u, decimal unsigned number
               if (num == 0)
                  u = (DWORD)arg1;
               else if (num == 1)
                     u = (DWORD)arg2;
                  else
                     break;             //ignore > 2 arg
                  num++;
                     b_itoab((char **)&buf_ptr, u, (unsigned int)10);
                     break;

               case 'c':                // %c, a single character
               if (num==0)
                  *buf_ptr++ = (char)arg1;
               else if (num == 1)
                  *buf_ptr++ = (char)arg2;
                  else
                     break;             //ignore > 2 arg
                  num++;
                       break;

               default:
                       break;
            }// end switch
            fmt++;
         }// end else
   } //end while

   *buf_ptr = 0;                        // end of string indicator

   // Output rearranged string
   buf_ptr = buf;
      for (index = 0 ; *buf_ptr != (char)0 & index < MAXCHARS ; index++)
      charput(*buf_ptr++);              // output

      return(index);
}
```

The function b_itoab(), which perform conversion from integer to ASCII, has limited the integers to be based either 16 or 10.

```
void b_itoab(char **buf, DWORD i, unsigned int base)
{
      BYTE index=0;
      DWORD rem;
      char conv[LEN];

      if (i == 0)
      {
      (*buf)[0] = '0';
         ++(*buf);
      return;
   }
      conv[index++] = 0;
      while (i)
      {
      rem = i % base;
      if (base == 10)
         conv[index++] = rem + '0';
      else if (base == 16)
      {
         if (rem < 10)
               conv[index++] = rem + '0';
            else
               conv[index++] = rem + 'A'- 0xA;
       }
      i /= base;
      }
      while (conv[--index])
      {
      (*buf)[0] = conv[index];
         ++(*buf);
      }
}
```

The Bprintf() functions can be further customized to

i. Increase the string size

ii. Include other arguments: %f, %3d, %o…

iii. Increase the number of arguments

iv. Increase/ Decrease the argument size (BYTE, WORD, DWORD)

6. Comparison

In order to make a comparison of the code size, the map files for the following project is generated (under Option/ Link Library/List). To simplify the comparison, the size of section P is used to gauge the function's size. Both the debug & release(with optimization) setting are used.

	Debug	Release
Printf LCD (without include <no_float.h>)	20.4KBytes	14.9KBytes
Printf LCD (with include <no_float.h>)	7.4KBytes	5.6KBytes
Bprintf LCD.	1.1KBytes	0.8KBytes

The estimated code size for printf() with floating point formatters support is about 20.4Kbytes. When <no_float.h> is included, it can be reduced to about 7.4Kbytes. However if a customized function is written, the code size is further reduced by 7 times to 1.1K. Another reminder to the reader is that SLP maximum ROM size is just a mere 32K!

For RAM size, the custom Bprintf() function will used about 30 Bytes of stack (when MAXCHAR=20), whereas the library printf() function will reserved 128 (H'80) Bytes of heap memory.

7. Conclusion

As SLP has small ROM size, and most of its targeted application do not required the complication of IO streams. Thus the usage of the library printf() function is not efficient. Programmers are advised to customize their own printf() function.

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Sep.03	—	First edition issued

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.