お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

H8/300L Super Low Power (SLP)シリーズ

LCD、シリアルポートに対する printf 関数

内容

本アプリケーションノートでは、printf 関数の書き方について説明します。ライブラリ関数から呼び出す方法とユーザ作成の方法を扱います。また、メッセージの出力先を SIM IO ウィンドウ、LCD、シリアルポートにする方法も示します。

はじめに

標準入力（キーボード）と標準出力（ディスプレイ）機器をもつ PC と異なり、組み込みシステムでは、開発者は入力元と出力先を定義する必要があります。

一例として、printf 関数があります。組み込みシステム開発者は、この関数の出力先を定義しなくてはなりません。広く用いられている出力としては、LCD パネルや、PC のハイパーターミナル（デバイスのシリアルポート経由）があります。

本アプリケーションノートでは、5 つの HEW2.1 プロジェクトファイルを用いて、SLP H8/38024F 向けの次の 5 種類の例を説明します。操作例は、SLP CPU ボード（ALE300L）とアプリケーションボードを用いた例です。

1. Simulated I/O
2. LCD パネルへの printf（標準ライブラリ関数を使用）
3. シリアルポートへの printf（標準ライブラリ関数を使用）
4. LCD パネルへの基本的な printf（ユーザ作成のプログラムを使用）
5. シリアルポートへの基本的な printf（ユーザ作成のプログラムを使用）

動作確認デバイス

H8/38024

目次

1. printf 関数の使用法

printf は組み込みシステム開発者が広く用いている関数で、情報を外部に出力する機能を持ちます。主な使用法は次の 2 つです。

i) ユーザインタフェースの提供（例：血圧関係の機器で LCD パネルに血圧を表示）
ii) デバッグ手段の提供（例：生の ADC 値をシリアルポートから PC のハイパーターミナルに送信）

【注】 以下の例は、HEW2.1（H8 TINY/Super Low Power ツールチェイン）でビルドしたものです。ここで用いるプロジェクトファイルには、以前のバージョンの HEW ではアクセスできません。ご使用のバージョンが異なる場合は、ご使用の HEW で新しいプロジェクトを作成し、その新しいプロジェクトディレクトリで必要な C プログラムとヘッダファイルを置き換えるのが、最も簡単な方法です。

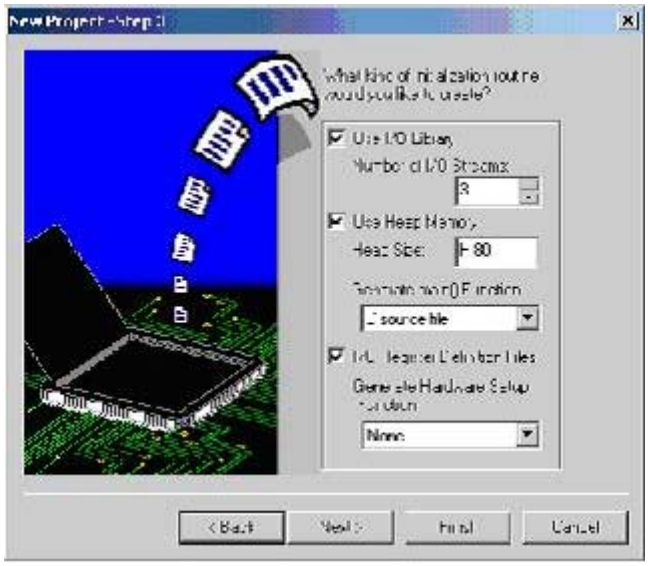
2. Simulated I/O

ハードウェアの準備ができる前に、HEW シミュレータで作業を開始できます。デバッグ環境の効率をさらに向上するために、HEW シミュレータでは Simulated I/O を導入しました。Simulated I/O は、開発者が結果（デバッグ情報）を HEW のデバッグ（SIM IO）ウィンドウに出力できるようにします。すなわち、printf 関数はメッセージをこのウィンドウに表示できます。

SIM I/O 機能の設定は、HEW プロジェクトジェネレータが自動的に行ないます。（この例では、プロジェクトは H8S, H8/300 標準ツールチェインでビルドしてあります。）無料の H8 TINY/Super Low Power ツールチェインにはシミュレータ機能がありません。

Simulated I/O の結果を確認する方法を以下に簡単に示します。

i) プロジェクト生成のステップ 3 で、**Use I/O Library** オプションと **Use heap memory** オプションを選択し、I/O ストリーム数を **3** のままにします。

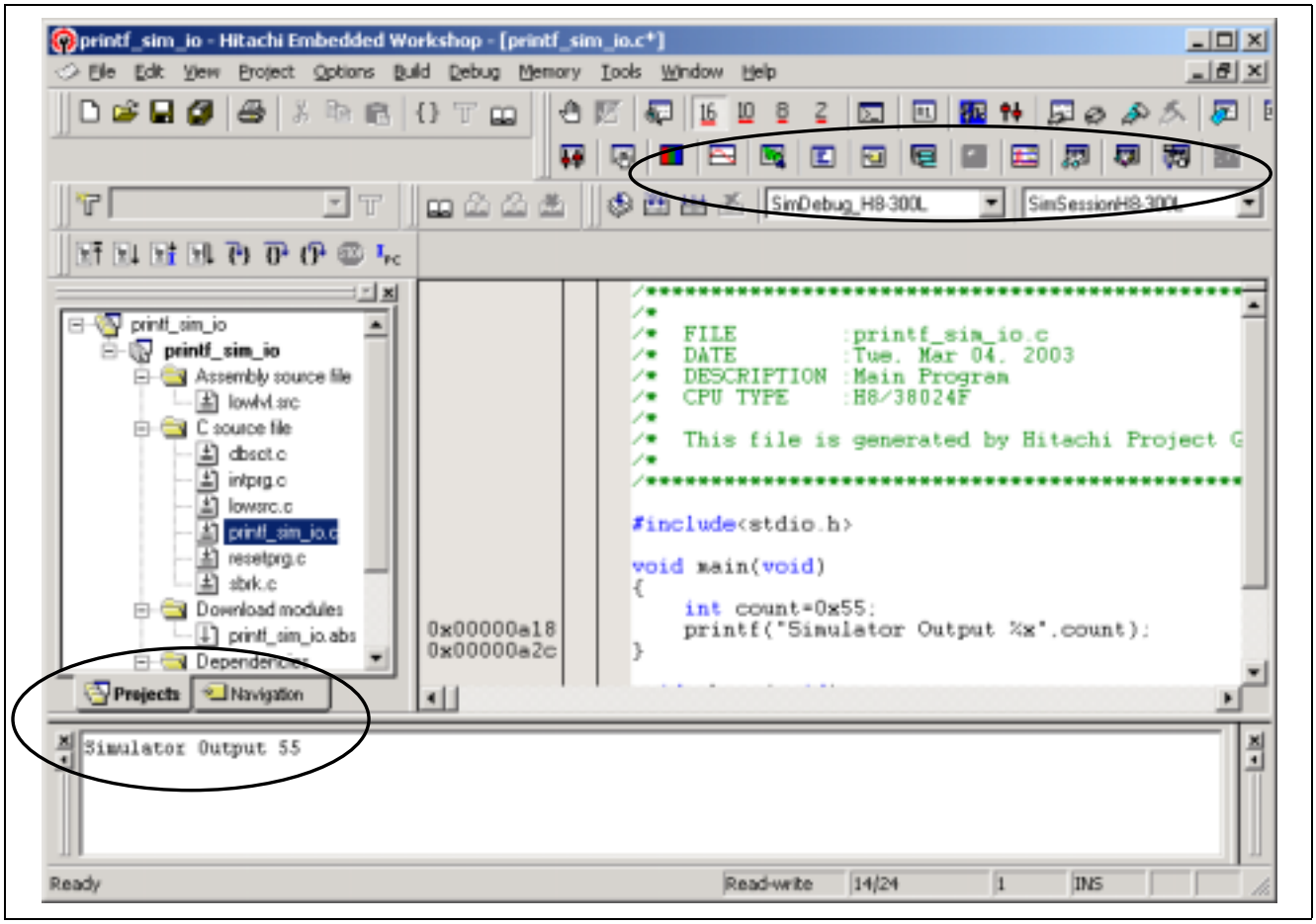


ii) これにより、次のファイルが生成／変更されます。
   a. Lowsrc.c（低レベル標準入出力機能）
   b. Lowlev.src（charget と charput の出力先を含んでいます – PC SIM IO ウィンドウ）
   c. Resetprg.c（関数_INIT_IOLIB()および_CLOSEALL()の呼び出しを追加し、標準 I/O を初期化します）
   d. Sbrk.c（ヒープメモリを割り当てます）

iii) printf()をメインルーチンに追加します（必ず#include <stdio.h>を追加してください）。
iv) デバッグセッションのデフォルトを”Debug”から“SimDebug_H8-300L”に変更します。
v) プロジェクトを**ビルド**します（F7）。
vi) Option/**Debug Setting** ウィンドウを開き、セッションを“SimSessionH8-300L”に設定すると、以下が自動的に設定されます。
   a. ターゲットを“H8/300L Simulator”に設定
   b. デバッグフォーマットのデフォルトを“Elf/Dwarf2”に設定
   c. ステップ v でのコンパイルファイルとしてダウンロードモジュールを追加

vii) セッションを“SimSession H8-300L”に変更します。
viii) Option/ simulator/ **Simulator Memory Resources** を開きます。
   a. メモリマップで、使用可能なメモリをリード可／ライト可を指定して追加します。割り当てたリソースは、システムメモリリソースとして表示されます（HEW が自動的に設定します）。

ix) Option/ simulator/ **Simulator System** を開き、システムコールアドレスをイネーブルにします（HEW が自動的に設定します）。
x) Debug/**Download Modules** を開き、ファイル（Debug setting であらかじめ設定したファイル）をロードします。
xi) View/ **Simulated IO** を開きます。
xii) Debug/**Reset Go** を開きます。
xiii) Simulated I/O ウィンドウのメッセージを確認します。



詳細な手順と説明は、HEW のオンラインユーザーズマニュアルを参照してください。

以下に HEW が生成したリセットルーチンを示します。

```
__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();
}
```

以下に HEW が生成した Lowlev.src を示します。文字を SIM IO に送出します。

```
          .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. charput 関数と charget 関数の変更

上に示した例から、printf()が SIM IO ウィンドウにメッセージを出力するのに必要な関数を、HEW 関数ジェネレータがすべて生成したことがわかります。メッセージをシリアルポートや LCD などの他の手段に出力したい場合は、lowsrc.src ファイルの charput 関数を変更します。この 2 つの関数を C 言語で記述して、他の.c ファイルに含めることができます。

以下に、次の 2 つ出力手段について例を示します。

i) シリアルポート
ii) LCD

3.1 シリアルポートへ出力するための printf

3.1.1 プログラムの説明

SLP アプリケーションボードのシリアルポート 3（SCI-3）を用いる例を説明します。

main ルーチンは汎用 I/O と SCI-3 を初期化します。次に、printf 関数が charput 関数を用いて一連の文字を出力します。charput 関数はシリアルポート経由でデータを出力します。

“no_float.h”は“stdio.h“より先に宣言しなくてはいけません。これにより、printf 関数のプログラムサイズを削減します（浮動小数点フォーマッタを用いない場合）。

```
#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);
}
```

以下に、main 関数が呼び出す 3 つの関数について詳しく説明します。

1. void init_io(void); //汎用初期化ルーチン

```
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|---|
    //AEVL = 0 : P37 as I/O
    //AEVH = 0 : P36 as I/O
    //TMOFH = 0: P32 as I/O
    //TMOFL = 0: P31 as I/O
    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|
    //Bits 7, 6, 4 and 3 are reserved and always read as 1 and cannot be
     //modified
    //POF1 = 0 (initial value)
    //Only 0 can be written to reserved bits 2 & 1.
    //IRQ0 = 1 : functions as IRQ0_N input pin
    P_IO.PMR2.BYTE = 0xD9;

    //PMR9 : |---|---|---|---|PIOFF|---|PWM2|PWM1|
    //PIOFF = 0 : large-current port step-up circuit is turned on
    //PWM1 = PWM2 = 0 : P90 and P91 functions as P_IO output pin
    P_IO.PMR9.BYTE = 0xF0;

    //PMRB : |---|---|---|---|IRQ1|---|---|---|
    //IRQ1 = 0 : PB3 functions as I/O or AN3
    //Bits 7 to 4 and 2 to 0 are reserved and always read as 1.
    P_IO.PMRB.BYTE = 0xF7;

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

```
    //IEGR : |---|---|---|---|---|---|IEG1|IEG0| : |1|1|1|0|0|0|0|0|
    //Bits 7 to 5 are reserved; they are always read as 1 and cannot be
    //modified
    //Bits 4 to 2 are reserved; only 0 can be written to these bits
    //IEG0 = 0 : Falling edge of IRQ0_N is detected
    P_SYSCR.IEGR.BYTE = 0xE0;

    //IENR1 : |IENTA|---|IENWP|---|---|IENEC2|IEN1|IEN0| : |0|0|0|0|0|
    //IENTA = 0 : Disable Timer A interrupt request
    //Bit 6 is reserved; only 0 can be written to it
    //IENWP = 0 : Disable WKP7_N TO WKP0_N interrupt requests
    //Bits 4 and 3 are reserved; only 0 can be written to these bits
    //IENEC2 = 0 : Disable IRQAEC interrupt request
    //IEN1 = 0 : Disable interrupt request from IRQ1_N
    //IEN0 = 1 : Enable interrupt request from IRQ0_N
    P_SYSCR.IENR1.BYTE = 0x01;
    P_SYSCR.IENR2.BYTE = 0x10;
    set_imask_ccr(0);
}
```

2. void init_sci(void); // SCI-3 を 2400bps、8 ビット、1 ストップビット、パリティなしに初期化する

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

   //SCR3 : |TIE|RIE|TE|RE|MPIE|TEIE|CKE1|CKE0|
   //TIE  : Transmit interrupt enable
   //RIE  : Receive interrupt enable
   //TE   : Transmit enable
   //RE   : Receive enable
   //MPIE : Multiprocessor interrupt enable
   //TEIE : Transmit end interrupt enable
   //CKE1 : Clock enable 1
   //CKE0 : Clock enable 0
   //CKE1 = CKE0 = 0
   //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|
   //COM : Communication Mode  : 0 : asynchronous mode
   //CHR : Character Length    : 0 : character length = 8 bits
   //PE  : Parity Enable       : 0 : parity bit addition and checking disabled
   //PM  : Parity Mode : 0 : even parity (no effect since parity is already
   //disabled)
   //STOP: Stop Bit Length     : 0 : 1 stop bit
   //MP  : Multiprocessor Mode: 0 : multiprocessor communication function
   //disabled
   //|CKS1|CKS0| : Clock Select: |0|0| : clock source for baud rate generator
   //= clk
   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|
   //SPC32 = 1 : P42 functions as TXD32 output pin
   //need to set TE bit in SCR3 after setting this bit to 1
   //SCINV3 = 0 : TXD32 output data is not inverted
   //SCINV2= 0 : RXD32 input data is not inverted
   //Bits 7 and 6 are reserved and always read as 1
   //Bits 4, 1 and 0 are reserved and only 0 can be written to these bits
   P_SCI3.SPCR.BYTE = 0xE0;

   //SSR  : |TDRE|RDRF|OER|FER|PER|TEND|MPBR|MPBT|
   //TDRE : transmit data register empty
   //RDRF : receive data register full
   //OER  : overrun error
   //FER  : framing error
   //PER  : parity error
   //TEND : transmit end
   //MPBR : Multiprocessor bit receive
   //MPBT : Multiprocessor bit transfer
   P_SCI3.SSR.BYTE = 0x84;               //Initialise upon reset to 0x84
}
```

3. void charput(char outputchar) // データをシリアルポート 3 に出力する

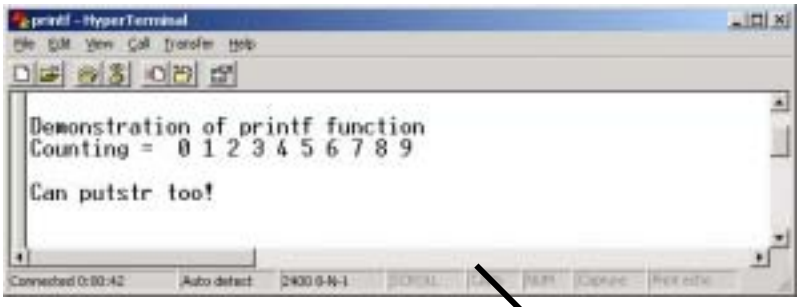
```
void charput(char OutputChar)              //Serial Port
{
   while ((P_SCI3.SSR.BIT.TDRE) == 0);

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

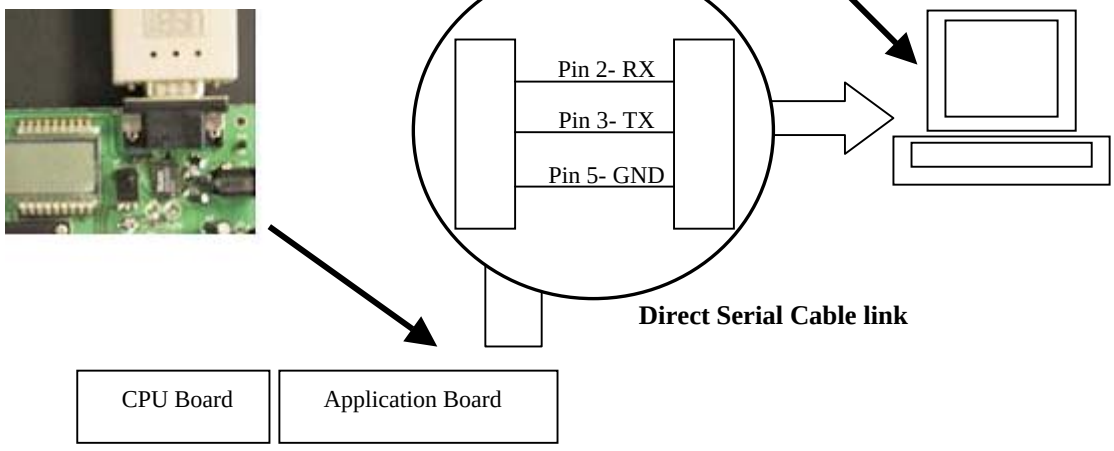
3.1.2 ハードウェアの設定

PC ウィンドウハイパーターミナルを設定して、このメッセージを表示させることができます。シリアルケーブル（ストレートケーブルタイプ）を用いてアプリケーションボードと PC シリアルポートを接続します。ハイパーターミナル設定の図を以下に示します。

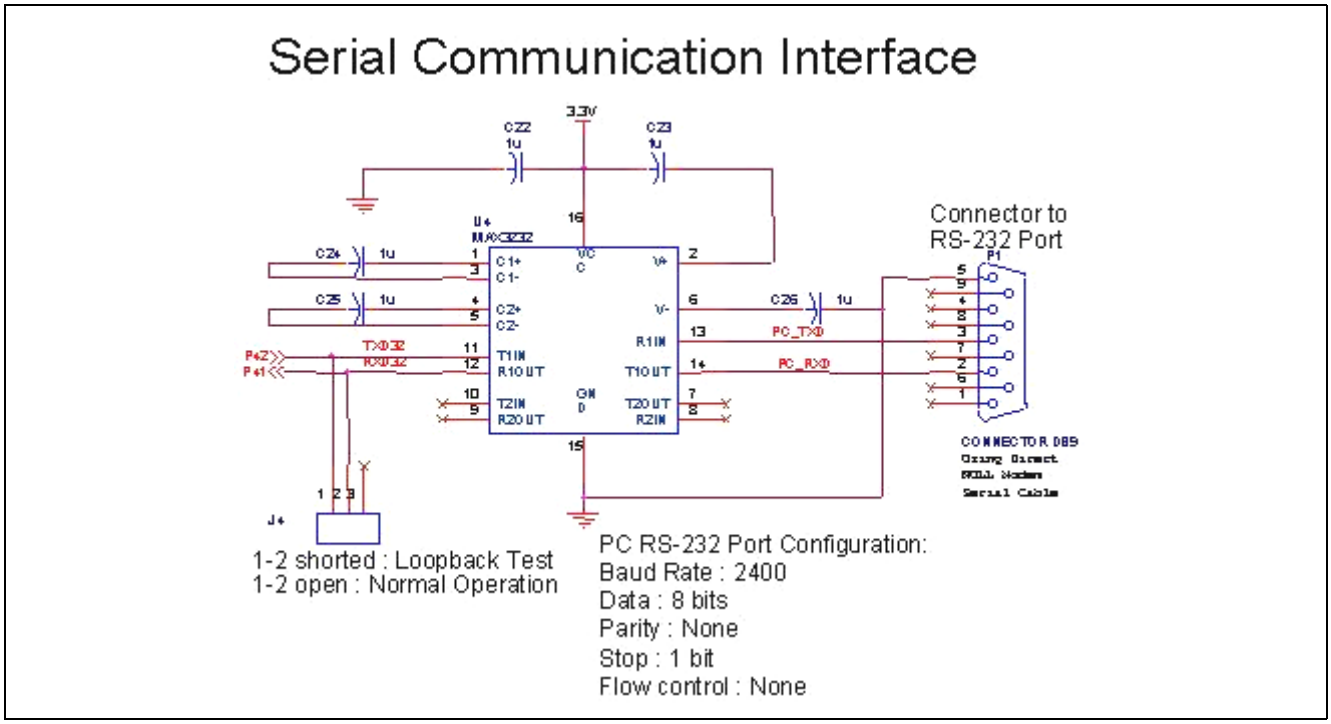
出力メッセージの図を以下に示します。



物理的な設定は次のとおりです。



アプリケーションボードには、MCU と PC をインタフェースするために RS232 ドライバが必要です。



3.2 LCD に出力するための printf

3.2.1 プログラムの説明

SLP アプリケーションボードの LCD を用いる例を説明します。

main ルーチンが汎用 I/O と LCD を初期化した後、標準ライブラリの printf 関数を呼び出すことができます。このとき、charput 関数が LCD（1 行 x 7 文字）に出力します。グローバル変数 *position* が LCD に文字を表示する位置を決めます。

```
#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);
 }
```

プログラム例は 4 つの主な関数を用います。

1. void init_io(void); //汎用初期化ルーチン（3.1 章の関数と同一）

2. void init_lcd(void) //LCD の初期化

```
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;
}
```

3. void Display_number(…) // LCD に数値を表示する。LCD は 7 文字のみ表示可能。

```
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];
}
```

4. void charput(char outputchar) // display_number()を呼び出して文字を表示する。

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

5. printf_lcd.h // LCD に表示する文字を変換するためのテーブルを検索する。

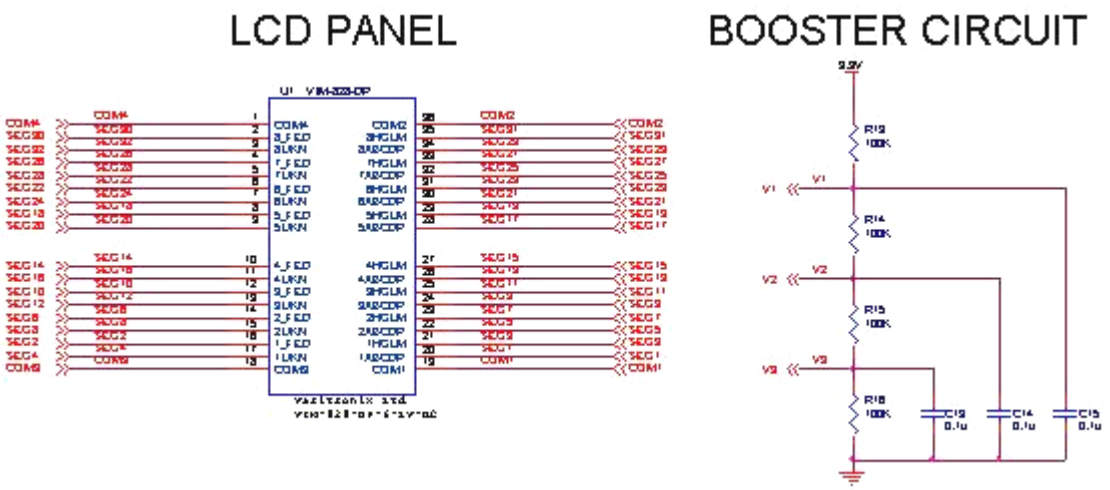
```
//for 1/4 duty cycle
const unsigned short   lcd_number_data[128]
= {                  //ASCII
                      0x0039,   //  00.   'NUL'     :Display
                      0x0039,   //  01.   'SOH'     :Display
                      0x0039,   //  02.   'STX'     :Display
                      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 ハードウェアの設定

LCD パネルを直接 SLP MCU に接続します。



アプリケーションボードの LCD 表示を示します。

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

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

4. printf()ライブラリ関数のデメリット

printf()は使いやすい反面、たいへん複雑なライブラリ関数です（関数の詳細については、H8 コンパイラユーザーズマニュアルを参照してください）。SLP MCU で用いる場合、次のようなデメリットがあります。

4.1 ROM サイズ

printf()関数は、I/O ストリーム、ヒープ、多数の引数を扱わなければならないために、ROM 領域を大量に必要とします（詳細な比較は、6 章を参照してください）。

【注】 SLP H8/38024 の ROM サイズは最大 32K バイトです。

4.2 RAM サイズ

printf()はヒープメモリを使う必要があるため、printf のためだけに H’80 バイトが割り当てられます（他の目的にヒープを用いない場合）。

【注】 SLP H8/38024 の RAM サイズは最大 1K バイトです。

5. ユーザ作成の printf 関数 – Bprintf()

上述の問題を解決するには、ユーザが関数を作成することです。本アプリケーションノートでは、基本 printf (Bprintf）関数を作成します。アプリケーションの必要に応じて、この例をさらにカスタマイズすることもできます。

2～3 章のセッションで説明した例と比べ、Bprintf 関数には以下が不要です。

i) Lowsrc.c
ii) Lowlev.src
iii) Resetprg.c での_INIT_IOLIB()および_CLOSEALL()関数呼び出し
iv) sbrk.c

言い換えると、I/O ストリームとヒープメモリが不要ということになります。これはユーザ作成関数だけの利点です。

5.1 関数の使用法

プロトタイプは以下のとおりです。

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

この関数は、普通は通常の printf 関数と同じように呼び出すことができますが、次の制限があります。

i) 表示文字数（20）
ii) 引数は 2 つのみ
iii) 第 1 引数は BYTE で、第 2 引数は DWORD
iv) サポートする引数の書式は%x、%c、%u

使用例：

i) Bprintf(“Hello world”,0,0);
ii) Bprntf(“\n\rGet data = %x from address %x”, (BYTE)data, (DWORD)address);

5.2 関数の説明

ライブラリの printf()と異なり、Bprintf()は printf 関数の簡略バージョンです。以下の制限があります。

i) MAXCHARS ：文字列のサイズを指定します。使用するスタックの深さを決める主な構成要素でもあります。
ii) case 条件を 3 つ持つ文 ：引数の書式を%x、%c、%u の 3 つに制限します。
iii) 変数 num ：引数の数を 2 つに制限します。

Bprintf()関数は charput 関数を呼び出してメッセージを出力先に送出します。

```
#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);
}
```

b_itoab()関数は整数を ASCII に変換します。整数は 16 進または 10 進に限られます。

```
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);
    }
}
```

Bprintf()関数は、さらに以下のようにカスタマイズできます。

i) 文字列のサイズを増加する
ii) 引数の他の書式を追加する：%f、%3d、%o など
iii) 引数の数を増加する
iv) 引数のサイズを拡大／縮小する（BYTE, WORD, DWORD）

6. 比較

プログラムサイズの比較をするために、以下のプロジェクトのマップファイルを生成します（Option/ Link Library/List を用います）。比較を簡単にするために、セクション P のサイズを用いて関数のサイズを評価します。最適化では、debug と release の両方の設定を用います。

	Debug	Release
printf LCD（<no_float.h>をインクルードしない）	20.4K バイト	14.9K バイト
printf LCD（<no_float.h>をインクルードする）	7.4K バイト	5.6K バイト
Bprintf LCD.	1.1K バイト	0.8K バイト

浮動小数点フォーマッタを用いた場合の printf()のプログラムサイズは、約 20.4K バイトと算出されます。<no_float.h> をインクルードした場合は、約 7.4K バイトに削減されます。しかし、ユーザが関数を作成した場合は、プログラムサイズはさらに 1.1K バイトと 7 分の 1 に削減されます。SLP の最大の ROM サイズは 32K バイトしかないことに、再度注目してください。

RAM サイズは、ユーザ作成の Bprintf()関数では約 30 バイトのスタックを使用します（MAXCHAR=20 のとき）。これに対して、ライブラリの printf()関数は、128（H’80）バイトのヒープメモリを使用します。

7. まとめ

SLP の ROM サイズは小さく、ほとんどのターゲットアプリケーションでは複雑な I/O ストリームを必要としないため、ライブラリの printf()関数は効率的ではありません。ユーザ側で printf()関数をカスタマイズすることを推奨します。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2003.09.19	—	初版発行

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ(http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。