MSP430程序库<四>printf 和 scanf 函数移植

printf 和 scanf 函数是 C 语言中最常用的输入出函数，从学习 C 语言开始，就开始使用这两个函数，然而当写用 C 语言写单片机程序时却不能使用这两个函数，总觉得单片机的 C 语言和一般的 C 语言差别很大，写起来不大方便；其实，单片机的 C 语言也是标准 C 语言上扩展或是改动的，都支持格式化输入输出函数（printf 和 scanf）；事实上，printf，scanf 只负责格式化输入输出的字符，至于从哪儿输入，输出到哪儿，他们分别依靠 getchar 和 putchar 函数，只要实现单片机上的 getchar 函数和 putchar 函数，即可正常使用 printf 函数和 scanf 函数，这可以给我们单片机的信息交互带来很多方便。下面我们就来实现他们的移置。

1.硬件介绍：

硬件部分只需字符型输入输出设备：scanf 从输入字符型设备读取字符，printf 输出到字符型输出设备。在这里，我选用的字符型输入设备是超级终端，通过串口与单片机连接，输入字符；输出设备是超级终端和12864的液晶。scanf 从串口读入字符，printf 输出字符到串口和液晶。

有关串口的预提信息参考：MSP430程序库<二>UART 异步串口。

有关液晶的具体信息参考：MSP430程序库<三>12864液晶程序库。

scanf 还可以从按键读取信息，可以参考移置方法自行移置。

2.程序实现：

- printf

单片机在调用 printf 时，printf 是负责将数据解析成 ASCII 码流，通过调用 putchar 函数依次将字符发出。如果在 putchar 内编写从串口发送一字节数据，则 printf 的结果将从单片机串口发送出；如果 putchar 是向液晶写字符，让液晶显示一个字符，则 printf 的结果将显示在液晶上。本程序实现 putchar 同时向串口和液晶同时发送一个字符（液晶是显示一个字符）。

putchar 函数如下：

```
intputchar(intch)
{
    putchar2Com(ch);
    putchar2Lcd(ch);
    return(ch);
}
```

程序先向串口发送一个字符，然后像向晶发送字符。

其中：putchar2Com，向串口发送一个字符，代码如下：

```
intputchar2Com(intch)
{
 if(ch == '\n')              //  '\n'(回车)扩展成 '\n''\r' (回车+换行)
   {
     UartWriteChar('\r') ;    //0x0d 换行
   }
 UartWriteChar(ch);          //从串口发出数据
```

```
    return(ch);
}
```

代码仅仅调用向串口写字符的函数 UartWriteChar(ch)（详见 Uart.c，在<二>中有介绍），当要输出换行时，需先输出'\n'将光标移至本行首位置，还需要'\r'（换行）才能将光标置于下一行起始位置，即将'\n'扩展为'\r','\n'两个字节依次发出。

purchar2Lcd 函数比较复杂，因为我所使用的12864液晶是中文字库的液晶，每行8个地址，可以显示8个中文字符或16个英文字符，而 putchar 只发出一个字节，需要判断每个地址的前半字还是后半字（因为每个字可以显示中文，如果中文的两个字节在相邻的两个地址上，将不会显示，或是显示乱码）。

上代码：

```
intputchar2Lcd(intch)
{
    charaddr,dat;

 if(ch == '\n')              //   '\n'(回车),换行
   {
       ChangeNextRow();
    }
 else
   {
       addr = LcdReadAddr();
       if(ch < 0x80)
       {
           LcdWriteData(ch);
       }
     else
       {
         LcdWriteData(0x20);       //写入一个空字符，根据地址判断是否为前半字
         if(addr == LcdReadAddr())    //前半字  从新写入 ch 字符
           {
               LcdWriteComm(addr);
               LcdWriteData(ch);
            }
          else
           {
               LcdWriteComm(addr);
               dat = LcdReadData();
             if(dat < 0x80)             //前一个字符是英文字符
              {
              LcdWriteData(0x20);                    //空格
              }
               LcdWriteData(ch);
            }
```

```
        }
    }
    if((addr != LcdReadAddr()) &&                    //写入的是行最后位的后半字则换行
       (addr==0x87 || addr==0x97 || addr==0x8F || addr==0x9F))
    {
        ChangeNextRow();
    }
    return(ch);
}
```

这个函数首先判断换行；然后处理其他一般字符，如果是英文字符，不用考虑前后半字，只需正常写入液晶即可；如果是中文字符，在判断是否是前半字，前半字则直接写入，后半字则判断之前写入的前半字是否是中文，是则直接写入，不是则把英文字符移入后半字，然后写入；最后判断是否到行尾，是则换行。

```
程序更新为：更新日期：20110821 18:51
目的是修复原来，行尾前半字为英文，再输入中文会显示乱码。
intputchar2Lcd(intch)
{
    charaddr,dat;
    charchangeRowFlag = 0;

    if(ch == '\n')           //   '\n'(回车),换行
    {
        ChangeNextRow();
        changeRowFlag = 1;
    }
    else if(ch == '\b')     // '\b' (退格)
    {
        BackSpace();
    }
    else
    {
        addr = LcdReadAddr();
        if(ch < 0x80)
        {
            LcdWriteData(ch);
        }
      else
       {
         LcdWriteData(0x20);       //写入一个空字符，根据地址判断是否为前半字
        if(addr == LcdReadAddr())    //前半字  从新写入 ch 字符
           {
                LcdWriteComm(addr);
                LcdWriteData(ch);
```

```
            }
          else
            {
                LcdWriteComm(addr);
                dat = LcdReadData();
              if(dat < 0x80)            //前一个字符是英文字符
                {
                  LcdWriteData(0x20);                  //空格
                }
              if((addr != LcdReadAddr()) &&                  //写入的是行最后位
的后半字则换行
                  (addr==0x87 || addr==0x97 || addr==0x8F ||
addr==0x9F))
                {
                    ChangeNextRow();
                    changeRowFlag = 1;
                }
                LcdWriteData(ch);
            }
        }
    }
   if((addr != LcdReadAddr()) &&    //写入的是行最后位的后半字则换行,且未换过
行
      (changeRowFlag == 0) &&
       (addr==0x87 || addr==0x97 || addr==0x8F || addr==0x9F))
    {
        ChangeNextRow();
    }
    return(ch);
}
```

前后半字判断方法如下：读液晶地址，向液晶写入一个空格，再读地址，两地址相同则是前半字，不同则是后半字。读地址函数在 Lcd12864.c 中，新加入函数，代码如下：

```
charLcdReadAddr()
{
    charch;

    WaitForEnable();

    CLR_RS;
    SET_RW;

    DATA_DIR_IN;

    SET_EN;
```

```
    _NOP();

 ch = DATA_IN;     //读数据
   CLR_EN;
    DATA_DIR_OUT;

    return(ch|0x80);
}
```

这个是读地址，ch|0x80是因为写入液晶地址首位应为1.。

液晶中新加入两个函数，一个是上边的读地址，另外一个是读数据；作用是读取液晶当前地址处的数据，从而判断之前半字是否是中文。代码如下：

```
charLcdReadData()
{
    charch;

    WaitForEnable();

    SET_RS;
    SET_RW;

    DATA_DIR_IN;

    SET_EN;
    _NOP();

 ch = DATA_IN;     //读数据
   CLR_EN;
    DATA_DIR_OUT;

    returnch;
}
```

另外 putchar 还调用了换行——ChangeNextRow 函数，完成液晶输出换至下一行。

代码如下：

```
voidChangeNextRow()
{
    charaddr;

 addr = LcdReadAddr();        //当前地址
   if(addr <= 0x88)
    {
        LcdWriteComm(0x90);
    }
    else if(addr <= 0x90)
```

```
    {
        LcdWriteComm(0x98);
    }
    else if(addr <= 0x98)
    {
        LcdWriteComm(0x88);
    }
  else
    {
      AddNewline();             //添加行，同时向上滚动
        LcdWriteComm(0x98);
    }
}
```

读取当前地址，判断在哪一行，然后写入下一行首地址；如果是最后一行，则所有安徽那个向上移，写入最后一行首地址。

AddNewLine 函数完成所有行向上滚动一行，然后地址定位至最后一行。

代码如下：

```
voidAddNewline()
{
    charstr[17];
    str[16] = 0;

  //第二行 移至第一行
    LcdWriteComm(0x90);
  LcdReadData();                //空读取
    for(inti = 0;i<16;i++)
    {
        str[i] = LcdReadData();
    }
    LcdWriteString(0x80,str);

  //第三行 移至第二行
    LcdWriteComm(0x88);
     LcdReadData();
     for(inti = 0;i<16;i++)
    {
        str[i] = LcdReadData();
    }
    LcdWriteString(0x90,str);

  //第四行 移至第三行
    LcdWriteComm(0x98);
     LcdReadData();
```

```
    for(inti = 0;i<16;i++)
    {
        str[i] = LcdReadData();
    }
    LcdWriteString(0x88,str);

  //第四行 空白
  LcdWriteString(0x98,"                ");     //十六个空格
  }
```

读出下一行数据，写入上一行，最后一行写入空格即可。

到此 putchar 函数全部完成，printf 移植的程序部分完成，使用方法详见使用示例。

- scanf

scanf 和 printf 类似，其只负责格式化输入的字符，字符来源是从 getchar 函数获取；同样，在使用 scanf 函数之前，要针对字符输入源自行编写 getchar 函数

最简 getchar：

```
intgetchar()
{
    return(putchar(UartReadChar()));
}
```

这是最简单的 getchar 函数，直接调用读取字符函数，输出并返回。

但是人的输入过程会偶尔犯错误的，为了支持退格键等，需要开辟一个缓存区。

详细代码如下：

```
#defineLINE_LENGTH 80           //行缓冲区大小，决定每行最多输入的字符数

/*标准终端设备中，特殊 ASCII 码定义，请勿修改*/
#defineInBACKSP 0x08            //ASCII  <--  (退格键)
#defineInDELETE 0x7F            //ASCII <DEL> (DEL 键)
#defineInEOL '\r'              //ASCII <CR>  (回车键)
#defineInSKIP '\3'             //ASCII control-C
#defineInEOF '\x1A'            //ASCII control-Z

#defineOutDELETE "\x8 \x8"     //VT100 backspace and clear
#defineOutSKIP "^C\n"          //^C and new line
#defineOutEOF "^Z"             //^Z and return EOF

intgetchar()
{
  static charinBuffer[LINE_LENGTH + 2];       //Where to put chars
  static charptr;                            //Pointer in buffer
    charc;
```

```
while(1)
{
 if(inBuffer[ptr])                          //如果缓冲区有字符
    return(inBuffer[ptr++]);                //则逐个返回字符
ptr = 0;                                    //直到发送完毕，缓冲区指针归零
while(1)                                    //缓冲区没有字符，则等待字符输入
   {
     c = UartReadChar();                      //等待接收一个字符
    if(c == InEOF && !ptr)                   //==EOF==  Ctrl+Z
    {                                        //只有在未入其他字符时才有效
        printf(OutEOF);                      //终端显示 EOF 符
        returnEOF;                           //返回 EOF（-1）
       }
     if(c==InDELETE || c==InBACKSP)       //==退格或删除键==
       {
         if(ptr)                             //缓冲区有值
           {
             ptr--;                          //从缓冲区移除一个字符
            printf(OutDELETE);               //同时显示也删掉一个字符
           }
        }
     else if(c == InSKIP)                    //==取消键 Ctrl+C ==
       {
        printf(OutSKIP);                     //终端显示跳至下一行
       ptr = LINE_LENGTH + 1;                 //==0 结束符==
          break;
       }
     else if(c == InEOL)                     //== '\r' 回车==
       {
        putchar(inBuffer[ptr++] = '\n');//终端换行
       inBuffer[ptr] = 0;                    //末尾添加结束符（NULL）
       ptr = 0;                              //指针清空
          break;
       }
     else if(ptr < LINE_LENGTH)              //== 正常字符 ==
       {
         if(c >= ' ')                        //删除 0x20以下字符
           {
             //存入缓冲区
               putchar(inBuffer[ptr++] = c);
            }
        }
     else                                    //缓冲区已满
       {
```

```
                putchar('\7');                    //== 0x07 蜂鸣符，PC 回响一声
            }
        }
    }
}
```

注释已经很详细了，这里不再详细解释。

scanf 的移植程序部分已经完成，如果需要从键盘读入字符，可以仿照上述函数写 getchar 函数。具体使用和设置见使用示例。

另外，iar 的安装文件夹下，430文件夹下有一个 src 文件夹，lib/clib 文件夹下（我的具体文件夹是：D:\Program Files\IAR Systems\Embedded Workbench 6.0 Evaluation\430\src\lib\clib\getchar.c），有一个 getchar.c 文件，这是 getchar 的函数，内容如下：

```
#include"stdio.h"

extern char_low_level_get(void);        /* Read one char from I/O */
                                        /* Should be supplied by user */

static voidput_message(char*s)
{
  while(*s)
    putchar(*s++);
}


#defineLINE_LENGTH 80            /* Change if you need */

#defineIn_DELETE 0x7F            /* ASCII <DEL> */
#defineIn_EOL '\r'              /* ASCII <CR> */
#defineIn_SKIP '\3'             /* ASCII control-C */
#defineIn_EOF '\x1A'             /* ASCII control-Z */

#defineOut_DELETE "\x8 \x8"    /* VT100 backspace and clear */
#defineOut_SKIP "^C\n"          /* ^C and new line */
#defineOut_EOF "^Z"             /* ^Z and return EOF */

intgetchar(void)
{
 static chario_buffer[LINE_LENGTH + 2];       /* Where to put chars */
static intptr;                                /* Pointer in buffer */
 charc;

  for(;;)
  {
```

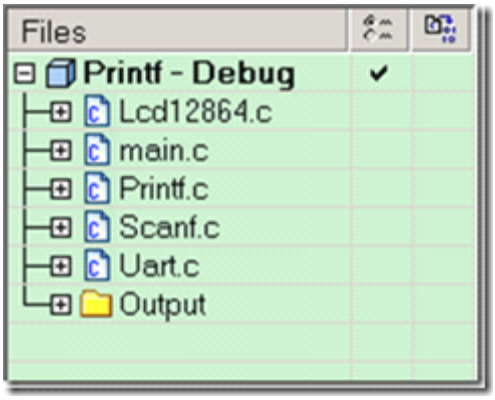
```
  if(io_buffer[ptr])
    return(io_buffer[ptr++]);
  ptr = 0;
  for(;;)
  {
    if((c = _low_level_get()) == In_EOF && !ptr)
    {
      put_message(Out_EOF);
      returnEOF;
    }
    if(c == In_DELETE)
    {
      if(ptr)
      {
        ptr--;
        put_message(Out_DELETE);
      }
    }
    else if(c == In_SKIP)
    {
      put_message(Out_SKIP);
   ptr = LINE_LENGTH + 1;   /* Where there always is a zero... */
     break;
    }
    else if(c == In_EOL)
    {
      putchar(io_buffer[ptr++] = '\n');
      io_buffer[ptr] = 0;
      ptr = 0;
      break;
    }
    else if(ptr < LINE_LENGTH)
    {
      if(c >= ' ')
      {
        putchar(io_buffer[ptr++] = c);
      }
    }
  else
    {
      putchar('\7');
    }
  }
}
```

```
}
```

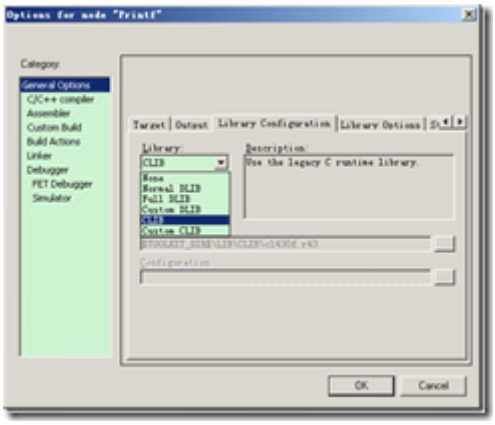
_low_level_get(void)；这个函数需用户定义，不过这个 getchar 函数不支持退格键，可以更改以支持；_low_level_get(void);这个函数可以直接调用 UartReadChar();这个函数，使用时，把 getchar.c 加入项目，同时在项目中添加_low_level_get(void);函数，函数体只有一句：return UartReadChar();即可。

3.程序调用示例：

程序使用方式，项目中添加 printf.c 文件和 scanf.c 文件（用 printf 函数则加 printf.c 文件，用 scanf 函数就添加 scanf.c 文件），在要使用函数的地方包含 stdio.h（编译器自带库——标准输入输出库）



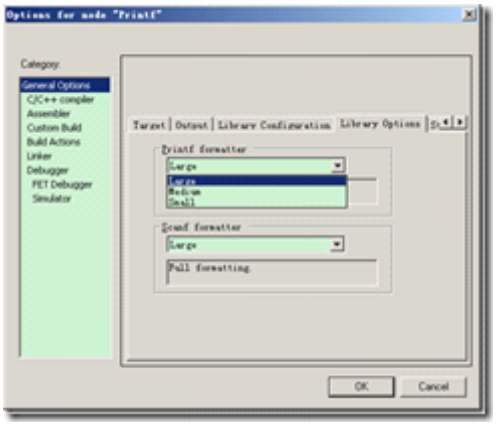
还要设置使用库和 printf 的大小：



如果不进行这项设置，使用 scanf 时将报错：

Error[e27]: Entry "getchar" in module Scanf (G:\work\程序库\Printf\Debug\Obj\Scanf.r43) redefined in module ?getchar (D:\Program Files\IAR Systems\Embedded；用的是 C 语言，这里选择 CLIB。

然后设置库选项：



这里选择大尺寸，目的是支持所有的格式，因为所用单片机有64kb 的程序存储空间，足够使用，如果程序存储空间不够大，推荐选择中尺寸或小尺寸。大尺寸 printf 占用空间 4.8kb、scanf ：2.3kb，中尺寸 printf：2.5kb、scanf：1.6kb，小尺寸 printf：1.6kb。实际使用时根据需要进行选择。

同时要加入 Lcd12864的使用（c 文件，h 文件（要调用 lcd12864的初始化函数））Uart 和液晶一样要调用初始化函数。

```
#include<msp430x16x.h>
#include<stdio.h>
#include"Uart.h"
#include"Lcd12864.h"
```

头文件包含。

```
voidmain( void)
{
 // Stop watchdog timer to prevent time out reset
   WDTCTL = WDTPW + WDTHOLD;
    ClkInit();
    LcdInit();
 UartInit(38400,'n',8,1);//串口初始化,设置成38400bps,无校验,8位数据,1位停止
 //int a;
   _EINT();
 //scanf("%d",&a);
 //printf("刘中原%d\n",a);
   printf("刘中原%f\n",23.6);
    printf("刘中原%1.2f\n",23.6);
}
```

使用时，先调用液晶和串口的初始化函数，然后开中断；就可以正常的调用 scanf 和 printf 函数了。

至此，printf 和 scanf 的移植全部完成，使用这两个函数将给单片机的输入输出带来极大方便。另外，Lcd12864的液晶使用是4行显示，空间较小，可能需要定位至具体位置，以使界面看起来更合理，为此，在 Printf 中再添加一个定位函数（GotoXY）：

```
voidGotoXY(charx,chary)
```

```
{
    charaddr;

    if(y==0)
    {
        addr = 0x80 + x / 2;
    }
    else if(y==1)
    {
        addr = 0x90 + x / 2;
    }
    else if(y==2)
    {
        addr = 0x88 + x / 2;
    }
 else
   {
        addr = 0x98 + x / 2;
    }
    LcdWriteComm(addr);
 if(x % 2)                    //是奇数，后移一位（写入空格）
   {
        LcdWriteData(0x20);
    }
}
```

这样就方便了液晶程序的编写。

又加入一个函数，在 printf.c 里，目的是支持退格键，内容如下：

```
voidBackSpace()
{
    charaddr,dat;

 addr = LcdReadAddr();        //当前地址
LcdWriteData(0x20);          //写入一个空字符，根据地址判断是否为前半字
if(addr == LcdReadAddr())    //前半字
   {
        if(addr == 0x80)
            return;
        else if(addr == 0x90)
            addr = 0x87;
        else if(addr == 0x88)
            addr = 0x97;
        else if(addr == 0x98)
            addr = 0x8F;
     else
```

```
            addr = addr - 1;

          LcdWriteComm(addr);
        LcdReadData();           //空读取
          dat = LcdReadData();
          LcdWriteComm(addr);
          if(dat < 0x80)
              LcdWriteData(dat);
      }
   else
     {
          LcdWriteComm(addr);
      }
}
```

退格完成功能：仅仅地址向前退一格，详细见源程序。

printf 和 scanf 移植全部完成，欢迎大家使用；

相关文章及附件下载：http://www.ideyi.org/bbs/article_1077_369541.html