如何在单片机中处理浮点数

Jacky　广西桂林

在设计单片机系统的时候经常会遇到需要显示浮点数的情况，本文将对浮点数的显示方法做一个总结。需要说明的是，本文所有的程序均是在 C 语言的环境下运行的，而且都是使用编译器提供的函数来处理的，不同的编译器会有不同的函数。在汇编的条件下，我目前还没想出好的解决方法。以前有一种说法，就是涉及到浮点运算的时候最好是用汇编，我觉得现在似乎已经过时了，因为在高速单片机及强大的 C 函数的支持下，浮点运算的速度已经没有多大的影响了（当然还是要看具体的使用场合，对于实时控制，你只能老老实实的用汇编），相反使用 C 可以极大的提高代码开发速度（以前用 ASM51 写了一个浮点数转换成 ASCII 码的程序，差点吐血了）。

首先介绍一下浮点数在单片机中的存储方法，有了 C，其实你基本无需考虑浮点数的存储问题，因为编译器已经为你做好了大部分的工作。但在这里还是有必要介绍一下，这里引用了 www.mcu51.com 上的一篇文章，作者不详。

float 类型标量用四个字节保存格式用下面的 IEEE-754 标准，一个浮点数用两个部分表示尾数和 2 的幂,尾数代表浮点上的数据二进制数,二的幂代表指数指数的保存形式是一个 0 到 255 的 8 位值.指数的实际值是保存值(0 到 255)减去 127,一个范围在 127 到-128 之间的值.尾数是一个 24 位值(代表大约 7 个十进制数),最高位 MSB 通常是 1,因此不保存.一个符号位表示浮点数是正或负.

浮点数保存的字节格式如下

地址　　+0　　+1　　+2　　+3

内容　　SEEE EEEE　　EMMM MMMM　　MMMM MMMM　　MMMM MMMM

这里

S 代表符号位 1 是负 0 是正

E 幂偏移 127

M 24 位的尾数保存在 23 位中

零是一个特定值表示幂是 0 尾数是 0

浮点数-12.5 作为一个十六进制数 0xC1480000 保存.在存储区中,这个值如下:

地址 +0　　+1　　+2　　+3

内容 0xC1　0x48　0x00　0x00

浮点数和十六进制等效保存值之间的转换相当简单. 下面的例子说明上面的值-12.5 如何转换. 浮点保存值不是一个直接的格式. 要转换为一个浮点数, 位必须按上面的浮点数保存格式表所列的那样分开, 例如:

地址	+0	+1	+2	+3
格式	SEEE EEEE	EMMM MMMM	MMMM MMMM	MMMM MMMM
二进制	11000001	01001000	00000000	00000000
十六进制	00	00	48	C1

从这个例子可以得到下面的信息：符号位是 1 表示一个负数，幂是二进制 10000010 或十进制 130 130 减去 127 是 3 就是实际的幂，尾数是后面的二进制数

10010000000000000000000

在尾数的左边有一个省略的二进制点和 1, 这个数在浮点数的保存中经常省略, 加上一个 1 和点到尾数的开头, 尾数值如下:

1. 10010000000000000000000

接着, 根据指数调整尾数. 一个负的指数向左移动小数点. 一个正的指数向右移动小数点. 因为指数是三, 尾数调整如下:

1100. 10000000000000000000

结果是一个二进制浮点数. 小数点左边的二进制数代表所处位置的二的幂.

小数点的右边也代表所处位置的二的幂, 但是, 幂是负的.

这些值的和是 12.5 因为有设置符号位这数是负的因此十六进制值 0xC1480000

是-12.5

好了，介绍完了。上面的介绍看起来有点复杂，花点时间啃一下吧，相信你会看懂的，如果看不懂也不要紧（前提是你用的是 C），我现在也记不起那些细节了，只知道 float 型数据是用 4 个字节来保存的。

一、 如何把浮点数转换成 ASCII 码

1、GCC 中如何实现

下面进入正题，首先介绍在 GCC 中如何显示浮点数，要想显示浮点数必须得把它转换

成 ASCII 码或者 BCD 码。编译器使用 ATmanAVR4.5，该编译器是基于 GCC 的，有着非常漂亮的 IDE 界面，所以在 winavr 中应该也能通过。芯片使用 ATmega16L，8M 外部晶振，串口调试软件 SSCOM3.2。熔丝配置如下：

GCC 提供了一个非常好用的函数－dtostrf，其原型为

```
char *dtostrf(double val, char width, char prec, char *s);
```

val －要转换的数值 ，也可以为 float 类型

width －输出字符串最小宽度

prec －精度，即小数点后的位数

s －转换结果

使用时需包含#include <stdlib.h>文件。该函数返回字符串指针，作用是把 double 型数值 *val* 转换为 ASCII 字符串存储到 *s* 。 调用者要保证 *s* 具有足够的空间。下面给出具体的程序（压缩文件中包含了工程文件）。AtmanAvr4.5 调试通过。

```
#include "float.h"
```

```
#include "floatUART.h"
#include <delay.h>
#include <stdlib.h>
#include <math.h>
#include <stdio.h>

static void io_init(void)
{
    wdt_disable();
    ACSR = 0x80;
 }
// 版本 1，不使用 dtostrf 的返回值，直接使用 S 指针
int main(void)
{
    io_init();
    uart_init();
    while(1)
    {
        char s[7];
        unsigned char i;
        float x = 123.456;
        dtostrf( x,7,3,s);

        for( i=0;i<7;i++ )    //   把数组中的数据通过串口发出
        {
            UDR = s[i];
            while(!(UCSRA&0X40));
            UCSRA |= 0x40;
        }
        delay( 2000,8000);
    }
}

/*
    版本 2，使用 dtostrf 的返回值

    while(1)
    {
        double x = 123.456;
        char s[7];
        char *result;
        unsigned char i;

        result = dtostrf( x,7,3,s);
```

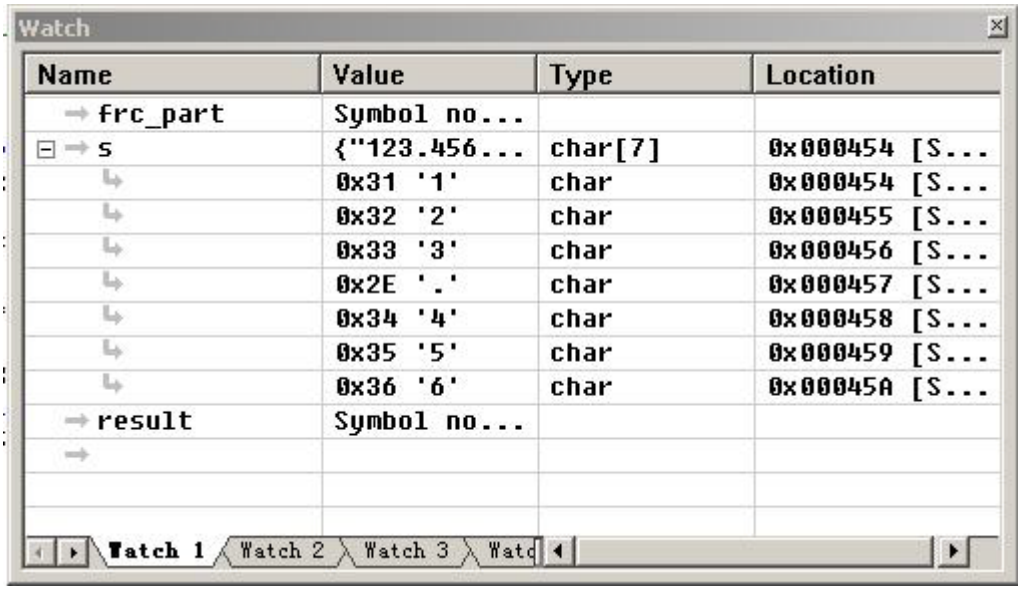
```
            for( i=0;i<7;i++ )      //   把数组中的数据通过串口发出
            {
                UDR = result[i];
                while(!(UCSRA&0X40));
                UCSRA |= 0x40;
            }
            delay( 2000,8000);
        }
    */
```

说明：float x = 123.456;

dtostrf(x,7,3,s); x 也可以定义成 double 型。For 循环用来把数据从串口发出，便于观察。在版本 2 中使用了 dtostrf 函数的返回值。

程序编译后所用的 code 为 2156 byte，还可以接受。

运行后结果如下，注意看数组 S 中各元素的值：



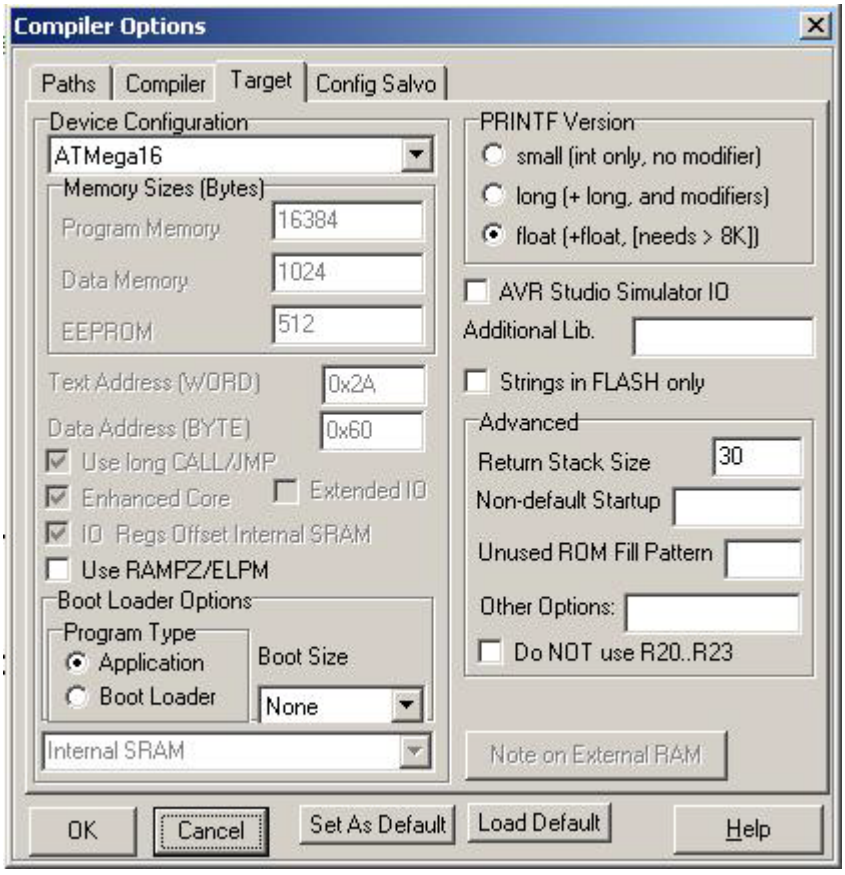
2、ICC 中如何实现

ICC 也是一个很常用的 AVR 编译器，可惜在 ICC 中没有 GCC 中的×××××函数，不过，ICC 提供了另外一个函数，就是 sprintf 函数，这是一个标准的 C 函数（ATmanAVR 似乎没有提供这个函数，反正我没找到）。有一点需要说明，在 ICC 中使用 sprintf 做浮点数转换必须要 8K 以上的 code 空间（也就是说 M8 无法使用 sprintf 函数），在 ICC 中设置如下：

在“Project－>Options－>Target－>PRINTF Version”选项中选中“float”。

在“Project－>Options－>Target－>Advanced－>Return Stack Size” 选项中填 30 以上。如下

图：



关于 sprintf 函数我就不罗嗦了，找一本标准 C 的参考数，看一下 printf 的用法就可以了，sprintf 与 printf 的用法完全相同，区别在于 sprintf 的结果是输出到由自己定义的一个缓冲区，printf 是输出到串口。ICC 中程序如下：

```
#include "FLOAT.H"
#include "MATH.H"
#include "iom16.h"
#include "stdio.h"
#include "macros.h"                    //  常用的宏定义
#define uchar unsigned char
#define uint unsigned int


//----------------------  延时函数  START  -------------------------
/*****************************************
延时 n 毫秒,不是很精确,估计的
晶振 8M
作者：蒋剑东  广西桂林
```

```
修改时间：2004-09-03
*****************************************/
void delay_nms(uint count)
{
    uint i,j ;
    for(j=1;j<=count;j++)
    {
        for(i=0;i<830;i++)
        {
            asm("wdr");
        }
    }
}

//-----------------------   延时函数   END     -------------------------


void cpu_init(void)
{
    CLI();
    //    串行口初始化     START
    UBRR=0x33 ;      //     波特率 9600
    UCR=0x18 ;

}

void main(void)
{
    float pi=3.141592;
    uchar buf[8];
    uchar i;
    cpu_init();
    puts("Hellow Word!");
    while(1)
    {
        sprintf(buf,"%f\r\n",pi ) ;
        for(i=0;i<8;i++)
        {
            UDR = buf[i];
            while(!(USR&0X40));
            USR |= 0x40;
        }
        delay_nms(1000);
    }
```

}

说明：在程序中定义了一个数组 buf 作为 sprintf 函数的输出缓冲区，当程序执行完毕之后 buf 数组中的值变成[0x31,0x2e,0x31, 0x34, 0x31, 0x35, 0x39, 0x32]，这就是 3.141592 的 ASCII 码值，为便于观察，我把它发到了串口上。

使用 C 就这么简单，几句话就搞掂了。先别急，这段代码经过编译之后竟然占了 M16 的 62%（9.92K），极大的影响了 sprintf 函数的实用性，而同样的程序在 keil 中只用了 2568 字节。说明 ICC 在 sprintf 函数上表现不尽如人意。

不过 ICC 提供了另外一个函数，这就是 modf 函数，ICC 的帮助文档说明如下：

float modf(float x, float *pint)

returns a fraction f and stores an integer into *pint that represents x. f + (*pint) equal x. abs(f) is in the interval [0, 1] and both f and *pint have the same sign as x.

这个函数的作用是把一个 float 型数据分解成整数和小数部分，当然这个函数不能直接用于浮点数的转换，为此我以 modf 为基础编写了另外一个函数 FloatToAscii，函数原型如下：

因这段时间太忙，

3、keil 中如何实现