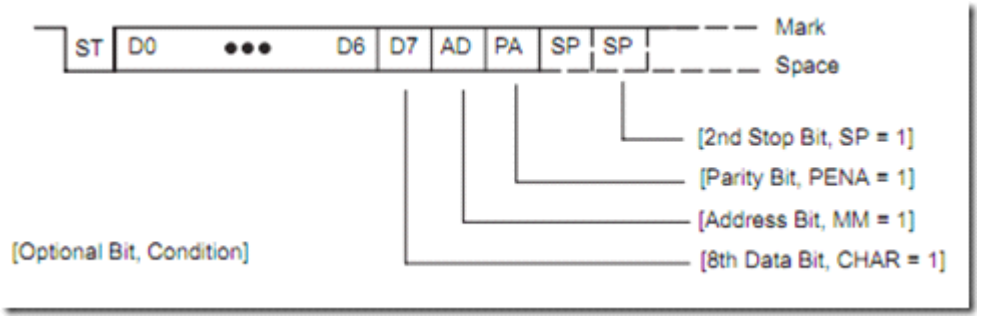
MSP430程序库<二>UART 异步串口

串行通信接口是处理器与其他设备进行数据通信最常用的方式之一。我的这个程序库是针对MSP430f14系列和 MSP430f16系列的，我常用的单片机是这两款：msp430f149，msp430f169。这两款单片机中均有两个增强型串行通信接口，都可以进行同步或是异步通信，甚至169的模块 USART0还能进行进行 I2C 协议通信。在这里，我们只讨论异步串行通信。

1.硬件介绍：

MSP 单片机的 USART 模块可以配置成 SPI（同步通信）模式或 UART（异步通信）模式，这里只讨论 UART 方式。UART 数据传输格式如下：



起始位，数据位由高到低7/8位，地址位 0/1位，奇偶校验位 奇偶或无，停止位1/2位。数据位位数、地址位、奇偶校验位、停止位均可由单片机内部寄存器控制；这两款单片机都有两个 USART 模块，有两套独立的寄存器组；以下寄存器命中出现 x 代表0或是1,0代表对应0模块的寄存器，1代表对应1模块的寄存器；其中，与串口模式设置相关的控制位都位于 UxCTL 寄存器，与接收相关的控制位都位于 UxRCTL 寄存器，与发送相关的控制位都位于 UxTCTL 寄存器；波特率设置用 UxBR0、UxBR1、UxMCTL 三个寄存器；接收与发送有独立的缓存 UxRXBUF、UxTXBUF，并具有独立的移位寄存器和独立的中断；中断允许控制位位于 IE1/2寄存器，中断标志位位于 IFG1/2寄存器。

波特率设置：430的波特率设置用三个寄存器实现，
UxBR0：波特率发生器分频系数低8位。
UxBR1：波特率发生器分频系数高8位。
UxMCTL：波特率发生器分频系数的小数部分实现。
设置波特率时，首先要选择合适的时钟源：USART 模块可以设置的时钟源有 UCLK 引脚、ACLK、SMCLK；对于较低的波特率（9600以下），可选 ACLK 作为时钟源，这样，在 LPM3（低功耗3）模式下，串口仍能正常发送接收数据；另外，由于串口接收过程有一个三取二判决逻辑，这至少需要三个时钟周期，因此分频系数必须大于3；波特率高于9600时，将不能使用 ACLK 作为时钟源，要调为频率较高的 SMCLK 作为时钟源；另外还可以外部输入 UCLK 时钟。分频系数计算公式如下：

$$N = \frac{BRCLK}{baud\ rate}$$

小数分频是 MSP430单片机的串口特色之一，UxMCTL 寄存器的作用就是控制小数的分频，控制方法如下：对应位是1，则分频系数加一，0则分频系数减一；小数分频器会自动依次取出每一位来调整分频系数。其计算方法：可以先计算小数部分一的个数，然后把1均匀的

放入 UxMCTL 的8位中，这样计算比较简单，分频系数的小数部分乘以8即得到1的位数，查表得到对应的 UxMCTL 值；另外一种通过计算每一位的错误率，交互计算，直到得到最小错误率的 UxMCTL 值，这种方法比较复杂，但得到的小数分频误差更小，这种方法也是 TI 给的计算方法，详细参考 UserGuide。

另外，有关寄存器，以及其他单片机硬件有关知识请参考德州仪器提供的用户指南和数据手册等资料。

2.程序实现：

- 宏定义：是程序具有更好的移植性。

对模块的寄存器进行宏定义，把0/1换成 x，使用时，只需更改宏定义即可更改程序是使用哪个模块；这样程序就具有了比较好的移植性。

```
/***********************************宏定义
*********************************/
    #defineUxCTL     U0CTL
    #defineUxRCTL    U0RCTL
    #defineUXTCTL    U0TCTL

    #defineUxBR0     U0BR0
    #defineUxBR1     U0BR1
    #defineUxMCTL    U0MCTL

    #defineUxRXBUF U0RXBUF
    #defineUxTXBUF U0TXBUF

    #defineUxME      U0ME
    #defineUxIE      U0IE
    #defineUxIFG     U0IFG

    #defineUTXEx     UTXE0
    #defineURXEx     URXE0

    #defineURXIEx    URXIE0
    #defineUTXIEx    UTXIE0

#defineUARTON   P3SEL |= 0X30              // P3.4,5 = USART0 TXD/RXD
/***********************************************************************
*************/
    程序改为 UART1时，只需把宏定义中的0改为1   UARTON 改为对应端口的即可
```

- 异步串口初始化（UartInit）：完成波特率，停止位以及其他相关的设置。

串口初始化，首先是波特率寄存器值的计算和设置：本程序选用第二种：通过运算，选取误差最小的寄存器所需值进行设置。

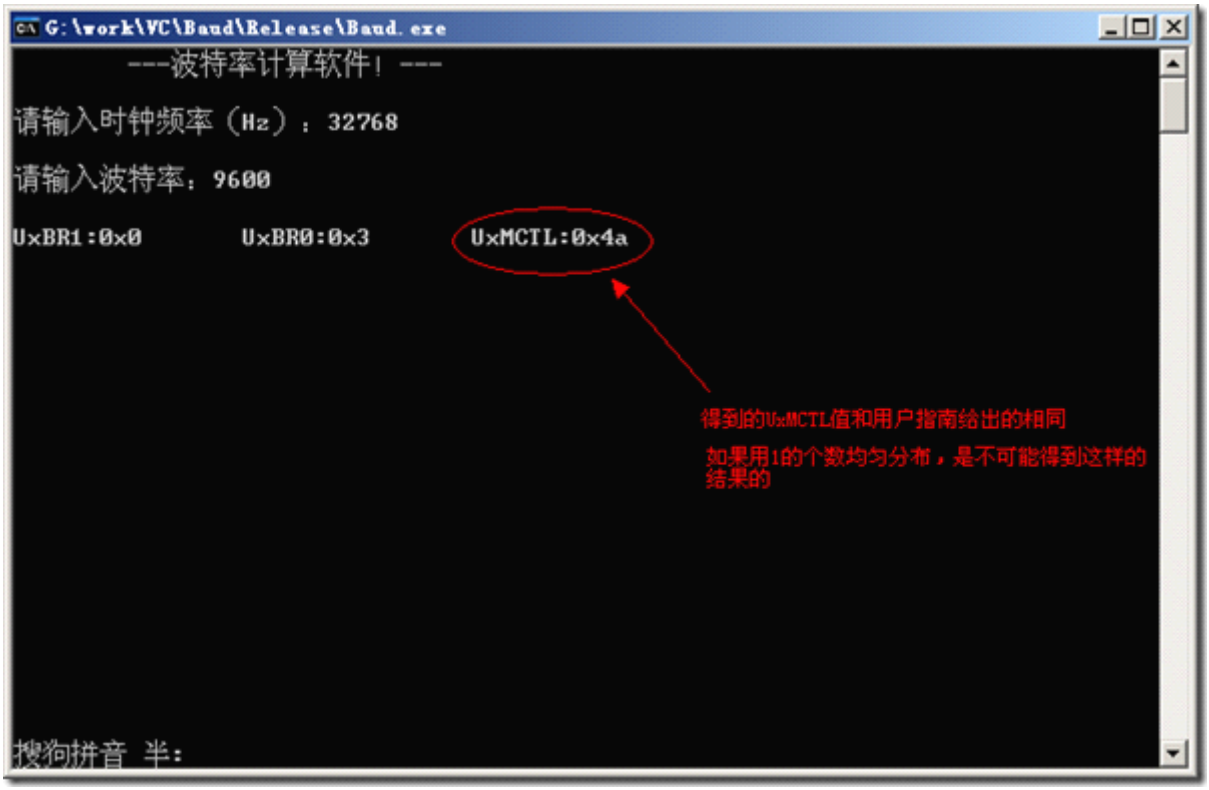
波特率寄存器值根据所选时钟频率和所需波特率值进行设置，计算方法：从 m0（UxMCTL 最低位）开始计算，根据这一位的误差（0或1时）误差较小的 bit 值，直到计算完成。

为了更好的写这个程序，我先用 C 语言写了一个简单的波特率计算软件，为了让设置波特率的函数能够在单片机程序中复用，程序用宏定义模拟的 MSP430单片机的波特率寄存器。完整程序如下：

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

//函数声明
   voidSetBaudRateRegisters(longclk,intbaud);

/************************宏定义************************/
   #defineUxBR1     a[0]
   #define    UxBR0     a[1]
   #define    UxMCTL     a[2]

unsigned chara[3];                        //数组模拟寄存器
   voidmain()
   {
    longclk;                             //时钟
   longbaud;                              //波特率
      printf("\t---波特率计算软件！---\n");
       printf("\n 请输入时钟频率（Hz）：");
       scanf("%ld",&clk);
       printf("\n 请输入波特率：");
       scanf("%ld",&baud);
    getchar();                             //读取多余回车符

   SetBaudRateRegisters(clk,baud);       //设置寄存器值

    //显示寄存器值

   printf("\nUxBR1:0x%x\tUxBR0:0x%x\tUxMCTL:0x%x\n",UxBR1,UxB
   R0,UxMCTL);

       getchar();
   }

/*****************************************************************
***************
* 名      称：SetBaudRateRegisters
* 功      能：根据时钟 波特率设置对应寄存器
* 入口参数：
*            clk:          所选时钟频率（如：32768）
```

```
            baud          波特率        (300~115200)
* 出口参数：无
* 范      例: SetBaudRateRegisters(32768,9600) //用时钟频率32768产生9600的
波特率
**********************************************************************
**************/
    voidSetBaudRateRegisters(longclk,longbaud)
    {
     intn = clk / baud;       //整数波特率
    charmSum = 0;               //Σmi
    inttxEr0;                  //对应位为0时错误率
    inttxEr1;                  //对应位为1时错误率
    chari = 0;                 //循环计数

    UxBR1 = n >> 8;            //高8位
    UxBR0 = n & 0xff;          //低8位
      UxMCTL = 0;

     //循环  比较错误率大小  设置 UxMCTL
      for(;i < 8;i++)
       {
           txEr0 = 100 * baud * ((i + 1) * n + mSum) / clk - 100 * (i + 1);
           txEr1 = 100 * baud * ((i + 1) * n + mSum + 1) / clk - 100 * (i
    + 1);
           if(abs(txEr1) < abs(txEr0))
           {
               mSum++;
               UxMCTL |= (1<<i);
           }
       }
    }
```

程序可以使用任何的 C 语言编译器编译运行，可供网友们复用此程序。我使用 vs2010编译运行的，运行结果如下：



运行效果很好，和官方给出的值一样，但是也不全都是这样，4800的波特率（时钟：32768）时就不一样，可能是我计算式只是用了发送时的误差计算，没有用接收误差，计算结果稍有出入，如果有兴趣，网友可以自行添加接收误差，判断；应该就和官方给出的数值完全一样了。

初始化函数：初始化函数完成串口时钟源选择，波特率初始化，奇偶校验，数据位，停止位，以及其他相关设置。

时钟源选择：根据波特率选取时钟源，波特率大于9600，选1M 的 SMCLK 时钟（需要初始化时钟系统对应函数参考使用示例），小于9600，选 ACLK（32768）以使功耗降低（低功耗3仍能正常收发数据）

```
UxTCTL &=~ (SSEL0+SSEL1);          //清除之前的时钟设置
if(baud<=9600)                     //brclk 为时钟源频率
   {
  UxTCTL |= SSEL0;                 //ACLK，降低功耗
 brclk = 32768;                   //波特率发生器时钟频率=ACLK(32768)
   }
else
   {
  UxTCTL |= SSEL1;                 //SMCLK，保证速度
 brclk = 1000000;                 //波特率发生器时钟频率=SMCLK(1MHz)
   }
```

波特率设置：直接调用之前实现的设置寄存器函数即可，当波特率在正常范围外时，返回0。

```
    //------------------------设置波特率-------------------------
   if(baud < 300||baud > 115200)    //波特率超出范围
      {
```

```
        return0;
    }
  SetBaudRateRegisters();          //设置波特率寄存器
```

奇偶校验、数据位位数、停止位数设置：比较简单，直接根据参数值设置对应寄存器即可。

```
//-----------------------设置校验位-----------------------
  switch(parity)
  {
   case'n':case'N': UxCTL &=~ PENA;                  break;  //无校验
  case'p':case'P': UxCTL |= PENA + PEV;            break;  //偶校验
  case'o':case'O': UxCTL |= PENA; UxCTL &=~ PEV; break;  //奇校验
  default:          return(0);                              //参数错误
  }

//-----------------------设置数据位-----------------------
  switch(dataBits)
  {
   case7:case'7': UxCTL &=~ CHAR; break;       //7位数据
  case8:case'8': UxCTL |= CHAR;  break;       //8位数据
  default:        return(0);                  //参数错误
  }
//-----------------------设置停止位-----------------------
  switch(stopBits)
  {
   case1:case'1': UxCTL &= ~SPB;  break;       //1位停止位
  case2:case'2': UxCTL |= SPB;   break;       //2位停止位
  default:        return(0);                  //参数错误
  }
```

其他：包括串口收发使能，串口接收和发送中断设置，第二功能打开等。

```
   UARTON;                        //端口使能
  UxME |= UTXEx + URXEx;       //发送 接收使能

  UCTL0 &= ~SWRST;              // Initialize USART state machine

  UxIE |= URXIEx + UTXIEx;     // Enable USART0 RX interrupt
```

到此，MSP430异步串行口的初始化工作全部完成，如果需要其他的方式，只需对应设置寄存器即可。

- 写字符（UartxWriteChar）：向 UARTx 模块写（发送）一个字符。

写字符：向串口写入一个字符，通过串口向终端发送一个字符。

```
voidUartWriteChar(charc)
{
```

```
 while(TxFlag==0) UartLpm();   // 等待上一字节发完，并休眠
TxFlag=0;                          //
 UxTXBUF=c;
}
```

这个函数根据程序标志 TxFlag 判断上一字符是否发送完成，此标志位将在发送中断中被置位，表示本字符发送完成。发送中断程序如下：

```
#pragmavector=UARTxTX_VECTOR
__interrupt voidUartTx ()
{
  TxFlag=1;
  __low_power_mode_off_on_exit();
}
```

发送字符时，先等待上一字符发送完成，然后把字符放入发送缓冲区，待发送完成，中断置标志位，指示发送完成。

- 读取字符（UartxReadChar）：从 UARTx 模块读取（获取）一个字符。

读取字符和写字符类似：调用读取函数后，等待标志位，接收到字符后，读出来。

```
charUartReadChar()
{
while(RxFlag==0) UartLpm(); // 收到一字节?
 RxFlag=0;
  return(UxRXBUF);
}
```

同样，RxFlag 指示收到一个字符，并且在中断中被置位。中断程序如下：

```
#pragmavector=UARTxRX_VECTOR
__interrupt voidUartRx()
{
  RxFlag=1;
/*在这里添加用户中断服务程序代码，如将数据压入接收缓冲等*/
 __low_power_mode_off_on_exit();
}
```

读取函数将阻塞，如果收不到字符，CPU 将一直处于低功耗状态。

- 写字符串（UartxWriteStr）：向 UARTx 模块写（发送）一个字符串。

写字符串只需调用写字符函数即可，比较简单，程序如下：

```
voidUartWriteStr(char*s)
{
    while(*s)
    {
        UartWriteChar(*s++);
    }
}
```

- 这样，即可调用这个函数通过串口发送字符串。

- 头文件：头文件把要调用的函数声明放进去，需要使用函数时只需包含此文件，不需要再进行函数声明。

头文件内容如下：

```
#ifndef__UART_H
#define__UART_H

charUartInit(longbaud,charparity,chardataBits,charstopBits);
voidUartWriteChar(charc);
voidUartWriteStr(char*s);
charUartReadChar();

#endif/* __UART_H */
```

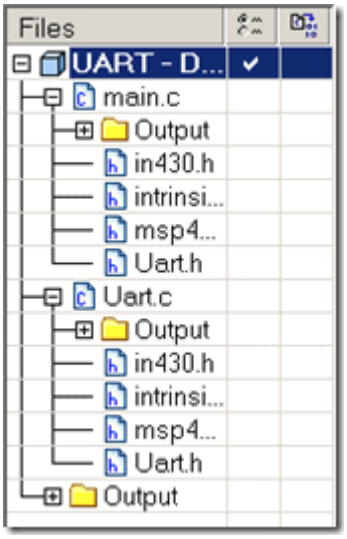
其中#ifndef 等预编译用来防止重复包含。

3.程序调用示例：

要调用这个函数库，首先要包含 Uart.h 头文件；把 Uart.h 拷到对应文件夹中，然后在要调用程序的源程序文件中添加文件包含：

```
#include"msp430x16x.h"  //430寄存器头文件
#include"Uart.h"          //串口通讯程序库头文件
```

然后，在项目中加入 Uart.c，把 Uart.c 拷入项目文件夹中，在项目中添加文件，加入后文件结构大致如下图：



如果要用9600以上的波特率，需要把 SMCLK 设为1M，我的程序调用了以下的这个函数，把 MCLK 设为8MHz，SMCLK 设为1MHz：

```
voidClkInit()
{
    chari;
  BCSCTL1 &= ~XT2OFF;                    //打开 XT2振荡器
```

```
  IFG1&=~OFIFG;                          //清除振荡错误标志
    while((IFG1&OFIFG)!=0)
     {
         for(i=0;i<0xff;i++);
       IFG1&=~OFIFG;                     //清除振荡错误标志
    }
   BCSCTL2 |= SELM_2+SELS+DIVS_3;   //MCLK 为8MHz，SMCLK 为1MHz
  }
```

调用示例程序如下：

```
  ClkInit();

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
  _EINT();
  UartWriteStr(str);
UartWriteChar(0x0d);     //发送"换行"(\r)"
UartWriteChar(0x0a);     //发送"回车"(\n)"

  UartWriteStr("下面测试串口收发函数\r\n");

while(1)                          //串口测试
  {
   chr=UartReadChar();       //收1字节
  UartWriteChar(chr);       //将收到的数据返回
  }
```

如果是9600以下的波特率，可以不调用时钟系统初始化函数，否则必须调用这个函数，或者用其他的方法把 SMCLK 频率设为1MHz；初始化完成之后，还要开中断（因为 Uart 函数库用到了中断）；然后才能正常的使用这些函数。

这样，这个程序库就完成了，欢迎大家下载使用。

另外，网上还有一个用预编译的方法计算波特率的，个人认为比较好，没有选用它的原因是：它只能在编译时确定寄存器内容，无法再运行的时候进行设置。
http://www.ideyi.org/bbs/article_1077_374533.html

还有一个用网页计算波特率的，可以上网时计算波特率很方便：
http://www.838dz.com/calculator/1805.html

相关文章及附件下载：http://www.ideyi.org/bbs/article_1077_368325.html