文章编号：1671-1041(2004)01-0079-02

基于 MSP430 的串行通信软件设计

李 卫，陶维青

（合肥工业大学 电气与自动化工程学院，安徽合肥 230009）

摘要： 文章在介绍 MSP430 系列单片机的基础上，着重论述了其在串行通信应用中的软件设计方法。就不同系列硬件集成度的不同，分别给出了直接利用 USART 硬件模块实现串行通信的过程和利用定时器 A(Timer A) 的比较 / 捕获功能来模拟 UART 的编程方法，后者则为实现多个串行通信口提供了解决思路。

关键词： MSP430; 串行通信； USART; 比较 - 捕获功能

中图分类号： TP311.52 **文献标识码：** B

1 MSP430 单片机简介

MSP430 系列单片机是 TI 公司近几年推出的一种 16 位的超低功耗的混合信号控制器。MSP430 工作在 1.8~3.6V 电压下，有正常工作模式和 4 种低功耗工作模式，单片机可以方便地在各种工作模式之间切换，用中断请求将 CPU 唤醒只要 6μs。MSP430 也具有非常高的集成度，单片集成了多通道 12bit 的 A/D 转换、片内精密比较器、多个具有 PWM 功能的定时器、斜边 A/D 转换、片内 USART、看门狗定时器、片内数控振荡器（DCO）、大量的 I/O 端口以及大容量的片内存储器，单片可以满足绝大多数的应用需要。MSP430 的这种高集成度使应用人员不必在接口外接 I/O 及存储器上花太多的精力，而可以方便的设计真正意义上的单片系统。MSP430 的片内存储器有 ROM、OTP、EPROM、Flash Memory 4 种型号，采用冯．诺伊曼结构，因此 RAM、ROM 和全部的外围模块都位于同一地址空间内。MSP430 的超低功耗特性使其在电池供电、便携式设备的应用中表现出非常优良的特性，在国内已经有了越来越多的应用。

2 MSP430 串行通信的特点

MSP430 系列中有 USART 模块的有 F12X、F13X、F14X、C13X1、X33X、F43X 和 F44X 等，其中 F14X 和 F44X 系列片内有两个 USART 模块，其他型号只有一个。而所有的型号都可以通过定时器实现软件串行口。与此对应，MSP430 系列的串行通信就有了两种差别较大的实现方式：一是利用硬件通用串行同步 / 异步模块（USART），通过对一系列的寄存器设置后，由硬件自动实现数据的移入移出；二是在定时器模块支持下，由用户软件控制，一位一位的将数据由端口送出或接收进来。在大部分的 MCU 硬件实现方式中，都使用预分频器和分频器的方法产生合适的波特率，而 MSP430 的波特率发生器不但有一个分频计数器，还有一个调整器。分频因子由所需波特率和和分频计数器的时钟频率决定，通常情况分频因子会有小数部分，分频计数器实现整其数部分，而由调整器实现对其小数部分的逼近。用分频器计数加调整的方法可以实现字节内的各位有不同的分频因子，从而保证在低时钟频率时实现高通波特率。

3 串行通信的实现

分为直接利用 USART 模块实现和利用定时器的比较捕获功能来模拟实现两种方法。

3.1 利用 USART 模块实现

3.1.1 相关寄存器设置

在 MSP430 中，每个 USART 模块由相关联的几个寄存器实现控制操作。

* 控制寄存器 UXCTL —— USART 模块的基本操作由此寄存器的控制位决定，如通信协议的选择、通信模式、字符长度及校验位等。

* 发送控制寄存器 UXTCTL ——波特率发生器时钟源的选择，接收和发送用同一个波特率发生器。

* 接收控制寄存器 URCTL ——控制与接收操作相关的串行口硬件。

* 波特率选择寄存器 UXBR0(或 UXBR1) ——分频器分频因子的整数部分。

* 波特率调整控制寄存器 UXMCTL ——反映分频器分频因子的小数部分。

* 接收发送数据缓存 URXBUF、UTXBUF ——接收的数据和待发送的数据放到这里。

USART 接收和发送使用两个独立的中断源，对应了两个独立的中断向量，所以发送和接收在不同的中断程序中处理。另外，中断的产生还需要对总的中断使能位和接收发送使能位进行置位，这两个控制位在特殊功能寄存器中定义。

通过对上面一系列寄存器正确设置后，就可以在中断处理程序中轻松实现通讯了。

3.1.2 通讯过程组织

发送和接收由于有独立的中断向量，故都可以采用中断方式处理。发送当然可以采用查询方式，但在多任务系统中为保证系统实时性，采用中断方式更为合理。接收过程应和定时器配合，在接收开始后在定时器中断里面计算延时时间，在超出最长的接收等待延时时间后就可判定一次完整的传输过程结束，转而进行报文处理。发送是在发送中断处理程序中进行发送计数，一旦完成，立即关中断。

3.2 利用定时器 A 的比较 / 捕获功能实现

3.2.1 工作原理

任选 TimerA 的一个比较 / 捕获寄存器（这里以 CCR0 为例），选择和定时器输出信号相复用的两个 I/0 口作为接收发送端口。波特率是用定时器定时产生中断来实现的，由定时器时钟源的频率和要求的波特率决定每个数据位收发所需的时间间隔 BitTime，有如下公式确定：

$$BitTime=T_{clk}/Band$$

其中，T_{clk} 为该定时器基准频率，Baud 为所需的波特率。

发送过程比较简单，首先赋给 CCR0 合适的初值，以产生与波特率对应的数据位时间间隔，然后在比较功能所触发的中断处理程序中将数据从输出单元的引脚一位一位的移出，就可以实现完整数据的发送。

接收时，首先将比较 / 捕获寄存器设为下降沿捕获模式，以实现接收数据的检测。图 1 显示了完整的一个字节接收的时序（UART8N1 协议）。

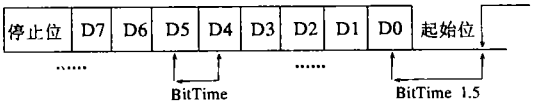

图 1 UART8N1 协议的字符时序

收稿日期：2003-09-02 电子邮件来稿

当检测到起始位时，硬件将计数器 TAR 中的数据写入 CCR0 中，并产生相应中断，在该中断处理程序中，将捕获功能转换为比较功能，并将 1 .5 位的时间间隔加到 CCR0 中，这样当到达 1 .5 位时间间隔时会由比较功能触发一次中断，从而可以读取输入端口的状态，接收到第一个数据位的信息。此后，由定时器产生一位的时间间隔，比较功能再次触发中断，读取第二个数据位的信息，如此反复，实现一个完整字节的接收。 由于 TimerA 的比较 / 捕获寄存器还有将输入信号锁存的功能，只要在下一个数据位到来之前将数据位接收，就可以实现输入信号的准确接收，而不必担心数据位的丢失。

3.2.2 通讯过程组织

需要注意的是，由于接收和发送的处理程序在同一个中断服务程序中，所以在定时器中断服务程序中就应当根据相应寄存器的状态来判断当前的收发状态。收发结束后，立即在中断服务程序中将中断关掉（图 2）。下面给出了 C 语言实现的发送和接收程序的部分例程，以供参考。

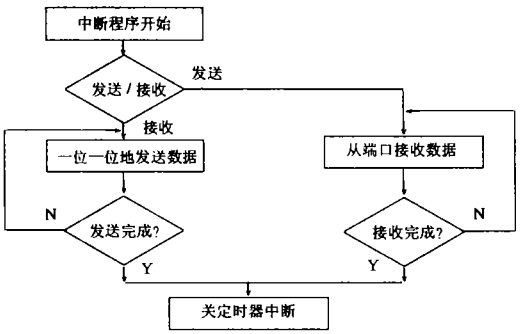

图 2 流程图

例程选用 MSP430F149 单片机，子系统时钟为 1MHz 。利用定时器 A 模拟软件串行口，选用定时器的 CCR0 寄存器，由 P1.1 口发送，P2.2 口接收，8N1 协议，9600 波特率的通讯，实现接收一个字节后再发送出去。

```
#define hfBitime    0x47// 0.5 位长度 + 调整
#define Bitime      0x6C// 9620 波特率
unsigned int  rtBuffer;// 发送接收缓冲
unsigned char BitCnt;  // 位计数

// 发送字符函数
void TX_Byte (void)
 {
  BitCnt = 0xA;          // 初始化位记数
  CCR0 = TAR; // TA 记数器的目前状态
  CCR0 += Bitime;        // 一位的等待时间
  rtBuffer |= 0x100;  // 增加标志停止位
  rtBuffer <<= 1;       // 增加 1 位起始位
  CCTL0 = OUTMOD0+CCIE;          // TXD = mark = idle
  while ( CCTL0 & CCIE );       // 等待发送完成
}

// 等待接收函数
void RX_Ready (void)
 {
  BitCnt=0x8;     // 初始化位记数
  CCTL0 = SCS+CCIS0+OUTMOD0+CM1+CAP+CCIE; // 同步，
CCI0B 为捕获事件中断源，置位模式
// 输出．下降沿，捕获模式，中断使能
}

//Timer A 中断服务程序
#if __VER__ < 200
interrupt[TIMERA0_VECTOR] void Timer_A (void)
#else
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
#endif
 {
  CCR0 += Bitime; // 加时间偏移到 CCR0
// 接收
  if (CCTL0 & CCIS0)       //  在接收状态?
  {
    if( CCTL0 & CAP )    //  在捕获模式?
    [
      CCTL0 &= ~ CAP;    // 捕获转为比较
      CCR0 += hfBitime; // 加 0.5 位的偏移
    }
    else
    {
      rtBuffer >>=1;
      if (CCTL0 & SCCI) // 从 SCCI 锁存中获得数据
       rtBufferl=0x80;
      BitCnt--;
      if ( BitCnt == 0)
      {
CCTL0 &= ~ CCIE;          // 接收完毕，关中断
      }
    }
  }
// 发送
  else
  {
    if ( BitCnt == 0)
      CCTL0 &= ~ CCIE;      // 发送完成，关中断
    else
    {
      CCTL0 |=  OUTMOD2;  // 发送 0
      if (rtBuffer & 0x01)
CCTL0 &= ~ OUTMOD2;          // 发送 1  置位模式
      rtBuffer >>= 1;
      BitCnt --;
    }
  }
}
```

4 小结

实验证明，应用以上的设计方法均能很好的实现 MSP430 单片机与微机之间的串行通讯，能够保证较好的实时性和较小的误差。其中，利用定时器的硬件特性所实现的串行通讯，与通常的 I/O 口扩展相比，还能减轻 CPU 的负担，降低功耗。进而与 MSP430 低功耗特性相结合，可设计出功耗极小的实时通讯系统。●

参考文献

[1] 胡大可．MSP430 系列 FLASH 型超低功耗 16 位单片机．北京航空航天出版社，2001.

[2] MSP430x1xx Family User's Guide (Rev. C), Texas Instruments Literature Number SLAU049C,2003.

[3] Mark Buccini,Implementing a UART Function With TimerA3 , Texas Instruments Incorporated, 2002.

作者简介：李卫，男，安徽安庆人，硕士研究生，研究方向为计算机控制系统，E-mail: wei_lea638@163.com; 陶维青，男，教授，现从事自动化仪表及配网自动化方面的研究工作。

作者声明：自愿将本文稿酬捐为“仪器仪表用户杂志爱心助学基金”