MSP430 串行异步通讯原理与实现

南京航空航天大学　魏小龙

本讲讲述串口功能与连接的实现。大多数 MSP430 芯片都有硬件异步通讯功能，有一些器件有两个通讯端口，也有少数没有。没有硬件串口的芯片可以实现软件（模拟）串口。下面表格为 430 系列芯片串口的情况。

系列芯片	F11 系列	F12 系列	F13 系列	F14 系列	F15 系列	F16 系列
串口数量	0	1	1	2	1	2

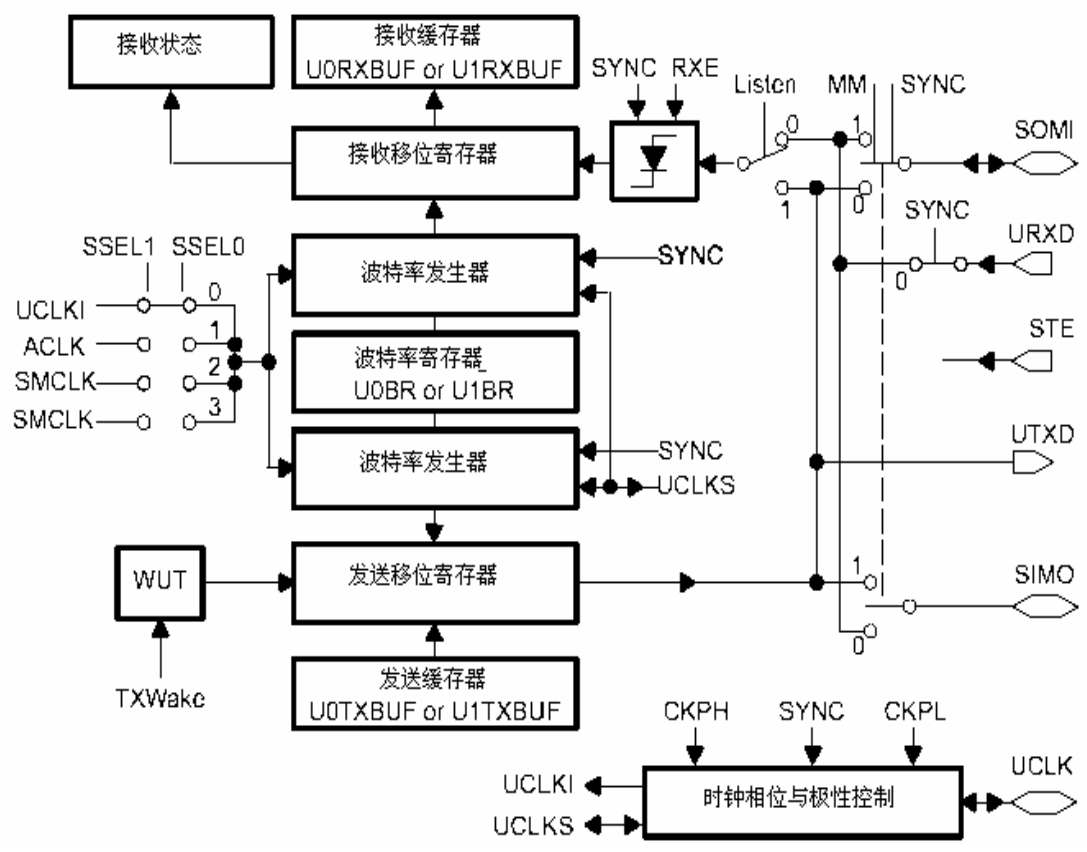
芯片系列	F2 系列	C31 系列	C32 系列	C33 系列	F41 系列	F42 系列
串口数量	1	0	0	1	0	1

芯片系列	FW42 系列	FE42 系列	FG43 系列	F43 系列	F44 系列	
串口数量	0	1	1	1	2	

对于没有硬件串口的芯片也可以实现软件串口，这里先讲硬件串口，后讲软件串口。然后再讲串口的链路实现。

先看串口功能的实现。

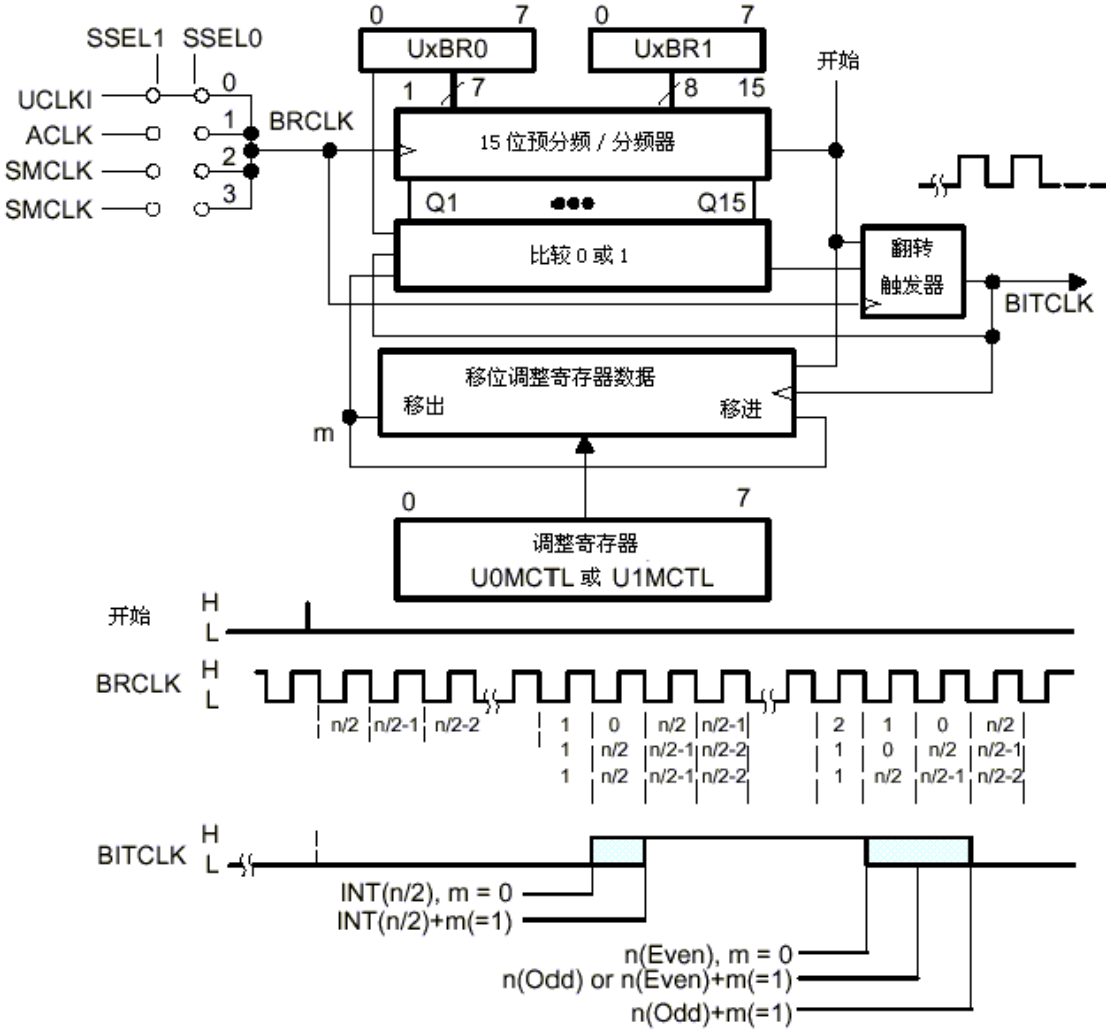
下图是 MSP430 系列芯片硬件串口的框图。



在该框图中，串口通讯由 3 部分构成：通讯速度的控制（数据位流的产生）、接收控制部分、

发送控制部分。

波特率生成部分由时钟输入选择与分频、波特率发生器、调整器、波特率寄存器等组成。串行通信时，接收与发送以什么样的速率将数据位收进或送出呢，这个速率就由波特率生成构件控制。下图为其较为详细的结构。



整个模块的时钟源来自内部 3 时钟或外部输入时钟，由 SSEL1、SSEL0 选择，以决定最终进入模块的时钟信号 BRCLK 的频率。时钟信号 BRCLK 送入一个 15 位的分频器，通过一系列的硬件控制，最终输出移出与移进两移位寄存器使用的移位位时钟 BITCLK 信号。那么这个信号（BITCLK）究竟是怎样产生的呢，该图的下半部分的一个波特率产生例子可以看出，是分频器在起作用。当计数器减计数到“0”时，输出触发器翻转，送给 BITCLK 信号。所以 BITCLK 信号周期的一半就是定时器（分频计数器）的定时时间。

接收控制部分与发送控制部分分别由两个移位寄存器构成。接收时，当接收到一个完整数据，产生一个信号（URXIFG0＝1），表示接收到完整数据，可以将此数据取走。而在发送时，当一个数据正在发送过程中，UTXIFG0＝1，此时，不能再发送数据，必须等当前数据发送完毕（UTXIFG0＝0）时，方可继续发送。

串口接收一般采用中断方式，而发送数据则多采用主动方式。下面是一段简单的完整通讯程序，实现功能：将接受的数据原样送回。

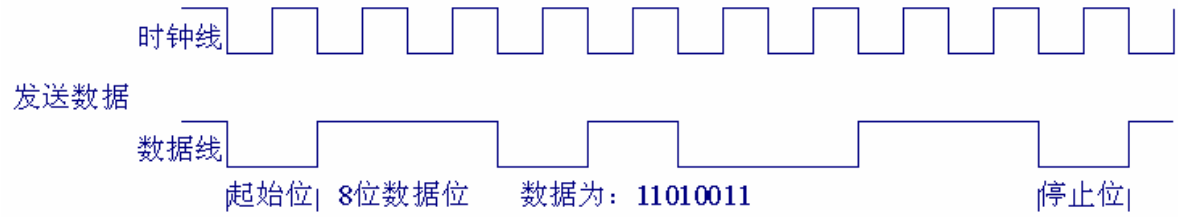
```
#include <msp430x44x.h>
void main(void)
```

```
{
  WDTCTL = WDTPW + WDTHOLD;   // 停止看门狗
  UTCTL0 = SSEL0;             // UCLK = ACLK，选择时钟来源
  UBR00 = 0x03;               // 32k/9600 - 3.41   波特率寄存器低字节
  UBR10 = 0x00;               // 32k/9600          波特率寄存器高字节
  UMCTL0 = 0x51;              // 由于波特率计算有余数，填写波特率调整寄存器
  UCTL0 = CHAR;               //  数据格式为 8 位数据
  ME1 |= UTXE0 + URXE0;       //  使能串口 TXD 与 RXD
  IE1 |= URXIE0;              // 让串口接收到数据后能产生中断
  P2SEL |= 0x30;              //  定义 P2.4,P2.5 为串口功能引脚
  P2DIR |= 0x10;              // 串口发送数据端口为输出，接收数据端口为输入
  _EINT();                    // 整个系统使能中断（开总中断）
  _BIS_SR(LPM3_bits);         //  初始化完毕，进入睡眠状态，主程序完毕
}

interrupt[UART0RX_VECTOR] void usart0_rx (void)
{
  while ((IFG1 & UTXIFG0) == 0);          // 当发送缓存为空时
  TXBUF0 = RXBUF0;                        //发送数据到串口
}
```

而对于没有硬件串口的型号，如何实现异步串口功能？

先分析异步串口的原理。下图是异步串口的时序图。



可以看出异步串口由一根口线构成：数据线，在数据发送时，数据线严格按照其时序将数据移位送至数据线，就可以了。图中的时钟是隐含的，由波特率确定。比如串口波特率为 9600，则时钟的周期为 1/9600 秒。在数据线上的数据按照：起始位、数据位、停止位等格式顺序排列。而起始位、数据位、停止位等的多少由通讯双方定义的通讯规约决定。

这样在没有硬件串口的情况下，完全可以模拟以上时序发送异步串行数据。下面的工作将完成上图的数据传送。

首先产生波特率。下面的延时程序可以完成此工作，延时时间为 1/9600 秒（系统时钟为 1M 时的延时循环参数）。

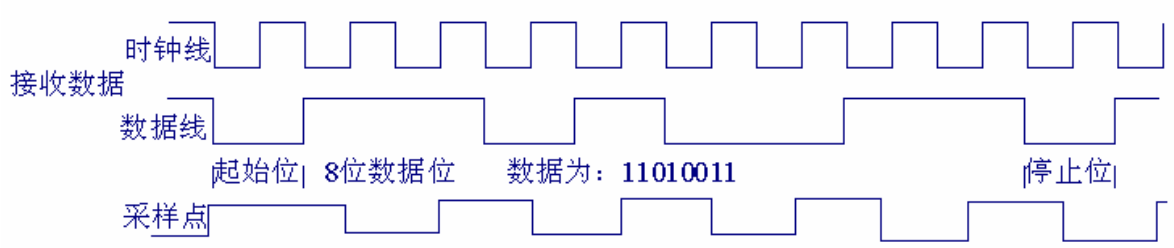
```
void Delay_9600(void)
    {
     unsigned int v=104;
     while(v!=0)v--;
    }
```

以上的延时程序用于产生通讯位率。下面定义 P1.0 为通讯数据发送端，P2.0 为通讯数据接收端。则按照通讯规约的时序图，每发送一位数据，调用一次延时程序：

调用以下程序之前已经进行了相应的端口方向设置。

```
void send_byte(char in)  //输入变量为即将发送的数据
    {
    char I=0;                       //定义一个循环变量，循环发送 8 个数据位
    P1OUT &= ~BIT0;                 //发送起始位
    Delay_9600();
    for(I=0;I<8;I++)
        {
        if(in&1)
              P1OUT |= BIT0;
        else    P1OUT &= ~BIT0;     //将数据位送到端口
        in = in>>1;                 //准备下位数据
        Delay_9600();               //位时钟
        }
    P1OUT &= ~BIT0;                 //发送停止位
    Delay_9600();
    }
```

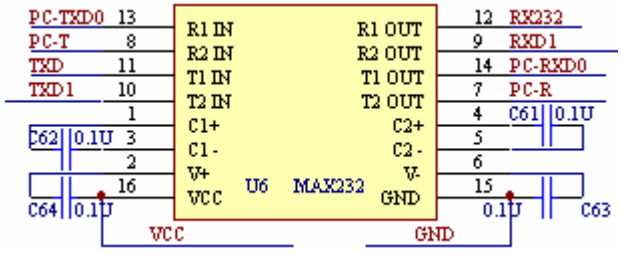
数据的接收可以使用中断的方式，设置 P2.0 为设为输入，并使能中断。当串行数据送达 P2.0 时，该端口将产生中断，在中断服务程序中，顺序接受 10 位数据，并去掉最先位（起始位）与最后位（停止位），再将中间 8 位组合成一个字节，即为接收到的数据，然后退出中断，等待其他数据的接收。这里要注意接收每一位数据的采样点。



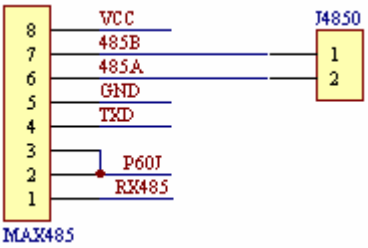
可以将起始位采取不予理睬的办法，如上图所示，则第一位数据的采样位于进入中断后的 1.5 位时钟处。然后延时一个位时钟的时间，再采样下一为数据。注意所有的采样点位于每一位数据的中间位置，其原因很明显：在时间上可以最大程度地容错。具体的接收程序略。

现在对于有无硬件串口，都可以进行串口通讯了，但这只能近距离 TTL 电平连接，对于远距离呢，必须使用对应的硬件电路实现通讯链路。常用的通讯链路有 RS232，RS485，红外线等。所有通讯链路的实现都只是将通讯双方以一定的电气规约联系起来。

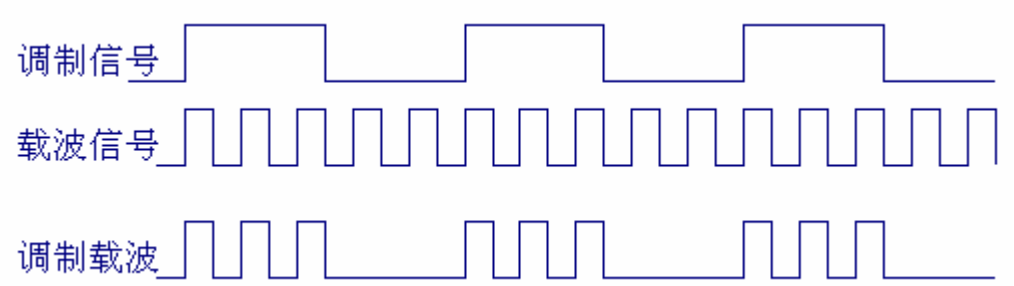
RS232 链路可以将通讯双方在 15 米以内有效连接。RS232 规定逻辑电平 0 为＋3～＋15V 电压，逻辑电平 1 为－15～－3V 电压，通常以 MAX232 芯片实现电平转换，具体电路如下图。



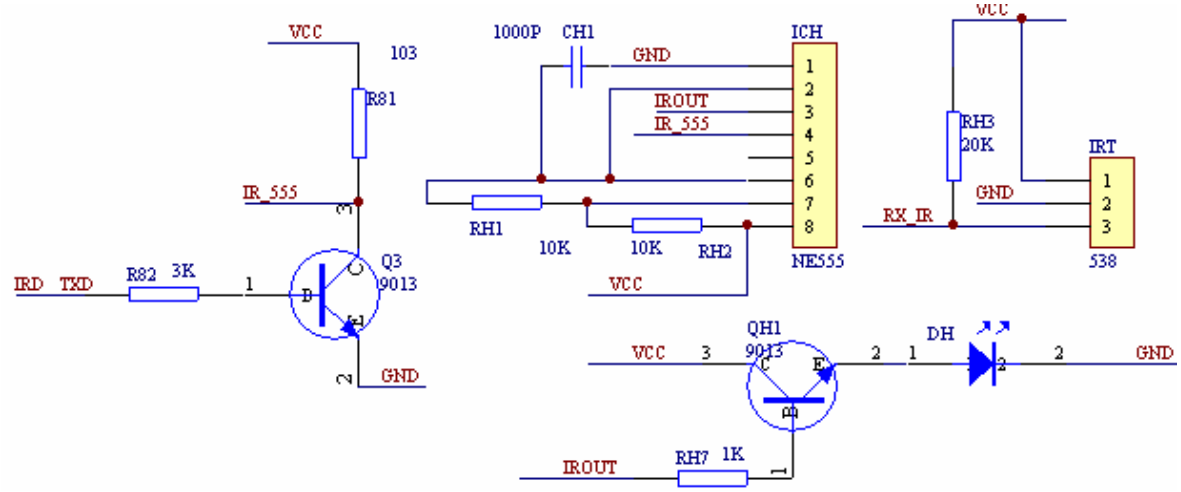
如果要实现较远的通讯距离，则通常选择 RS485 总线。RS485 通讯距离能达到 1.2Km，使用双绞线，但只能半双工通讯（即不能同时发送与接收数据）。使用 RS485 可以方便进行单片机网络构成，所有节点都挂在 RS485 总线上。RS485 总线使用差分电压，具有很高的抗干扰能力。规定总线 A 高于 B0.2V 时为数据 1，总线 B 高于 A0.2V 时为数据 0。下面是典型的 RS485 电路如下图所示。其中 P60J 为总线方向控制，因为 MAX485 器件半办双工，在同一时间只能是数据发送或数据接收，所以需要控制器件所处的工作状态，要么是发送数据，要么是接收数据。



前面两种通讯链路都是有线连接，下面讲讲最常见的红外通讯链路。红外通讯是将红外线作为通讯的载体。下图是红外线调制原理，当需要送处的信号为 1 时，就将调制信号送出；当要送出信号 0 时，就不发送任何信号。



下图是常用的红外线调制解调电路。常用的红外载波信号为 38K 的方波信号，下图使用 NE555 产生 38K 方波信号；而接收部分采用一体红外接收器件，将红外接收管、放大电路、解调制等集成为一体，给使用带来极大方便，只需要一个输出上来电阻即可。



红外通讯可实现近距离无线连接，但不能绕过障碍物，还有很多其他通讯链路，这里不一一介绍。