第 5 章

串 行 通 信

5.1 串行通信概述

MSP430F169 单片机的串行通信模块 USART 包括三个部分：UART 异步串行通信、SPI 同步串行通信和 I2C 同步串行通信。本章仅针对 UART 异步串行通信。

串口是系统与外界联系的重要手段，在嵌入式系统开发和应用中，经常需要上位机实现系统调试及现场数据的采集和控制。一般是通过上位机本身配置的串行口，借助串行通信技术，与嵌入式系统进行连接通信。

MSP430 系列的单片机可实现的串行通信功能有 USART 硬件直接实现和通过定时器软件实现两种。片内具有 USART 模块的 MSP430 系列单片机，由于系列不同，片内可包含一个或多个 USART 块。USART 模块可以自动从任何一种低功耗模式 LPMx 开始工作。所有 USART0 和 USART1 都可以实现两种通信方式：USRT 异步通信和 SPI 同步通信。另外，MSP430F16X 系列单片机的 USART0 还可以实现 I2C 通信。其中，UART 异步通信和 SPI 同步通信的硬件是通用的，经过适当的软件设计，这两种通信方式可以交替使用。

MSP430 串行异步通信模式通过两个引脚（即接收引脚 URXD 和发送引脚 UTXD）与外界相连。

异步通信的特点如下：

（1）异步模式，包括线路空闲/地址位通信协议。

（2）两个独立移位寄存器：输入移位寄存器和输出移位寄存器。

（3）传输 7 位或 8 位数据，可采用奇校验或偶校验或者无校验。

（4）从最低位开始的数据发送和接收。

（5）可编程实现分频因子为整数或小数的波特率。

（6）独立的发送和接收中断。

（7）通过有效的起始位检测将 MSP430 从低功耗唤醒。

（8）状态标志检测错误或者地址位。

1. 异步通信字符格式

异步通信字符格式由四部分组成：起始位、数据位、奇偶校验位和停止位，如图 5-1 所示。

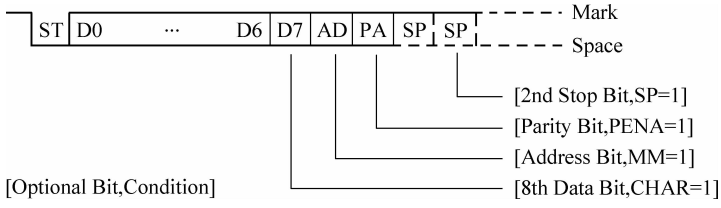

图 5-1　异步通信字符格式

用户可以通过软件设置数据位、停止位的位数，还可以设置奇偶位的有无。通过选择时钟源的波特率寄存器的数据来确定位周期。

接收操作以收到有效起始位开始。起始位由检测 URXD 端口的下降沿开始，然后以 3 次采样多数表决的方法取值。如果 3 次采样至少两次是 0 才表明是下降沿，然后开始接收初始化操作，这一过程实现错误起始位的拒收和帧中各数据的中心定位功能。

MSP430 可以处于低功耗模式，通过上述过程识别正确起始位后，MSP430 可以被唤醒，然后按照通用串口接口控制寄存器中设定的数据格式，开始接收数据，直到本帧采集完毕。

异步模式下，传送数据以字符为单位传送。因为每个字符在起始位处被唤醒，所以传输时多个字符可以一个接一个地连续传送，也可以断续发送和接收。

2. 异步多机通信模式

在异步模式下，USART 支持两种多机通信模式，即线路空闲多机模式和地址位多机模式。信息以一个多帧数据块，从一个指定的源传送到一个或者多个目的位置。在同一个串行链路上，多个处理机之间可以用这些格式来交换信息，实现了在多处理器通信系统间的有效数据传输。它们也用于使系统的激活状态压缩最低，以节省电流消耗或处理器所用资源。控制寄存器的 MM 位用来确定这两种模式。这两种模式采用唤醒发送、地址特性和激活等功能。

1）线路空闲多机模式

在这种模式下，数据块被空闲时间分割。在字符的第一个停止位之后，收到 10 个以上的 1，则表示检测到接收线路空闲，如图 5-2 所示。

如果采用两位停止位，则第二个停止位被认为空闲周期的第一个标志。空闲周期的第一个字符是地址字符。RXWAKE 位可用于地址字符的标志。当接收到的字符是地址字符时，RXWAKE 被置位，并送入接收缓存。用发送空闲帧来识别地址字符的步骤如下：

（1）TXWAKE＝1，将任意数据写入 UTXBUF（UTXIFG＝1）。当发送移位寄存器为空时，UTXBUF 的内容将被送入发送移位寄存器，同时 TXWAKE 的值移入 WUF。

（2）如果此时 WUT＝1，则发送的起始位、数据位及校检位等被抑止，发送一个正好 11

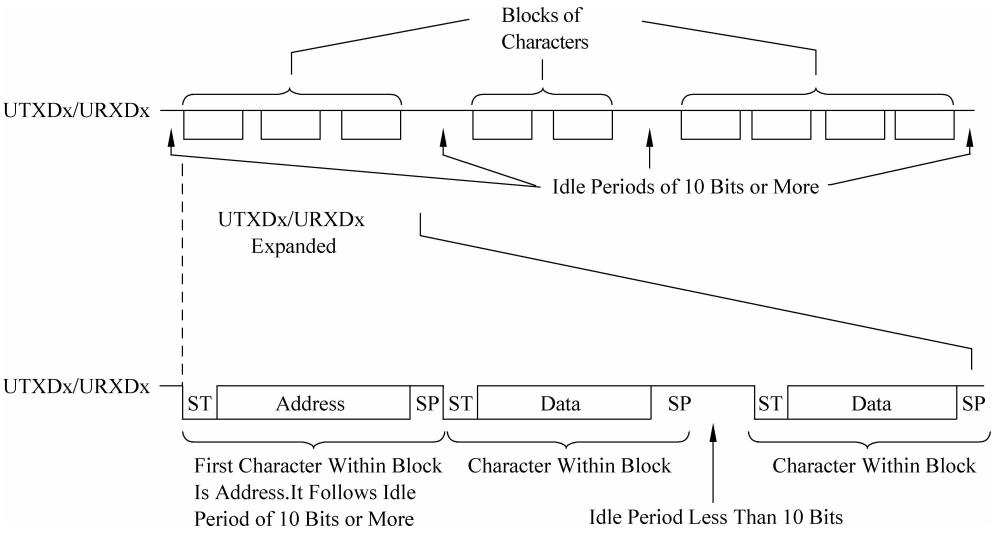

图 5-2 线路空闲多机模式

位的空闲周期。

(3) 在地址字符识别空闲周期后移出串口的下一个数据是 TXWAKE 置位后写入 UTXBUF 中的第二个字符。当地址识别被发送后,写入 UTXBUF 中的第一个字符被抑制,并在以后被忽略。这时须随便往 UTXBUF 中写入一个字符,以便能将 TXWAKE 的值移入 WUF 中。

2) 地址位多机模式

地址位多机模式的格式如图 5-3 所示。在这种模式下,字符包含一个附加的位作为地

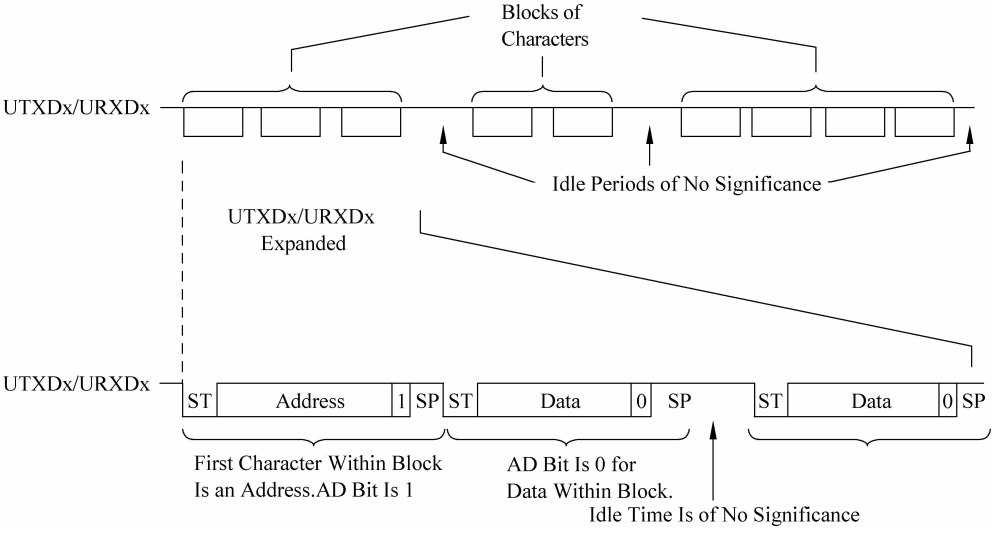

图 5-3 地址位多机模式

址标志。数据块的第一个字符带有一个置位的地址位，用以表明该字符是一个地址。当接收字符是地址时，RXWAKE 置位，并且将接收的字符送入接收缓存 URXBUF。

当 USART 的 URXWIE＝1 时，数据字符在通常方式的接收器内拼装成字节，但它们不会被送入接收缓存，也不产生中断。只有当接收到一个地址位为 1 的字符时，接收器才被激活，接收到的字符被送往 URXBUF，同时 URXIFG 被置位。如果有错误，则相应的错误标志被设置。应用软件在判断后作出相应的处理。在地址位多机模式下，通过写 TXWAKE 位控制字符的地址位。每当字符由 UTXBUF 传送到发送器时，TXWAKE 位装入字符的地址位，再由 USART 将 TXWAKE 位清除。

3．串行操作自动错误检测

USART 模块接收字符时，能够自动进行校验错误、帧错误、溢出错误和打断状态检测。各种检测错误的含义标志如下：

(1) FE：标志帧错误。当一个接收字符的停止位为 0 并被装入接收缓存时，则认为接收到一个错误帧。

(2) PE：奇偶校验错误。当接收的 1 的个数与它的效验位不相符时，则认为接收到一个错误帧。

(3) OE：溢出错误标志。当一个字节写入 UxRXBUF 时，前一个字符还没有被读出，则溢出。

(4) BRK：打断检测标志。当发生一次打断且 UxRXEIE 置位时，该位为 1。

当上述任何一种错误出现，位 RXERR 置位，表明有一个或者多个出错标志(FE、PE、OE 和 BRK 等)被置位。

如果 URXEIE＝0 并且有错误被检测到，那么接收缓存不接收任何数据。如果 URXEIE＝1，接收缓存接收字符，相应的错误检测标志置位，直到用户软件复位或者接收缓存内容被读出，才能复位。

4．波特率的产生

在异步通信中，波特率是很重要的指标，表示每秒钟传送二进制数码的位数。波特率反映了异步串行通信的速度。波特率发生器产生同步信号表明各位的位置。波特率部分由时钟输入选择和分频、波特率发生器、调整器和波特率寄存器组成。串行通信时，数据接收和发送的速率就由这些构件控制。

整个模块的时钟源来自内部 3 个时钟或外部输入时钟，由 SSEL1 和 SSEL0 选择，以决定最终进入模块的时钟信号 BRCLK 的模式。

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，最终输出两移位寄存器使用的移位时钟 BITCLK 信号。这个信号(BITCLK)的产生，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 信号周期的一半就是定时器(分频计数器)的定时时间。

5. 波特率的设置与计算

采集每位数据时，在每位数据的中间都要进行 3 次采样，以多数表决的原则进行数据标定与移位接收操纵，依次逐位采集。由此看出，分频因子要么很大，要么是整数，否则，由于采集点的积累偏移，会导致每帧后面的几位数据采样点不在其数据的有效范围内。

MSP430 的波特率发生器使用一个分频计数器和一个调整器，能够用低时钟频率实现高速通信，从而在系统低功耗的情况下实现高性能的串行通信。使用分频因子加调整的方法可以实现每一帧的各位有不同的分频因子，从而保证每个数据钟的 3 个采样状态都处于有效的范围内。

分频因子 N 由分频计数器的时钟(BRCLK)的频率和所需的波特率来决定：

$$N = \text{BRCLK}/\text{波特率} \tag{5-1}$$

如果使用常用的波特率与常用晶体产生的 BRCLK ，则一般得不到整数的 N，还有小数部分。分频计数器实现分频因子的整数部分，调整器使得小数部分尽可能准确。

波特率可由下式计算：

$$\text{波特率} = \text{BRCLK}/\text{N} = \text{BRCLK}/(\text{UBR} + (\text{M7} + \text{M6} + \cdots + \text{M0})/8) \tag{5-2}$$

其中，N 为目标分频因子；UBR 为 UXBR0 中的 16 位数据值；MX 为调整寄存器(UXMTCL)中的各数据位。

例如：BRCLK＝32.768kHz，要产生 BITCLK＝2400Hz，分频器的分频系数为 32768/2400＝13.65，所以设置分频器的计数值为 13。接下来用调整寄存器的值来设置小数部分的 0.65。调整器是一个 8 位寄存器，其中每一位分别对应 8 次分批情况，如果对应位 0，则分频器按照设定的分频系数分频计数；如果对应位为 1，则分频器按照设定的分频系数加 1 分频计数。按照这个原则 0.65 * 8＝5，也就是说，8 次分频计数过程中应有 5 次加 1 计数，3 次不加 1 计数。调整寄存器的数据由 5 个 1 和 3 个 0 组成。调整器的数据每 8 次周而复始循环使用，最低位最先调整，比如设置调整寄存器的值为 6BH(01101011)，当然也可以设置其他值，必须有 5 个 1，而且 5 个 1 要相对分散。即分频器按顺序 13、14、14、13、14、13、14、14 来分频。在 8 位调整器调整位都使用后，再重复这一顺序。

常用波特率设置如表 5-1 所示。

表 5-1 常用波特率设置

Baud Rate	Divide by		A：BRCLK＝32,768Hz						B：BRCLK＝1,048,576Hz				
	A：	B：	UxBR1	UxBR0	UxMCTL	Max. TX Error %	Max. RX Error %	Synchr. RX Error %	UxBR1	UxBR0	UxMCTL	Max. TX Error %	Max. RX Error %
1200	27.31	873.81	0	1B	03	−4/3	−4/3	±2	03	69	FF	0/0.3	±2
2400	13.65	436.91	0	0D	6B	−6/3	−6/3	±4	01	B4	FF	0/0.3	±2
4800	6.83	218.45	0	06	6F	−9/11	−9/11	±7	0	DA	55	0/0.4	±2
9600	3.41	109.23	0	03	4A	−21/12	−21/12	±15	0	6D	03	−0.4/1	±2
19,200		54.61							0	36	6B	−0.2/2	±2
38,400		27.31							0	1B	03	−4/3	±2
76,800		13.65							0	0D	6B	−6/3	±4
115,200		9.1							0	09	08	−5/7	±7

在后面的例程中，对波特率设置有详细的解释。

5.2　USART 相关寄存器

1. 串口控制寄存器——UxCTL

其各位定义如下：

位	7	6	5	4	3	2	1	0
	PENA	PEV	SPB	CHAR	LISTEN	SYNC	MM	SWRST

PENA：校验允许位。0：校验禁止；1：校验允许。校验允许时，发送端发送校验，接收端接收该校验。地址位多机模式中，地址位包含校验操作。

PEV：奇偶位校验位，该位在校验允许时有效。0：奇校验；1：偶校验。

SPB：停止位选择。决定发送的停止位数，但接收时接收器只检测 1 位停止位。0：1 位停止位；1：2 位停止位。

CHAR：字符长度。0：7 位字符长度；1：8 位字符长度。

LISTEN：反馈选择。选择是否将发送数据由内部反馈给接收器。0：无反馈；1：有反馈，发送信号由内部反馈给接收器。

SYNC：USART 模块的模式选择。0：UART 模式(异步)；1：SPI 模式(同步)。

MM：多机模式选择位。0：线路空闲多机协议；1：地址多机协议。

SWRST：控制位。

该位的状态影响其他一些控制位和状态位的状态。在串行口的使用过程中，这一位是比较重要的控制位。一次正确的 USART 模块初始化应该是这样的顺序：先在 SWRST＝1 时设置串行口；然后设置 SWRST＝0；最后如果需要中断，则设置相应的中断使能。

2. 串口发送寄存器——UxTCTL

其各位定义如下：

位	7	6	5	4	3	2	1	0
	Unused	CKPL	SSELx		URXSE	TXWAKE	Unused	TXEPT

CKPL：时钟极性控制位。0：UCLKI 信号与 UCLK 信号极性相同；1：UCLKI 信号与 UCLK 信号极性相反。

SSELx：时钟源选择位。

这两位确定波特率发生器的时钟源，其关系如表 5-2 所示。

表 5-2 时钟源选择位

SSEL1、SSEL0	输入分频选择	宏 定 义
0 0	外部时钟 UCLKI	SSEL_0
0 1	辅助时钟 ACLK	SSEL_1
1 0	子系统时钟 SMCLK	SSEL_2
1 1	子系统时钟 SMCLK	SSEL_3

URXSE：接收触发沿控制位。0：没有接收触发沿检测；1：有接收触发沿检测。

TXWAKE：传输唤醒控制。0：下一个要传输的字符为数据；1：下一个要传输的字符为地址。

TXEPT：发送器空标志。它在异步模式与同步模式时不一样。0：正在传输数据或者发送缓冲器(UTXBUF)有数据；1：发送移位寄存器和 UTXBUF 空或者 SWRST=1。

3. 串口接收寄存器——UxRCTL

其各位定义如下：

位	7	6	5	4	3	2	1	0
	FE	PE	OE	BRK	URXEIE	URXWIE	RXWAKE	RXERR

FE：帧错误标志。0：没有帧错误；1：帧错误。

PE：校验错误标志位。0：校验正确；1：校验错误。

OE：溢出标志位。0：无溢出；1：有溢出。

BRK：打断检测位。0：没有被打断；1：被打断。

URXEIE：接收出错中断允许位。0：不允许中断，不接收出错字符并且不改变 URXIFG 标志位；1：允许中断，接收出错字符并且能够置位 URXIFG。

URXWIE：接收唤醒中断允许位。当接收到地址字符时，此位能够置位 URXIFG；当 URXEIE=0 时，如果接收内容有错误，此位不能置位。

URXIFG：传输中断标志位。0：所有接收的字符能够置位 URXIFG；1：只有接收到地址字符才能置位 URXIFG。

RXWAKE：接收唤醒检测位。在地址位多机模式，接收字符地址位置位时，该机被唤醒，在线路空闲多机模式，在接收到字符前检测到 URXD 线路空闲时，此位被唤醒，RXWAKE 置位。0：没有被唤醒，接收的字符是数据；1：唤醒，接收的字符是地址。

RXERR：接收错误标志位。0：没有接收错误；1：有接收错误。

4. 波特率控制寄存器

波特率控制寄存器 0——UxBR0，其各位定义如下：

位	7	6	5	4	3	2	1	0
	UxBR7～0							

波特率控制寄存器 1——UxBR1，其各位定义如下：

位	15	14	13	12	11	10	9	8
	UxBR15～8							

UxBR0 和 UxBR1 两个寄存器用于存放波特率分频因子的整数部分。其中 UXBR0 为低字节，UXBR1 为高字节。两字节合起来为一个 16 位字，成为 UBR。在异步通信时，UBR 的允许值不小于 3。如果 UBR＜3，则接收和发送会发生不可预测的错误。

5. 波特率调整寄存器——UxMCTL

其各位定义如下：

位	7	6	5	4	3	2	1	0
	M7	M6	M5	M4	M3	M2	M1	M0

如果波特率发生器的输入频率 BRCLK 不是所需的波特率的整数倍，带有一小数，则整数部分写入 UBR 寄存器，小数部分由调整控制寄存器 UxCTL 的内容反映。波特率由以下公式计算：

$$\text{波特率} = \text{BRCLK}/(\text{UBR} + (\text{M7} + \text{M6} + \cdots + \text{M0})/8) \tag{5-3}$$

其中 M0，M1，…，M6 及 M7 为控制器 UxMCTL 中的各位。调整寄存器的 8 位分别对应 8 次分频，如果 M＝1，则相应次的分频增加一个时钟周期；如果 Mi＝0，则分频计数器不变。

6. 串口接收缓冲寄存器——UxRXBUF

其各位定义如下：

位	7	6	5	4	3	2	1	0
UxRXBUF7～0								

接收缓存从接收移位寄存器最后接收的字符，可由用户访问。

当接收和控制条件为真时，接收缓存装入当前接收到的字符，如表 5-3 所示：

表 5-3　接收缓存装入当前接收到的字符

条　件		结　果			
URXEIE	URXWIE	装入 URXBUF	PE	PE	BRK
0	1	无差错地址字符	0	0	0
1	1	所有地址字符	×	×	×
0	0	无差错字符	0	0	0
1	0	所有字符	×	×	×

7. 串口发送缓冲寄存器——UxTXBUF

其各位定义如下：

位	7	6	5	4	3	2	1	0
UxTXBUF7～0								

发送缓存内容可以传送至发送移位寄存器，然后由 UxTXDx 传输。对发送缓存进行写操作可以复位 UxTXIFGx。

8．串口模块控制寄存器——ME1、ME2

其各位定义如下：

ME1 模块允许寄存器 1

位	7	6	5	4	3	2	1	0
	UTXE0	URXE0						

ME2 模块允许寄存器 2

位	7	6	5	4	3	2	1	0
			UTXE1	URXE1				

UTXE0：串口 0 的发送允许。

URXE0：串口 0 的接收允许。

UTXE1：串口 1 的发送允许。

URXE1：串口 1 的接收允许。

0：禁止；1：允许。

9．串口中断标志控制寄存器——IFG1、IFG2

其各位定义如下：

IFG1 中断标志寄存器 1

位	7	6	5	4	3	2	1	0
	UTXIFG0	URXIFG0						

IFG2 中断标志寄存器 2

位	7	6	5	4	3	2	1	0
			UTXIFG1	URXIFG0				

UTXIFG0：串口 0 的发送中断标志。

URXIFG0：串口 0 的接收中断标志。

UTXIFG1：串口 1 的发送中断标志。

URXIFG1：串口 1 的接收中断标志。

0：无中断请求标志；1：有中断请求标志。

10. 串口中断控制寄存器——IE1、IE2

其各位定义如下：

IE1 中断控制寄存器 1

位	7	6	5	4	3	2	1	0
	UTXIE0	URXIE0						

IE2 中断控制寄存器 2

位	7	6	5	4	3	2	1	0
			UTXIE1	URXIE1				

UTXIE0：串口 0 的发送中断允许。

URXIE0：串口 0 的接收中断允许。

UTXIE1：串口 1 的发送中断允许。

URXIE1：串口 1 的接收中断允许。

0：中断请求禁止；1：中断请求允许。

这里总结一下 USART 使用方法：①初始化，包括工作模式、帧结构等；②波特率设置；③中断的相关设置；④编写中断服务函数。

5.3　串行通信协议

1. RS-232 协议简介

RS-232 是目前被广泛使用的异步串行数字通信电气标准，由美国电子工业协会(Electronics Industry Association，EIA)于 1962 年公布，于 1969 年最后修订而成，RS(Recommended Standard)表明它是一种被推荐的标准。该标准定义了数据终端设备(DTE)和数据通信设备(DCE)间按位串行传输的接口信息，合理安排了接口的电气信号和机械要求，过去数十年中，RS-232 在低速数据通信领域出尽了风头。这种传输速度不快、传输距离也不远的接口能够在几乎所有民用通信设备中占据主要角色，一个原因是早期用户对通信速度和距离的要求不高；另一个原因是它被所有 PC、服务器认同为标准串行接口，成为计算机与桌面设备之间最简单、有效、通用的连接通道之一。出于同样原因，在多单片机之间的通信中 RS-232 也占据着重要的位置。

2. RS-232 接口的引脚定义

RS-232 有 25 芯和 9 芯两种，9 芯的 EIA-RS-232C 的接口如图 5-4 所示。

9 芯信号线定义如下：

3 脚 TXD：发送数据(输出)。

2 脚 RXD：接收数据(输入)。

7 脚 RTS：请求发送数据(输出)。

8 脚 CTS：允许发送数据(输入)。

6 脚 DSR：对方准备好(输入)。

5 脚 SG(GND)：地脚。

1 脚 DCD：对方接收另一端(远地)数据时状态(输入)。

4 脚 DTR：本方准备好(输出)。

9 脚 RI：对方收到振铃时状态(输入)。

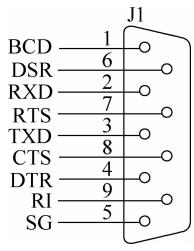

图 5-4 RS-232C 的接口

常用的引脚有 3 根，分别是 2 脚 RXD，3 脚 TXD 和 5 脚 GND。

3. 电气特性

RS-232 协议规定最大的通信速度为 20kb/s，这种速率基本可以满足早期信息传输的需要，但是随着科技的进步，人们对速度的要求越来越高，于是出现原始 RS-232 标准的修订版本，通信速率不断被加快。作为单片机系统，由于其处理能力有限，工作频率不是很高，一般可实现的最高通信速率约为 112kb/s。

RS-232 协议对数字信号的真值与电平的对应关系作了定义，即大于＋3V 的信号被认为是逻辑 0，小于－3V 的信号则被认为是逻辑 1。这里的“电平”是指相对于传输线“信号地”(Signal Ground)的电压。需要说明的是，一般单片机上的 UART 接口虽然在位格式上与 RS-232 协议的定义一样，但是它们在电平定义上是完全不同的。

AVR 单片机的输出信号实际上并不符合 RS-232 的标准，因为其串行通信管脚上的电压为 TTL 标准，即 0～5V 之间的两个状态，传输距离一般在 1～2m 以内。另一方面 RS-232 信号的电压一般在－12～＋12V 之间；另外，彼此对于逻辑 1 和逻辑 0 的定义也完全不同，因此，二者进行通信时，中间必须插入一个电平和逻辑转换环节。

EIA- RS-232C 电平：逻辑 1　－3～－15V

逻辑 0　＋3～＋15V

TTL 电平：逻辑 1　＋2.7～＋5V

逻辑 0　0～＋0.5V

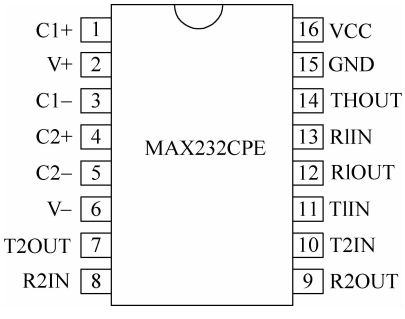

图 5-5 MAX232 芯片引脚图

近年来出现了许多单电源电平转换芯片，其中最为流行的是 MAXIM 公司的 MAX232 芯片，如图 5-5 所示。它提供 4 路转换通道，其中两路用于将 RS-232 电平转换为 TTL 电平，另外两路用于将 TTL 电平转换为 RS-232 信号。一般说来，该芯片需要 4 个外接电容，根据芯片型号的后缀不同，电容的最小值有不同的取值。MAX232 需要外接最小 1μF 的电容，而 MAX232A 只需要接 0.1μF 的电容即可。

进行具体设计时，将 9 芯的 RS-232 的 2 脚与 MAX232 芯片的输出引脚 7 脚 T2out 相连，9 芯的

RS-232 的 3 脚与 MAX232 芯片的输入引脚 8 脚 R2IN 相连，9 芯的 RS-232 的 5 脚接地。

但对于两个单片机直接连接而言，由于两者都是 TTL 电平，则不需要电平转换芯片，不过两者的 TXD 和 RXD 需要相互反接才能正常收发；此外，两者的波特率还需要一致。

4. 通信软件介绍

在完成各硬件连接后，使用友善串口调试助手进行软件调试。友善串口调试助手界面如图 5-6 所示。

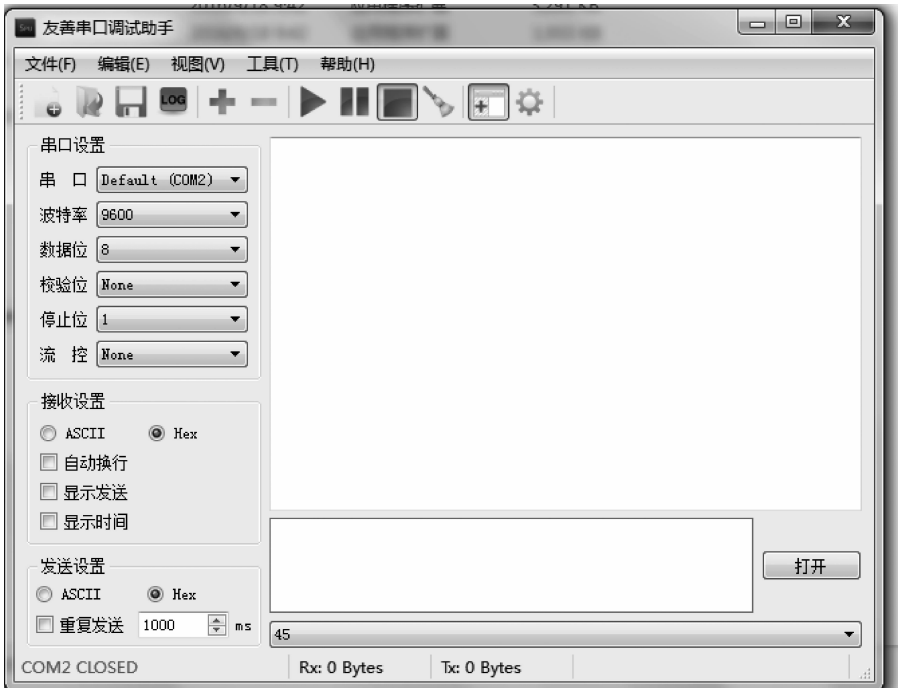

图 5-6　友善串口调试助手界面

当数据线一端接单片机 RS-232 接口，另一端接计算机 USB 端口，单片机上电后，设置正确的串口、波特率等就可以实现串行通信了，注意的是友善串口接收或发送装置的 Hex 就是十六进制数。

5.4　串行通信系统设计

设计举例：打开串行助手软件，设置串口号，十六进制显示以及输入十六进制发送，下位机（单片机）以十进制显示在 12864 上第一行。

硬件电路图如图 5-7 所示。

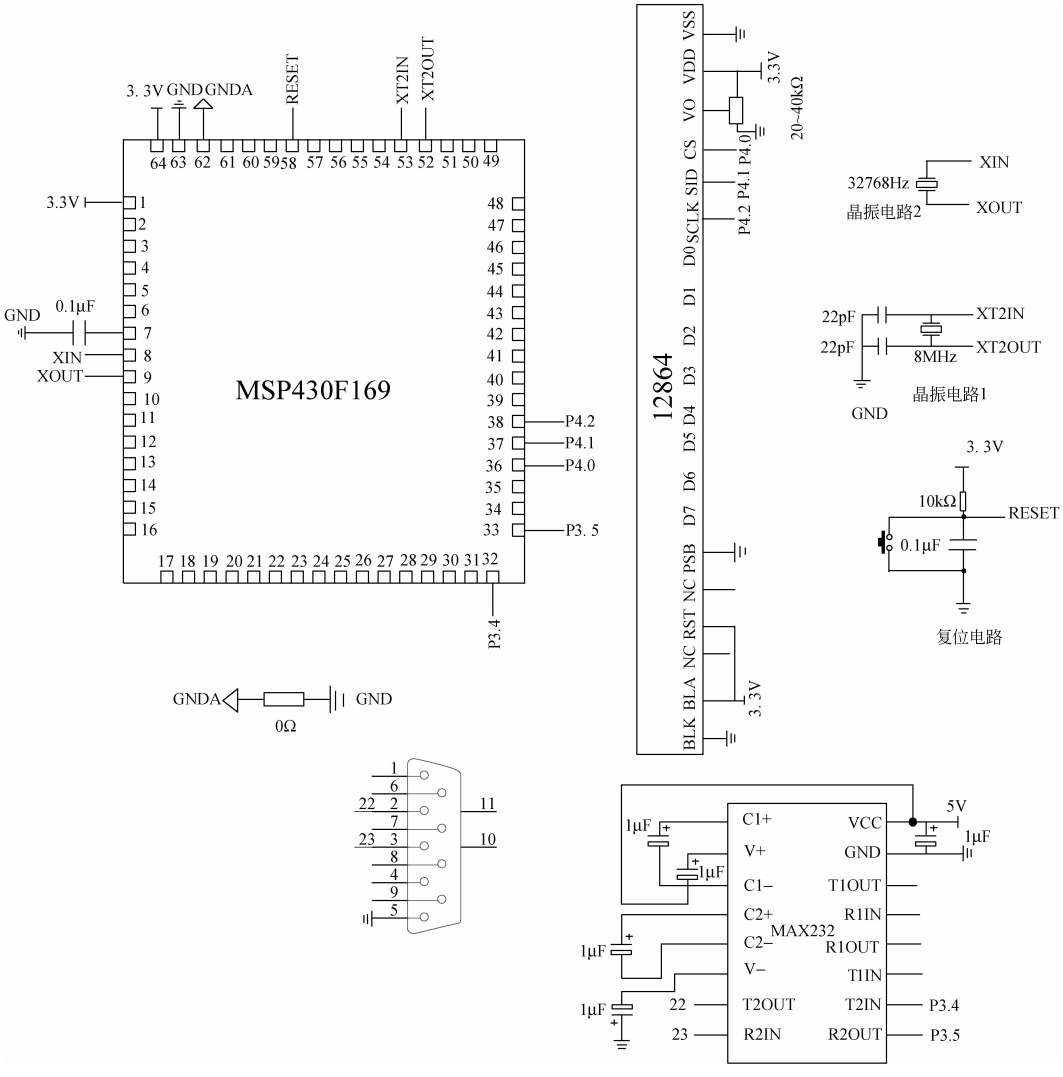

图 5-7 基于单片机的串行通信系统设计电路图

用 ACLK 作为时钟源，波特率为 9600bps，程序清单如下：

```
#include <MSP430x14x.h>
#include "string.h"
#define uint unsigned int
#define uchar unsigned char
#define ulong unsigned long
#define SID BIT1
#define SCLK BIT2
#define CS BIT0
#define LCDPORT P4OUT
```

```
#define SID_1     LCDPORT |= SID          //SID 置高
#define SID_0     LCDPORT &= ~SID         //SID 置低
#define SCLK_1    LCDPORT |= SCLK         //SCLK 置高
#define SCLK_0    LCDPORT &= ~SCLK        //SCLK 置低
#define CS_1      LCDPORT |= CS           //CS 置高
#define CS_0      LCDPORT &= ~CS          //CS 置低
uchar lcd_bus;
uchar table[7] = "       ";
uchar dis_flag;
int data,time1 = 0,k;
void delay(unsigned char ms)
{
    unsigned char i,j;
    for(i = ms;i > 0;i -- )
    for(j = 120;j > 0;j -- );
}

void delay_ms(uint aa)
{
    uint ii;
    for(ii = 0;ii < aa;ii++)
    __delay_cycles(8000);
}

void delay_us(uint aa)
{
    uint ii;
    for(ii = 0;ii < aa;ii++)
    __delay_cycles(8);
}

void sendbyte(uchar zdata)                  //数据传送函数
{
    uint i;
    for(i = 0;i < 8;i++)
    {
       if((zdata << i)&0x80)
       {
          SID_1;
       }
       else
       {
          SID_0;
       }
       delay(1);
       SCLK_0;
       delay(1);
```

```
        SCLK_1;
        delay(1);
    }
}
/*************************************************************
* 名    称:LCD_Write_cmd()
* 功    能:写一个命令到 LCD12864
* 入口参数:cmd 为待写入的命令,无符号字节形式
* 出口参数:无
* 说    明:写入命令时,RW = 0,RS = 0 扩展成 24 位串行发送
* 格    式:11111 RW0 RS 0 xxxx0000 xxxx0000
*           |最高的字节 |命令的 bit7～4|命令的 bit3～0|
*************************************************************/
void write_cmd(uchar cmd)
{
    CS_1;
    sendbyte(0xf8);
    sendbyte(cmd&0xf0);
    sendbyte((cmd<<4)&0xf0);
}

/*************************************************************
* 名    称:LCD_Write_Byte()
* 功    能:向 LCD12864 写入一个字节数据
* 入口参数:byte 为待写入的字符,无符号形式
* 出口参数:无
* 范    例:LCD_Write_Byte('F') //写入字符'F'
*************************************************************/

void write_dat(uchar dat)
{
    CS_1;
    sendbyte(0xfa);
    sendbyte(dat&0xf0);
    sendbyte((dat<<4)&0xf0);
}
/*************************************************************
* 名    称:LCD_pos()
* 功    能:设置液晶的显示位置
* 入口参数:x 为第几行,1～4 对应第 1 行～第 4 行
*          y 为第几列,0～15 对应第 1 列～第 16 列
* 出口参数:无
* 范 例:LCD_pos(2,3) //第 2 行,第 4 列
*************************************************************/
void lcd_pos(uchar x,uchar y)
{
    uchar pos;
```

```
    if(x == 0)
    {x = 0x80;}
    else if(x == 1)
    {x = 0x90;}
    else if(x == 2)
    {x = 0x88;}
    else if(x == 3)
    {x = 0x98;}
    pos = x + y;
    write_cmd(pos);
}
/ ************************************************** /
//LCD12864 初始化
void LCD_init(void)
{
    write_cmd(0x30);                    //基本指令操作
    delay(5);
    write_cmd(0x0C);                    //显示开,关光标
    delay(5);
    write_cmd(0x01);                    //清除 LCD 的显示内容
    delay(5);
    write_cmd(0x02);                    //将 AC 设置为 00H,且游标移到原点位置
    delay(5);
}

void display(int num)
{
    if(num <= 9&&num > 0)
    {
        dis_flag = 1;
        table[0] = num % 10 + '0';
        table[1] = ' ';
    }
    for(k = 0;k < 2;k++)
    {
        lcd_bus = table[k];
        write_dat(lcd_bus);
        delay(5);
    }
    else if(num <= 99&num > 9)
    {
        dis_flag = 2;
        table[0] = num/10 + '0';
        table[1] = num % 10 + '0';
        for(k = 0;k < 2;k++)
        {
            lcd_bus = table[k];
```

```
            write_dat(lcd_bus);
            delay(5);
        }
    }
    else if(num<=999&num>99)
    {
        dis_flag=3;
        table[0]=num/100+'0';
        table[1]=num/10%10+'0';
        table[2]=num%10+'0';
        for(k=0;k<3;k++)
        {
            lcd_bus=table[k];
            write_dat(lcd_bus);
            delay(5);
        }
    }
    else if(num<=9999&num>999)
    {
        dis_flag=4;
        table[0]=num/1000+'0';
        table[1]=num/100%10+'0';
        table[2]=num/10%10+'0';
        table[3]=num%10+'0';
    }
    for(k=0;k<4;k++)
    {
        lcd_bus=table[k];
        write_dat(lcd_bus);
        delay(5);
    }
  }
}

#pragma vector = UART0RX_VECTOR
__interrupt void UART0_RXISR(void)
{
    data=RXBUF0;
    time1=data;
    IFG1&=~URXIFG0;
}

int main( void )
{
    WDTCTL = WDTPW + WDTHOLD;
    P4OUT=0x0f;
    P4DIR = BIT0 + BIT1 + BIT2;
```

```
    LCD_init( );
    P3SEL | = 0x30;                          // 选择 P3.4 和 P3.5 作 UART 通信端
    ME1 | = UTXE0 + URXE0;                   // 使能 USART0 的发送和接收
    UCTL0 | = CHAR;                          // 选择 8 位字符
    UTCTL0 | = SSEL0;                        // UCLK = ACLK
    UBR00 = 0x03;                            // 波特率 9600
    UBR10 = 0x00;                            //
    UMCTL0 = 0x4A;                           // 波特率修正
    UCTL0 &= ~SWRST;                         // 初始化 UART 状态机
    IE1 | = URXIE0;                          // 使能 USART0 的接收中断
    IFG1& = ~ URXIFG0;
    _EINT( );
    delay(40);
    while(1)
    {
        while (!(IFG1 & URXIFG0));
        lcd_pos(0,0);
        display(time1);
    }
}
```

因为是采用 ACLK 作为时钟源，其频率为 32768Hz，设置波特率为 9600bps，则 32768/9600＝3.4133，整数部分为 3，故 UBR00＝0x03；UBR10＝0x00；小数部分为 0.4133，则 0.4133×8＝3.3，取整即为 3。即说明 UMCTL0 寄存器必须有 3 个 1，其他均为 0。把 1 分散排列，故 UMCTL0＝0x4A 。

现在看串行接收中断函数 #pragma vector = UART0RX_VECTOR 的含义。

```
__interrupt void UART0_RXISR(void)
{
    data = RXBUF0;
    time1 = data;
    IFG1& = ~ URXIFG0;
}
```

发生中断时，寄存器 RXBUF0 把收到的数据传给 data，后又传给 time1，然后清除接收中断标志位(fIFG1&＝～ URXIFG0)。在主函数中 while (! (IFG1 & URXIFG0))看串口 0 接收数据是否完成，若未完成，则继续等待。若完成，则显示收到的数据。

现用 SMCLK 作为时钟源，波特率为 9600bps，程序清单如下(为节省篇幅，程序相同部分省略)：

```
#include <MSP430x14x.h>
#define uint unsigned int
#define uchar unsigned char
#define ulong unsigned long
#define SID BIT1
```

```
#define SCLK BIT2
#define CS BIT0
#define LCDPORT P4OUT
#define SID_1     LCDPORT |= SID            //SID 置高
#define SID_0     LCDPORT &= ~SID           //SID 置低
#define SCLK_1    LCDPORT |= SCLK           //SCLK 置高
#define SCLK_0    LCDPORT &= ~SCLK          //SCLK 置低
#define CS_1      LCDPORT |= CS             //CS 置高
#define CS_0      LCDPORT &= ~CS            //CS 置低
uchar lcd_bus;
uchar table[7] = "     ";
uchar dis_flag;
int k,data,time = 0;;
void delay(unsigned char ms)
{
    …
}

void Clk_Init( )
{
    unsigned char i;
    BCSCTL1& = ~XT2OFF;                    //打开 XT2 振荡器
    BCSCTL2| = SELS;
    BCSCTL2| = DIVS0 + DIVS1;
    do
    {
        IFG1 &= ~OFIFG;                    //清除振荡错误标志
        for(i = 0; i < 0xff; i++) _NOP();  //延时等待
    }
    while ((IFG1 & OFIFG) != 0);           //如果标志为 1,继续循环等待
    IFG1& = ~OFIFG;
}

void delay_ms(uint aa)
{
    …
}

void delay_us(uint aa)
{
    …
}
void sendbyte(uchar zdata)                 //数据传送函数
{
    …
}
```

```
void write_cmd(uchar cmd)
{
    …
}
void write_dat(uchar dat)
{
    …
}

void lcd_pos(uchar x,uchar y)
{
    …
}

void LCD_init(void)
{
    …
}
void display(int num)
{
    …
}

#pragma vector = UART0RX_VECTOR
__interrupt void UART0_RXISR(void)
{
    data = RXBUF0;
    time = data;
    IFG1& = ~ URXIFG0;
}

int main( void )
{
    WDTCTL = WDTPW + WDTHOLD;
    Clk_Init();                          //时钟初始化
    P4OUT = 0x0f;
    P4DIR = BIT0 + BIT1 + BIT2;
    LCD_init();
    P3SEL | = 0x30;                      //选择 P3.4 和 P3.5 作 UART 通信端
    ME1 | = UTXE0 + URXE0;               //使能 USART0 的发送和接收
    UCTL0 | = CHAR;                      //选择 8 位字符
    UTCTL0 | = SSEL1;                    //UCLK = ACLK
    UBR00 = 0x68;                        //波特率为 9600
    UBR10 = 0x00;
    UMCTL0 = 0x02;                       //Modulation
    UCTL0 & = ~SWRST;                    //初始化 UART 状态机
    IE1 | = URXIE0;                      //使能 USART0 的接收中断
```

```
    IFG1&=~URXIFG0;
    _EINT( );
    delay(40);
    while(1)
    {
        while (!(IFG1 & URXIFG0));
        lcd_pos(0,0);
        display(time);
    }
}
```

因为是采用 SMCLK 作为时钟源，其频率为 8MHz，8 分频（BCSCTL2|=DIVS0+DIVS1）后为 1MHz。设置波特率为 9600bps，则 1000 000/9600=104.166，整数部分为 104，故 UBR00=0x68；UBR10=0x00；小数部分为 0.166，则 0.166×8=1.328，取整即为 1。即说明 UMCTL0 寄存器必须有 1 个 1，其他均为 0，故 UMCTL0=0x02。

现用 SMCLK 作为时钟源，波特率为 38400bps，程序清单如下。为节省篇幅，程序相同部分省略。

```
#include<MSP430x14x.h>
#define uint unsigned int
#define uchar unsigned char
#define ulong unsigned long
#define SID BIT1
#define SCLK BIT2
#define CS BIT0
#define LCDPORT P4OUT
#define SID_1     LCDPORT|=SID            //SID 置高
#define SID_0     LCDPORT&=~SID           //SID 置低
#define SCLK_1    LCDPORT|=SCLK           //SCLK 置高
#define SCLK_0    LCDPORT&=~SCLK          //SCLK 置低
#define CS_1      LCDPORT|=CS             //CS 置高
#define CS_0      LCDPORT&=~CS            //CS 置低
uchar lcd_bus;
uchar table[7]="     ";
uchar dis_flag;
int k,data,time=0;;
void delay(unsigned char ms)
{
    …
}

void Clk_Init( )
{
    unsigned char i;
    BCSCTL1&=~XT2OFF;                    //打开 XT2 振荡器
    BCSCTL2|=SELS;
```

```
    do
    {
      IFG1 &= ～OFIFG;                        //清除振荡错误标志
        for(i = 0; i < 0xff; i++) _NOP();     //延时等待
    }
    while ((IFG1 & OFIFG) != 0);              //如果标志为 1,继续循环等待
    IFG1&= ～OFIFG;
}

void delay_ms(uint aa)
{
    …
}

void delay_us(uint aa)
{
    …
}

void sendbyte(uchar zdata)                    //数据传送函数
{
    …
}

void write_cmd(uchar cmd)
{
    …
}

void write_dat(uchar dat)
{
    …
}

void lcd_pos(uchar x,uchar y)
{
    …
}

void LCD_init(void)
{
    …
}

void display(int num)
{
    …
```

```
}

#pragma vector = UART0RX_VECTOR
__interrupt void UART0_RXISR(void)
{
    data = RXBUF0;
    time = data;
    IFG1& = ~ URXIFG0;
}

int main( void )
{
    WDTCTL = WDTPW + WDTHOLD;
    Clk_Init( );                          //时钟初始化
    P4OUT = 0x0f;
    P4DIR = BIT0 + BIT1 + BIT2;
    LCD_init( );
    P3SEL | = 0x30;                       // 选择 P3.4 和 P3.5 作 UART 通信端
    ME1 | = UTXE0 + URXE0;                // 使能 USART0 的发送和接收
    UCTL0 | = CHAR;                       // 选择 8 位字符
    UTCTL0 | = SSEL1;                     // UCLK = smclk
    UBR00 = 0xd0;                         // 波特率为 38400
    UBR10 = 0x00;
    UMCTL0 = 0x92;                        // Modulation
    UCTL0 & = ~SWRST;                     // 初始化 UART 状态机
    IE1 | = URXIE0;                       // 使能 USART0 的接收中断
    IFG1& = ~ URXIFG0;
    _EINT( );
    delay(40);
    while(1)
    {
        while (!(IFG1 & URXIFG0));
        lcd_pos(0,0);
        display(time);
    }
}
```

因为是采用 SMCLK 作为时钟源，其频率为 8MHz，设置波特率为 38400bps，则 8000000/38400＝208.33，整数部分为 208，故 UBR00＝0xd0；UBR10＝0x00；小数部分为 0.33，则 0.33×8＝2.64，四舍五入取整为 3。即说明 UMCTL0 寄存器必须有 3 个 1，其他均为 0。故 UMCTL0＝0x92。

请读者思考一下，如果仍采用 SMCLK 的 8MHz 作为时钟源，不分频，设置波特率为 9600bps，寄存器 UBR00、UBR10 和 UMCTL0 如何设置，这里提示一下，把 UBR00 装满，剩余部分装入 UBR10 即可。

图 5-8 是把串行助手软件设定的数据（十六进制）发送到 12864 显示屏上（十进制）。

图 5-8　基于单片机的串行通信系统设计实验结果