微机单片机串口通信技术

北京航空航天大学
xia jie

2020年2月

内容

1. 通信协议、串口通信原理
   - 1.1 通信协议
   - 1.2 串口通信原理
     - RS-232C标准（协议）美国电子工业协会推荐标准
2. 基于串口助手进行串口通信
3. ATmega128的通用同/异步串行口USART

1.1 通信协议

- 通信协议
- 是指通信双方的一种约定。约定包括对数据格式、同步方式、传送速度、传送步骤、检纠错方式以及控制字符定义等问题做出统一规定，通信双方必须共同遵守。

软件协议中的串行通信协议

① 异步通信协议——起止式异步协议

- 是一个字符一个字符传输，并且传送一个字符总是以起始位开始，以停止位结束，字符之间没有固定的时间间隔要求。
- 异步通信是按字符传输的，每传输一个字符，就用起始位来通知收方，以此来重新核对收发双方同步。
- 异步双方不需要共同的时钟，也就是接收方不知道发送方什么时候发送。
- 特点：简单、点对点、通信效率低

② 同步通信协议

- 有面向字符的和面向比特的两种协议
- 同步就是双方有一个共同的时钟，当发送时，接收方同时准备接收。
- 特点：复杂、点对多点、通信效率高

我们采用：RS-232串口通信协议

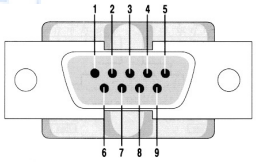
1.2 串口通信原理

- 串行接口
  - ❖将接收来自CPU的并行数据字符转换为连续的串行数据流发送出去
  - ❖将接收的串行数据流转换为并行的数据字符供给CPU的器件

TXD(pin 3):串口数据输出

RXD(pin 2):串口数据输入

GND(pin 5): 地线



25针—〉9针—〉3针

物理接口标准中 串行通信接口的基本任务:

a) 实现数据格式化
b) 进行串－并转换
c) 控制数据传输速率
d) 进行错误检测
e) 进行TTL与EIA电平转换
f) 提供EIA-RS-232C接口标准所要求的信号线

RS—232C

EIA 电平：

-3～-25V 电平表示逻辑“1”

+3～+25V 电平表示逻辑“0”

TTL 电平：

+5V 电平表示逻辑“1”

0V 间电平表示逻辑“0”

计算机内部流动的信号

2.基于串口助手进行串口通信

2.1 版本1——串口助手V2.2.exe

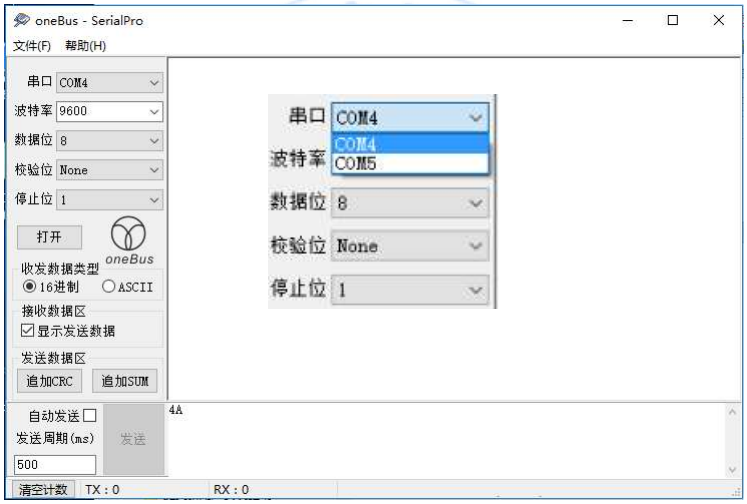
2.2 版本2——串口助手SerialPro.exe

2.1版本1——串口助手V2.2.exe

- 特点：只能COM1~COM4

2.2 版本2——串口助手SerialPro.exe

- 特点：自动探测使用的串口号



3. ATmega128的通用同/异步串行口USART

3.1 特点、构成(2个)： USART0和USART1

3.2 对应状态和控制寄存器

3.3 USART 初始化

3.4 发送数据－ USART 发送器

3.5 接收数据－ USART 接收器

3.6 程序收发串行数据的方式

3.7 如何发送10位数据

3.1 特点、构成
通用同/异步串行口USART

- USART是一个高度灵活的串行通信设备，特点：
  - 全双工操作（独立的串行接收和发送寄存器）
  - 异步或同步操作
  - 高精度的波特率发生器
  - 支持5、6、7、8、9个数据位和1、2个停止位
  - 支持奇偶校验操作、数据过速检测、帧错误检测
  - 3个独立中断：发送结束中断、发送寄存器空中断、接收结束中断
  - 倍速异步通信模式
- 有2个USART： USART0和USART1，分别有不同的I/O寄存器

3.2 对应状态和控制寄存器

① I/O 数据寄存器：UDR0、UDR1
② 控制和状态寄存器A：UCSR0A、 UCSR1A
③ 控制和状态寄存器B：UCSR0B、 UCSR1B
④ 控制和状态寄存器C：UCSR0C、 UCSR1C
⑤ USART波特率寄存器：
- UBRR0L和UBRR0H
- UBRR1L和UBRR1H

① I/O 数据寄存器：UDR0、UDR1

Bit	7	6	5	4	3	2	1	0	
	RXBn[7:0]								UDRn (读)
	TXBn[7:0]								UDRn (写)
读 / 写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

- 写入UDR ——发送
  - 只有当UCSRA寄存器的UDRE标志置位后才可以对发送缓冲器进行写操作。否则写入UDR 的数据会被USART发送器忽略。
- 读UDR ——接收
  - 接收缓冲器包括一个两级FIFO，一旦接收缓冲器被寻址FIFO 就会改变它的状态。因此不要对这一存储单元使用读- 修改- 写指令(SBI 和CBI)。使用位查询指令(SBIC和SBIS)时也要小心，因为这也有可能改变FIFO 的状态。

② 控制和状态寄存器A －UCSRnA

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
读 / 写	R	R/W	R	R	R	R	R/W	R/W	
初始值	0	0	1	0	0	0	0	0	

③ 控制和状态寄存器B －UCSRnB

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
读 / 写	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
初始值	0	0	0	0	0	0	0	0	

④ 控制和状态寄存器C －UCSRnC

Bit	7	6	5	4	3	2	1	0	
	–	UMSELn	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
读 / 写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	1	1	0	

⑤ USART波特率寄存器

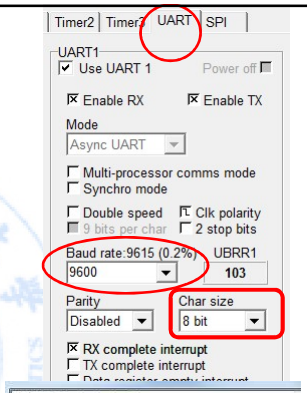
Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
读 / 写	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

串行数据帧格式（注意要与上位机一致）

- USART接收以下30种组合的数据帧格式
  1. 1个起始位
  2. 5、6、7、8、9个数据位
  3. 无校验位、奇校验或偶校验位
  4. 1个或2个停止位
- ❖ 数据帧的结构由寄存器UCSEB和UCSRC中的UCSZ2：0、UPM1：0与USBS（停止位选择）设定，接收与发送使用相同的设置。

3.3 USART 初始化

- 进行通信之前首先要对USART 进行初始化：
  - ❖波特率的设定
  - ❖帧结构的设定
  - ❖以及根据需要使能接收器或发送器。
  - ❖对于中断驱动的USART操作，在初始化时首先要清零全局中断标志位(全局中断被屏蔽)。
- 重新改变USART 的设置应该在没有数据传输的情况下进行




```
//UART1 initialize
// desired baud rate:9600
// actual baud rate:9615 (0.2%)
// char size: 8 bit
// parity: Disabled
void uart1_init(void)
{
 UCSR1B = 0x00; //disable while set
 UCSR1A = 0x00;
 UCSR1C = 0x06;
 UBRR1L = 0x67; //set baud rate lo
 UBRR1H = 0x00; //set baud rate hi
 UCSR1B = 0x98;
}

#pragma interrupt_handler uart1_rx_
void uart1_rx_isr(void)
{
 //uart has received a character in
}
```

3.4 发送数据
以5~8 个数据位的方式发送帧

- 将需要发送的数据加载到发送缓存器UDR将启动数据发送。加载过程为CPU 对UDR 寄存器的写操作

```
void USART_Transmit( unsigned char data )
{
   // 若发送缓冲器为空，将数据放入缓冲器，发送数据
   while ( !( UCSR0A & (1<<UDRE0)) ) ;
   UDR0 = data;
}
```

- 本代码假定已经包含了合适的头文件
- 这个程序只是在载入新的要发送的数据前，通过检测UDRE 标志等待发送缓冲器为空。

3.5 接收数据 (以5~8 个数据位的方式接收帧)

- 一旦接收器检测到一个有效的起始位，便开始以所设定的波特率进行接收，直到收到一帧数据的第一个停止位。接收到的数据被送入接收缓冲器中。通过读取UDR 就可以获得接收缓冲器的内容的。

采用查询方式

```
unsigned char USART_Receive( void )
{
   while ( !(UCSR0A & (1<<RXC0)) ); /* 等待接收数据/
   return UDR0; /* 从缓冲器中获取并返回数据*/
}
```

```
#pragma interrupt_handler
uart1_rx_isr:iv_USART1_RXC
void uart1_rx_isr(void)
{
    dataIn= UDR1;
}
```

高版本

采用中断方式

```
//UART1 initialize
// desired baud rate:9600
// actual baud rate:9615 (0.2%)
// char size: 8 bit
// parity: Disabled
void uart1_init(void)
{
 UCSR1B = 0x00; //disable while set
 UCSR1A = 0x00;
 UCSR1C = 0x06;
 UBRR1L = 0x67; //set baud rate lo
 UBRR1H = 0x00; //set baud rate hi
 UCSR1B = 0x98;
}

#pragma interrupt_handler uart1_rx_
void uart1_rx_isr(void)
{
 //uart has received a character in
}
```

- 本代码假定已经包含了合适的头文件

采用中断方式接收数据的C 代码例程

```
#pragma interrupt_handler uart0_rx_isr:19
void uart0_rx_isr(void)
{  //uart has received a character in UDR
    n_in=UDR0;
}
```

低版本

- 本代码假定已经包含了合适的头文件

- 注意：
  - 每次只能读取一个字节
  - 执行本段代码之前首先要初始化USART。

3.6 程序收发串行数据的方式

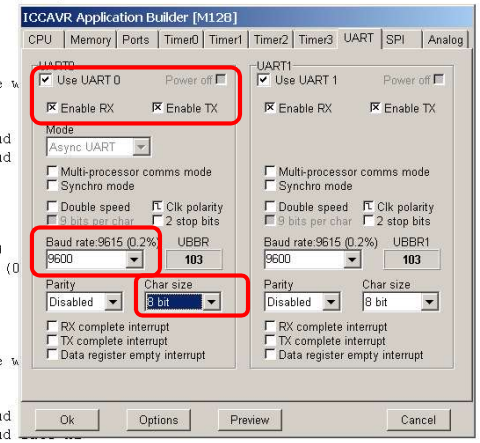
- 查询法：缺陷——MCU不可干别的事
- 中断法：在对应中断服务子程序中进行
  - 应注意中断初始化
- 同时需要串口收发的可行方法：
  1. 发送采用查询法，接收采用中断法
     - 查询USR中的UDRE位
  2. 接收和发送均采用中断法。
     - 发送的数据寄存器空中断方式：可能会没完没了
     - 发送的发送结束中断方式
     - 接收的接收中断方式：必须将数据读出，否则也可能没完没了

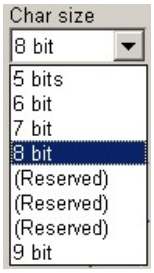
UART串口菜单

- 选用以及设置串口模式、通信波特率、数据位、RX和TX是否中断等。

```
//UART0 initialize
// desired baud rate: 9600
// actual: baud rate:9615 (0.2%)
// char size: 8 bit
// parity: Disabled
void uart0_init(void)
{
 UCSR0B = 0x00; //disable w
 UCSR0A = 0x00;
 UCSR0C = 0x06;
 UBRR0L = 0x67; //set baud
 UBRR0H = 0x00; //set baud
 UCSR0B = 0x18;
}

//UART1 initialize
// desired baud rate:9600
// actual baud rate:9615 (0
// char size: 8 bit
// parity: Disabled
void uart1_init(void)
{
 UCSR1B = 0x00; //disable w
 UCSR1A = 0x00;
 UCSR1C = 0x06;
 UBRR1L = 0x67; //set baud
 UBRR1H = 0x00; //set baud
 UCSR1B = 0x18;
}
```





21

3.7 如何发送10位数据

- 方法1
  - ❖数据分成：高5位+低5位
  - ❖串口发送两次将它们发送出去
- 方法2
  - ❖数据分成：高7位+低3位 or高3位+低7位
  - ❖串口发送两次将它们发送出去