汇编语言课程设计报告

远程控制软件 BSER/BCLI

串行通信应用

作者：广东工业大学计算机一系 971 班陈铭华

日期：2000/1/15

目录:

一、设计目标（Aim）

本软件是要运用 x86 汇编语言对 PC 机的串行通信端口进行编程，在通过串行通信线连接的两台 PC 中实现 DOS 环境下，甲机对乙机的控制与监视。

功能包括：

由甲机发出命令，对乙机进行操作；

在甲机中能够观察到乙机的所有命令行操作及操作后的输出结果。

二、基本概念（Basic Concepts）

1、UART

Universal Asynchronous Receiver/Transmitter 通用异步接收/发送器，用于控制 RS-232 串行通信接口工作。现在大多数的 PC 使用的芯片是 16550，而在早期的机器中，对应的芯片是 8250 或 16450，16550 在引脚和软件上与它们是兼容的。16550 有多个 8 位的寄存器，用于各种控制参数的设置、数据的发送接收、状态反馈等功能。在程序中可通过不同的 I/O 端口对相应的寄存器进行存取。这些寄存器的作用以及它们的端口号列表如下：

端口	说　明
3F8/2F8	THR/RBR - 发送、接收数据用的缓冲区 (read/write)
	LSB - 当LCR的第7位为1时，这也是用来设置波特率的除数因子的低位字节 (read/write)
3F9/2F9	IER - 中断允许寄存器 (read/write)
	MSB - 当LCR的第7位为1时，这是波特率除数因子的高位字节 (read/write)
3FA/2FA	IIR - 中断识别寄存器 (read only)
	FCR - 16550的FIFO队列的控制寄存器 (write only)
3FB/2FB	LCR - 线路（串行通信口）控制寄存器 (read/write)
3FC/2FC	MCR - Modem 控制寄存器 (read/write)
3FD/2FD	LSR - 线路状态寄存器 (read only)
3FE/2FE	MSR - Modem 状态寄存器 (read only)
3FF/2FF	Scratch Pad Register 这并不用于串行I/O (read/write)

这里，端口 3Fx 是 PC 中串口 1 缺省的端口号，而 2Fx 则是对应串口 2。

各个寄存器的详细用法参见附录 3。

2、BIOS 数据区

PC 中内存最低端的 1.5k 字节(0000:0000h-0000:05FFh)由 BIOS 保留，其中开头的 1024 个字节作为中断向量表，而剩下的由地址 0000:0400h 开始的 512 个字节则用来存放 BIOS 的一些重要数据和系统信息。其中，0000:0400h 开始的 4 个字(8 个字节)的内容存放了 4 个串口(从 COM1 到 COM4)的基地址，即它们所对应的第一个 I/O 端口号。缺省情况下，COM1 对应的是 3F8h，COM2 对应的是 2F8h。

另外，0000:0505h 开始的 250 个字节为保留，BIOS 未用。

3、PSP

Program Segment Prefix 程序段前缀。DOS 在每装入一个程序的时候都会建立一个 256 字节的数据区，称为 PSP。对于一个 COM 文件，PSP 就在代码段的开头，即 cs:0000h-cs:00FFh 处。程序执行时的命令行参数就存放在 PSP 中偏移为 81h 开始的 127 个字节的缓冲区里，以字符 0Dh 结束。而 PSP 中偏移为 80h 的字节单元则指示了命令行参数的长度。

三、实现说明（Implement Description）

为了要实现前面所提出的目标，软件由两个部分组成。一个为服务器端程序(BSER)，它将运行在被控制的机器上，负责接收从串行端口送来的命令并且执行，然后再把输出送回到串口。另一个为客户端程序(BCLI)，运行在作为控制端的机器上，通过与服务器端程序通信，对被控机进行控制与监视。

(一)、BSER.COM

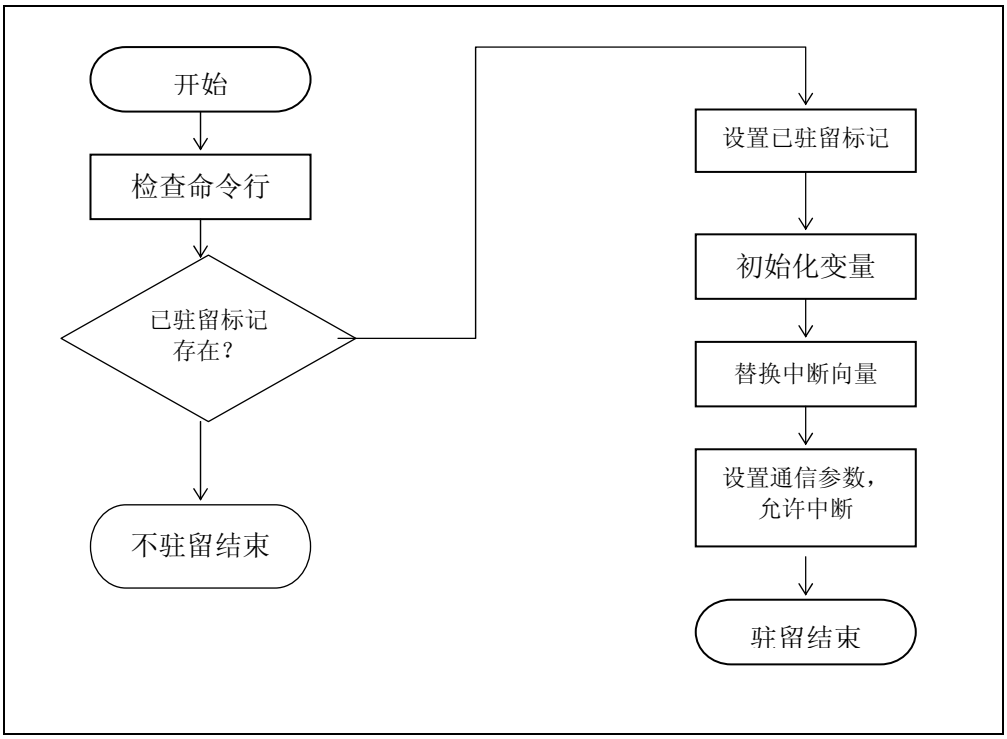
为了要在后台对被控机进行控制并且截取输出，BSER 是一个 TSR 内存驻留程序。考虑到对内存空间的占用，BSER 按 COM 文件的格式进行编写。其基本框架是：

```
cseg    segment
        assume  cs:cseg,ds:cseg
        org 100h
begin:
        jmp     main
        .
        .       ; 数据定义
        .       ; 其它过程定义
        .
main    proc
        .
        .       ; 主程序
        .
        .
main    endp
cseg    ends
        end     begin
```

在这里把主程序 main 放在最后的目的是为了方便计算驻留所需要的空间。main 前面的程序代码是要驻留的，而 main 本身及其后面的代码并不需要驻留，通过 main 的地址来求出前面部分代码的长度，然后驻留退出。这样可以减少不必要的内存开销。

主程序 main 的框架如图：

1、变量说明



程序中主要用到的一些变量如下。它们主要用于存放相应的端口地址，中断向量等。缺省时，它们的值都是基于 COM2 的。

```
comINT   db      0Bh              ;串口对应的中断号，COM2的为0Bh
intMask db       11110111b        ;用于设置中断屏蔽寄存器(21h)时使用的屏蔽值
BaseAdd dw       0402h            ;指向BIOS数据区中存放相应串口基址的地方
                                  ;这里是0000:0402h，存放COM2的基址
THR      dw      ?                ;这些都是指向UART各个寄存器的端口号
RBR      dw      ?                ;根据实际使用的串口来设置相应值
LSB      dw      ?
IER      dw      ?
MSB      dw      ?
IIR      dw      ?
FCR      dw      ?
LCR      dw      ?
MCR      dw      ?
LSR      dw      ?
MSR      dw      ?
org_intCom       dd      ?        ;保存串口中断的原入口
org_int21        dd      ?        ;DOS功能调用(int 21h)的入口地址
```

2、检查命令行

这部分由 param 过程进行处理。程序直接对 PSP 进行访问来获取命令行。从 PSP 的偏移 81h 开始逐个字符进行检查，遇到 1、2、I 等关键字时则修改相关变量值，直至命令行结束。

1 代表使用 COM1，2 代表使用 COM2，I 代表不进行已驻留标记的检查而强制驻留。

3、检查已驻留标记

BSER 利用 BIOS 数据区中的保留空间设置了已驻留标记。如果 0000:05FEh 和 05FFh 两个字节的

内容为'MW'则说明之前已经有一个 BSER 驻留了，程序将不再驻留而直接退出。调用 DOS 功能 4Ch，返回码为 1。若没有'MW'标记，则先设置标记，再继续后面的程序。

4、初始化变量

这里主要是对前面提到的各个指向 UART 寄存器的变量(THR、RBR、LCR 等)赋值。

5、替换中断向量

BSER 要使用自己的程序代替原来的串口中断处理程序和 DOS 的 INT 21h 处理程序，以到达接收命令和截取输出的目的。替换中断向量使用了 DOS 功能 25h 和 35h。这里定义了两个宏：

```
SetVect macro num,handler
        mov     dx,offset handler
        mov     ah,25h
        mov     al,num
        int     21h
        endm
GetVect macro num,handler
        mov     ah,35h
        mov     al,num
        int     21h
        mov     word ptr cs:handler+2,es
        mov     word ptr cs:handler,bx
        endm
```

这里，num 是中断号，handler 是符号地址，对于 SetVect，handler 是 BSER 自己的中断处理程序入口，对于 GetVect，它则是用来保留原向量的变量名。由于 BSER 是 COM 程序，所以在 SetVect 中没有显式地给 DS 赋值。

6、设置通信参数

通过向 LCR 写入数据可以设置串口的通信参数，包括波特率、数据字长等。LCR 各位的作用为：

```
|7|6|5|4|3|2|1|0|
 |  |  |  |  |  +-+---- 数据字长 00 = 5位，01 = 6位，10 = 7位，11 = 8位
 |  |  |  |  | +------- 停止位长度，  0 = 1 位，1 = 1.5 或 2 位
 |  |  |  |  +-------- 0 = 无校验位，1 = 有校验位
 |  |  |  +---------- 0 = 奇校验，1 = 偶校验
 |  |  +------------ 0 = 禁止校验，1 = 允许校验
 |  +------------- 0 = 无间断，1 = 强制间断
 +---------------- 当为1时用于设置波特率，正常工作时应为0
```

当 LCR 的第 7 位设为 1 时，通过分别向 MSB、LSB 寄存器写入除数因子 K 的高、低位字节来设定通信时的波特率。因子 K 与波特率的关系为：

K=1843200 / (波特率 * 16)

程序中先把 LCR 的第 1、0 位置 11(数据字长 8 位)，第 2 位置 0(1 位停止位)，第 5、4、3 位置 000(无校验)，第 6 位置 0，而第 7 位置 1。然后向 LSB 写 0Ch，向 MSB 写 00h(即 K=000Ch，波特率为 9600)。最后再把 LCR 的第 7 位置回 0。程序如下：

```
mov     dx,LCR                  ;set communication parameters
mov     al,10000011b             ;set bit-7, 8N1
out     dx,al

mov     dx,LSB                  ;K=0ch, 9600Baud
mov     al,0ch
```

```
out     dx,al

mov     dx,MSB
mov     al,0
out     dx,al

mov     dx,LCR
mov     al,00000011b
out     dx,al                   ;reset bit-7
```

7、允许中断

先向 IER 写入引起 UART 中断的原因，然后再设置 PC 的中断屏蔽寄存器(端口 21h)便可令串口工作在中断方式下。IER 中各位的说明如下：

```
|7|6|5|4|3|2|1|0|
  | |  | |  | |  | +--- 1 = 数据到达时中断 (或者是 16550 超时中断)
  | |  | |  | |  +----- 1 = THR 空闲时中断
  | |  | |  | +------ 1 = 线路状态改变时中断
  | |  | |  +-----   1 = MODEM 状态改变时中断
  +-+-+-+-------- 保留 (应为0)
```

*注：要使上面的中断条件起作用，还要把MCR的第3位设为1。并不是所有的参考资料都提及了这点。

在 BSER 中，我们只需要设置成有数据到达时才中断就可以了。这部分程序如下：

```
cli                     ; 由于下面将要对中断的触发条件进行设置，所以先禁止外中断的发生

mov     dx,MCR                  ;enable interrupt
mov     al,00001011b
out     dx,al

mov     dx,IER                  ;enable interrupt
mov     al,00000001b
out     dx,al

in      al,21h                  ;set IntMaskReg
and     al,intMask
out     21h,al

sti
```

8、驻留结束

程序驻留前，先要计算驻留所需要的空间，然后再调用 DOS 功能 31h，驻留并返回 DOS 提示符状态。进行驻留部分的代码为：

```
mov     dx,offset main  ; 取main的偏移地址
mov     cl,4
shr     dx,cl           ; 右移4位，因为驻留所需空间是以节(16字节)为单位的
inc     dx
mov     ax,3100h        ; 调用DOS的31h功能，返回码0
```

```
        int     21h
```

9、串口中断处理程序

当有数据从串口到达(有命令从 BCLI 发送过来)时，就会引发该中断。对于 COM1，该中断号是 0Ch，对于 COM2 它是 0Bh。在程序中，这一部分是 int_com 过程。BSER 对它的要求很简单，就是把接收到的数据仿真成键盘的输入。为此，使用了 BIOS 的键盘服务 INT 16h 的 05h 功能。程序代码如下：

```
int_com proc
        .
        .
        .
        mov     dx,cs:RBR       ;从串口接收数据
        in      al,dx

        mov     ah,05h          ;调用INT 16h的05h功能
        mov     ch,0            ;扫描码设为0，忽略特殊键
        mov     cl,al           ;以刚收到的数据为键的ASCII码
        int     16h

        mov     al,20h           ;发出EOI(中断结束)命令
        out     20h,al
        .
        .
        .
        iret
int_com endp
```

10、INT 21h 中断处理程序

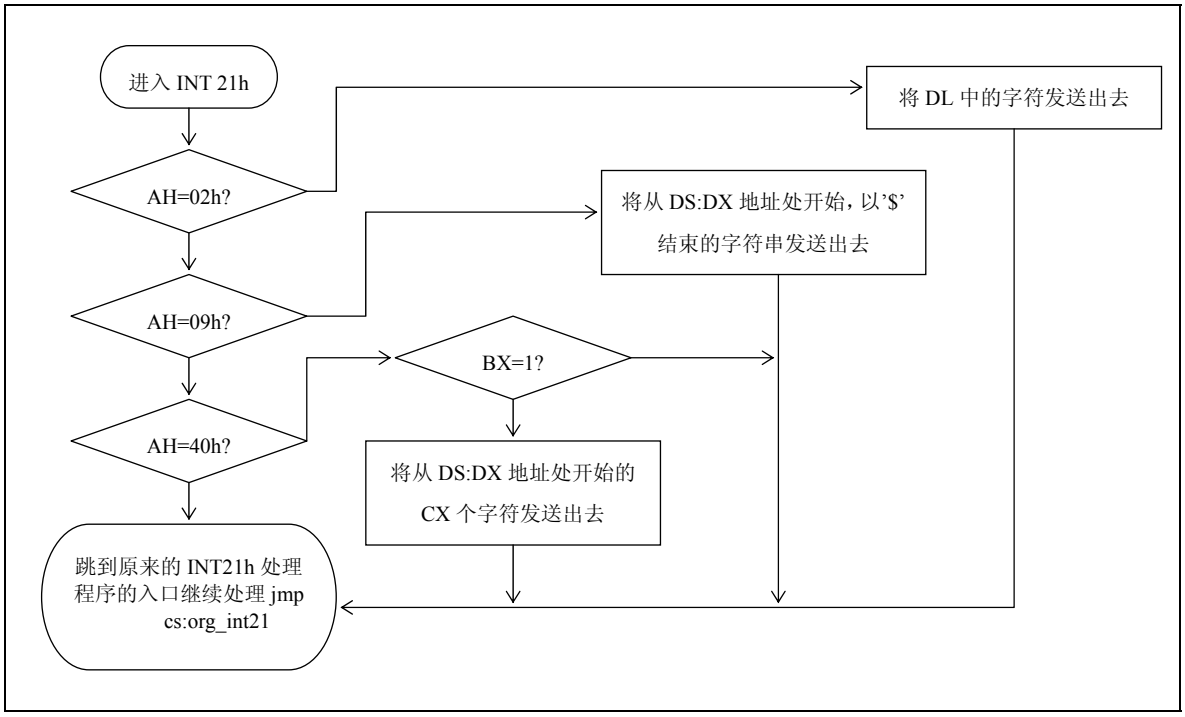
经过分析，大多数的 DOS 应用程序都是通过调用 INT 21h 来进行屏幕输出的。为了要到达截取屏幕输出的目的，就要截取 INT 21h 中向屏幕输出的操作。INT 21h 提供的所有功能中，与屏幕输出有关的是：

02h　　显示输出单个字符；

09h　　显示字符串；

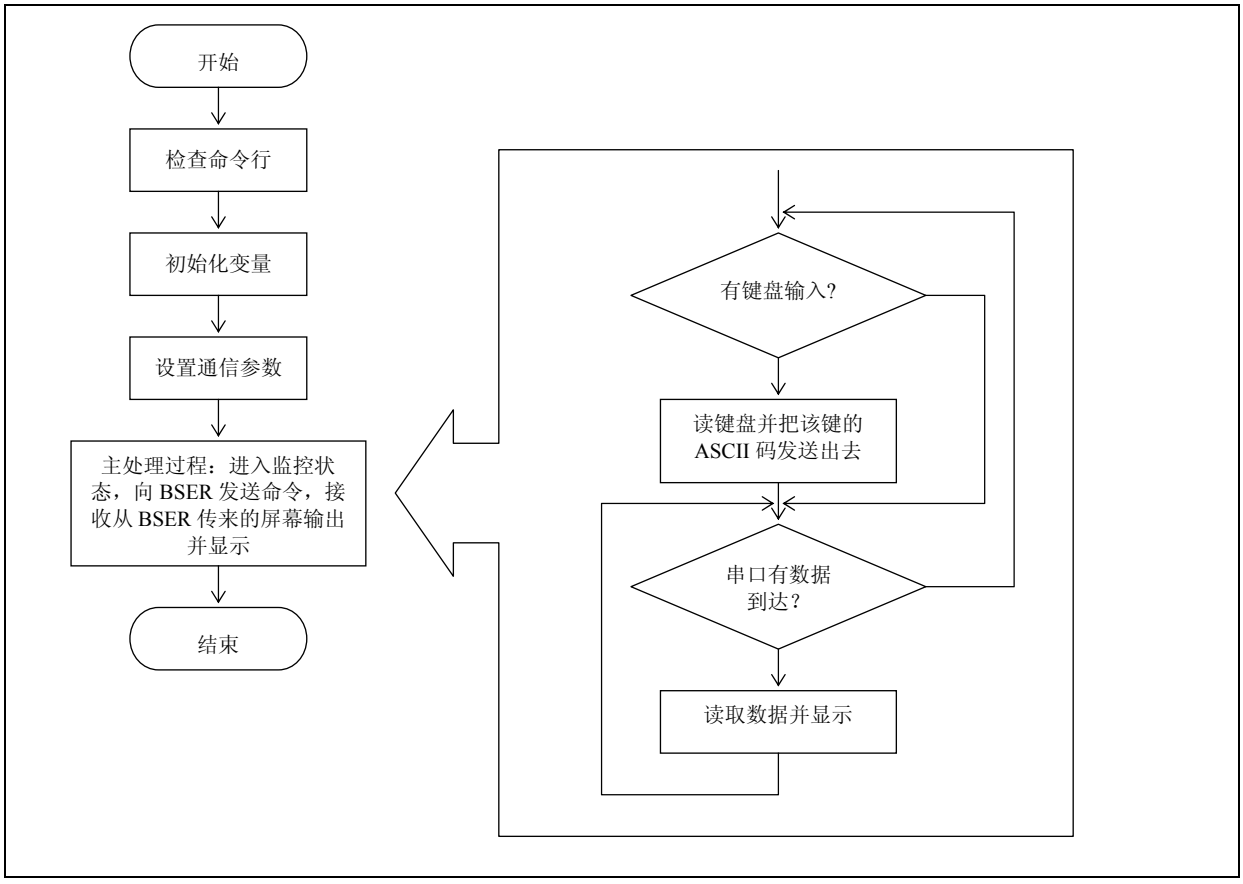
40h　　写文件或设备，当文件句柄为 1 时向 STDOUT 输出，缺省时就是屏幕。

该段程序是过程 int_21，流程为：



(二)、BCLI.COM

BCLI 同样是一个 COM 程序。其基本框架，大部分的变量说明都与 BSER 类似。因为没有修改任何的中断向量，BCLI 显得相对简单很多。它的流程图如下：



四、存在问题（Known Bugs）

1、不能截取”Bad command or file name”**等出错信息**

BSER 在截取 int 12h 的 40h 功能时，只对文件句柄为 1(BX=1)，即 STDOUT 的输出进行截取，而输出”Bad command or file name”等错误信息时，DOS 使用的是 STDERR(文件句柄为 2)。在新版本中已修改。

2、在旧的 DOS 版本中，客户端看不见服务器端的键盘输入

这是由于 Win98 中的 DOS 与之前的版本(包括 Win95)在对命令行输入的处理上有所差异造成的。而本程序是在 Win98 下进行开发的，编写时没有仔细考虑。有待改进。

3、使用三线简易接法的串行线似乎不能正常运行(未经充分证实)

开发程序时使用的是一条完整接线的 9 针串行通信线。引起问题的原因可能是对通信端口进行初始化时激活了 MODEM 的 RTS 和 DTR 信号。见程序段：

```
mov     dx,MCR
mov     al,00001011b        ; 改为 mov al,00001000b 应该可以
out     dx,al
```

最初把MCR这样设置是因为参照了kbfake.c，见附录4。

五、参考资料（References）

1、《深入 DOS 编程》求伯君主编 / 北京大学出版社
2、《IBM-PC 汇编语言程序设计》沈美明、温冬婵 / 清华大学出版社
3、《PC 软硬件技术资料大全》清华大学出版社
4、《COMMAND 结构分析教程》郭嵩山 / 电子工业出版社
4、《对 EXE2BIN.EXE 文件的剖析》王振祥
5、《How to Make a TSR》Jannes Faber / http://www.fys.ruu.nl/~faber/Amain.html
6、《Help PC 2.10》David Jurgens
7、《Norton Guides Database – Assembly Language Database》Peter Norton Computing, Inc.
8、KBFAKE author unknown / Simtel.net

六、补充信息（Additional Info）

暂无。

七、附录（Appendix）

1、 BSER.ASM 源程序
2、 BCLI.ASM 源程序
3、 HELPPC 2.1 中关于 UART 的说明
4、 KBFAKE.C 源程序