EL-308

Microprocessor Based System Design

Talk Program: Chat between two computers over serial port

Term Project Report

Spring 2007

Supervisor: Ayhan Bozkurt

by

E. Selin Baytok 8088

R. Yağız Mungan 8288

1. INTRODUCTION

TALK program project was chosen from various suggested projects in order to have the chance to be introduced to serial communication principles in practice. Expected features of the program to be written are as follows:

- Two computers will be interconnected over RS 232.
- Once the call of the calling party is answered by running the TALK program on the remote computer, both computer screens will be divided into two.
- Keystrokes entered from the local computer will be displayed on the upper half of the screen and sent to the remote machine, while remotely typed characters will appear on the lower part.
- A proper word processing capability including functions of “enter” and “backspace” keys.

2. DESIGN PROCEDURE

2.1 Hardware Design

Communication between two computers was maintained over serial port with RS 232 standard. The cable used to connect these terminals were prepared to be in “null modem” configuration in which modem is not active, allowing the computers to communicate directly. As it is shown in Figure 1, the transmitted data (pin3) and the received data pins (pin 2) of the computers are cross connected while request (pin 7) -clean to send (pin 8) data pins and data terminal (pin 4)-data set ready pins (pin 6) are connected to each other at each side. Finally signal ground pins (pin 5) are connected to each other.

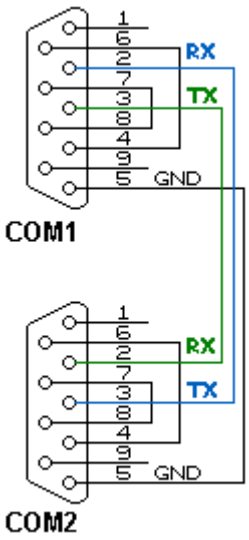

Figure 1: Modem-Nulling Configuration

2.2 Software Design

The software design consists of two parts. The application program will perform the data transfer. And a service routine will notify the user if the other computer requests to “talk”.

2.2.a Application Program

The basic flow of the code will be explained in the following.

- When the program starts it first disables interrupt mechanism
- When a user runs the program TALK.EXE the program must poke the other computer, to indicate that the user wants to talk, the signal used to poke the other computer is then blocked by the program.
- Initialization of the screen mode, and framing for the “talk” screen, and setting the cursor position
- The program will wait if a key is pressed from the keyboard, if a key is pressed the program will do the required action for the key, and send key ASCI to the other computer
- If no key is pressed the code will check if any data has arrived from the other computer, if any the code will do the necessary actions with respect to the character arrived
- Special actions: If the key is “escape” the program will exit. If the key is “backspace” the program will clear the previous character, with never touching the frames. If the character is “enter” the program will move the cursor one row down.

- If any user would fill up the space reserved, the part of the screen reserved will be refreshed, opening up space to continue talking
- Each cursor positions are stored in DB's restoring after each send and receive action. And set to their target values after "enter" and "backspace" action.
- If no key is received the code will loop back for searching if a key is pressed.
- Before exiting program the code enables the interrupts to be further poked, if any user wants to "talk" again.

The code is:

```
        .MODEL SMALL

        .STACK 100h

        .DATA
INTA1           EQU 21H
COM1BASE        EQU 03F8H
LSR             EQU COM1BASE+5
MCR             EQU COM1BASE+4
THR             EQU COM1BASE
RBR             EQU COM1BASE
IER             EQU COM1BASE+1
LCR             EQU COM1BASE+3
out_col DB 2
out_row DB 1
in_col  DB 2
in_row  DB 14


            .CODE

Start:

        mov ax,@Data
        mov ds,ax
```

```
                        ;Disable interrupts when talk program is
                        ;executing

mov dx,INTA1            ;Disable IR4 on 8259
in al,dx
or al,00010000b
out dx,al

mov dx,LCR              ;Set DLAB=0
in al,dx
and al,01111111b
out dx,al

mov dx,IER              ;Disable recevied data ready interrupt
in al,dx
and al,11111110b        ;by D0 of interrupt enable register
out dx,al

mov dx,RBR
mov al,0                ;send 0 to the receive buffer register
out dx,al



mov al,88h              ;dummy start data
call send



mov dx,184fh            ;clear the overall page
call clear


mov ah,00
mov al,03h              ;set video mode to CGA with 80*25 dimensions
int 10h
```

```
        mov ah,02                ;set cursor position to
        mov bh,0
        mov dl,0                 ;column number
        mov dh,12                ;row number
        int 10h


line:                            ;framing for the talk window
        mov ah,02h               ;draw a line on the 12th row
        mov dl,"-"
        int 21h
        inc cx
        cmp cx,38                ;write "TALK" starting from 38th column
        jne line
        inc cx
        mov dl,"T"
        int 21h
        inc cx
        mov dl,"A"
        int 21h
        inc cx
        mov dl,"L"
        int 21h
        inc cx
        mov dl,"K"
        int 21h
line2:                           ;finishes the line, at the 12th row there is a line
                                 ;with
       mov ah,02h                ;"TALK" written at the middle of it
       mov dl,"-"                ;the line is the border of the spaces for the users
       int 21h
       inc cx
       cmp cx,79
       jne line2

      mov ah,02
      mov bh,0
      mov dl,2                   ;col
      mov dh,0                   ;row
```

```
int 10h

mov cx,0                ;on the first row, first 3 columns are empty
mov dl," "              ;this is the start for the upper part of the screen
int 21h
inc cx
mov dl," "
int 21h
inc cx
mov dl," "
int 21h
inc cx

mov dl,"y"              ;after the 3rd column "you said:" is written
int 21h
inc cx
mov dl,"o"
int 21h
inc cx
mov dl,"u"
int 21h
inc cx
mov dl," "
int 21h
inc cx
mov dl,"s"
int 21h
inc cx
mov dl,"a"
int 21h
inc cx
mov dl,"i"
int 21h
inc cx
mov dl,"d"
int 21h
inc cx
mov dl,":"
int 21h
```

```
mov ah,02          ;on the 13th row the lower part of screen begins
mov bh,0
mov dl,2
mov dh,13
int 10h
mov cx,0
mov dl," "
int 21h
inc cx
mov dl," "
int 21h
inc cx
mov dl," "
int 21h
inc cx

mov dl,"h"         ;"he/she said:" appears on the 13th row
int 21h
inc cx
mov dl,"e"
int 21h
inc cx
mov dl,"/"
int 21h
inc cx
mov dl,"s"
int 21h
inc cx
mov dl,"h"
int 21h
inc cx
mov dl,"e"
int 21h
inc cx
mov dl," "
int 21h
inc cx
mov dl,"s"
```

```
    int 21h
    inc cx
    mov dl,"a"
    int 21h
    inc cx
    mov dl,"i"
    int 21h
    inc cx
    mov dl,"d"
    int 21h
    inc cx
    mov dl,":"
    int 21h                    ;the framing ends here




    mov dl,2                   ;set cursor to column 2 row 1 when the
    mov dh,1                   ;program is first initiated
    call set_cursor
wait_key:
    mov ah,01                  ;control whether there is a key stroke
    int 16h
    jz incoming                ;if not check whether a data received or not
    mov ah,0                   ;if there is a key stroke
    int 16h
    cmp al,1Bh                 ;check whether it is "ESC"
    jne go_on
    call escape                ;if it is, quit program

go_on:                         ;other wise print and send the data
    mov dl,out_col             ;go to the last cursor position(column) you left

    mov dh,out_row             ;last cursor position(row)
    call set_cursor
```

```
    mov ah,02h
    mov dl,al                ;print the data to the upper side
    int 21h

    call send                ;also send the data to the remote computer

    cmp al,08h               ;if the data is "backspace"
    jne notback
    call back
    jmp endback              ;call "back" procedure and end this time in the
                             ;loop

notback:
    cmp al,0Dh               ;if data isn't "backspace" but "enter"
    jne skip

    call entered             ;call "entered" procedure

    mov al,0Ah               ;send linefeed to set cursor to the beginning of
                             ;the line
    call send

    mov al,0Dh               ;set al back to ascii code of enter

skip:
    cmp out_row,11           ;if it is the end line of the upper half
    jne normal
    cmp al,0Dh               ;and the new data is "enter"
    je enter1
    cmp out_col,79           ;check whether it is the end of the row
    jne normal

enter1:
    mov dx,0B4fh             ;if it is, clean the upper half
    call clear1
```

```
    mov dl,2                ;and set the cursor back to its initial condition
    mov dh,1
    call set_cursor



normal:
    call get_cursor         ;get the final position of the cursor

    mov out_row,dh          ;store it
    mov out_col,dl
endback:
    jmp wait_key            ;wait for the next stroke



incoming:
    mov ah,03               ;get status of
    mov dx,00               ;COM1 port
    int 14h

    and ah,1                ;check the last bit for a received data
    cmp ah,1
    jne wait_key            ;if not wait for a key stroke


    mov dl,in_col           ;set the cursor to the last place left

    mov dh,in_row
    call set_cursor


    mov ah,02               ;retrieve the received data
    mov dx,0
    int 14h

    cmp al,88h              ;if it is the dummy start signal
    je endback2             ;quit this time in the loop
```

```
        cmp al,08h              ;if received data is "backspace"
        jne notback2
        call back2              ;call back2 procedure and clear the lower half of
                                ;the screen
        jmp endback2

notback2:                       ;if it is not
        mov ah,02               ;display the character in the lower half
        mov dl,al
        int 21h

        cmp in_row,24           ;if it reaches to the end of the 25th column
        jne normal2
        cmp al,0Dh              ;and the received data is enter
        je enter2
        cmp in_col,78
        jne normal2

enter2:
        mov dx,184fh            ;clear the lower half
        call clear2

        mov dl,2                ;set cursor to the initial condition in the lower
                                ;half
        mov dh,14
        call set_cursor

normal2:
        call get_cursor         ;update the cursor variables

        mov in_row,dh
        mov in_col,dl

endback2:

        jmp wait_key            ;wait for a next key stroke


                                ;enable interrupts before quiting
```

```
        mov dx,INTA1            ;enable 8259
        in al,dx
        and al,11101111b
        out dx,al

        mov dx,LCR
        in al,dx
        and al,01111111b        ;Dlab=0
        out dx,al

        mov dx,IER
        in al,dx
        or al,00000001b         ;Interrupt enabled on IER
        out dx,al

        mov dx,MCR
        in al,dx
        or al,00001000b         ;Interrupt enabled on modem
        out dx,al


        mov ah,4Ch
        int 21h


clear   proc                    ;clears the whole screen
        mov ax,0600h            ;before putting the frame
        mov bh,07               ;after leaving and before entering the program
        mov cx,0
        int 10h
        ret
clear   endp
clear1  proc                    ;clears the upper part of the
        mov ax,0600h            ;screen reserved for the user
        mov bh,07               ;does not touch the frame
        mov cx,0100h
        int 10h
        ret
clear1  endp
```

```
clear2  proc                    ;clears the lower part of the
        mov ax,0600h            ;screen reserved for the user
        mov bh,07               ;does not touch the frame
        mov cx,0E00h
        int 10h
        ret
clear2  endp

set_cursor      proc
        mov ah,02               ;sets cursor position
        mov bh,0                ;DH will decide row
        int 10h                 ;DL will decide column

        ret
set_cursor      endp

send            proc
        mov ah,01               ;sends the data in al register
        mov dx,00               ;through serial port COM1
        int 14h
        ret
send            endp

get_cursor      proc
        mov ah,03h               ;get current cursor position
        mov bh,00               ;DH has row
        int 10h                 ;DL has column
        ret
get_cursor      endp

back            proc            ;if backspace is pressed

        cmp out_col,0           ;if the cursor is not at the 0th column of a row
        jne samerow             ;backspace will not cause change of the row
        cmp out_row,1           ;the user is not allowed above the first row
        je skiprow              ;so pressing backspace at 0th column of the first
                                ;row
                                ;does not cause anyhing
```

```
        mov out_col,79          ;if row has to change, the column will be the
                                ;end column
        sub out_row,1           ;and we will go up one row

        mov dl,out_col          ;dl will have the new column number
        mov dh,out_row          ;dh will have the new row number
        call set_cursor         ;for the set_Cursor procedure to apply ;the
                                ;bacskpacing

        mov ah,02               ;displaying 'blank' instead of the character which
        mov dl,0                ;is to be cleared, however ah=02h int 21h moves
        int 21h                 ;the cursor to the next position

        mov dl,out_col          ;column number and
        mov dh,out_row          ;row number is used to put the cursor back
        call set_cursor         ;again using the procedure set_cursor



samerow:
        sub out_col,1           ;if only the column number needs to be changed
        mov dl,out_col          ;column number is decreased and loaded into dl
        mov dh,out_row          ;row number is also loaded into dh for
        call set_cursor         ;for the procedure set_cursor
        mov ah,02               ;displaying 'blank' instead of the character which
        mov dl,0                ;is to be cleared, however ah=02h int 21h moves
        int 21h                 ; the cursor to the next position

        mov dl,out_col          ;column number and
        mov dh,out_row          ;row number are used to put the cursor back
        call set_cursor          ;again using the procedure set_cursor
skiprow:                        ;if no change is needed jumps here
        ret
back    endp


entered     proc
        mov ah,02h              ;enter is applied
        mov dl,0Dh
```

```
        int 21h


        mov dl,0Ah                  ;line feed
        int 21h
        ret
entered         endp

back2           proc                ;works like the procedure back
                                    ;except for the info that is received
        cmp in_col,0
        jne samerow2
        cmp in_row,14
        je skiprow2
        mov in_col,79
        sub in_row,1

        mov dl,in_col               ;col
        mov dh,in_row               ;row
        call set_cursor

        mov ah,02
        mov dl,0
        int 21h

        mov dl,in_col               ;col
        mov dh,in_row               ;row
        call set_cursor



samerow2:
        sub in_col,1
        mov dl,in_col               ;col
        mov dh,in_row               ;row
        call set_cursor
        mov ah,02
        mov dl,0
```

```
        int 21h

        mov dl,in_col           ;col
        mov dh,in_row           ;row
        call set_cursor
skiprow2:
        ret
back2       endp


escape  proc                    ;if esc is pressed enters the code



        mov dx,184fh            ;the screen is cleared to the end
        call clear
        mov dl,2                ;and the cursor is set to
        mov dh,1
        call set_cursor         ;second column of the first row

                                ;enable interrupts before quiting

        mov dx,INTA1            ;enable 8259
        in al,dx
        and al,11101111b
        out dx,al

        mov dx,LCR              ;Dlab=0
        in al,dx
        and al,01111111b
        out dx,al

        mov dx,IER
        in al,dx
        or al,00000001b         ;Interrupt enable on IER
        out dx,al

        mov dx,MCR
        in al,dx                ;Interrupt enabled on modem control
```

```
        or al,00001000b
        out dx,al

        mov dx,184fh                    ;clear the previous notify message
        call clear


        mov ah,4Ch
        int 21h
        ret
escape  endp


        END Start
```

Following figure is a sample example, created by the code above.

Figure 2: Sample screen created by TALK.EXE

2.2.b Interrupt Service Routine Program

The basic of the flow of the code can be summarized as follows:

- The code will first initialize the serial port with the required specifications:
  - No parity
  - Two stop bits
  - Baud rate of 4800bps
  - 8 bit data
- The code will enable the interrupts on 8259, in order to be poked by the other computer.
- The code will wait for the start signal send by the program TALK.EXE, when the signal is found the string “TALK” will be displayed on the upper-right corner of the screen.

,

The code is:

```
.MODEL SMALL

.DATA

INTA0   EQU 20h

INTA1   EQU 21h

EOI     EQU 64h
```

```
COM1BASE EQU 03F8h
LSR     EQU COM1BASE+5
MCR     EQU COM1BASE+4
THR     EQU COM1BASE
RBR     EQU COM1BASE
IER     EQU COM1BASE+1
LCR     EQU COM1BASE+3
        .CODE
Start:
        mov ax,@Data
        mov ds,ax

; Zero ES
        mov ax,0000h
        mov es,ax

;Enable IR4 in 8259
       mov dx,INTA1
       in al,dx
       and al,11101111b
       out dx,al

;Initiate connection
        mov dx,LCR
```

```
mov al,10000000b        ;DLAB=1 ==>Divisor latch activated
out dx,al

mov ax,24               ;Baud rate is set to be 1.8432Mbps/(16*24)=4800bps
mov dx,RBR
out dx,ax

mov dx,LCR
mov al,00000011b        ;No parity,1 stop bit,8 bit data
out dx,al               ;DLAB=0 ==>Divisor latch inactivated

mov dx,IER
in al,dx
or al,00000001b         ;Received data available interrupt enabled
out dx,al


mov dx,MCR
in al,dx
or al,00001000b         ;OUT2 in modem control is set to enable interrupt
out dx,al


mov ax,OFFSET HardInt   ;vector pointing for hardware interrupt
```

```
        mov es:[4*12+0],ax          ;COM1 port invokes an interrupt
        mov ax,cs                   ;with vector number of 12      (4+8)
        mov es:[4*12+2],ax


; Stay resident and terminate
        mov ah,31h
        mov dx,OFFSET Last
        inc dx
        int 21h



HardInt:

        push ds
        push ax
        push dx
        push bx
        push cx
        push es                             ;necessary infos are saved for later use

        mov ax,cs
        mov ds,ax
```

```
mov ch,"k"                    ;show "Talk"at upper right corner
mov dx,0b800h
mov es,dx
mov es:[158],ch

mov ch,"l"
mov dx,0b800h
mov es,dx
mov es:[156],ch

mov ch,"a"
mov dx,0b800h
mov es,dx
mov es:[154],ch

mov ch,"T"
mov dx,0b800h
mov es,dx
mov es:[152],ch




mov dx,INTA0                  ;end of interrupt
mov al,EOI
```

```
        out dx,al


        pop es
        pop cx
        pop bx
        pop dx
        pop ax
        pop ds
        iret

Last:
        END Start
```

The following figure shows a sample screen created by the “terminate and stay resident” program.

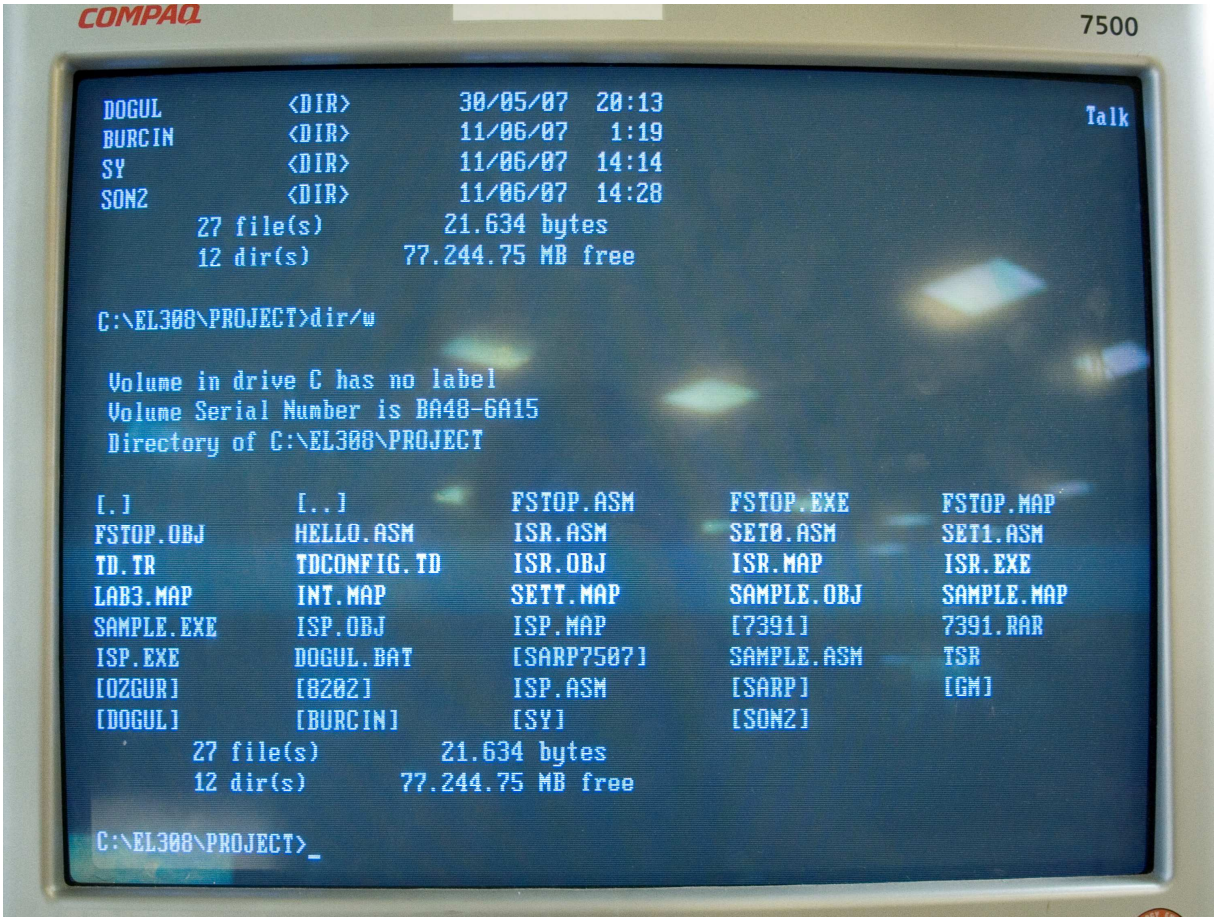

Figure 3: A sample screen created by TSR.EXE