RMV ELECTRONICS INC. **Application Note:**	**Application #:** 00002
	Date: September 1994
Description: High Speed Serial Communications Between the PC and the ITC232-A.	**Status:** Draft Version

The ITC232-A is capable of RS232 communication speeds up to 115200 baud. To enable a PC with a standard 8250 UART to communicate at a sustained rate of 115200 baud, x86 assembly language is used. For transmission software, a polled routine is provided. For reception software, a polled routine and a interrupt driven routine are provided. These routines that are in a single object module may be linked with a users high level program (in the examples provided, C applications). The assembler used is Microsoft's Macro Assembler version 6.0

To configure the communications port, the Setup routine is given parameters such as: baud rate divisor, line control parameters, and port selection. This routine saves a copy of the original parameters so that the communications port can be restored to its original state. If the user desires interrupt driven communications, the SetInt routine should be used to configure the interrupt.

The two polling routines provided are SendChar and ReceChar. SendChar sends a character and returns a flag indicating if a successsful transmission occured. ReceChar receives a character and returns a flag indicating if a successful reception occured.

The routine Receive obtains a character from a buffer. This buffer fills with characters whenever a receive data available interrupt occurs. The size of this buffer is adjustable.

To disable the receive interrupt, the RemInt routine may be used. To restore the previous communication port state, the Restore routine may be used.

Four sample programs written in Microsoft C version 6.0 are provided to illustrate all the routines features. The four samples are: Send, Rece, Receint, and Itctalk. The routine Itctalk sends the help command and outputs the incoming help lines.

```
/* SEND.C */
/* Program example */

/* Assembly procedures are in itc_rout.asm  */
/* Assembler:     Microsoft Assembler version
6.0 */
/* Assemble with: ML /c itc_rout.asm

/* C compiler:    Microsoft C version 6.0 */
/* Compile with:  CL send.c itc_rout.asm */

/* Description:   This program sends out
alternating */
/*              + and - characters at 115200 baud.
*/


/* PROCEDURE Setup(Divisor, Parms,
SelBase) */
/* Divisor = 115200/desired baud rate */

/* PROCEDURE Restore(void) */
#include <stdio.h>

extern void far Setup( unsigned int Divisor,
unsigned char Parms, unsigned char SelBase );
extern unsigned int far SendChar( unsigned
char Sfl, unsigned char Tvalue );
extern unsigned int far ReceChar( unsigned
char Rfl, unsigned char Rvalue );
extern void far Restore( void );

void main()
{
   unsigned int tval = 0;
   unsigned long count = 0;

   printf( "Setup the ports\n");
```

```
   Setup( 1, 3, 2 );              // 115200,8,NP,1,
Com2
   printf( "Now transmitting 4096 characters\n" );

   while(count<4096)
      {
      do
         {
         tval = SendChar( 0, 43 );
//         printf("plus tval = %x\n",tval);
         }
      while(tval == 65280);       // repeat if Tx reg
is full
      do
         {
         tval = SendChar( 0, 45 );
//         printf("minus tval = %x\n",tval);
         }
      while(tval == 65280);       // repeat if Tx reg
is full
      ++count;
      }

   Restore();                     // restore old comm.
parameters
   printf( "Transmitted 4096 characters\n" );
}
```

```
/* Parms   = 0 | a | bcd | e | fg     */
/*           a   = 0 for break */
/*               = 1 for space */
/*           bcd = 000 none parity */
/*               = 001 odd parity  */
/*               = 011 even parity */
/*           e   = 0 for 1 stop bit */
/*               = 1 for 2 stop bits */
/*           fg  = 10 for 7 bits per character */
/*               = 11 for 8 bits per character */
/* SelBase = 1 for COM1 */
/*           2 for COM2 */
/*           3 for COM3 */
/*           4 for COM4 */

/* PROCEDURE SendChar(Sfl, Tvalue) */
/* Sfl     = FF if Tx register is full */
/*         = 00 if Tx register is empty */
/* Tvalue  = character to transmit */
/* returns a word with the MSB containing Sfl
```

```
/* RECE.C */
/* Program example */

/* Assembly procedures are in itc_rout.asm  */
/* Assembler:     Microsoft Assembler version
6.0 */
/* Assemble with: ML /c itc_rout.asm

/* C compiler:    Microsoft C version 6.0 */
/* Compile with:  CL rece.c itc_rout.asm */

/* Description:   This program receives */
/*                20 characters at 115200 baud. */

/* PROCEDURE Setup(Divisor, Parms,
SelBase) */
/* Divisor = 115200/desired baud rate */
/* Parms   = 0 | a | bcd | e | fg     */
/*           a   = 0 for break */
/*               = 1 for space */
/*           bcd = 000 none parity */
/*               = 001 odd parity  */
/*               = 011 even parity */
/*           e   = 0 for 1 stop bit */
/*               = 1 for 2 stop bits */
/*           fg  = 10 for 7 bits per character */
/*               = 11 for 8 bits per character */
/* SelBase = 1 for COM1 */
/*           2 for COM2 */
/*           3 for COM3 */
/*           4 for COM4 */

/* PROCEDURE ReceChar(Rfl, Rvalue) */
/* Rfl     = FF if Rx register is full */
/*         = 00 if Rx register is empty */
/* Rvalue  = character received */
/* returns a word with the MSB containing Rfl
/* and the LSB containing the received
character

/* PROCEDURE Restore(void) */


#include <stdio.h>

extern void far Setup( unsigned int Divisor,
unsigned char Parms, unsigned char SelBase );
extern unsigned int far SendChar( unsigned
```

```
char Sfl, unsigned char Tvalue );
extern unsigned int far ReceChar( unsigned
char Rfl, unsigned char Rvalue );
extern void far Restore( void );

void main()
{
   unsigned int rval = 0;
   unsigned long count = 0;

   printf( "Setup the ports\n");
   Setup( 1, 3, 2 );        // 115200,8,NP,1, Com2
   printf( "Waiting for reception of 20
characters\n");
   while(count<20)
      {
      do
         {
         rval = ReceChar( 0, 0 );
         }
      while(rval == 0);
         printf("plus rval = %x\n",rval);
      ++count;
      }

   Restore();              // restore old baud and
parameters
   printf( "Received all 20\n" );
}
```

```
/* RECEINT.C */
/* Program example */

/* Assembly procedures are in itc_rout.asm  */
/* Assembler:     Microsoft Assembler version
6.0 */
/* Assemble with: ML /c itc_rout.asm

/* C compiler:    Microsoft C version 6.0 */
/* Compile with:  CL receint.c itc_rout.asm */

/* Description:   This program receives 1024
characters */
/*                at 115200 baud using an interrupt
routine. */



/* PROCEDURE Setup(Divisor, Parms,
SelBase) */
/* Divisor = 115200/desired baud rate */
/* Parms   = 0 | a | bcd | e | fg     */
/*           a   = 0 for break */
/*               = 1 for space */
/*           bcd = 000 none parity */
/*               = 001 odd parity  */
/*               = 011 even parity */
/*           e   = 0 for 1 stop bit */
/*               = 1 for 2 stop bits */
/*           fg  = 10 for 7 bits per character */
/*               = 11 for 8 bits per character */
/* SelBase = 1 for COM1 */
/*           2 for COM2 */
/*           3 for COM3 */
/*           4 for COM4 */

/* PROCEDURE SetInt(void) */

/* PROCEDURE RemInt(void) */

/* PROCEDURE Receive(Rfl, Rvalue) */
/* Rfl     = FF if Rx buffer is full */
/*         = 00 if Rx buffer is empty */
/* Rvalue  = character received */
/* returns a word with the MSB containing Rfl
/* and the LSB containing the received
character

/* PROCEDURE Restore(void) */


#include <stdio.h>
#include <stdlib.h>

extern void far Setup( unsigned int Baud,
unsigned char Parms, unsigned char SelBase );
extern unsigned int far Receive( unsigned char
Rfl, unsigned char Rvalue );
extern void far Restore( void );
extern void far SetInt( void );
extern void far RemInt( void );


void main()
{
   unsigned int values[1023];
   unsigned int rval = 0;
   unsigned long count = 0;

   printf( "Setup the ports\n");
   Setup( 1, 3, 2 );           // 115200,8,NP,1,
Com2
```

```
   SetInt();
   printf( "Waiting for reception of 1024
characters\n");
   while(count<1024)
      {
      do
         {
         rval = Receive( 0, 0 );   //
         }
      while(rval == 0);
      rval = rval & 255;
      values[count]=rval;            // store in a
1024 element array
      printf("rval = %x\n",rval);     // output
received character
      ++count;
      }

   RemInt();                  // disable serial
interrupts
   Restore();                 // restore old comm.
parameters


   printf( "Received all 1024\n" );
}
```

```
/* ITCTALK.C */
/* Program example */

/* Assembly procedures are in itc_rout.asm  */
/* Assembler:     Microsoft Assembler version
6.0 */
/* Assemble with: ML /c itc_rout.asm

/* C compiler:    Microsoft C version 6.0 */
/* Compile with:  CL itctalk.c itc_rout.asm */

/* Description:   This program receives 700
characters */
/*                at 9600 baud using an interrupt
routine. */



/* PROCEDURE Setup(Divisor, Parms,
SelBase) */
/* Divisor = 115200/desired baud rate */
/* Parms   = 0 | a | bcd | e | fg     */
/*           a   = 0 for break */
/*               = 1 for space */
/*           bcd = 000 none parity */
/*               = 001 odd parity  */
/*               = 011 even parity */
/*           e   = 0 for 1 stop bit */
/*               = 1 for 2 stop bits */
/*           fg  = 10 for 7 bits per character */
/*               = 11 for 8 bits per character */
/* SelBase = 1 for COM1 */
/*           2 for COM2 */
/*           3 for COM3 */
/*           4 for COM4 */

/* PROCEDURE SetInt(void) */

/* PROCEDURE RemInt(void) */

/* PROCEDURE Receive(Rfl, Rvalue) */
/* Rfl     = FF if Rx buffer is full */
/*         = 00 if Rx buffer is empty */
/* Rvalue  = character received */
/* returns a word with the MSB containing Rfl
/* and the LSB containing the received
character

/* PROCEDURE Restore(void) */


#include <stdio.h>
#include <stdlib.h>

extern void far Setup( unsigned int Baud,
unsigned char Parms, unsigned char SelBase );
extern unsigned int far Receive( unsigned char
Rfl, unsigned char Rvalue );
extern unsigned int far SendChar( unsigned
char Sfl, unsigned char Tvalue );
extern void far Restore( void );
extern void far SetInt( void );
extern void far RemInt( void );


void main()
{
   unsigned char values[1023];
   unsigned int tval,rval = 0;
   unsigned long count = 0;

   count = 0;
   while(count<700)
      {
      values[count] = 0;
```

```
        ++count;
        }

    printf( "Setup the ports\n");
    Setup( 12, 3, 2 );            // 9600,8,NP,1,
Com2
    SetInt();

    do
       {
       tval = SendChar( 0, 63 );     // send out '?'
       }
    while(tval == 65280);       // repeat if Tx reg is
full

    do
       {
       tval = SendChar( 0, 13 );     // send out
carriage return
       }
    while(tval == 65280);       // repeat if Tx reg is
full

    printf( "Waiting for reception of 700
characters\n");

    count = 0;
    while(count<700)
       {
       do
          {
          rval = Receive( 0, 0 );   //
          }
       while(rval == 0);
       rval = rval & 255;
       values[count]=(unsigned char)rval;  // store
in a 700 element array
       ++count;
       }

    RemInt();                  // disable serial
interrupts
    Restore();                 // restore old comm.
parameters

    count = 0;
    while(count<700)
       {
       printf("%c",values[count]);
       ++count;
       }
    printf( "Received all 700 characters\n" );
}
```

```
; C Compiler:   Microsoft C version 6.0
;
; To assemble itc_rout.asm:
;
;             ML /c itc_rout.asm
;
; To link assembly object file with a C
program:
;
;             CL filename.c itc_rout.obj
;
;--------------------------------------------------------
; Procedures:
;
; SETUP         Selects base address, saves old
comm.
;             parameters and sets new comm
parameters.
;
; RESTORE       Restores the old comm
parameters.
;
; SETINT        Sets up the interrupt handler
address.
;             Only receive line status and receive
data
;             ready interrupts are processed.
;
; REMINT        Disables comm. interrupts.
;
; INSERTBUF     Inserts a character into the
buffer.
;
; REMOVEBUF     Removes a character from
the buffer.
;
; RECEIVE       Gets a character from the
buffer.
;
; SENDCHAR      Transmits a character directly
to the UART.
;
; RECECHAR      Receives a character directly
from the UART.
;
;--------------------------------------------------------

Com1            EQU     3F8h            ; comm. base
```

```
addresses
Com2            EQU     2F8h
Com3            EQU     3E8h
Com4            EQU     2E8h

CR              EQU     13
LF              EQU     10

IntEnableReg    EQU     1               ; offset of
register from
IntIdReg        EQU     2               ; base register
LineControlReg  EQU     3
ModemControlReg EQU     4
LineStatusReg   EQU     5
ModemStatusReg  EQU     6

ModemControlOn  EQU     08h
ModemControlOff EQU     0
IntEnableAll    EQU     05h
IntDisableAll   EQU     0

MDMSTATUS       EQU     0               ;
identifying numbers for
TXREGEMPTY      EQU     2               ; cause of
interrupt
RXDATAREADY     EQU     4               ; only
number 4 and 6 are checked
RLINESTATUS     EQU     6
                PUBLIC  MDMSTATUS
                PUBLIC  TXREGEMPTY
                PUBLIC  RXDATAREADY
                PUBLIC  RLINESTATUS

BUFSIZE         EQU     2048            ; size of
storage
CharBuffer      STRUCT
   InsertHere   DW      0               ; insertion
pointer
   RemoveHere   DW      0               ; removal
pointer
   BufferCount DW       0               ; count of
characters
   DataArea     DB      BUFSIZE DUP(0)
CharBuffer      ENDS

                PUBLIC
BUFSIZE,IntEnableReg,IntIdReg,LineControlRe
g
                PUBLIC
ModemControlReg,LineStatusReg,ModemStatu
sReg
                PUBLIC
ModemControlOn,ModemControlOff
                PUBLIC  IntEnableAll,IntDisableAll
;---------------------------------------------------------

                .MODEL  MEDIUM,c

                .STACK  2048

;---------------------------------------------------------
                .DATA
; declare variables

                PUBLIC  ComBase,IntEnableMask
                PUBLIC
IntDisableMask,ComIntNumber,OldHandlerSeg
                PUBLIC
OldHandlerOff,NewHandler,ComBuffer,BufferSi
ze
                PUBLIC  BufferSize,Rflag

ComBase         DW      ?               ; base
address of serial port
IntEnableMask   DB      ?               ; masks for
8259
IntDisableMask  DB      ?               ;
ComIntNumber    DB      ?               ;
OldHandlerSeg   DW      ?               ; old
Interrupt handler's address
OldHandlerOff   DW      ?
NewHandler      DD      SerIntHandler   ; far
pointer to new handler
OrgParms        DB      ?               ; original
parameters
OrgDivi         DW      ?               ; original divisor
OrgMod          DB      ?

Rflag           DB      0               ; receive buffer
flag 0=empty ff=full

ComBuffer       CharBuffer 1 DUP (<>)
;receive buffer
BufferSize      =       ($-ComBuffer)   ; size of
buffer
;---------------------------------------------------------

                .CODE

;---------------------------------------------------------
;Procedure:     Setup
;
;Description:   Selects comm. port and sets the
parameters such as
;               parity, number of stop bits, and word
length and
```

```
;               baud rate. Also saves previous reg.      contents.
;
;Input:         Divisor  Baud rate divisor,
115200/Divisor = desired baud rate.
;               Parms    Parity, number of stop bits
and word length.
;               SelBase  Base address select n =
COMn.
;-------------------------------------------------------
Setup           PROTO   C
Divisor:WORD,Parms:BYTE,SelBase:BYTE
Setup           PROC    FAR C PUBLIC
Divisor:WORD,Parms:BYTE,SelBase:BYTE

                mov     ax, @data       ; set up DS for
addressing data
                mov     ds, ax

                mov     al, SelBase     ; get selected
comm port (base)

; determine which comm port to use

                cmp     al, 1                   ; Com1
                jne     Next1
                mov     ComBase, Com1
                mov     IntEnableMask, 0EFh
                mov     IntDisableMask, 10h
                mov     ComIntNumber, 12
                jmp     Next
Next1:          cmp     al, 2                   ; Com2
                jne     Next2
                mov     ComBase, Com2
                mov     IntEnableMask, 0F7h
                mov     IntDisableMask, 8h
                mov     ComIntNumber, 11
                jmp     Next
Next2:          cmp     al, 3                   ; Com3
                jne     Next3
                mov     ComBase, Com3
                mov     IntEnableMask, 0EFh
                mov     IntDisableMask, 10h
                mov     ComIntNumber, 12
                jmp     Next
Next3:          cmp     al, 4                   ; Com4
                jne     Done
                mov     ComBase, Com4
                mov     IntEnableMask, 0F7h
                mov     IntDisableMask, 8h
                mov     ComIntNumber, 11

; set the comm parameters directly
```

```
Next:           mov     dx, ComBase
                add     dx, LineControlReg       ; get
Lcr address

                in      al, dx
                mov     OrgParms, al             ; save
original parameters
                mov     al, 80h
                out     dx, al                   ; set DLAB

                mov     dx, ComBase
                in      ax, dx
                mov     OrgDivi, ax              ; save
original divisor

                mov     ax, Divisor              ; set LSB
of divisor
                out     dx, al

                mov     al, ah                   ; set MSB of
divisor
                inc     dx
                out     dx, al

                mov     dx, ComBase
                add     dx, LineControlReg       ; get
Lcr address
                mov     al, Parms                ; set the
comm parms
                out     dx, al                   ; reset DLAB
Done:
                ret
Setup           ENDP


;--------------------------------------------------------
;Procedure:     Restore
;
;Description:   Restores old comm. settings
;--------------------------------------------------------
Restore         PROTO   C
Restore         PROC    FAR C PUBLIC
                push    ax
                push    dx
                mov     dx, ComBase
                add     dx, LineControlReg       ; get
Lcr address
                mov     al, 80h
                out     dx, al                   ; set DLAB

                mov     dx, ComBase              ; set
original divisor
                mov     ax, OrgDivi              ; set LSB
of divisor
                out     dx, al

                mov     al, ah                   ; set MSB of
divisor
                inc     dx
                out     dx, al

                mov     dx, ComBase              ; set
original parameters
                add     dx, LineControlReg       ; get
Lcr address
                mov     al, OrgParms             ; set the
comm parms
                out     dx, al
                pop     dx
                pop     ax
                ret
Restore         ENDP


;--------------------------------------------------------
;Procedure:     SetInt
;
;Description:   Installs a new interrupt handler
while saving
;               old handler.  Enables serial interrupts.
;--------------------------------------------------------
SetInt          PROTO   C
SetInt          PROC    FAR C PUBLIC
; get and save old handler's address
                mov     ah, 35h
                mov     al, ComIntNumber
                int     21h
                mov     OldHandlerOff, bx
                mov     OldHandlerSeg, es
; install new interrupt handler
                mov     ah, 25h
                mov     al, ComIntNumber
                push    ds
                mov     dx, WORD PTR Newhandler
                mov     ds, WORD PTR
Newhandler+2
                int     21h
                pop     ds
; enable interrupts from the serial port
                cli                      ; first diable all
interrupts
                mov     dx, ComBase
                add     dx, ModemControlReg
                in      al, dx
                mov     OrgMod, al
                mov     al, ModemControlOn
```

```
                out     dx, al
; set the Interrupt Enable Register
                mov     dx, ComBase
                add     dx, IntEnableReg
                mov     al, IntEnableAll
                out     dx, al
; program the 8259 to acknowledge interrupts
                in      al, 21h
                and     al, IntEnableMask
                out     21h, al
                sti                     ; reenable interrupts
                ret
SetInt          ENDP


;--------------------------------------------------------
;Procedure:     RemInt
;
;Description:   Removes interrupt handler and
restores old one.
;--------------------------------------------------------
RemInt          PROTO   C
RemInt          PROC    FAR C PUBLIC
; clean up...
                cli                     ; first diable all
interrupts
                mov     dx, ComBase
                add     dx, ModemControlReg
                mov     al, ModemControlOff
                out     dx, al
                mov     al, OrgMod
                out     dx, al
; reset the Interrupt Enable Register
                mov     dx, ComBase
                add     dx, IntEnableReg
                mov     al, IntDisableAll
                out     dx, al
; turn off interrupts at the 8259
                in      al, 21h
                and     al, IntDisableMask
                out     21h, al
                sti                     ; reenable interrupts
; restore old interrupt vector
                mov     ah, 25h
                mov     al, ComIntNumber
                push    ds
                mov     ds, OldHandlerSeg
                mov     dx, OldHandlerOff
                int     21h
                pop     ds
                ret
RemInt          ENDP

;--------------------------------------------------------
;Procedure:     SerIntHandler
;
;Description:   Serial interrupt routine processes
receive data
;               available and receive line status
interrupts.
;--------------------------------------------------------
SerIntHandler   PROC    FAR
                sti                     ; reenable interrupts

                push    ds
                push    ax
                push    bx
                push    cx
                push    dx
                push    di
; set up DS for data
                mov     ax, @data
                mov     ds, ax
                ASSUME  DS:@data
; check if any interrupts pending?
AnyIntPending:
                mov     dx, ComBase
                add     dx, IntIdReg
                in      al, dx
                mov     cl, al
                test    al, 1
                jz      ProcessAnother
; tell 8259 that processing of this interrupt is
complete
                mov     al, 20h
                out     20h, al
                jmp     ExitSerInt

ProcessAnother:


; check which interrupt
                cmp     cl, MDMSTATUS
                jne     IsItRLINESTATUS
; read modem status register
                mov     dx, ComBase
                add     dx, ModemStatusReg
                in      al, dx
                jmp     AnyIntPending

IsItRLINESTATUS:
                cmp     cl, RLINESTATUS
                jne     IsItRXDATAREADY
; read line status register
                mov     dx, ComBase
                add     dx, LineStatusReg
```

```
            in      al, dx          ; get line status
            jmp     AnyIntPending   ; ignore status
for now

IsItRXDATAREADY:
            cmp     cl, RXDATAREADY
; read and place character in buffer
            mov     dx, ComBase
            in      al, dx
            mov     ah, 0
; save character in buffer
            call    InsertBuf
            jmp     AnyIntPending

ExitSerInt:
; done, restore registers and return using an
IRET
            pop     di
            pop     dx
            pop     cx
            pop     bx
            pop     ax
            pop     ds

            iret
SerIntHandler   ENDP


;-------------------------------------------------------
;Procedure:     InsertBuf
;
;Description:   Inserts a character into a circular
buffer.
;
;Input:         AL = Character to insert into buffer.
;
;Output:        Rflag = FF if buffer is full.
;               Rflag = 7F if a char. exists in the
buffer.
;-------------------------------------------------------
InsertBuf       PROC    FAR PUBLIC
; Fills buffer with received chars
; AL has character to insert
            push    di
            push    bx
            push    dx

            mov     di, OFFSET ComBuffer
            cmp     [di].CharBuffer.BufferCount,
BUFSIZE
            jne     NotFull
            mov     Rflag,0FFh
            mov     dx, ComBase
            add     dx, IntEnableReg
            in      al, dx                  ; if buffer
full
            and     al, NOT RXDATAREADY
  ; disable receive int
            jmp     ExitInsertBuf
NotFull:                                    ; if
something in
            mov     Rflag,07Fh
            mov     bx, [di].CharBuffer.InsertHere
            mov     BYTE PTR
[di+bx+CharBuffer.DataArea], al
            inc     [di].CharBuffer.BufferCount
; adjust Insert Pointer
            inc     [di].CharBuffer.InsertHere
            cmp     [di].CharBuffer.InsertHere,
BUFSIZE
            jnz     ExitInsertBuf
; wrap around
            mov     [di].CharBuffer.InsertHere, 0
ExitInsertBuf:
            pop     dx
            pop     bx
            pop     di
            ret
InsertBuf       ENDP


;-------------------------------------------------------
;Procedure:     RemoveBuf
;
;Description:   Removes a character from the
circular buffer.
;
;Output:        AL = Character to remove from the
buffer.
;               Rflag = 0  if buffer is empty.
;               Rflag = 7F if a char. exists in the
buffer.
;-------------------------------------------------------
RemoveBuf       PROC    FAR PUBLIC
; character returned in AL.
            push    di
            push    bx
            mov     dx, ComBase
            add     dx, IntEnableReg
            in      al, dx
            or      al, RXDATAREADY             ;
enable receive
            xor     al, al

            mov     di, OFFSET ComBuffer
            cmp     [di].CharBuffer.BufferCount, 0
```

```
; is buffer empty ?
            jne     NotEmpty
            mov     Rflag,0
            jmp     ExitRemoveBuf
NotEmpty:                                   ; if
something in
            mov     Rflag,07Fh
            mov     bx,
[di].CharBuffer.RemoveHere
            mov     al, BYTE PTR
[di+bx+CharBuffer.DataArea]
            dec     [di].CharBuffer.BufferCount
; adjust Remove Pointer
            inc     [di].CharBuffer.RemoveHere
            cmp     [di].CharBuffer.RemoveHere,
BUFSIZE
            jnz     ExitRemoveBuf
; wrap around
            mov     [di].CharBuffer.RemoveHere,
0
ExitRemoveBuf:
            pop     bx
            pop     di
            ret
RemoveBuf       ENDP


;-------------------------------------------------------
;Procedure:     SendChar
;
;Description:   Sends a character to the transmit
holding
;           register if it is empty.
;
;Input:         Tvalue = Character to transmit.
;
;Ouput:         Sfl = FF if Tx reg. is full.
;           Sfl = 00 if Tx reg. is empty.
;-------------------------------------------------------
SendChar        PROTO   C           Sfl:BYTE,
Tvalue:BYTE
SendChar        PROC    FAR C PUBLIC
Sfl:BYTE, Tvalue:BYTE

            mov     bl,Tvalue       ; bl=send char
            cli
                                ; ah=transmit status
flag
            mov     ah, 0FFh        ; ah=FF Tx reg
full, ah=00 Tx empty
            mov     dx, ComBase
            add     dx, LineStatusReg
            in      al, dx
            test    al, 20h
            jz      Done            ; if Tx reg full,
return

            xor     ah, ah          ; zero the flag
            mov     dx, ComBase
            mov     al, bl
            out     dx, al          ; send the char

Done:           xor     al, al
            sti
            ret
SendChar        ENDP


;-------------------------------------------------------
;Procedure:     ReceChar
;
;Description:   Receives a character from the
receive
;           register if it is not empty.
;
;Output:        Rvalue = Character to receive.
;           Rfl = FF if Rx reg. is full.
;           Rfl = 00 if Rx reg. is empty.
;-------------------------------------------------------
ReceChar        PROTO   C           Rfl:BYTE,
Rvalue:BYTE
ReceChar        PROC    FAR C PUBLIC
Rfl:BYTE, Rvalue:BYTE
            xor     ax,ax
            cli
            xor     ah, ah          ; ah=receive
status flag
                                ; ah=00 Rx reg empty,
ah=FF Rx full
            mov     dx, ComBase
            add     dx, LineStatusReg
            in      al, dx
            test    al, 01h
            jz      Done            ; if Rx reg empty,
return

            mov     ah, 0FFh        ; set the flag
            mov     dx, ComBase
            in      al, dx          ; receive the char
            jmp     Done2

Done:           xor     al,al
Done2:          sti
            ret
ReceChar        ENDP
```

```
;-------------------------------------------------------
;Procedure:     Receive
;
;Description:   Receives a character from the circular
;              buffer.
;
;Output:        Rvalue = Character to receive.
;              Rfl = FF if Rx buffer is full.
;              Rfl = 00 if Rx buffer is empty.
;-------------------------------------------------------


Receive         PROTO   C           Rfl:BYTE, Rvalue:BYTE
Receive         PROC    FAR C PUBLIC Rfl:BYTE, Rvalue:BYTE
                xor     ax,ax
                cli
                call    RemoveBuf       ; al=returned char
                mov     ah,Rflag        ; ah=receive buffer status flag
                cmp     ah, 0h
                jne     Done
                xor     al, al
Done:
                sti

                ret
Receive         ENDP

                END
```