

```

{*****}
{          S E R I R Q . P A S          }
{*****}
{ Task      : Uses chat program to demonstrate }
{          serial port IRQ programming         }
{*****}
{ Author      : Michael Tischer / Bruno Jennrich }
{ Developed on : 04/08/1994                     }
{ Last update : 04/07/1995                     }
{*****}
{$X+} { Function results optional }

uses CRT,ARGS,WIN,SERUTIL,IRQUTIL;

{- Global variables -----}
var
Screen,
Remote,
Local,
Status      : WINDOW;

iSerPort,
iSerIRQ,
iUART,
iCom        : Integer;
c           : Array[0..1] of Char;
lBaud       : Longint;
lpOldIRQ    : Pointer;
pSaved      : ^Byte;           { Saved screen }
s           : String;

{*****}
{ GetSer : Interrupt routine, initiated by serial port }
{*****}
{ Info : This function accepts data from the serial }
{       port and displays it in the "remote" window. }
{       This function also contains requirements for }
{       all other port events that initiate }
{       an IRQ. }
{*****}
Procedure GetSer; Interrupt;

var bIRQID : Byte;
    c      : String;
    i      : Integer;

Begin
    bIRQID := port[iSerPort + SER_IRQ_ID]; {Get port status}
    if ( bIRQID and SER_ID_PENDING ) = 0 then { Is IRQ pending ? }
        Begin { Yes }
            case ( bIRQID and SER_ID_MASK ) of
                SER_ID_RECEIVED: { After data received }
                    Begin
                        i := 0;
                        c := '';
                        while ( ser_IsDataAvailable( iSerPort ) and ( i < 16 ) ) do
                            Begin
                                c := c + char ( port[iSerPort + SER_RXBUFFER]);
                                Inc( i );
                            End;
                        win_Print( Remote, c );
                    End;

                SER_ID_SENT: { After data sent }
                    Begin
                        End;

                SER_ID_MODEMSTATUS: { After line status change }
                    Begin
                        End;
                    End;
            end;
            irq_SendEOI( iSerIRQ ); { End of interrupt }
        End;
    End;

{*****}
{ Hz : Converts a hex number to a hex string }
{*****}
{ Input : HZ - the hex number }
{ Output : the converted hex string }
{*****}
Function Hex( hz : Word ) : String;

var i : Integer;

```

```

hs : String;
begin
  hs := '';
  Repeat
    i := hz mod 16;
    hz := hz div 16;
    case i of
      0..9 : hs := chr( i + ord( '0' ) ) + hs;
      10..15: hs := chr( i - 10 + ord( 'A' ) ) + hs;
    end;
  Until hz = 0;
  if Length( hs ) mod 2 <> 0 then
    hs := '0' + hs;
  hex := hs;
End;

{*****}
{ M A I N   P R O G R A M }
{*****}
Begin
  if GetArg( '?', _none, Nil, 1 ) <> 0 then
    Begin
      Writeln( 'Call:' );
      Writeln( 'SERIRQ [-COM:comport] [-BAUD:baudrate]' );
      Writeln( 'COM port = 1 or 2 (Default: 1)' );
      Writeln( 'Baud rate = 50 - 115200 (Default: 9600)' );
      Halt(0);
    End;

  {-- Evaluate command line -----}
  if GetArg( '-COM:', _int, @iCom, 1 ) <> 0 then
    Begin
      if iCom = 1 then
        Begin
          { Only COM1 and COM2 are "standardized" }
          iSerPort := SER_COM1;
          iSerIRQ := SER_IRQ_COM1;
        End
      else
        if iCom = 2 then
          Begin
            { Only COM1 and COM2 are "standardized" }
            iSerPort := SER_COM2;
            iSerIRQ := SER_IRQ_COM2;
          End
        else
          Begin
            Writeln( 'Unsupported COM port!' );
            Halt(0);
          End;
        End
      else
        Begin
          { Only COM1 and COM2 are "standardized" }
          iSerPort := SER_COM1;
          iSerIRQ := SER_IRQ_COM1;
        End;
      End

  if GetArg( '-BAUD:', _long, @lBaud, 1 ) = 0 then
    lBaud := 9600;
    { Maximum baud rate for UART 8450A }

  if lBaud > SER_MAXBAUD then
    Begin
      Writeln( 'Baud rate too high!' );
      Writeln( 'Maximum: ', SER_MAXBAUD, ' Bd' );
    End;

  {-- Initialize port with parameters obtained --}
  iUART := ser_Init( iSerPort, lBaud, SER_LCR_8BITS or
                    SER_LCR_1STOPBIT or
                    SER_LCR_NOPARITY );

  if iUART = NOSER then
    Begin
      Writeln( 'No port!' );
      Halt( 0 );
    End;

  if iUART > INS8250 then { For 14450 activate FIFO buffer }
    ser_FIFOLevel( iSerPort, SER_FIFO_TRIGGER14 );

  win_GetScreenSettings( Screen );
  pSaved := win_Save( Screen );
  { Save screen }

  { Format output window }
  win_Init( Remote, 0, 0, 80, 9, $17, $1f, WIN_SCROLL or WIN_CRLF );
  win_Init( Local, 0, 10, 80, 9, $17, $1f, WIN_SCROLL or WIN_CRLF or
            WIN_ACTIVE or WIN_HASCURSOR );
  win_Init( Status, 0, 20, 80, 2, $17, $1f, WIN_SCROLL or WIN_CRLF );

```

```

win_Clr( Screen );
win_Clr( Remote );
win_Clr( Local );
win_Clr( Status );
str(lBaud,s);
win_print( Status, 'COM port: $' + Hex( iSerPort ) +
               ', IRQ: $' + Hex( iSerIRQ ) +
               ', Baud: '+s+#10 );

win_print( Status, '<ESC> = End ' );
win_print( Status, 'Use "?" to display options ' );
win_print( Status, '(c) BHJ, MITI' );

{-- Install serial interrupt handler. --}
lpOldIRQ := ser_SetIRQHandler( iSerPort,
                               iSerIRQ,
                               @GetSer,
                               SER_IER_RECEIVED or SER_IER_SENT );

{-- Initiate chat mode ----}
c[0] := #0;
c[1] := #0;

win_GotoXY( Local, 0, 0);
While c[0] <> #27 do
Begin
  c[0] := ReadKey;
  if c[0] <> #0 then
    Begin { Output entered character through port... }
      ser_WriteByte( iSerPort, Byte ( c[0] ), $8000, 0, 0 );
      if ( c[0] > #31 ) and ( c[0] < #128 ) then
        win_Print( Local, c[0] ) { ...and display in window. }
      else
        win_print( Local, '<' + Hex ( Byte ( c[0] ) ) + '>' );
    End
  else
    ReadKey;
End;
{--- Restore old interrupt handler and screen contents ----}
ser_RestoreIRQHandler( iSerPort, iSerIRQ, lpOldIRQ );
win_Restore( pSaved, TRUE );

if iUART > INS8250 then ser_FIFOLevel( iSerPort, 0 ); {Reset FIFO.}
End.

```