

PC 16-line RS232 port I/O card (SIO-1)

User's Manual

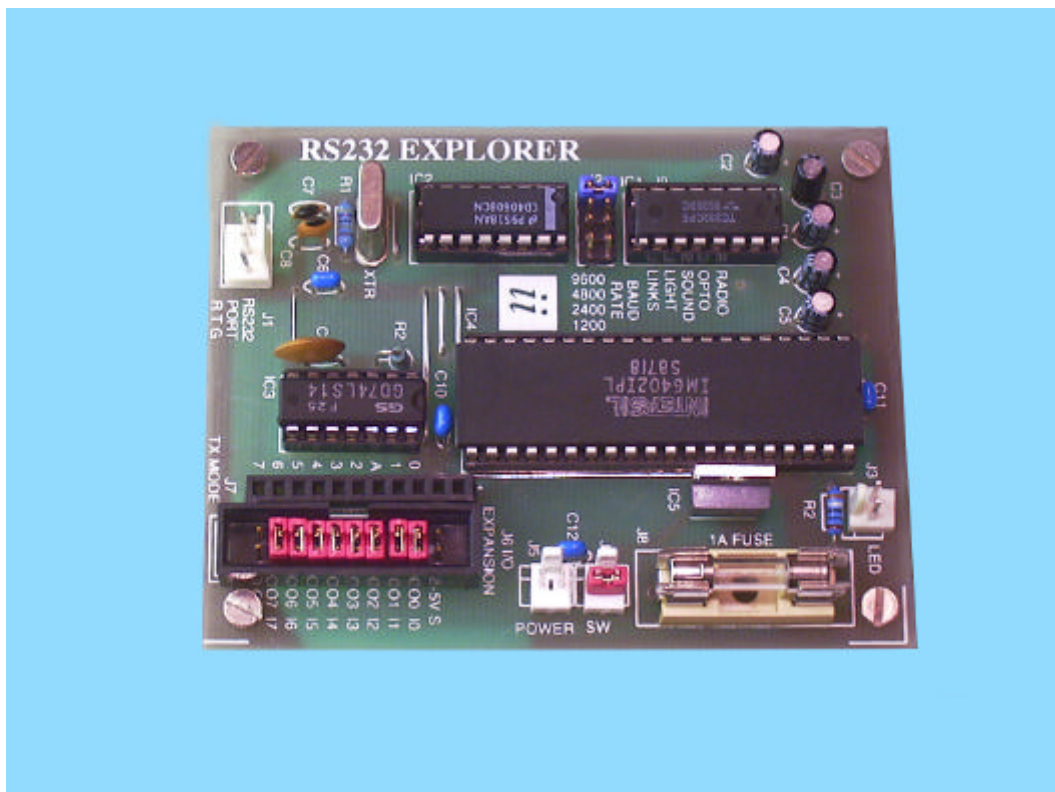
This I/O card is connected to a computer via a serial port (RS232 port). The card provides 8 output lines and 8 input lines. Each line is TTL compatible. The 16 I/O lines are available from a 20-way male DIL PCB header for easy connection.

The I/O card is connected to a PC via only three wires of the RS232 port (namely TX, RX and Ground). The Baud rate is programmable (1200, 2400, 4800 and 9600). There is no parity check bit. Data bit length is 8 and the number of stop bit is 1. A byte transmitted from the host computer appears at the output lines of the card. After each successful transmission, the card sends a byte, which appears at the inputs, back to the computer.

A DOS-based program "SIODos" and a Windows95/98 program "SIOWin" are supplied with the I/O card. They allow users to check functionality of the card.

A Turbo Pascal 6 unit is supplied with the card, which contains basic I/O functions for controlling the I/O card. After declaring the unit in a TP6 programs, users can call the functions anywhere in the TP6 programs. The "SIOWin" is written in Visual Basic 5. Since VB5 has a serial port control "MSCOMM", we can use this to control the I/O card directly without using any external DLLs.

The card requires a 7-15V DC (1A) power supply. An on-board 1A fuse is on the card.



RS232 16-line I/O card - SIO-1 card

1. Hardware

1.1 I/O lines

The card has 16 input/output lines. 8 lines are output and the other 8 lines are input. All I/O lines are TTL compatible and are available from a 20-way DIL male header (see Figure 1).

1.2 RS232 connection

The SIO card is connected to a computer via three wires of the RS232 port (TX, RX and Ground). The Baud rate is programmable (1200, 2400, 4800 and 9600). There is no parity check. Data bit length is 8 and the number of stop bit is 1.

When the Baud rate is set at 9600, the maximum data write rate will be 960 bytes per second.

1.3 Write data to and read data from the SIO card

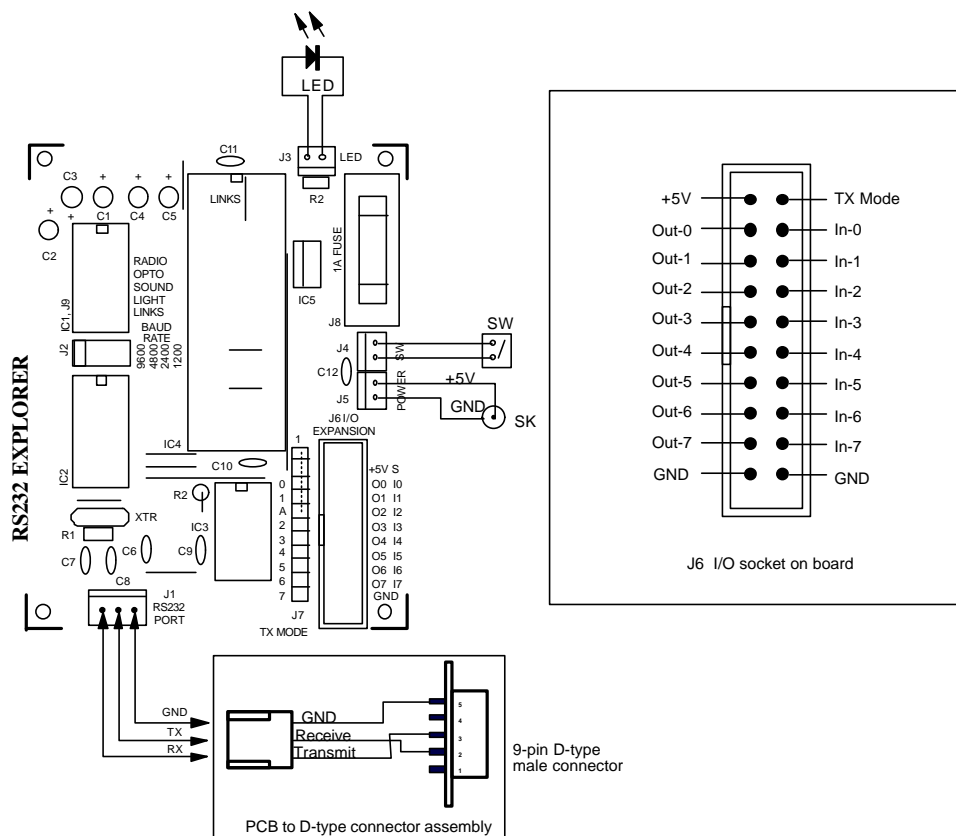
The card is always ready to receive data from the host PC via its RS232 port. A binary data transmitted from the PC will appear at the output of the card. Every time a data is sent to the card, the card sends back a data to the computer. The data comes from the input of the card.

1.4 Power supply

The card requires a 7-15V DC (1A) voltage supply. An on-board 1A fuse protects the over load of current of the power supply. The card provides connectors for power input, a switch and an LED indicator. The SIO card consumes 10 mA current. The regulated +5V power supply is available from the 20-way DIL socket.

1.5 Sizes and basic connections

The size of the SIO card is 7.5 cm by 9.5 cm. Figure 1 shows the layout and basic connections of the card.



2. Application note

The RS232 *port*, also known as the *serial port*, is an industrial standard interface designed for connecting peripherals to a computer. A computer at least has one such a port installed. The port may come with computer's motherboards or come with plug-in I/O cards. Adding more RS232 ports is easy and inexpensive. The ports have logic name COM1 to COMn (n=2,3 or 4).

Originally, the port was designed for connecting PCs to printers, mice, modems. They can also be used for other interfacing applications. Devices designed for the RS232 port not only provide the easiest way of connection to computers but also offer a universal hardware solution for all computers.

The SIO-1 card allows the RS232 port to be used. It serves as an interface between computers and external circuits. The card utilises an industrial standard UART 6402. This has an 8-bit output port and an 8-bit input port. The SIO card is connected to a computer via three wires of the RS232 port (TX, RX and Ground). The Baud rate is programmable (1200, 2400, 4800 and 9600). There is no parity check. Data bit length is 8 and the number of stop bit is 1.

The card is always ready to receive data from the host PC via its RS232 port. A binary data transmitted from the PC will appear at the output of the card. Every time the card receives a data, it sends back a data to the computer. This data comes from the input of the card.

Inside a computer, a COM port is always associated with an UART. The UART has 8 registers. To control data flow through a COM port is a matter of writing data to and reading data from these registers. In Turbo Pascal 6, direct port access command can be used to achieve this. In Visual Basic 4 and 5, MSCOMM control is used.

3. Software

The CDROM supplied with the card contains the following directories:

- \Manual
- \Drivers
- \Dos
- \Win16

Files in each directory are listed as follows:

Files	Description	Explanation
\Manual Directory		
Manual.doc	Manual in MSWord 97 format	
SIOTP6.pas	Turbo Pascal 6 source codes for I/O functions. This program produces CIOTP6.tpu	For users which want to convert the TP6 program into other programming languages such as Basic or C++
\Drivers Directory		
SIOTP6.tpu	A TP6 unit containing I/O functions	They contain all I/O functions to control the card.
\Dos Directory		
SIODos	Dos-based user program	Test the functionality of the I/O card in DOS environment
SIOTP6.tpu	A TP6 unit containing I/O functions	
\Win16 Directory		
This directory contains necessary files for installation.		Test the functionality of the I/O card in Windows environment

4. TP6 I/O functions and VB5 COMM control

Turbo Pascal 6 I/O functions are listed below. They are contained in the SIOTP6.tpu.

Function Name	Description
<i>RS232(x)</i>	X=0: return number of COM installed on a PC X=1: return base address of 1 st COM port (in decimal) X=2: return base address of 2 nd COM port (in decimal) X=3: return base address of 3 rd COM port (in decimal)
<i>Write_transmit_buffer(RS232_addr, byte)</i>	Output a byte from a specified COM port
<i>Read_receive_buffer(RS232_addr)</i>	Read a byte from the buffer of a specified COM port
<i>Write_data_format(RS232_addr, Baudx, Parity, Data_bit, Stop_bit)</i>	Configure a specified COM port. To achieve the following setup: 9600,n,8,1, the following command is used: <i>Dummy=Write_data_format(RS232_addr, 96, 0, 8, 1)</i>
<i>Write_interrupt_enable(RS232_addr, Output_byte)</i>	Write data into the interrupt register of a selected UART to enable an interrupt
<i>Read_interrupt_identification (RS232_addr)</i>	Read data from the interrupt identification register of the UART to check the source of interrupt
<i>Write_modem_status(RS232_addr, RTS, DTR):</i>	Change status of RTS and DTR lines. RTS and DTR is either 0 or 1.
<i>Read_modem_status(RS232_addr, x)</i>	Read status of DCD, DSR and CTS lines (X=1 to read DCD bit, x=2 to read DSR bit, x=3 to read CTS bit)

Note:

1. *RS232_addr* is the base address of the selected COM port (in decimal). *RS232_addr* is obtained using the following command: *RS232_addr = RS232(x)* (x=1 or 2 or 3 for COM1, COM2 or COM3).
2. All parameters are integers

The TP6 source program to produce "SIOTP6.tpu" is given in Program List 1. How the functions are used in a program is demonstrated in Program List 2.

4.1 TP6 I/O functions

TP6 program "SIOTP6.Pas" contains all I/O functions to control the SIO card. The complete program source code is listed as follows. This file generates "SIOTP6.tpu"

Program List 1

```
(*****
*
* Module:          Intec 16-line Serial I/O card - SIO card
*
* Author:         Pei An
*                 Intec Associates Limited
*                 11 Sandpiper Dirve
*                 Stockport
*                 SK3 8UL UK
*                 Tel. +44-(0)161 477 9583
*                 Fax. +44-(0)161 477 5755
*                 Email: pan@intec-group.co.uk
*
* Copyright 1999 Intec Associates Limited
*
* Description:
*
* This module provides functions to drive the Intec RS232 I/O card
* Functions included:
```

```

-Function RS232(x:integer):integer;
-Function Write_interrupt_enable(RS232_address, Output_byte: integer):integer;
-Function Read_interrupt_identification(RS232_address:integer):integer;
-Function Write_data_format(RS232_address, Baudx, Parity, Data_bit,
Stop_bit:integer):integer;
-Function write_transmit_buffer(RS232_address, Output_byte: integer):integer;
-Function Write_modem_status(RS232_address, RTS, DTR:integer):integer;
-Function Read_receive_buffer(RS232_address:integer):integer;
-Function Read_modem_status(RS232_address, x:integer):integer;
***** *)
{$DEFINE MAIN}
{$UNDEF MAIN}

{$IFDEF MAIN}
program SIOTP6;

{$ELSE}
unit SIOTP6;

Interface
    Function RS232(x:integer):integer;
    Function Write_interrupt_enable(RS232_address, Output_byte: integer):integer;
    Function Read_interrupt_identification(RS232_address:integer):integer;
    Function Write_data_format(RS232_address, Baudx, Parity, Data_bit,
Stop_bit:integer):integer;
    Function write_transmit_buffer(RS232_address, Output_byte: integer):integer;
    Function Write_modem_status(RS232_address, RTS, DTR:integer):integer;
    Function Read_receive_buffer(RS232_address:integer):integer;
    Function Read_modem_status(RS232_address, x:integer):integer;

Implementation
{$ENDIF}

Function RS232(x:integer):integer;
(* Universal auto detection of COM base address
$0000:$0400 holds the printer base address for COM1
$0000:$0402 holds the printer base address for COM2
$0000:$0404 holds the printer base address for COM3
$0000:$0406 holds the printer base address for COM4
$0000:$0411 number of parallel interfaces in binary format *)
var
    number_of_COM, COM1, COM2, COM3, COM4 :integer;
begin
    number_of_COM:=mem[$40:$11]; {read number of parallel ports}
    number_of_COM:=(number_of_COM and (8+4+2)) shr 1;
    COM1:=0; COM2:=0; COM3:=0; COM4:=0;
    COM1:=memw[$40:$00];      {Memory read procedure}
    COM2:=memw[$40:$02];
    COM3:=memw[$40:$04];
    COM4:=memw[$40:$06];
    case x of
        0:    RS232:=number_of_COM;
        1:    RS232:=COM1;
        2:    RS232:=COM2;
        3:    RS232:=COM3;
        4:    RS232:=COM4
    end;
end;

Function Write_interrupt_enable(RS232_address, Output_byte: integer):integer;
begin
    Port[RS232_address+1]:=Output_byte;
end;

Function Read_interrupt_identification(RS232_address:integer):integer;
begin
    Read_interrupt_identification:=Port[RS232_address+2]
end;

Function Write_data_format(RS232_address, Baudx, Parity, Data_bit, Stop_bit:integer):integer;
var
    byte1, byte2, output_byte: byte;
    divisor: integer;
    baud:longint;
begin
    Baud:=baudx * 100 ;
    divisor:=115200 div Baud;

```

```

    if divisor<=255 then begin byte1:=divisor; byte2:=0 end;
    if divisor>255 then begin byte2:=divisor div 256; byte1:=divisor mod 256; end;
    output_byte:=(data_bit-5) + 4*(stop_bit-1) + 8*(parity);
    port[RS232_address+3]:=128;{Loading serial data format, first bit of the register is 1}
    port[RS232_address+0]:=Byte1;      {LSB of the divisor is 1}
    port[RS232_address+1]:=Byte2;      {MSB of the divisor is 0}
    port[RS232_address+3]:=output_byte; {Load divisor and other parameters}
end;

Function write_transmit_buffer(RS232_address, Output_byte: integer):integer;
begin
    port[RS232_address]:=Output_byte;
end;

Function Write_modem_status(RS232_address, RTS, DTR:integer):integer;
(* RTS and DTR = 0 or 1, RTS and DTR are inverted by MAX238 on the experimental board *)
(* RTS=bit 1, DTR=bit 0 of Modem control register, offset 04 *)
begin
    RTS:=1-RTS;
    DTR:=1-DTR;
    Port[RS232_address+4]:=RTS*2 + DTR (* to output to the register 04 *)
end;

Function Read_receive_buffer(RS232_address:integer):integer;
begin
    Read_receive_buffer:=port[RS232_address];
end;

Function Read_modem_status(RS232_address, x:integer):integer;
(* X=1 select DCD bit, x=2 select DSR bit, x=3 select CTS bit *)
(* DCD=bit 7, DSR=bit 5, CTS=bit 4 of Modem status register, offset 06h *)
(* All bits are inverted by the Max238 on the experimental board *)
var
    input_byte:byte;
begin
    input_byte:=port[RS232_address+6];
    case x of
    1:   Read_modem_status:=1-round((input_byte and 128)/128);
    2:   Read_modem_status:=1-round((input_byte and 32)/32);
    3:   Read_modem_status:=1-round((input_byte and 16)/16);
    end;
end;
begin
end.

```

4.2 How a Turbo Pascal 6 program calls I/O functions

In a TP6 program, SIOTP6.tpu unit has to be declared at the beginning of the program. Then I/O functions can be called elsewhere in the program. A sample program is shown in Program list 2. The user interface of this program is shown in Figures 2 and 3

Program List 2

```

Program SIO_Card;
uses
    dos,crt,SIOTP6;
    { SIOTP6 contains the I/O procedures controlling the card }
var
    bitnumber,outputbyte,dummy,i:byte;
    RS232_address,delaynumber,com_number:integer;
    number_of_COM,code:integer;

Procedure find_delay_number;
{Check pc speed and find the delaynumber for delay}
var
    time1,time2,dt:real;
    t,h1,m1,s1,s1001,h2,m2,s2,s1002:word;
begin
    clrscr;
    gotoxy(25,24); write('Checking computer speed');
    gettime(h1,m1,s1,s1001);
    time1:=3600*h1+60*m1+s1+s1001/100;
    for t:=1 to 30000 do delay(1);

```

```

    gettime(h2,m2,s2,s1002);
    time2:=3600*h2+60*m2+s2+s1002/100;
    dt:=time2-time1;
    delaynumber:=round(30000/dt*0.002);
    clrscr;
    gotoxy(30,24); write('Finished...');
    clrscr;
end;

Procedure Select_COM_address;
(* select a com port *)
var
    Number_COMs,i,Selected_COM:integer;
begin
    textbackground(blue);
    clrscr;
    writeln;
    writeln;
    if RS232(RS232(0))=0 then Number_COMs:=RS232(0)-1 else Number_COMs:=RS232(0);
    if RS232(RS232(0)-1)=0 then Number_COMs:=RS232(0)-2 else Number_COMs:=number_COMs;
    textcolor(Lightred);
    writeln('                RS232 Port Report of your PC');
    writeln;
    writeln;
    textcolor(yellow);
    write('                ', Number_COMs,' RS232 port(s) installed on your PC:');
    writeln;
    writeln;
    textcolor(green);
    for i:=1 to Number_COMs do writeln('                Base address of COM',i,' =
',RS232(i),' Decimal ');
    textbackground(Green);
    textcolor(magenta);
    repeat
        gotoxy(24,20); write('Input 1,2 or 3 to select a COM port : '); readln(Selected_COM);
    until (Selected_COM>=0) and (selected_COM<=4);
    Com_number:=Selected_COM;
    RS232_address:=RS232(Selected_COM);
end;

Procedure find_bit_weight;
{TO FIND THE BIT WEIGHT OF THE BIT}
begin
    if bitnumber=1 then outputbyte:=1; {find the bit weight of the selected bit}
    if bitnumber=2 then outputbyte:=2;
    if bitnumber=3 then outputbyte:=4;
    if bitnumber=4 then outputbyte:=8;
    if bitnumber=5 then outputbyte:=16;
    if bitnumber=6 then outputbyte:=32;
    if bitnumber=7 then outputbyte:=64;
    if bitnumber=8 then outputbyte:=128;
end;

{*****MAIN PROGRAM*****}
begin
    find_delay_number;
    Select_COM_address;
    dummy:=Write_data_format(RS232_address,96,0,8,2);
    { COM configuration 9600/4800/2400/1200 (9600 is selected)
      Data bit length:8, Parity Check: None, Stop Bit: 2 }
repeat
    textbackground(blue); textcolor(yellow);
    clrscr;
    writeln('                RS232 I/O card (SIO Card) testing program');
    writeln;
    write('                Input the bit number to be tested (1 to 8, 9 to quit) : ');
    readln(Bitnumber); writeln;
    textcolor(green+blink);
    gotoxy(1,24);
    write('                The selected bit changes from 0 to 1 to 0 repeatedly');
    textcolor(yellow);
    if bitnumber<>9 then
        begin
            Find_bit_weight;
            repeat
                dummy:=write_transmit_buffer(RS232_address,outputbyte);
                delay(200*delaynumber);
                gotoxy(30,20); writeln('Input byte into PC:
',read_receive_buffer(RS232_address):3);

```

```
        delay(200*delaynumber);
        dummy:=write_transmit_buffer(RS232_address,0);
        delay(200*delaynumber);
        gotoxy(30,20); writeln('Input byte into PC:
',read_receive_buffer(RS232_address):3);
        delay(200*delaynumber);
    until keypressed or (bitnumber=9);
    end;
    if bitnumber<>9 then readln;
    until bitnumber=9;
    clrscr;
end.
```

4.3 Visual Basic program

The card can be controlled using the MSCOMM control that is supplied with the Visual Basic. So there is no need to use external DLLs! Program in List 3 is an example of how to use some functions of the MSCOMM control. The user interface of this program is shown in Figures 4 and 5.

Program List 3

```
Dim Com_number As Variant

Private Sub Command1_Click()
Dim outbyte(1) As Byte

'Transmitting a byte from PC to the board
If Text1.Text > 255 Then Text1.Text = 255
outbyte(0) = (Text1.Text)
MSCOMM1.Output = outbyte

'Receiving a byte form the board
Do
Loop Until MSCOMM1.InBufferCount >= 1 'check if a byte is received by the PC
MSCOMM1.InputLen = 1 'read one byte a time
Label2.Caption = AscB(MSCOMM1.Input) 'AscB to convert input string into a binary
End Sub
Private Sub Command2_Click()
End
End Sub
Private Sub Form_Load()
Com_number = 0
Do
Com_number = (InputBox$("Input 1,2,3 or 4 to select" & Chr(13) & "COM1, COM2,COM3 or COM4", "Select a COM port"))
If Com_number = "" Then End
Com_number = Val(Com_number)
Loop Until Com_number <> 0

MSCOMM1.CommPort = Com_number
MSCOMM1.OutBufferSize = 1 'output buffer size=1
MSCOMM1.InputMode = comInputModeBinary
MSCOMM1.PortOpen = True
Label6.Caption = "Selected COM port: COM" & Com_number
End Sub
```

4.4 Using other programming languages

If other programming languages such as BASIC or C++ are used, users have to write their own I/O drivers.

5. DOS program - SIODos

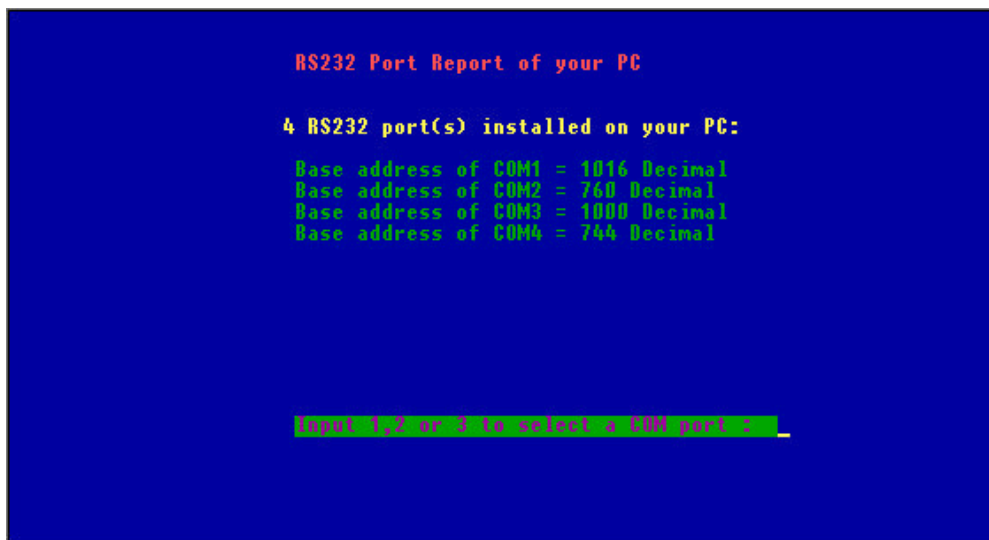
“SIODos” is a DOS-based user software. It allows users to test the functionality of the SIO-1 card.

3.1 Start the program

You start the “SIODos” program in the usual way. In DOS environment, type in “SIODos” in the directory containing the file and press Return. In Window 95/98 environment, the program can be started from START/RUN or from EXPLORER.

3.2 Select a Centronic port

The first screen reports the number of RS232 ports installed on your computer and allows you to select a port to be used with the SIO card. Key in “1”, “2” or “3” followed by a RETURN to select COM1, COM2 or COM3 (see Figure 2).



```
RS232 Port Report of your PC

4 RS232 port(s) installed on your PC:
Base address of COM1 = 1016 Decimal
Base address of COM2 = 760 Decimal
Base address of COM3 = 1000 Decimal
Base address of COM4 = 744 Decimal

Input 1,2 or 3 to select a COM port: _
```

Figure 2 The window reports number of COM ports installed and allows you to select one for the card

3.3 Main screen

After you have selected the COM port, a window shown in Figure 3 appears.

This window allows users to test each bit of the output ports. Input "1", "2" or "8" to select a bit to be tested. The selected bit produces a square wave (about 1 Hz) continuously. An LED, a logic probe or a multimeter can be used to check the logic status.

Data present at the input is read into the computer after a data transmission from the computer to the card is completed. The value of the received data is shown on the screen. If the output lines are connected the corresponding input lines, a loop test can be performed.

To test another bit, just press a key and a new bit is keyed in. Input "9" to quit the program.

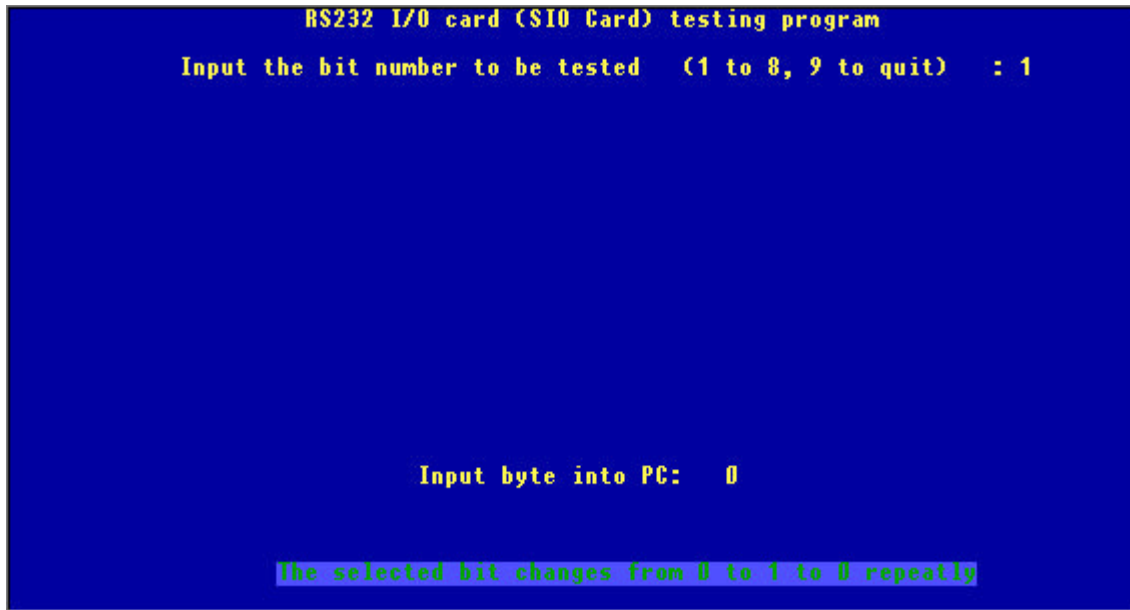


Figure 3 Window for testing output and input ports on the SIO card

6. Windows program - SIOWin

"SIOWin" is a Windows95/98 application software. It allows users to check the functionality of the SIO card in Windows environment.

6.1 Start the program

In Windows 95/98 environment, SioWin program is started by clicking **Start** and is followed by selecting **Run** in the Start menu. Then type in the program name with the full path (for example C:\SIO\SioWin). SioWin can be also started from the Windows Explorer.

6.2 Run the program

After the program starts, the window shown in Figure 4 appears on the screen. It allows users to select a COM port to be used with the SIO card. Click OK to continue.

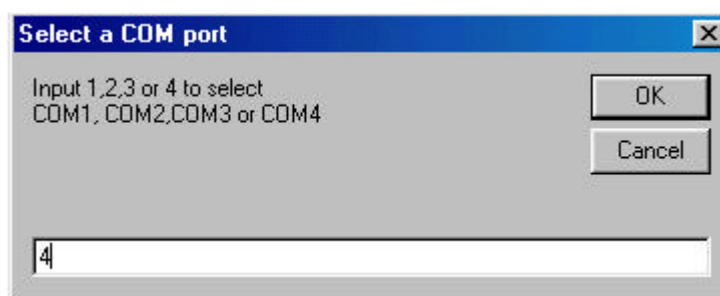


Figure 4 Window allowing users to select a COM port

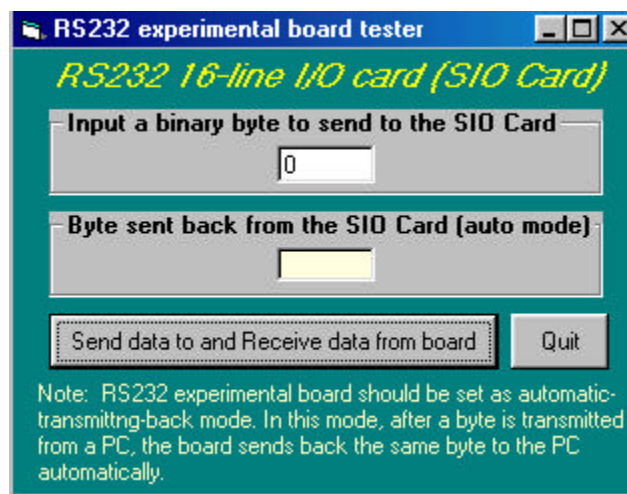


Figure 5 Windows for testing output and input ports of the SIO card

Figure 5 is the window which allows users to output data to the SIO card and read data from it. Data to be sent to the card is first keyed in. Then, click the command button "Send data to and Receive data from board". The program will send data to the SIO card and read data from the card.