serial i/o port

RS232 port goes parallel

Although a serial port is usually intended for, well, serial communication, it may be used in a different way. This article describes a simple BASIC program which turns the RS232 port on your PC into an interface with four parallel inputs and two parallel outputs. The investment: a few pence for components, and a little type work.

Computers normally have a parallel port (the Centronics interface to which the printer is attached) and a serial port (identified as 'RS232 interface'). In actual fact, both ports are parallel in essence. On the serial port, the control signals are, in fact, ordinary I/O lines whose logic level is under the control of instructions written to a serial controller IC. The logic levels of the control lines on the RS232 interface is an indication about the progress of the data transfer process. On most serial ports identified as 'RS232', the following digital I/O lines are used: DCD, DSR, RTS, CTS, DTR and RI.

The BASIC program shown here writes commands directly into a number of registers con-

1

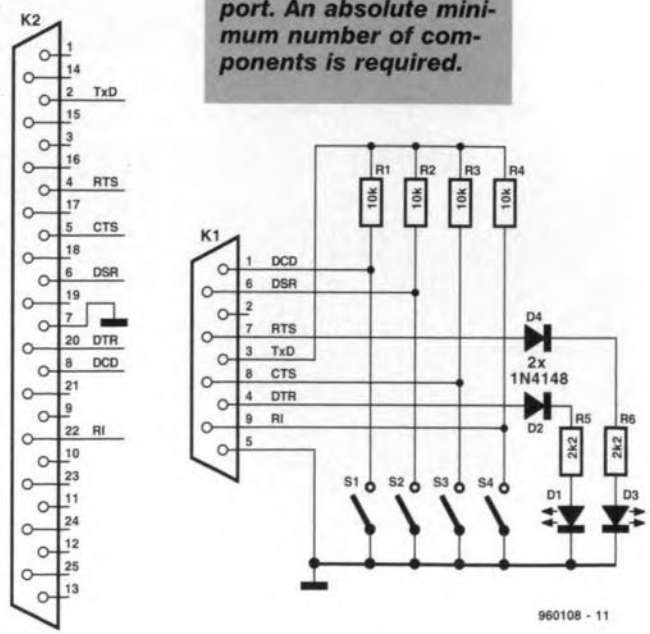
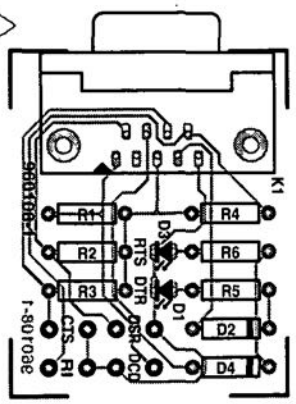


Figure 1. Circuit diagram of the serial I/O port. An absolute minimum number of components is required.

Design by J. Schuurmans

2



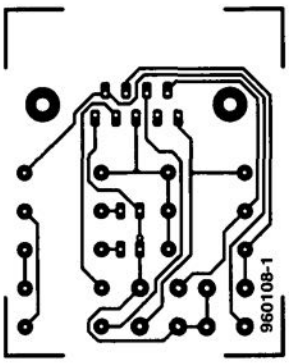


Figure 2. Track layout and component mounting plan of the printed circuit board designed for the project (board not available ready-made).

COMPONENTS LIST

Resistors:
R1,R2,R3,R4 = 10kΩ
R5,R6 = 2kΩ2

Semiconductors:
D1,D3 = LED
D2,D4 = 1N4148

Miscellaneous:
S1,S2,S3,S4 = single-pole on/off switch
K1 = 9-way sub-D socket, angled

tained in the serial controller, which is usually a UART type 16450, 16550 or 8250. These three controllers are identical as far as the register structure is concerned. The registers contain the bits that determine the logic level of each of the above mentioned I/O lines, allowing simple software to be applied to run the lot.

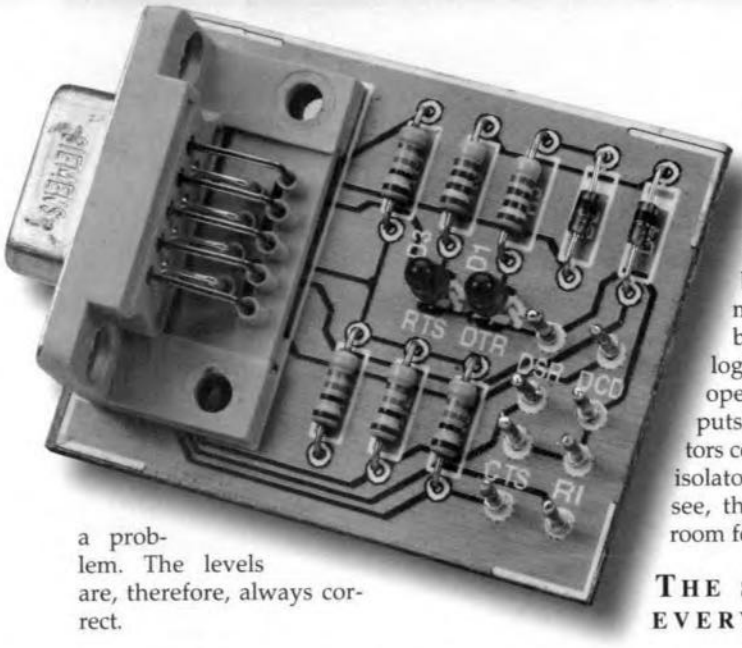
The drawing in **Figure 1** illustrates the use of the serial interface as an I/O port. The two outputs are formed by the RTS and DTR lines, the inputs, by the lines RI, CTS, DSR and DCD. Finally, TxD and GND are used to generate the necessary supply voltage. Only the RxD line is not used.

The electrical levels used by this I/O port depend on the hardware contained in the PC. A real RS232 interface uses logic levels which swing between +12 V and −12 V. There are, however, also PC manufacturers who use TTL levels instead of the ±12 V swing. Because the input levels at switches S1 through S4 is derived from the supply voltage, in other words, from the switching levels available on the RS232 interface, the absolute value will not be

Figure 3. Listing of the BASIC program which arranges all the control functions. Making your own modifications should not be a problem (program not available on disk).

3

```
10  CLS
20  ComAddress = &H2F8
40  OUT ComAddress + 3, INP(ComAddress + 3) OR 64
50  PRINT
60  PRINT TAB(24); CHR$(201); STRING$(30, CHR$(205)); CHR$(187)
70  PRINT TAB(24); CHR$(186); STRING$(30, CHR$(32)); CHR$(186)
80  PRINT TAB(24); CHR$(186); "    RS-232 Input/Output Port    "; CHR$(186)
90  PRINT TAB(24); CHR$(186); STRING$(30, CHR$(32)); CHR$(186)
100 PRINT TAB(24); CHR$(200); STRING$(30, CHR$(205)); CHR$(188)
110 PRINT
120 PRINT "Instructions:"; TAB(20); "Press 'R' to alter RTS"
130 PRINT "-------------"; TAB(20); "Press 'D' to alter DTR"
140 PRINT TAB(20); "Press 'Q' to quit"
150 PRINT : PRINT
160 PRINT "OUTPUT:", "RTS", "DTR"
170 PRINT "-------", "---", "---"
180 PRINT "State:",
190 IF INP(ComAddress + 4) AND 2 THEN PRINT "HIGH",  ELSE PRINT "LOW",
200 IF INP(ComAddress + 4) AND 1 THEN PRINT "HIGH" ELSE PRINT "LOW"
210 PRINT : PRINT
220 PRINT "INPUT", "DCD", "RI", "DSR", "CTS"
230 PRINT "-----", "---", "--", "---", "---"
240 PRINT "State:",
250 State = INP(ComAddress + 6) AND 240
260 IF State AND 128 THEN PRINT "HIGH",  ELSE PRINT "LOW",
270 IF State AND 64 THEN PRINT "HIGH",  ELSE PRINT "LOW",
280 IF State AND 32 THEN PRINT "HIGH",  ELSE PRINT "LOW",
290 IF State AND 16 THEN PRINT "HIGH" ELSE PRINT "LOW"
300 DO
310     LOCATE 20, 15
320     Previousstate = State AND 240
330     IF State <> Previousstate THEN
340         IF State AND 128 THEN PRINT "HIGH",  ELSE PRINT "LOW",
350         IF State AND 64 THEN PRINT "HIGH",  ELSE PRINT "LOW",
360         IF State AND 32 THEN PRINT "HIGH",  ELSE PRINT "LOW",
370         IF State AND 16 THEN PRINT "HIGH" ELSE PRINT "LOW"
380     END IF
390     A$ = UCASE$(INKEY$)
400     IF A$ = "R" THEN
410         LOCATE 15, 15
420         RTSState = INP(ComAddress + 4) AND 2
430         IF RTSState THEN
440           OUT ComAddress + 4, INP(ComAddress + 4) XOR 2
450           PRINT "LOW "
460         ELSE
470           OUT ComAddress + 4, INP(ComAddress + 4) XOR 2
480           PRINT "HIGH"
490         END IF
500     ELSEIF A$ = "D" THEN
510         LOCATE 15, 29
520         DTRState = INP(ComAddress + 4) AND 1
530         IF DTRState THEN
540           OUT ComAddress + 4, INP(ComAddress + 4) XOR 1
550           PRINT "LOW "
560         ELSE
570           OUT ComAddress + 4, INP(ComAddress + 4) XOR 1
580           PRINT "HIGH"
590         END IF
600     END IF
610 LOOP UNTIL A$ = "Q"
```



a problem. The levels are, therefore, always correct.

PRACTICAL MATTERS

The circuit is built on a small printed circuit board of which the track layout and component mounting plan are given in **Figure 2**. Considering the simplicity of the circuit, construction should not cause problems. Once finished, the board may be connected to the serial port on your PC. Where applicable, the switches may be replaced by end contacts, logic ports with open-collector outputs or phototransistors contained in opto-isolators. As you can see, there is plenty of room for experiments.

THE SOFTWARE: EVERYTHING IN BASIC

The (invisible) heart of the project is actually the serial controller in the PC. This device is usually mapped at one of the familiar base addresses like $2F8_H$ for COM1, and $3F8_H$ for COM2. In case you are unsure about the base address(es) used in your computer, use a hardware diagnosis program like MSD (MicroSoft Diagnostics) or CheckIt to collect relevant information. The register addresses that matter in the present application are [base+4] and [base+6]. **Table 1** tells you how the various bits contained in the registers are used to switch the I/O lines. Bit 6 at [base+3] enables the TxD line to be made high permanently ('set break'). This initialisation is performed at the start of the program.

The complete listing of the BASIC program we have in mind is given in **Figure 3**. Typing it into BASIC will not take too much time, we reckon. The base address of the COM port you intend to use is determined in line 20. The interface is switched on in line 30, when the TxD output is made permanently high.

The signal levels on the DTR and RTS lines are read in lines 190 and 200 respectively. Lines 260 through 290 enable the status of the various levels to be displayed on the monitor. The routine between lines 300 and 610 is repeated until the 'Q' key is pressed. In line 330, the software checks if the level at the inputs has changed. If so, the new states are copied to the screen. Line 390 checks to see if a key is pressed. In case the 'R' or 'D' key is pressed, the level of RTS or DSR is inverted, respectively. (960108)

Table 1. UART bit/register overview.

Address	bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
3	WLS0	WLS1	STB	PEN	EPS	Stick parity	**Set break**	DLAB
4	**DTR**	**RTS**	Out1	Out2	Loop	0	0	0
6	Delta CTS	Delta DSR	Trailing edge RI	Delta DCD	**CTS**	**DSR**	**RI**	**DCD**

CONSTRUCTION GUIDELINES

Elektor Electronics (Publishing) does not provide **parts and components other than** PCBs, fornt panel foils and software on diskette or IC (not necessarily for all projects). Components are usually available form a number of retailers – see the adverts in the magazine.

Large and small values of components are indicated by means of one of the following prefixes :

E (exa) = 10^{18}	a (atto) = 10^{-18}
P (peta) = 10^{15}	f (femto) = 10^{-15}
T (tera) = 10^{12}	p (pico) = 10^{-12}
G (giga) = 10^{9}	n (nano) = 10^{-9}
M (mega) = 10^{6}	μ (micro) = 10^{-6}
k (kilo) = 10^{3}	m (milli) = 10^{-3}
h (hecto) = 10^{2}	c (centi) = 10^{-2}
da (deca) = 10^{1}	d (deci) = 10^{-1}

In some circuit diagrams, to avoid confusion, but contrary to IEC and BS recommandations, the value of components is given by substituting the relevant prefix for the decimal point. For example,

3k9 = 3.9 kΩ 4μ7 = 4.7 μF

Unless otherwise indicated, the tolerance of resistors is ±5% and their rating is ⅓–½ watt. The working voltage of capacitors is ≥ 50 V.

In **populating a PCB**, always start with the smallest passive components, that is, wire bridges, resistors and small capacitors; and then IC sockets, relays, electrolytic and other large capacitors, and connectors. Vulnerable semiconductors and ICS should be done last.

Soldering. Use a 15–30 W soldering iron with a fine tip and tin with a resin core (60/40) Insert the terminals of components in the board, bend them slightly, cut them short, and solder: wait 1–2 seconds for the tin to flow smoothly and remove the iron. Do not overheat, particularly when soldering ICS and semiconductors. Unsoldering is best done with a suction iron or special unsoldering braid.

The value of a resistor is indicated by a **colour code** as follows.

color	1st digit	2nd digit	mult. factor	tolerance
black	–	0	–	–
brown	1	1	$\times 10^{1}$	±1%
red	2	2	$\times 10^{2}$	±2%
orange	3	3	$\times 10^{3}$	–
yellow	4	4	$\times 10^{4}$	–
green	5	5	$\times 10^{5}$	±0.5%
blue	6	6	$\times 10^{6}$	–
violet	7	7	–	–
grey	8	8	–	–
white	9	9	–	–
gold	–	–	$\times 10^{-1}$	±5%
silver	–	–	$\times 10^{-2}$	±10%
none	–	–	–	±20%

Examples:
brown-red-brown-gold = 120 Ω, 5%
yellow-violet-orange-gold = 47 kΩ, 5%

Faultfinding. If the circuit does not work, carefully compare the populated board with the published component layout and parts list. Are all the components in the correct position? Has correct polarity been observed? Have the powerlines been reversed? Are all solder joints sound? Have any wire bridges been forgotten?

If voltage levels have been given on the circuit diagram, do those measured on the board match them – note that deviations up to ±10% from the specified values are acceptable.

Possible corrections to published projects are published from time to time in this magazine. Also, the readers letters column often contains useful comments/additions to the published projects.