RS-232 SPLITTER

A. Rigby

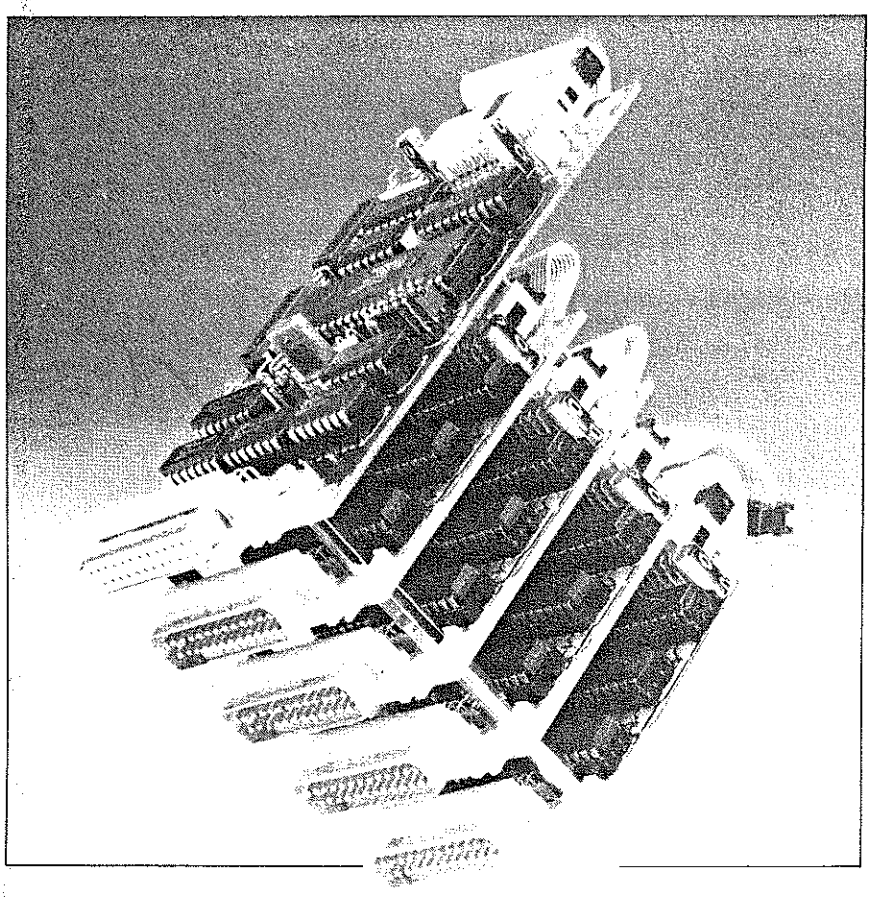
Among the computer peripherals that are typically connected to an RS232 port are mice, plotters, tracker-balls, digitizers, printers, scanners and modems. Not surprisingly, therefore, many computer users are forced to switch off their machine and perform the plug exchange trick when another peripheral is to be used. The RS232 splitter eliminates this annoying problem by allowing up to 256 (yes) serially controlled devices to be selected on 1 (yes) RS232 port. Switches and the like are not required, since the selection of the peripherals is effected by software.

There is a clear discrepancy between the serial interface capabilities of the average PC and the ever growing number of peripherals purchased or built by computer users. Most PCs offer only one RS232 port, while any extension beyond two of these is relatively expensive. As result, tangled cable nests may be found behind many a PC, since a serial port may usually serve one peripheral only.

Not surprisingly, many PC users would like to see a compact, inexpensive and simple-to-use switching system for a large number of peripherals. So-called switch-boxes offered commercially generally allow two peripherals to share one RS-232 port. Apart from the fact that an increase from one to two peripherals is not exactly spectacular, these boxes must be switched by hand, which requires that they are placed near the computer.

The RS-232 splitter has none of these advantages because:

- it can handle many more than two peripherals;
- it is controlled by the computer;
- it may be installed at a considerable distance from the computer.



RS-232, a flexible interface standard

The RS-232 port has been in use for many years, mainly by virtue of its flexibility, its ability to cover long distances, its transmission speed options and good noise immunity. The most practical boon is, however, that data may be carried over a single wire. In the most rudimentary configuration, a peripheral may be connected to a computer by three wires only: TXD for transmitted data, RXD for received data, and GND (ground). By contrast, a parallel connection requires at least 10 wires: 1 for ground, 8 for the data signals and 1 for the STROBE signal. At least 11 wires are required if handshaking is used, since in that case either BUSY or ACKNOWLEDGE must be added. Although the 3-wire serial link may use the XON/XOFF protocol —i.e., software — for handshaking, a hardware alternative is often preferred in view of speed and the use of simpler I/O routines. A hardware handshaking arrangement generally uses a number of control signals, while software handshaking is based on transmission and reception of certain words that control the dataflow in accordance with the speed of the computer and the peripheral.

Figure 1 shows the most commonly used RS-232 link. A computer is generally classified as DTE (data terminal equipment), and a peripheral as DCE (data communication equipment). A zero-modem connection may be used in those cases where a number of control signals is not available. Control- and data-lines may also be crossed to enable two computers (both DTE) to communicate via a serial link. The best known zero-modem connections are shown in Fig. 2. The RING INDICATOR (RI) line is not shown in these drawings because it is used on some modems only to signal to the computer that the auto-answer function has been actuated. The RI line is not normally used in applications other than with modems.

Switching by software

To guarantee compatibility with many RS-232 peripherals, all eight control lines on the interface must be switched automatically. Obviously, the ground line need not be switched since it is common to all equipment.

If, for instance, the computer is to be connected to a particular printer with a serial input, all control and data signals of a particular RS-232 outlet on the splitter

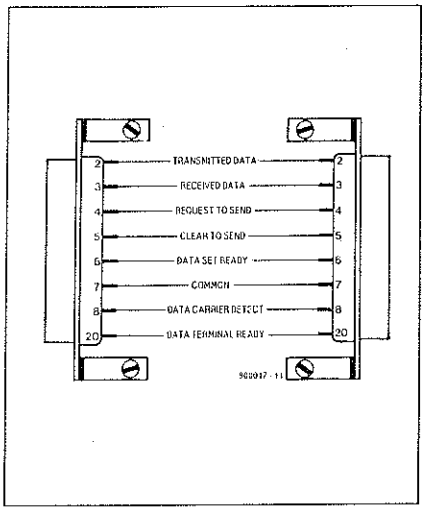


Fig. 1. Standard RS-232 connections.

(to which the printer is connected) are passed to the relevant pins of the computer's RS-232 port. All other equipment connected to the splitter is disconnected from this port.

The passing of signals between the computer and the selected peripheral must be done in the simplest way. In the case of the RS-232 splitter, this has been achieved by the use of software that transmits the peripheral selection code via the RS-232 port itself. Such a code is essentially a short trigger pulse which is sent by the computer. It is not recognized as data by the peripherals since the TXD line is briefly switched to a much higher transmission speed. The RS-232 splitter, however, does recognize the trigger pulse and is switched to receive and process the databyte that follows it. On reception of the trigger pulse, the splitter prevents the subsequent databyte being passed to the peripherals. Thus, channel selection happens unnoticed by the peripherals.

There are several ways to generate the trigger pulse. The actual method used depends on the computer type and your programming skills. Many computers allow the RS-232 port lines to be controlled by machine code or, say, a program language like C. A small program may be developed, for instance, to make the TXD line low for one clock pulse. If this does not work out, an alternative method may be to briefly select the highest bit rate on the RS-232 port and transmit character FF_H, which consists of a single low level (the start bit). The disadvantages of this approach are that the highest bit rate offered by the computer is no longer available for the peripherals, and that two *R*-*C* combinations must be changed in the RS-232 splitter. Some computers, such as the Commodore Amiga 500, offer a maximum bit rate as high as 250 kbit/s, which results in a single 4-µs long TXD pulse. This setting may be sacrificed to the above purpose of splitter channel control, since peripherals operating at 250 kbaud are rare.

Users of IBM PCs and compatibles are in the fortunate position of being able to order a disk which contains a simple peripheral channel selection program. The practical use of this program is simple: just enter the filename followed by the channel number of the desired peripheral.

How it works

The circuit in Fig. 3 contains the trigger pulse decoder and the channel selection logic. Every one serial channel requires one extension circuit, which is shown in Fig. 4. This is constructed on a small board and connected to the main board by a short length of flatcable. The number of extension boards may increase as the need for more serial channels develops.

The RS-232 interface of the circuit is shown to the left in Fig. 3. A total of ten buffers is used: five for input and five for output. The buffers are needed for signal level conversion since the circuit works with TTL levels (max. 5 V), while the RS-232 signals may swing between +12 V and −12 V. Note, however, that although a growing number of computers is capable of accepting TTL levels at the RS-232 port, there are good reasons to stick to positive and negative levels with a maximum of 12 V, mainly because these afford a larger noise margin.

Of all signals on the RS-232 connector, only TXD of the computer is used by the circuit —all other signals are simply switched through to the peripherals. The TXD signal, called S-IN after level conversion, triggers monostable multivibrator (MMV) IC_{5a} with each falling pulse edge. External components R_6-C_1 give the MMV a monotime of about 6 µs. If S-IN is still low after this monotime has lapsed, the pulse edge that triggered the MMV did not belong to a trigger pulse. The trigger pulse detection is, therefore, based on pulse-width comparison with the monotime. This is achieved with bistable IC_{7a}, whose clock input is connected to the Q output of IC_{5a}. During the falling pulse edge, i.e., at the end of the monotime, the bistable outputs take on levels determined by the information at the J- and K-inputs. The K-input is tied to ground. When the J-input is low owing to pulses shorter than 6 µs, the bistable output levels do not change. If, however, the pulse is shorter than 6 µs, the J-input will be high at the end. Output Q goes high to signal that the trigger pulse has been detected.

The MMV monotime is set to 6 µs to allow a safety margin of about 1.5 times between the pulse time (max. 4 µs) and the pulse detection time.

Gates IC_{6a}–IC_{6d} in combination with IC_{5b}, IC_{9a} and IC_{7b} keep the trigger pulse from being passed to the peripheral(s). Bistable IC_{5b} receives a clock pulse via IC_{6a}–IC_{6d} at every level change of S-IN. This clock pulse has a length of about 60 ns owing to the propagation delays of the HCT gates. Exactly 6 µs after the clock pulse, the Q output of IC_{5b} goes low again, so that J-K bistable IC_{7b} latches the levels present at its J- and K-input. Pulses shorter than 4 µs are not noted since they have been eliminated already and can not, therefore, appear in the serial output signal.

All other pulses that form serial data on the S-IN line are passed by the bistable. From the $\overline{Q}$ output, the S-IN signal is supplied to the peripherals via IC_{10a}. Since the Q output of IC_{7a} goes high on every trigger pulse, IC_{10a} prevents the subsequent data-

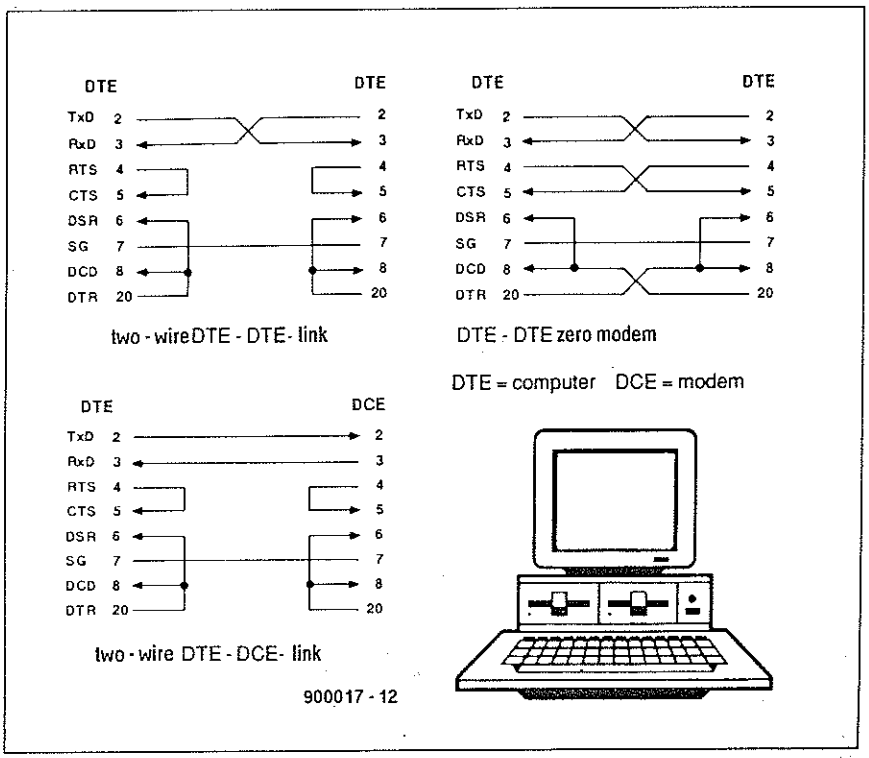


Fig. 2. In many cases, a full-blown RS-232 connection is not required. Shown here are three simple links between DTE and DCE, and between DTE and DTE (zero-modem).

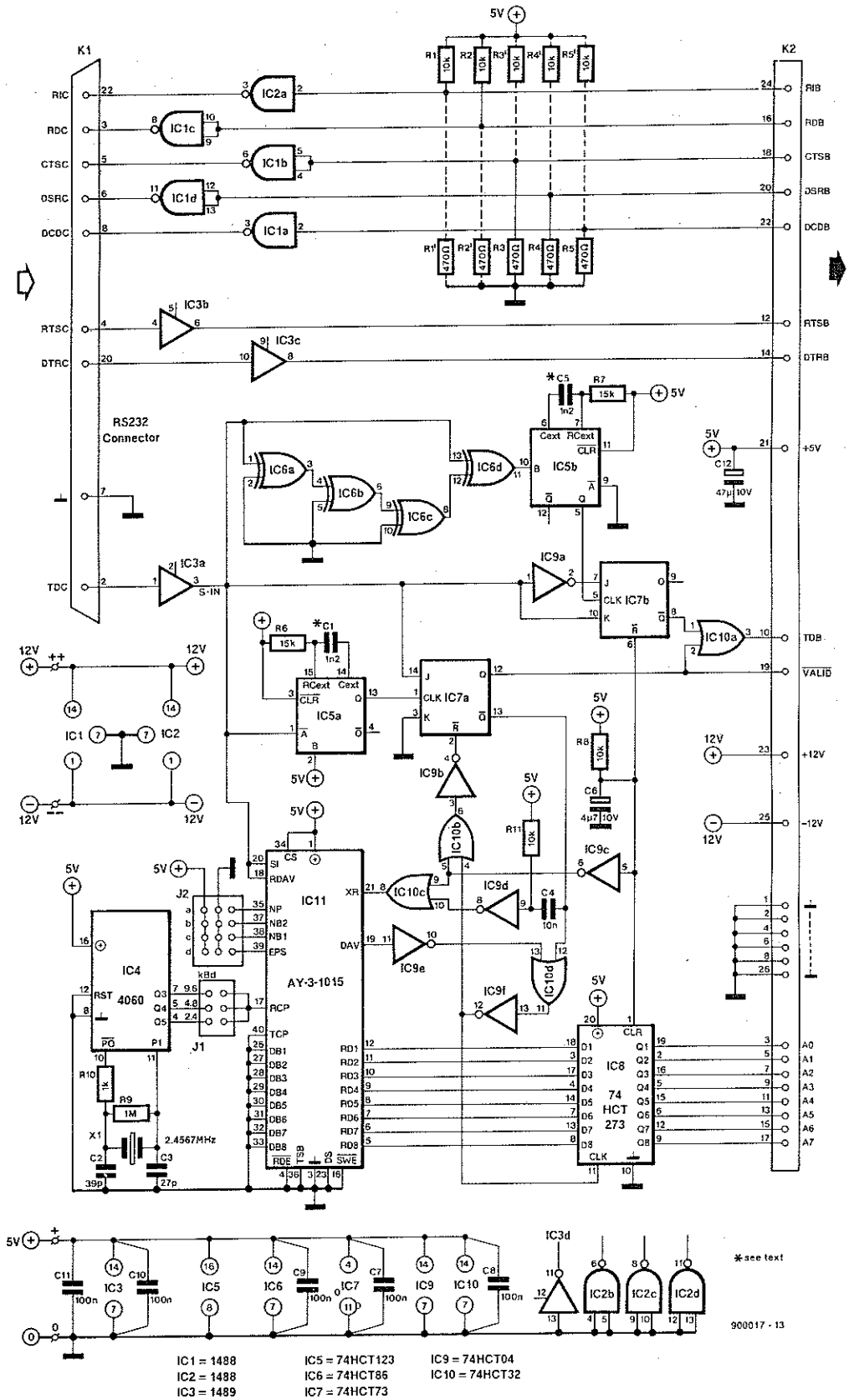


Fig. 3. Circuit diagram of the motherboard, which contains the trigger pulse decoder and channel selection logic to address individual extension boards.

word reaching the peripherals. This condition is ended as IC_{7a} is reset.

The channel number that follows the trigger pulse is decoded by a UART (universal asynchronous receiver/transmitter) Type AY-3-1015, which is an industry standard chip. After IC_{7a} has been set by the trigger pulse, the UART, IC_{11}, is reset via network R_{11}-C_4 and inverter IC_{9d}. This is done to ensure that it is cleared and ready for the reception of the dataword which determines the channel selection. The bit rate at which the dataword is sent by the computer is set with jumper J_1. Three bit rates, 2400 baud, 4800 baud and 9600 baud, are available from a quartz-crystal controlled oscillator/divider, IC_4. In practice, it will be convenient to use the same data rate for the channel selection code and the normal communication with the peripheral(s).

The DAV (data available) output of the UART goes high on reception of the channel selection code. The positive pulse edge causes the code available in parallel form to be latched into IC_8. The Q output of IC_{7a} prevents error pulses from the DAV output reaching IC_8. Error pulses may occur since the UART receives all serial data, whether these are channel selection codes or not. The high level at the $\overline{Q}$ output of bistable IC_{7a} prevents IC_{10d} blocking the DAV pulses. A valid DAV pulse resets IC_{7a}, so that the next pulses on the S-IN line are sent as normal data to the peripheral(s).

On power-up, network R_8-C_6 resets bistables IC_{7a}, IC_{7b}, register IC_8 and UART IC_{11}. This is done to make sure that the computer is connected to the peripheral on channel 0 when the system is switched on. The peripherals and the computer are not reset, however, so that the splitter may be switched on the moment it is required.

The circuit diagram of an extension board is given in Fig. 4. Every peripheral connected to the RS-232 splitter has its own extension board. Word comparator IC_{20} forms the channel code detector. Its P-inputs accept the decoded dataword from the computer, while its Q-inputs accept the pre-set channel number to be assigned to the associated peripheral. If the words match, pin 19 goes low. The channel preset is defined as a binary value with the aid of jumpers. Channel code 0 ($0000\ 0000_2$)is achieved by connecting all jumpers to ground, and channel code 255 ($1111\ 1111_2$) by connecting all jumpers to +5 V.

The actuated P=Q output of the 8-bit word comparator enables buffers IC_{23} and $IC_{21a\text{-}b\text{-}c}$ to establish the connection between the computer and the selected peripheral.

Level converters $IC_{22a\text{-}b\text{-}c}$, $IC_{24a\text{-}b\text{-}c}$ and IC_{25} ensure the correct voltage levels on the RS-232 output lines.

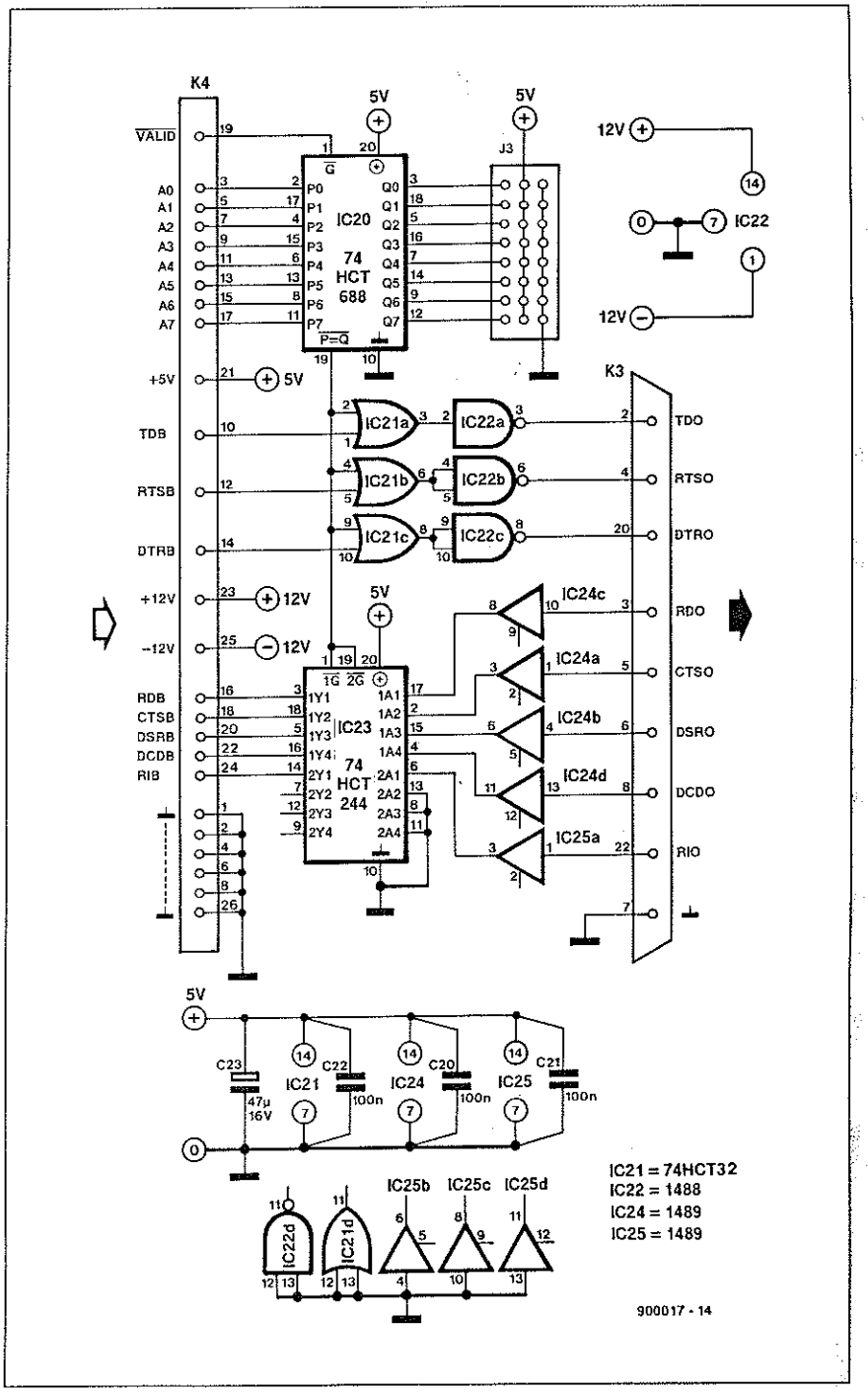


Fig. 4. Circuit diagram of extension board.

Construction

The track lay-outs and the component mounting plan of the main board are shown in Fig. 5. This board is double-sided, but not through-plated. On the ready-made printed-circuit board, the locations where a short piece of wire must be soldered as a through contact are indicated by silver-coloured pads at the component side. A few of these through contacts are made by soldering resistor terminals at the track side and the component side. If ICs are soldered direct on to the board, the white print on a few pads has to be removed. Fortunately, this is achieved almost automatically as a result of the heat developed during the soldering operation. The advantage of soldering the ICs direct is that the pins form the through-contacts, so that wires near the pins are not required. These holes must, however, have through-wires if IC sockets are used.

Do not start fitting parts on to the board before all through-contacts have been checked and found to be in order. The actual population of the main PCB is straightforward.

The extension board is shown in Fig. 6. It is single-sided and has 17 wire links, which must be fitted first. As with the main board, the decision whether or not to use IC sockets is up to you. The pin header block for the channel address setting is made either from three single-row pin headers or one dual-row and one single-row type. The extension board is

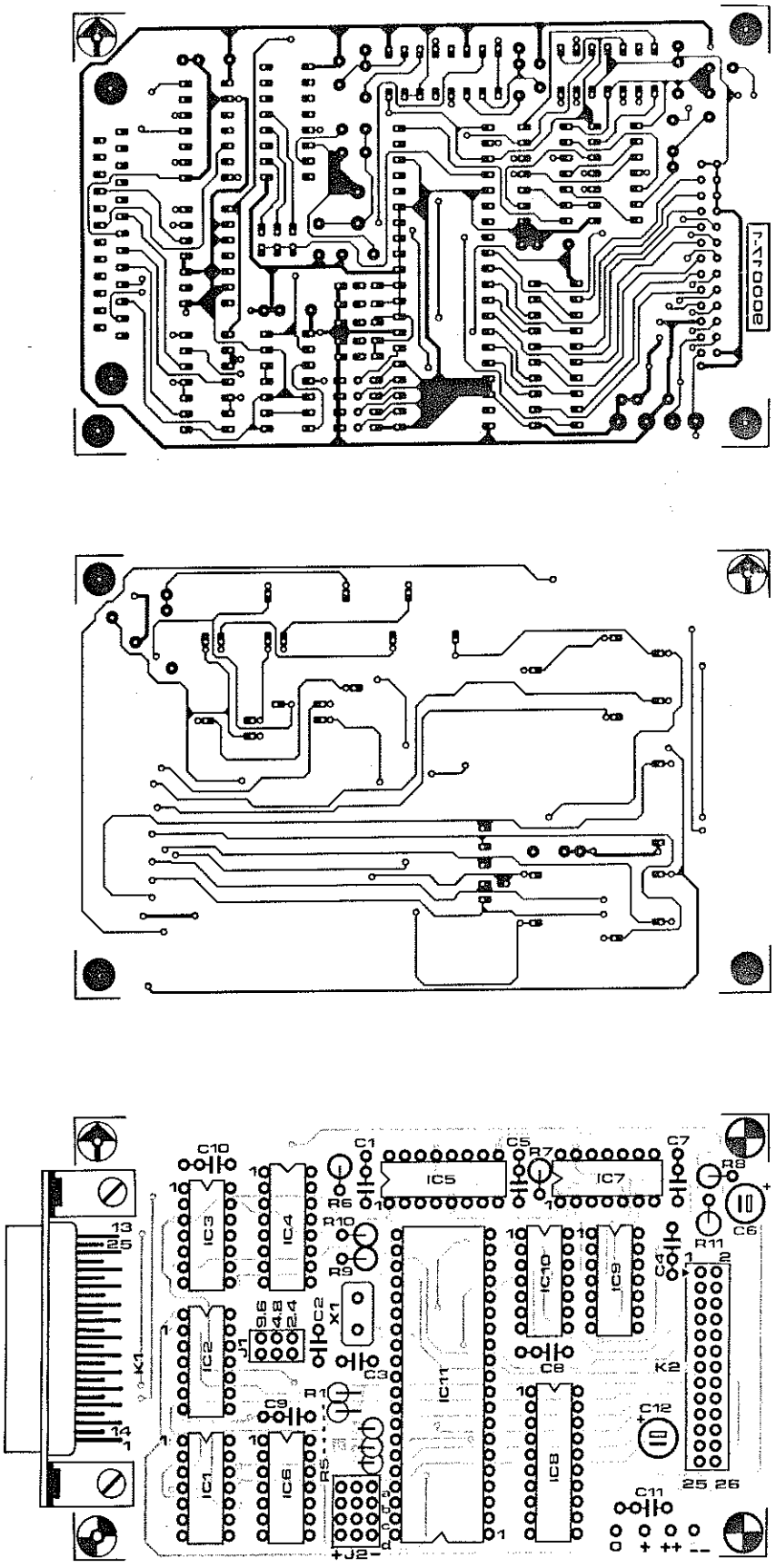


Fig. 5. Double-sided printed-circuit board for the central unit. Note that this board is not through-plated.

COMPONENTS LIST

MAIN BOARD

Resistors:

2	10k	R_1;R_2
3	470Ω	R_3;R_4;R_5
4	10k	R_6;R_7;R_8;R_{11}
1	1M0	R_9
1	1k0	R_{10}

Capacitors:

1	1n2	C_1
1	39p	C_2
1	27p	C_3
1	10n	C_4
1	1n2	C_5
1	4µ7 16 V radial	C_6
5	100n	C_7–C_{11}
1	47µ 16 V radial	C_{12}

Semiconductors:

2	1488	IC_1;IC_2
1	1489	IC_3
1	4060	IC_4
1	74HCT123	IC_5
1	74HCT86	IC_6
1	74HCT73	IC_7
1	74HCT273	IC_8
1	74HCT04	IC_9
1	74HCT32	IC_{10}
1	AY-3-1015	IC_{11}

Miscellaneous:

1	quartz crystal 2.4576 MHz	X_1
1	25-way angled sub-D connector (female)	K_1
1	26-way angled pin header	K_2
1	6-way pin header	J_1
1	pin header block; 3 off 4-pin contact strips	J_2
8	jumper	
1	PCB	900017-1

COMPONENTS LIST

EXTENSION BOARD
(one required for each RS-232 peripheral)

Capacitors:

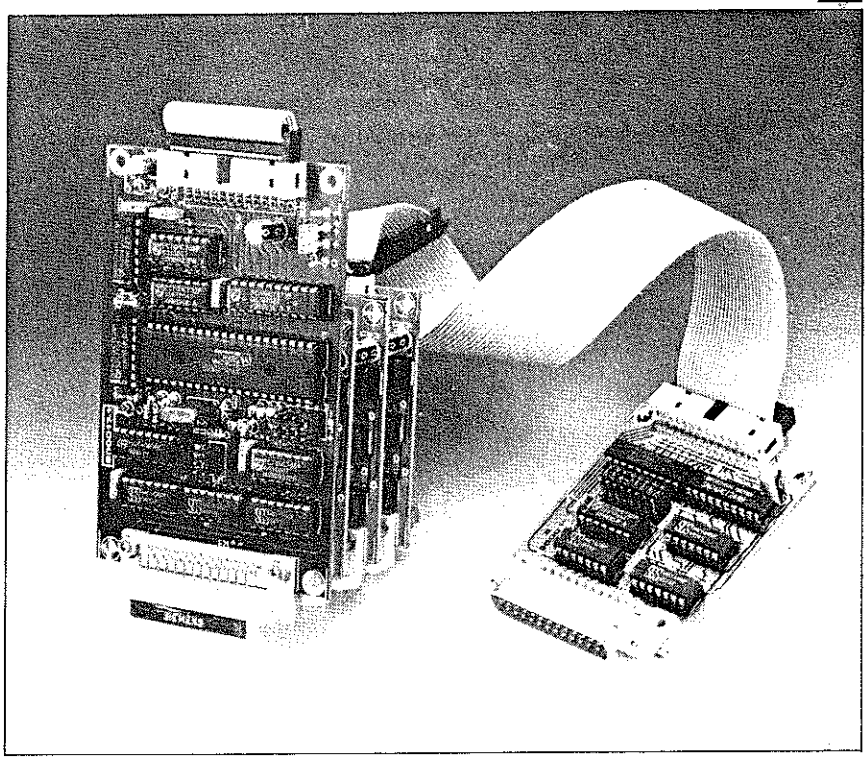
3	100n	C_{20};C_{21};C_{22}
1	47µ 16 V tantalum	C_{23}

Semiconductors:

1	74HCT688	IC_{20}
1	74HCT32	IC_{21}
1	1488	IC_{22}
1	74HCT244	IC_{23}
2	1489	IC_{24};IC_{25}

Miscellaneous:

1	25-way angled sub-D connector (male)	K_3
1	26-way angled PCB header	K_4
1	pin header block; 3 off 8-pin contact strips	J_3
1	PCB	900017-2



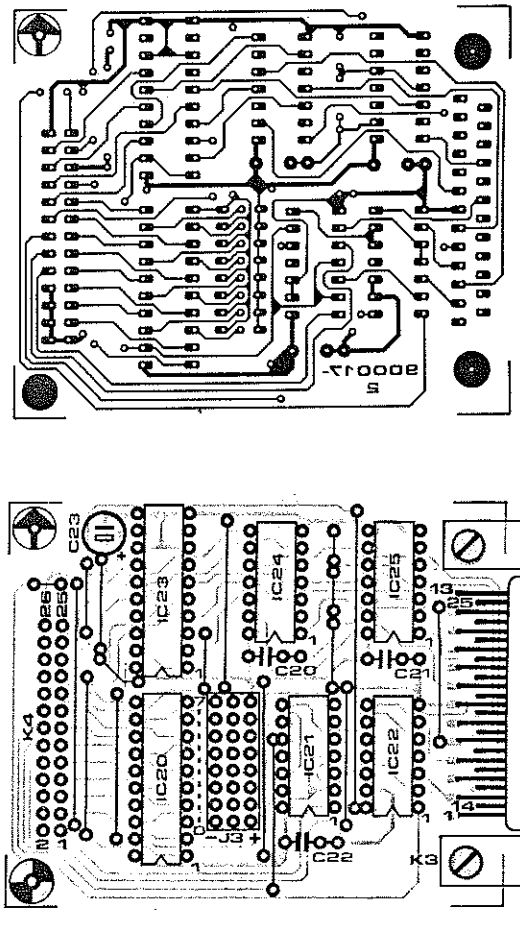


Fig. 6. Printed-circuit board for the extension (one required for each RS-232 peripheral).

connected to the main board via K_4, and to the peripheral via K_3.

Power supply and serial data format

The power supply for the circuit must have +5-V, +12-V and –12-V outputs. The current required by the circuit depends on the number of extension boards installed. In some cases, the supply voltages may be obtained from the PC.

The serial data format (start/stop bits, and parity bit) is set with jumpers J_{2b}, J_{2c} and J_{2d}. The most widely used format, one startbit, eight databits, no parity and 1 stopbit, is set with jumpers J_{2b}, J_{2c} and J_{2d} to +5 V, and J_{2a} to ground.

Software and component values

The hardware is not complete without a computer program that provides the required trigger pulse and the channel selection code to actuate a particular channel (0–255). Since the control of the TXD line is specific to the type of computer and its RS-232 interface, the model program given in Fig. 8 is based on the use of the highest baud rate for the generation of the trigger pulse. The BASIC program is simple by almost any standard, and should not be too difficult to adapt for a particular type of computer or interpreter. The bit rate for the trigger pulse is set to 9,600, so that the highest possible bit rate for normal use of the peripheral(s) is 4,800 at the highest.

If bit rates other than the ones mentioned above are used, networks R_7-C_5 and R_6-C_1 must be modified accordingly. The values shown in the circuit diagram are for a trigger pulse shorter than 4 µs. If a longer trigger pulse is used, the new *R*-*C* values may be calculated on the basis of the monotime, τ, obtained from

$$\tau = 0.45RC \text{ (ns)}$$

where *R* is in kilo-ohms and *C* in pico-farads. Remember that *C* must be greater than 10,000 pF (10 nF). Also note that the calculation is valid for a 74HCT123 only, **not** for a 74123 or 74LS123, which must not be used here.

If, for example, a bit rate of 9,600 is chosen, the associated pulse time is

$$1/9600 \text{ s} = 104 \text{ µs}$$

To ensure a safe margin between the pulse time and the monotime, the latter is made roughly 1.5 times longer, i.e., 150 µs. The shortest pulse time that can occur in a datastream of 4,800 bits/s is 210 µs, i.e., much longer than the monotime of 150 µs. Starting from a capacitor value of 12 nF (12,000 pF), an associated resistance of

$$150{,}000/(0.45 \times 12{,}000)$$

or roughly 27 kΩ should give adequate results.

In conclusion, for a system in which the trigger pulse is transmitted at 9,600 baud, and the peripheral data at 4,800 baud, both networks R_6-C_1 and R_7-C_5 consist of 27-kΩ resistors and 12-nF capacitors. ■

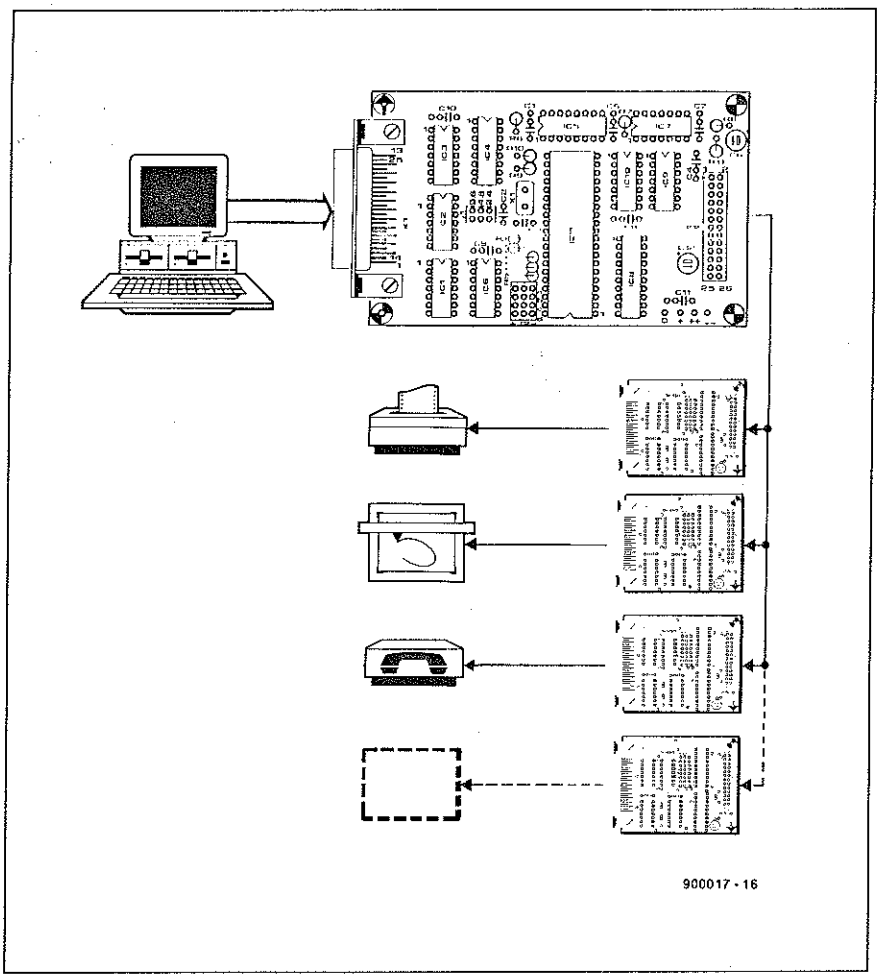


Fig. 7. Basic connections between the PC, the motherboard, the extension cards and the RS-232 peripherals (example).

UART SETTINGS

data format	J2b	J2c
5 bit	gnd	gnd
6 bit	gnd	+
7 bit	+	gnd
8 bit	+	+

parity bit	J2d
enable	+
disable	gnd

parity	J2a
even	+
odd	gnd

```
100 CLOSE
110 REM 9600 baud, no parity, 8 bits, 1 stopbit
120 OPEN "com1:9600,N,8,1" AS #1
130 INPUT "RS232 address",J
140 PRINT #1,CHR$(255); : REM SEND PULSE
150 PRINT #1,CHR$(J); :   REM SEND ADDRESS
160 CLOSE
170 REM 4800 baud, no parity, 8 bits, 1 stopbit
180 OPEN "COM1:4800,N,8,1" AS #1
190 FOR I=&H20 TO 80
200 PRINT #1,CHR$(I); : REM PRINT CHARACTER
210 NEXT
220 PRINT #1,CHR$(13),CHR$(10) : REM END OF LINE
230 GOTO 100
```

900017 - 15

Fig. 8. Model BASIC control program to control the serial port of an IBM PC or compatible.