ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/256537289

# PC-to-PC communication via RS-232 serial port using C

**Article** *in* Electronics World and Wireless World · January 2006

CITATIONS
0

READS
21,075

**1 author:**

Varun Jindal
Indian Institute of Management Calcutta
**11** PUBLICATIONS **1** CITATION

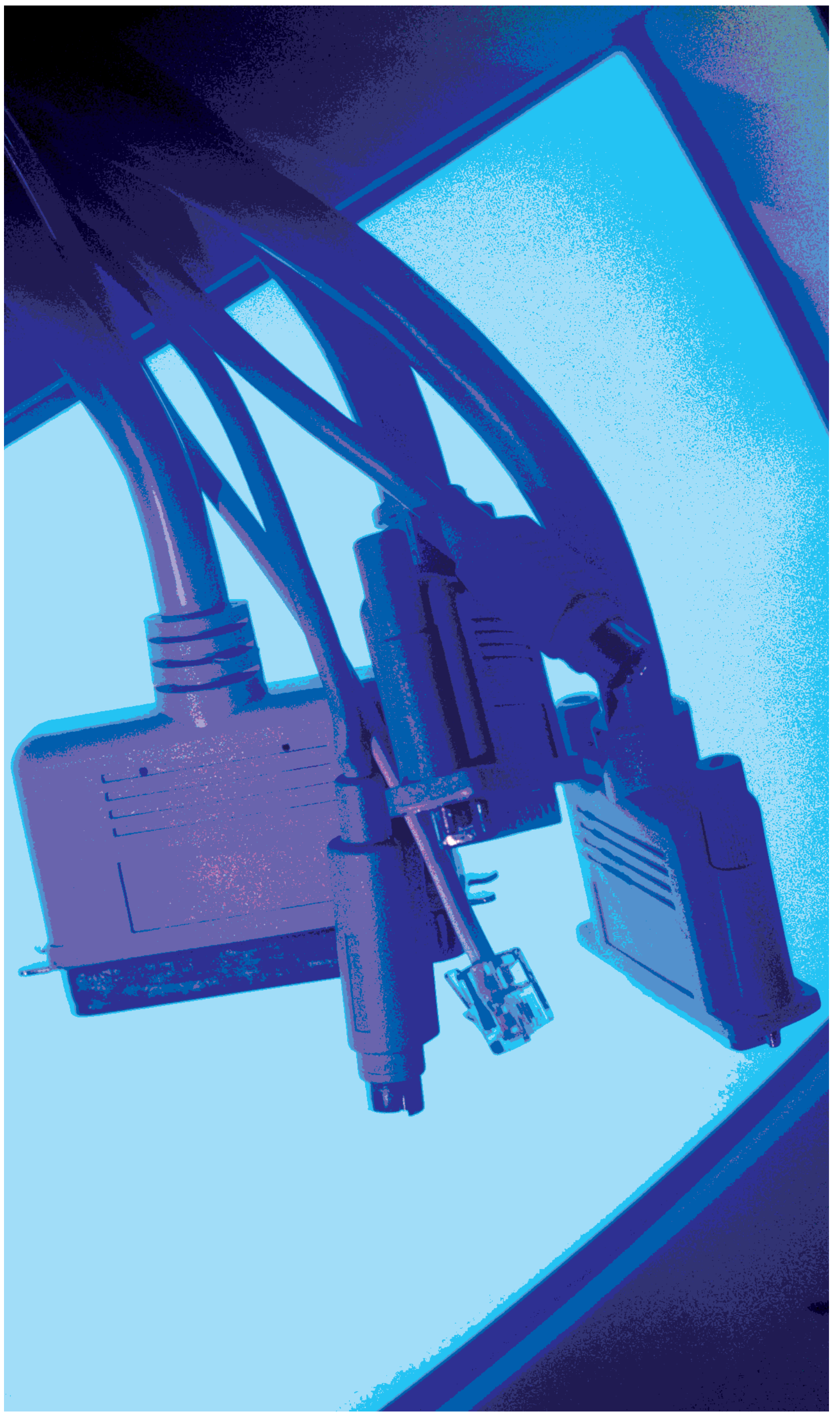
SEE PROFILE

All content following this page was uploaded by Varun Jindal on 27 January 2015.

The user has requested enhancement of the downloaded file.

# PC-to-PC Communication via RS-232 Serial Port Using C

**Varun Jindal** from the Panjab University's Institute of Engineering and Technology delves into programming via an RS-232 port in C



An endeavor has been made in this article to bring forth a simple, easy and novel way of implementing PC-to-PC communication via RS-232 serial port using C language. Implementing the asynchronous serial communication this way does not require the reader to be familiar with serial port registers and their programming, and there is no need for constructing user-defined functions for setting the baud rate and format of data, parity and stop bits. Moreover, the speed of data transfer is also greater and the function used for serial programming along with its arguments makes its purpose self-explanatory.

A PC can accommodate, at most, four serial ports but usually a PC has two RS-232 serial ports, COM1 and COM2. Any one of the serial ports can be used in each PC for linking them together. A serial port at the back of a PC is in the form of 9-pin (or sometimes 25-pin) D-type male connector. **Table 1** shows pin configurations of 9-pin D-type male connector, which is depicted in **Figure 1**.

## Serial communication

Data transfer within a system is generally in parallel. All the bits of the data word are transferred in parallel at the same instant. In some cases, particularly in transferring data over long distances, it is preferred to transfer the data in serial form. The data word from a transmitting system is converted to stream of bits by parallel to serial conversion, and one bit at a time is transferred on a single line to a receiving system. At the receiving end, the word is reconstructed by serial to parallel conversion. The speed of data transfer in serial communication is specified by baud.

The baud unit is named after Jean Maurice Emile Baudot, who was an officer in the French Telegraph Service. He is credited with devising the first uniform-length 5-bit code for characters of the alphabet in the late 19th century. What baud really refers to is modulation rate or the number of times per second that a line changes state. This is not always the same as bits per second (bps). If we connect two serial devices together using direct cables then baud and bps are, in fact, the same. But when modems are in question, this isn't the case.

### ➢ Asynchronous serial communication

Asynchronous data transfer is used for low speed communication, typically at standard rates such as

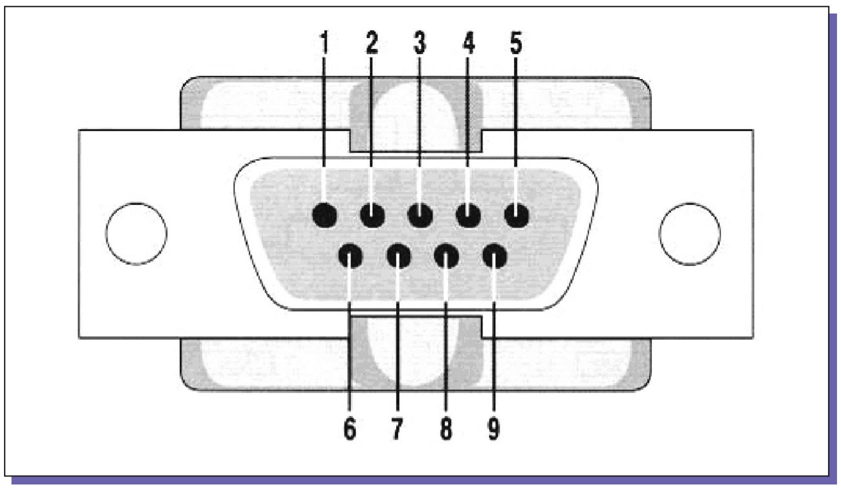
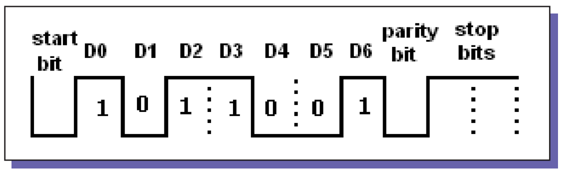


**Figure 1 (Above)**: 9-pin D-type connector of an RS-232 serial port

**Figure 2 (Left):** Asynchronous data transmission format



2400, 4800, 9600, 19200 baud etc. The asynchronous communication format does not use any synchronising clock or timing signal.

**Framing** – Transmission of a character starts with a start bit (logic 0), followed by the character bits (LSB first), a parity bit and ends with one or two stop bits (logic 1). This is referred to as one frame. Process of adding the start, parity and stop bits with character bits is referred to as framing. When no character is sent, the transmitter outputs logic high. The line remains in logic 1 (idle state) till the transmission of next character begins with another start bit. **Figure 2** shows transmission of a 7-bit ASCII character 'M'.

The parity bit is included in the frame for the receiver to check errors that may occur during transmission. The bit is made 0 or 1, so that the number of 1s in the character plus the parity bit is always odd in odd parity systems or even in even parity systems. Since, the character 'M' has even number (4) of 1s, the parity is made 0 for the even parity system and 1 for the odd parity system.

**Error detection** – Error in asynchronous communication is detected in three ways, by checking parity error, framing error and overrun error. The parity error informs that the received data has wrong parity, indicating that the noise was encountered during reception. The framing error informs that the received data does not have the start and stop bits at their proper places. The overrun error indicates that a new data has been received before the previous data could be taken away.

**Table 1: Pin functions of a 9-pin connector of RS-232 serial port**

| *Pin number* | *Description* |
|---|---|
| 1 | DCD (data carrier detect) |
| 2 | RxD (receive data) |
| 3 | TxD (transmit data) |
| 4 | DTR (data terminal ready) |
| 5 | GND (signal ground) |
| 6 | DSR (data set ready) |
| 7 | RTS (request to send) |
| 8 | CTS (clear to send) |
| 9 | RI (ring indicator) |

## RS/EIA-232

Short for Recommended Standard-232, a standard interface approved by EIA (Electronic Industries Association) for connecting serial devices, specifies signal voltages, signal timing, signal function, a protocol for information exchange and mechanical connectors. To ensure reliable communication and to enable the interconnection of equipment produced by different manufacturers, the interfacing standard RS-232 was set by EIA in 1960. Since then it has gone through a number of modifications, including a change in its name. RS-232A, RS-232B, RS-232C, EIA-232D and EIA-232E are the subsequent versions of this standard. The standard has been referred to as RS-232 (instead of EIA-232) throughout this article due to its popularity.

The RS-232 standard supports two types of connectors – a 25-pin D-type connector (DB-25) and a 9-pin D-type connector (DB-9). The type of serial communications used by PCs requires only nine pins, so either type of connector will work equally well. Since modern PCs employ only 9-pin D-type connectors, only this configuration has been discussed in this article, including connections and programming.

In RS-232 parlance, the device that connects to the interface is called Data Communications Equipment (DCE) and the device to which it connects is called Data Terminal Equipment (DTE). This standard was mainly designed to connect DTE that is sending and receiving serial data (such as a computer) and DCE that is used to send data over long distances (such as a modem).

**To distinguish between DTE and DCE:**

- Measure the DC voltages between (DB-9) pins 3 and 5 and between pins 2 and 5. Be sure that the black lead is connected to pin 5 (GND) and the red lead to whichever pin you are measuring.
- If the voltage on pin 3 (TxD) is more negative than -3V, then it is a DTE, otherwise it should be near zero volts.
- If the voltage on pin 2 (RxD) is more negative than -3V, then it is a DCE.
- If both pins 3 and 2 have a voltage of at least 3V,

EW Jan p25-29 PCtoPC 17/11/05 12:10 Page 27

then either you are measuring incorrectly, or your device is not a standard RS-232 device. Call technical support.

- In general, a DTE provides a voltage on TxD, RTS & DTR, whereas a DCE provides voltage on RxD, CTS, DSR & DCD.

## Programming an RS-232 serial port using C

**Library File Inclusion**
#include<bios.h>

**Function**
_bios_serialcom();

**Function Declaration/Syntax**
Unsigned _bios_serialcom(int cmd, int port, char abyte);

**Brief Description**
The function _bios_serialcom() uses BIOS interrupt 0x14 to perform various RS-232 communications over the I/O port given in the port. The function arguments along with their significance are given in **Table 2** below.

**Table 2: Various function arguments and their significance**

| *Argument* | *Significance* |
|---|---|
| abyte | OR combination of bits that specifies COM port settings |
| cmd | Specifies the I/O operation to perform |
| port | Identifies the I/O port |

**Function Argument Specifications**
**Port argument** – The serial port that is selected for RS-232 communication is specified in the port argument as given in **Table 3**.

**Table 3: Port argument specification**

| *Value of 'port' argument* | *Port selected* |
|---|---|
| 0 | COM1 |
| 1 | COM2 |
| 2 | COM3 |
| 3 | COM4 |

**Cmd Argument** – The I/O operation to be performed is specified by means of 'cmd' argument as given in **Table 4**. When the value of 'cmd' argument is set to either _COM_RECEIVE or _COM_STATUS, the value in 'abyte' argument is ignored.
**Abyte argument** – When the value of 'cmd' argument is set to _COM_INIT, the COM port settings are specified by the 'abyte' argument. The 'abyte' argument is an OR combination of the following values (one from each group in **Table 5**).

**Table 4: Cmd argument specifications**

| *Value of 'cmd' argument* | *Function performed* |
|---|---|
| _COM_INIT | Sets the communication parameters to the value in 'abyte' argument |
| _COM_SEND | Sends the character in 'abyte' argument out over the communications line |
| _COM_RECEIVE | Receives a character from the communications line |
| _COM_STATUS | Returns current status of the communications port |

**Table 5: Abyte argument specifications**

| *Value of 'abyte' argument* | *Meaning* |
|---|---|
| _COM_CHR7 | 7 data bits |
| _COM_CHR8 | 8 data bits |
| _COM_STOP1 | 1 stop bit |
| _COM_STOP2 | 2 stop bits |
| _COM_NOPARITY | No parity |
| _COM_ODDPARITY | Odd parity |
| _COM_EVENPARITY | Even parity |
| _COM_110 | 110 baud |
| _COM_150 | 150 baud |
| _COM_300 | 300 baud |
| _COM_600 | 600 baud |
| _COM_1200 | 1200 baud |
| _COM_2400 | 2400 baud |
| _COM_4800 | 4800 baud |
| _COM_9600 | 9600 baud |

## Return value

For all values of 'cmd' argument, the function _bios_serialcom() returns a 16-bit unsigned integer.
The upper 8 bits of the return value are status bits.

✔ If one (or more) error status bit(s) is (are) set to 1, an error has occurred.
✔ If no error status bit is set to 1, the byte was received without error.

The lower 8 bits vary depending upon the value of 'cmd' argument specified as given in **Table 6**.
The following format shows the details of all the return bits:

**Lower Byte of Return Value**
D0 = Received line signal detect
D1 = Ring indicator
D2 = Data set ready
D3 = Clear to send
D4 = Change in receive line signal detector
D5 = Trailing edge ring indicator
D6 = Change in data set ready
D7 = Change in clear to send
**Upper Byte of Return Value**
D8 = Data ready

**Table 6: Significance of lower byte of return value**

| *Value of 'cmd' argument* | *Lower 8 bits of return value* |
|---|---|
| _COM_INIT or _COM_STATUS | Lower bits are defined as shown in format given below. |
| _COM_SEND | . . . |
| _COM_RECEIVE | Byte read is in the lower bits of the return value (if there is no error i.e. no upper error-bits are set to 1) |

| | |
|---|---|
| D9 | = Overrun error |
| D10 | = Parity error |
| D11 | = Framing error |
| D12 | = Break detect |
| D13 | = Transmit holding register empty |
| D14 | = Transmit shift register empty |
| D15 | = Time out (set to 1 if abyte value could not be sent) |

## Description steps

**1. Objective** – To implement PC-to-PC communication by inputting the data through one port and receiving the same through another and vice-versa (Turbo C++ version 3.0).

**2. Connection diagram** – The connection diagram (**Figure 3**) shows the wiring of the null-modem that is intended for RS232 asynchronous communications (most PC-based systems). This configuration is called so because each PC terminal detects as if some modem is connected to it rather than the other PC.

The two PC terminals are connected through TxD, RxD and GND pins. The Data Terminal Ready (DTR, pin 4) is looped back to Data Set Ready (DSR, pin 6) and Data Carrier Detect (DCD, pin 1) on both PCs. When DTR is asserted active, then the DSR and DCD immediately become active. At this moment, the computer thinks the virtual modem to which it is connected is ready and has detected the carrier of the other modem. The lines Request to Send (RTS, pin 7) and Clear to Send (CTS, pin 8) have been linked together. When the computer wishes to send data, it asserts RTS high, which in turn asserts CTS high, meaning thereby that the virtual modem has the room for storing the data and the computer can send it.

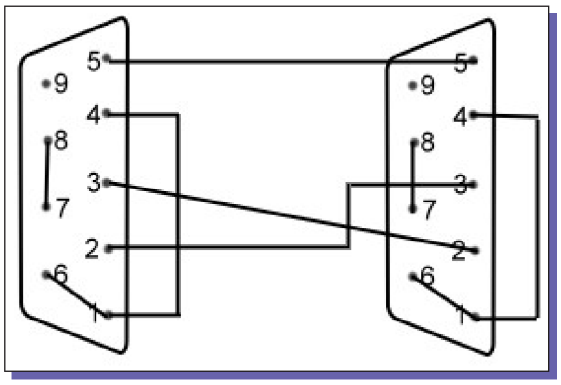


**Figure 3**: Null modem cable configuration

**3. Source code**

```
/* PC-to-PC communication – by VARUN JINDAL */
/* B.E. (E&C) - final year, Panjab University, Chandigarh */
#include<stdio.h>
#include<conio.h>
#include<bios.h>

#define SETTINGS (_COM_9600 | _COM_CHR8 | _COM_NOPARITY |
_COM_STOP1)
/* baud rate = 9600, 8 data bits, no parity bit, 1 stop bit */

void main(void)
{
 unsigned in,out,status;
 int port;

 clrscr();
 printf("Select Port(Enter '0' for COM1 and '1' for COM2):");
 scanf("%d",&port);
 printf("Press ESC to exit");
 textcolor(YELLOW);
 cprintf("\n\rData Received:");

 _bios_serialcom(_COM_INIT,port,SETTINGS);

 for(;;)
 {
 status=_bios_serialcom(_COM_STATUS,port,0);
 if (status & 512)
  printf("\n\t\a Overrun Error");
 if (status & 1024)
  printf("\n\t\a Parity Error");
 if (status & 2048)
  printf("\n\t\a Framing Error");
 if(status & (512|1024|2048)) /* if any error */
  break;

 if(status & 256)  /* if data ready */
 {
  if((out=_bios_serialcom(_COM_RECEIVE,port,0) & 255)!=0)
           putch(out);
 }
 if(kbhit()) /* if a keystroke is currently available */
 {
  in=getch(); /* get a character without echoing onto the screen */
  if(in==27) /* if ESC */
           break;
  _bios_serialcom(_COM_SEND,port,in);
 }
 }
}
```

EW Jan p25-29 PCtoPC 17/11/05 12:10 Page 29

**4. Testing** – It is usually difficult to work on both PCs when a programmer wishes to check his/her source code for PC-to-PC communication. The best possible way to overcome this problem is to use a loopback connector (shown in **Figure 4**), which enables the programmer to write source-code for programming serial port with single PC.

A loopback connector usually consists of a connector without a cable and includes internal wiring to re-route signals back to the sender. When the computer receives data, it will not know whether the signals it receives come from a remote DCE device set to echo characters, or from a loopback connector. Using loopback connector, proper operation of the computer's serial port can be checked.

**5. Data transfer procedure**

- Connect the two PCs together using the 3-wire link.
- Run the program given in the source code on both PCs.
- Before sending the data, set both the sending and receiving PC terminals to the same baud rate, and same format of data bits, parity bits and stop bits using the macro (pre-processor directive) 'SETTINGS' in the source code.
- When the source code is compiled and run on both PCs, the characters typed in one computer should appear on the other computer's screen and vice-versa.

**6. Limitations** – Here, we are using RS-232 serial asynchronous communication, so the communication speed is less than that of parallel data transfer, where 8-bit data is sent at a time rather than bit by bit.

**7. Applications** – On the guidelines presented in this article, useful functions such as file transfer, chatting etc can be implemented. By using hardware circuitry employing infrared/laser diodes, even wireless PC-to-PC communication can be easily implemented.

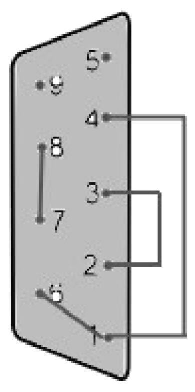


**Figure 4**: Loopback connector

**Designers' note:**
**The executable code is available on request**

Telnet

**Quality second-user test & measurement equipment**
**Tel: 02476 650 702 Fax: 02476 650 773**
**Web: www.telnet.uk.com Email: sales@telnet.uk.com**
**All equipment is used – with 30 days guarantee and 90 days in some cases. Add carriage and VAT to all goods.**
**1 Stoney Court, Hotchkiss Way, Binley Industrial Estate Coventry CV3 2RL ENGLAND**

TELNET

| | |
|---|---|
| Agilent (HP) 3314A Function Generator 20 MHz | £650 |
| Agilent (HP) 3325A and B function gen. from | £550 |
| Agilent (HP) 435A/B, 436A, 437B, 438A Power Meters from | £100 |
| Agilent (HP) 3561A Dynamic Signal Analyser | £2950 |
| Agilent (HP) 3562A Dual Ch. Dynamic Sig. Analyser | £3000 |
| Agilent (HP) 3582A Spectrum Analyser Dual Channel | £1200 |
| Agilent (HP) 3585A and B Spec. An. (40MHz) from | £2950 |
| Agilent (HP) 35660A Dynamic Sig. An | £2950 |
| Agilent (HP) 4191A R/F Impedance analyzer (1 GHz) | £2995 |
| Agilent (HP) 4192A L/F Impedance Analyser (13MHz) | £4000 |
| Agilent (HP) 4193A Vector Impedance Meter | £2750 |
| Agilent (HP) 4274A LCR Meter | £1750 |
| Agilent (HP) 4275A LCR Meter | £2750 |
| Agilent (HP) 4276A LCR Meter | £1400 |
| Agilent (HP) 4278A Capacitance Meter (1KHz / 1MHz) | £2950 |
| Agilent (HP) 5342A Frequency Counter (18GHz) | £850 |
| Agilent (HP) 5351B Frequency Counter (26.5GHz) | £2750 |
| Agilent (HP) 5352B Frequency Counter (40GHz) | £4950 |
| Agilent (HP) 53310A Mod. Domain An (opt 1/31) | £3450 |
| Agilent (HP) 54810A Infinium Scope 500MHz | £2995 |
| Agilent (HP) 8116A Function Gen. (50MHz) | £1750 |
| Agilent (HP) 8349B (2- 20GHz) Amplifier | £1950 |
| Agilent (HP) 8350B Mainframe sweeper (plug-ins avail) | £750 |
| Agilent (HP) 85024A High Frequency Probe | £1000 |
| Agilent (HP) 8594E Spec. An. (2.9GHz) opt 41,101,105,130) | £3995 |
| Agilent (HP) 8596E Spec. An. (12.8 GHz) opt various | £8000 |
| Agilent (HP) 89410A Vector Sig. An. Dc to 10MHz | £7500 |
| Agilent (HP) 89440A Vector Signal Analyser 2MHz – 1.8GHz | £8950 |
| Agilent (HP) 33120A Function/Arbitrary Waveform Generator 15MHz | £850 |
| Agilent (HP) 53131A Frequency Counter | £750 |
| Agilent (HP) 53181A Frequency Counter | £750 |
| Agilent (HP) 4284A Precison LCR Meter | £5750 |
| Agilent (HP) 6031A Power Supply (20V – 120A) | £1250 |
| Agilent (HP) 6032A Power Supply (60V – 50A) | £2000 |
| Agilent (HP) 6671A Power Supply (8V – 200A) | £1350 |
| Agilent (HP) E4411A Spectrum Analyser (9kHz – 1.5GHz) | £3500 |
| Agilent (HP) 8924C CDMA Mobile Station Test Set | £6000 |
| Agilent (HP) E8285C CDMA Mobile Station Test Set | £6000 |

| | |
|---|---|
| Agilent (HP) 54520A 500MHz 2 Channel Oscilloscope | £1000 |
| Agilent (HP) 54645D 100MHz Mixed Signal Oscilloscope | £3000 |
| Agilent (HP) 8713B 300kHz – 3GHz Network Analyser | £4500 |
| Agilent (HP) 8566B 100Hz – 22GHz High Performance Spec. An. | £7000 |
| Agilent (HP) 8592B 9kHz – 22GHz Spectrum Analyser | £7500 |
| Agilent (HP) E4418B EPM series Power Meter –single channel | £1500 |
| Agilent (HP) E9300A EPM series sensor for above(18GHz- 100mW) | £750 |
| Agilent (HP) 8648C Signal generator (100kHz-3.2GHz) | £4000 |
| Agilent (HP) 8347A R/F Amplifier (100kHz-3GHz) | £2000 |
| Agilent (HP) 33250A Arbitrary Function Generator (80MHz) | £2150 |
| Agilent (HP) E4406A (opt BAH) Vector Signal Generator (7MHz-4GHz) | £11000 |
| Agilent (HP) E4404B (opts 1D5,1DN,A4H) Spectrum An. (9kHz-6.7GHz) | £10000 |
| Agilent (HP)34401A 6.5 Digit Bench DMM | £550 |
| Agilent (HP) 4194A (50 ohm) Impedance/Gain Phase Analyser | £10750 |
| Agilent (HP)5350B Microwave Frequency Counter (20 GHz) | £1200 |
| Agilent (HP) 5343A Frequency Counter ( 26.5 GHz) | £1400 |
| Amplifier Research 10W1000B Power Amplifier (1 GHz) | £4700 |
| ENI 320L Power Amplifier (250kHz 110MHz) 20 Watts 50dB | £1200 |
| IFR (Marconi) 2051 10kHz-2.7GHz) Sig. Gen. | £5000 |
| Rohde & Schwarz SMY01 9kHz – 1040 MHz Signal Generator | £1750 |
| Rohde & Schwarz CMD 57 Digital Radio Comms Test Set | £4250 |
| Rohde & Schwarz XSRM Rubidium Frequency Standard | £3750 |
| Rohde & Schwarz CMD 80 Digital Radio Comms Test Set | £3500 |
| R&S SMIQ-03B Vector Sig. Gen. (3 GHz) | £7000 |
| R&S SMG (0.1 – 1 GHz) Sig. Gen. | £1750 |
| Seaward PAT 1000S Computerised PAT Tester(New in Box) normally £845 now | £550 |
| Tektronix THS 720A 100MHz 2 Channel Hand-held Oscilloscope | £1250 |
| Tektronix TDS 220 100MHz – 2 Channel Real – Time Scope | £650 |
| Tektronix TDS 524A 500 MHz – 500Ms/s 2 Channel Scope | £3000 |
| Tektronix TDS 724A 500 MHz – 1 Gs/s 2+2 Channels | £3250 |
| Tektronix 2465B 400 MHz 4 Channel Scope | £1000 |
| Tektronix 11402 (Digitizing Mainframe) + 11A33 + 11A34 plug-ins | £1650 |
| Tektronix 571Curve Tracer | £1250 |
| Wayne Kerr 3260A+3265A Precision Mag. An. with Bias Unit | £5500 |
| Wayne Kerr 3245 Precision Ind. Analyser | £1750 |
| Wayne Kerr 6425 Precison Component Analyser | £2000 |
| Wavetek 9100 Universal Calibrator (Opts 100 / 250) | £9000 |
| W&G PFJ 8 Error & Jitter Test Set | £6500 |
| **Various other calibrators in stock. Call for stock / prices** | |

View publication stats