Instruction Manual
Deluxe RS 232 Interface
© 1985 by Omnitronix

Scanned by: Andrew Wiskow 2008 of The Famous Cottonwood BBs !
http://cottonwood.servebbs.com/wiskow/

Choped up'd }
and wacked } Rick Youngman
to hell and } 2008
into PDF by' }

Many Thanks to Andrew for the original scan's

A note from Reeko:

#1: Please direct all flames to me, for the quality of this PDF file. You can email me at Reeko@BYTEME.COM, 24 hours aday.

WHY DO YOU NEED AN RS232 INTERFACE?

Because of the peculiar way the VIC-20 and Commodore 64 RS232 port is designed, most RS232 equipment which works with other computers will not directly connect up to the VIC-20/C64 computer. The design of the accessory connections on the Commodore is not necessarily bad, but it is different from what is commonly used. Modems and serial printers which can be connected directly up to a more technically standard computer can only be connected up to your VIC-20/C64 computer with the use of special interfaces. This probably has something to do with the fact that Commodore would like you to buy their modems and serial printers, which do connect directly up without the need of an interface. If you already have a modem or serial printer, you may not want to spend the extra money. All you have to do is figure out how to connect the modem or printer you already have, and you are in business. That is where the RS232 Interface comes in.

WHAT DOES THE RS232 INTERFACE DO?

The computer RS232 connection is called the USER I/O port. You have probably noticed that the standard cables available for your modem or printer won't fit onto the USER I/O port of your VIC-20 or C64. The USER I/O port is found on the back of the computer, behind and below the Commodore logo on the left hand side. This is where you would connect up a modem or serial printer. The connection found in this spot is called a "cardedge". It is part of the main computer PC Board. It sets in a slot in the back side of the computer, and a "cardedge connector" plugs onto it. A regular modem or serial printer does not have a cardedge connector, so the first incompatability is "How do you plug it in?".

Even if you could plug it directly in, the signals which are sent out from the USER I/O port are not compatible with standard modems or serial printers, nor are the signals sent back from a printer or modem compatible with the computer. Without getting too technical, the signals sent out of the computer are called TTL type, and the signals your modem or serial printer requires are called RS232 type. They are incompatible with each other.

The RS232 Interface is designed to handle the incompatibility of the connectors, and the incompatibility the two types of signals. It connects onto the USER I/O port of your computer, and the DB25 connector coming out of the RS232 Interface then can connect onto your modem or serial printer. When a TTL signal is sent out of the computer, the RS232 Interface converts it to RS232 and sends it out to your printer or modem. When the printer or modem sends an RS232 signal into the computer, the RS232 Interface converts it into a TTL signal and sends it into the computer. This solves most all points of incompatibility between your computer and serial printer or modem.

The connector most commonly used when working with RS232 type signals is called a DB25 type connector. It has either 25 pins in a oblong sort of socket, or 25 holes where corresponding pins would be plugged in. A connector with pins in it is called a male connector, and a connector with holes for pins is called a female connector. The RS232 Interface can be supplied with either of these types of connectors. With the description above, you can recognize which one yours has.

CONNECTING THE RS232 INTERFACE

The RS232 Interface plugs into the USER I/O port of the VIC-20 or C64 computer. As far as the USER I/O port goes, there is basically no difference between the two computers. In all cases, the computer power should be turned off before plugging in the RS232 Interface, otherwise you could short out the computer and blow the internal fuse or worse. The cable from the RS232 Interface plugs into the RS232 connection of your modem or serial printer.

HOW IS A MODEM DIFFERENT FROM A SERIAL PRINTER?

Besides the obvious ways a modem is different from a printer, there is a definite difference between the way they are hooked up. As was described above, you have a 25 pin plug - the DB25. If you look on the connector, you will see these pins are marked 1 thru 25. This connector plugs into the modem

or printer DB25, and it's pins are numbered also. The difference between a modem and a printer is the way the pins in the connector are set up. The way a modem's pins are set up is called DCE (Data Communication Equipment). The way a printer's pins are set up is called DTE (Data Terminal Equipment). Below is a diagram which shows the difference between the two:

DCE		DTE printer	
Receive Data	Pin 2	Transmit Data	Pin 2
Transmit Data	Pin 3	Receive Data	Pin 3
RTS	Pin 4	RTS	Pin 4
CTS	Pin 5	CTS	Pin 5
DSR	Pin 6	DSR	Pin 6
Signal Ground	Pin 7	Signal Ground	Pin 7
DCD	Pin 8	DCD	Pin 8
DTR	Pin 20	DTR	Pin 20
RI	Pin 22	RI	Pin 22

By looking at the chart above, you can see that the two are identical, with the exception of pins 2 and 3. The data you send a modem (DCE) is supposed to go into the modem's pin 2. The data you send to a serial printer (DTE) is supposed to go into the printer's pin 3. The data the modem sends into the computer comes out on the modem's pin 3. If a printer wants to send data into the computer (some do, some don't), it goes out on the printer's pin 2.

When hooking up two RS232 type devices, the idea is that one is DCE and the other is DTE. This means that DTE Pin 2 (Transmit Data) is hooked into DCE Pin 2 (Receive Data), and DCE Pin 3 (Transmit Data) is hooked into DTE pin 3 (Receive Data). This allows the two units to send data back and forth to each other.

The term DTE would generally apply to the device that you use, the part you type on. A computer is almost always set up to be the DTE (Data Terminal Equipment). A modem is almost always set up to be DCE. They will plug right into each other and be correctly connected up. The reason a printer is different is, printers have evolved from teletypes. You can still find printers around that have keyboards on them. A teletype would be DTE, and since printers have evolved from teletypes, a printer is usually DTE.

If your computer is DTE and the printer is also DTE, by looking at the chart above, you can see that there will be a conflict on pins 2 and 3. When both units are DTE, both pins 2 are trying to send data into each other. Both pins 3 are trying to receive data from each other. This will not work.

WHAT ARE THE SWITCHES ON THE RS232 FOR?

The Omnitronix Deluxe RS232 Interface has three switches on the top of the case. These switches allow you to set up the RS232 Interface for using a printer or a modem. The first switch is marked DTE/DCE. This switch sets up the pin output for DCE or DTE as shown in the chart above. Setting the switch on DTE makes pin 2 Transmit and pin 3 Receive. Setting the switch on DCE makes pin 3 Transmit and pin 2 Receive. If you have a modem, you want the the first switch set on DTE. If you have a printer, you want the first switch set on DCE. Remember, whenever you are connecting two RS232 devices together, one should be DTE and the other should be DCE.

The second switch is marked 20 HSK/HSK 5. This switch would only be important if you are using a printer. If you are using a modem, set this switch to HSK5. If you are using a printer, see the later section on PRINTER HANDSHAKING for this switch setting.

The third switch is marked LO BUSY/BUSY HI. This switch would also only be important if you are using a printer. If you are using a modem, set this switch to BUSY HI. If you are using a printer, see the later section on PRINTER HANDSHAKING for this switch setting.

USING THE RS232 INTERFACE

Just simply having the RS232 Interface in place does not give you the means to send and receive data thru the USER I/O port. In all cases this requires some sort of program commands to do so. The software we have written in this manual will not necessarily be suitable in every case with every modem or serial printer. When working with serial communications (to and from a serial printer, for example), you are dealing with baud rates, stop bits, parity, handshaking, and all the different ways those things can be set for a given device. We have done our best in this manual to describe what you might encounter and how you can set things up correctly. However, we are unable to supply examples for every situation possible, and in the event that you have any difficulty setting up the computer and serial device to work together, no examples or instruction will be able to take the place of you, or someone with you, understanding what settings are needed on your printer or modem and how to set the computer for those settings. Besides the information given here, complete instructions on program commands are given in the section on "RS232 Interface Description" found in the PROGRAMMERS REFERENCE GUIDE available for your VIC-20 or C64, published by Commodore. Full information on the necessary and available settings for your modem or printer should be found in the manuals supplied with those devices, and from the companies which manufactured them. If you have any difficulty, make sure that you a) understand the information given in these instructions, b) Understand the data given in the PROGRAMMERS REFERENCE GUIDE on the RS232 Interface Description, and c) Understand the required baud rate, stop bits, and parity used by your device and have set it to be compatible with how you have set the computer.

It is usually not as difficult as it may sound above. It is probable that the simple data that we supply here will be more than adaquate to get you up and running. An understanding of how you set up the computer and how you need to set up your modem or printer will have you printing or telecomunicating. First off we will start out with telecomunications.

BASIC TERMINAL PROGRAM ROUTINE

There are several ways to send and receive data thru the USER I/O port. An example of one way to do so is the BASIC Terminal Program supplied below. An OPEN statement in line 20 opens a channel to the USER I/O port, and the PRINT# and GET# basic commands found in lines 110 - 200 send and receive the modem data.

```
1 REM USE ONE OF THE TWO LINE 10s FOR EITHER VIC-20 OR C64
10 POKE53280,1:POKE53281,1:POKE53272,23:REM USE THIS FOR THE 64
10 POKE36869,242:POKE36879,25:REM USE THIS FOR THE VIC-20
19 REM SET UP VARIABLES AND GOSUB STANDARD/COMMODORE ASCII CONVERSION
20 OPEN2,2,3,CHR$(38)+CHR$(96):POKE36869,242:PRINTCHR$(144)
30 C$=CHR$(147):K$=CHR$(20):J$=CHR$(187):CR$=CHR$(13)
40 PRINTC$;"LOADING DATA...":GOSUB300
100 PRINTC$;"TERMINAL MODE":PRINTJ$;
105 REM MAIN PROGRAM LOOP
110 GET#2,A$:IFA$=""THEN130
120 A=I%(ASC(A$)):PRINTK$+CHR$(A)+J$;:GOTO110
130 GETA$:IFA$=""THEN110
200 PRINT#2,CHR$(O%(ASC(A$)));:GOTO110
295 REM THIS SETS UP STANDARD/COMMODORE ASCII CONVERSION
300 DIMI%(255),O%(255)
310 FORZ=32TO64:O%(Z)=Z:NEXT:O%(13)=13:O%(20)=8:O%(160)=32
320 FORZ=65TO90:Y=Z+32:O%(Z)=Y:NEXT:FORZ=91TO95:O%(Z)=Z:NEXT
330 FORZ=193TO218:Y=Z-128:O%(Z)=Y:NEXT
340 O%(133)=03:O%(134)=19:O%(135)=17:O%(136)=16
350 FORZ=0TO255:Y=O%(Z):IFY<>0THENI%(Y)=Z
360 NEXTZ
390 RETURN
```

The above program allows you to use the VIC-20 or C64 with a 300 Baud modem. It has nothing fancy in it, but it works. In the lines above, the O% is the letter O, not the number zero. In the lines above, the I% is the letter I, not the number one. Be sure to get that correct. What we will do here is examine

the parts that specifically involve the RS232 channel, so you will understand it and can modify it if you want to.

THE OPEN COMMAND

The OPEN command is the key to using the USER I/O port for serial communications. It determines the baud rate, word length, parity, stop bits, and handshaking. The OPEN command for RS232 communication is fully described in the PROGRAMMERS REFERENCE GUIDE for your computer, and we will assume in the following instructions that you have one. If you don't, you should get one. The most important data is found on two charts in the Reference Guide, under the section called "RS232 Interface Description". If you get the settings in the OPEN command wrong, your modem or printer won't be able to make any sense out of the data you send it.

You see the open command in line 20 of the program above. The exact command is:

```
OPEN2,2,3,CHR$(38)+CHR$(96)
```

The first 2 in the command is the LOGICAL FILE NUMBER. This can be any number from 1 to 255 that you wish to use. Once the program has executed your OPEN command, the Logical File number is then used in your PRINT# and GET#s to specifiy where PRINT and GET from (example: PRINT#2 or GET#2). That is pretty much all you need to know about the first 2.

The second 2 is the device number for the USER I/O port. If it was an 8, the data would go to a disk drive, and a 1 would send the data to the cassette recorder. As we are dealing with the USER I/O port, the number is always a 2.

The third number is a 3. When opening the RS232 port for for modem or printer, this three serves no purpose at all, and it could be any number at all. But some number has to be there.

The last two items in the OPEN command are two CHR$ bytes. The first of these is called the CONTROL REGISTER, and the second is called the COMMAND register. The Control register determines the baud rate of the incoming and outgoing data, the word length to be used, and the number of stop bits to be used. A 38 in binary works out to:

00100110

Which you can see by the control register chart (Programmers Reference Manual) works out to a 300 baud signal with a seven bit word and one stop bit. The 96 in the command register is a binary:

01100000

Which sets the command register for even parity, full duplex, and a 3 line handshake. You don't have to understand what all these things mean in order to be able to use them. More important is to know how your device should be set up, and what number you have to put in the open command to set things up that way. The Command register is optional in many cases.

If you wanted to change the above terminal routine for 1200 baud, you would change the 38 in the open command to a 40. If you work out the first four bits af a binary 40, you see by the Control Register chart that it changes to 1200 baud. Changing the OPEN command for different settings is easy, if you have the two charts given in the PROGRAMMERS REFERENCE GUIDE.

PRINT# AND GET#

The main program loop which sends the data you type and which receives incoming data is found with lines 110 - 200. Starting at line 110, the program looks for a byte coming in from the modem, using GET#2,A$. If a byte is coming in, it gets the character and puts it in A$. If it sees one, line 120 prints it on the screen and then goes back to line 110 to look for another. If there are no incoming bytes, the program goes to line 130 to look for keyboard input. If

you are typing something, it puts a character in A$ and goes to line 200 to print it out to the modem using PRINT#2,A$. The additional data in line 200 converts the character you typed from Commodore ASCII to Standard ASCII. The program keeps on sending the data you type until it doesn't see any more. It then goes back to line 110 and starts all over.

These are the simple basics for a terminal program. There can be many fancy modifications you can do to it, but the basic routine is right there. Additionally, two other program modules have been included in this manual, to let you see more ways to program using the PRINT# command. These modules have line numbers specifically planned to allow them to be merged directly into the above program. If you wish to add a module, be sure you type in the BASIC Terminal program with exactly the same line numbers as above, and the routine with exactly the same line numbers as given below.

```
     AUTODIAL PHONE # ROUTINE FOR A HAYES SMARTMODEM
     ACCESSED BY PRESSING FUNCTION 1 KEY
     FUNCTION 5 KEY REPEATS LAST COMMAND SENT OUT

140 IFA$=CHR$(133)THEN400
160 IFA$=CHR$(135)THENGOSUB450
370 READNP:FORZ=1TONP:READPH$(Z):NEXTZ
400 PRINTC$;"PHONE LOG":PRINT
410 FORZ=1TONP:PRINTZ;PH$(Z):NEXTZ:PRINT
420 INPUT"SELECTION";PH
430 IFPH=0ORPH>NPTHENGOTO110
440 PRINTC$:TP$=CR$+"ATDT"+PH$(PH)+CR$
450 PRINT#2,TP$:GOTO100
999 REM THE FIRST NUMBER IN THE DATA STATEMENT IS THE NUMBER OF PHONE NUMBERS
1000 DATA 5,5551234,6669876,3337654,4444321,8882345

     ROUTINE TO SEND YOUR OWN LOGON COMMANDS USING FUNCTION 3 KEY
     FUNCTION 5 KEY REPEATS LAST COMMAND SENT OUT

150 IFA$=CHR$(134)THENGOSUB500
160 IFA$=CHR$(135)THENGOSUB510
380 READNC:FORZ=1TONC:READCM$(Z):NEXT
500 CN=CN+1:TP$=CM$(CN)+CR$
510 PRINT#2,TP$
520 IFCM$(CN+1)="END"THENCN=0
530 RETURN
1997 REM THE 3 IN THE DATA STATEMENT IS THE NUMBER OF COMMANDS
1998 REM INCLUDING THE "END", WHICH MUST BE INCLUDED
1999 REM PUT THE COMMANDS USED IN QUOTES, AS DONE BELOW
2000 DATA 3,"SAMPLE STRING","SAMPLE STRING","SAMPLE STRING","END"
```

Given the data above, you could build yourself a suitable terminal program for your needs, and adjust it for any modem you might wish to use.

USING A PRINTER

The first thing needed to communicate with a printer is to use the OPEN command, as was done in the terminal program above. A serial printer should have the baud rate, word length, parity, and all other items set correctly, or it may not funtion correctly. Usually most newer printers have a series of switches which let you set the printer for all of these items, just as you can set the computer for them. What you need to do is set them both the same way. Although you should check your printer manual first, most newer printers often don't use parity signals, and have parity disabled as shown in the Command Register chart in your PROGRAMMERS REFERENCE GUIDE.

PRINTER HANDSHAKING

In most cases, the computer can send data to the printer faster than the printer can print it. When the printer has as much data as it can take, it sends a signal to the computer, telling the computer to stop sending data. When printer is temporarily unable to accept more data, it is refered to as being BUSY. When the printer is ready to receive and print more data, it sends

a signal to the computer telling it to send more data. This is called "Handshaking". If handshaking does not occur between the printer and the computer, sometimes data will be lost and difficulties will occur.

There are two kinds of handshaking generally used. One is called XON/XOFF handshaking, and the other is called DTR or BUSY LINE handshaking. With XON/XOFF handshaking, when the printer wants the computer to stop sending data, is sends an XOFF code chr$(17) to the computer on the printer Transmit data line. When the printer wants the computer to resume sending data, is sends an XON code CHR$(19) to the computer on the printer Transmit data line. Neither the VIC-20 or C64 is normally designed to use this type of handshaking. If the computer receives an XOFF code, it will still keep on sending data. If your printer only uses XON/XOFF handshaking, see the section on NO HANDSHAKING.

Actually, most serial printers use DTR or BUSY LINE handshaking. If you look at the pin chart given previously, you will see the DTR (Data Terminal Ready) signal comes out on pin 20 of the printer. This signal indicates whether the printer is ready to receive data or not (BUSY). RS232 signals go from a plus voltage to a minus voltage. A plus voltage is refered to as being HI, and a minus voltage is refered to as being LO. Some printers send out a constant LO signal on on pin 20 when they are ready to receive data, and the signal changes to a constant HI when the printer is BUSY. The signal will stay HI while the printer is BUSY, and then pin 20 will change back to a constant LO when the printer is ready again. Some printers are just the opposite, being HI when the printer is ready, and going LO when the printer is BUSY.

The VIC-20 does not look for BUSY line handshaking either. If the line goes BUSY, it just keeps on sending data. On the C64, BUSY line handshaking can be selected by specifying X-line handshaking with the Command Register of the OPEN command.

The pins on the RS232 interface which are checked by the computer when looking for a BUSY signal are pins 5 and 6. If either of these pins receive a BUSY signal, transmission of data out of the computer will cease, until the pin returns to it's previous state. The second switch on the RS232 box is labled 20 HSK/HSK 5. If you are using a printer, you want to switch this line to 20 HSK (Handshake). This sets the unit so that the signal coming out of pin 20 of the printer actually ends up going into the Pin 5 input of the RS232 Interface and the Pin 5 of the printer actually goes into the Pin 20 input of the RS232 Interface. You should set the RS232 to 20 HSK only if you expect to be getting your BUSY signal from pin 20 of what ever you hook up.

The third switch allows you to select whether a HI or a LO on pin 20 means BUSY. If HI means BUSY, set the switch to BUSY HI. If LO means busy, set the switch to LO BUSY. Determining which of these your printer uses is necessary, and usually it can be found in the printer manual. However, you can alway just try it and see which way works. First, enable the handshaking by setting the Command Register for X-Line handshaking, and set the RS232 switch to 20 HSK. Set the LO BUSY/BUSY HI switch to BUSY HI. Send some data to the printer. If the printer prints, then that is the right position for the switch for that printer. If the printer doesn't print, then switch the BUSY switch to LO BUSY. As soon as you switch the swich, the printer should print. If it does, the LO BUSY is the correct setting for that printer.

NO HANDSHAKING

The main reasons you might not be able to use handshaking are a) if your printer does not have BUSY LINE handshaking, or b) if you have a VIC-20. In either of these cases, there is a way to get around it. Since the reason you need handshaking in the first place is because the computer can send the data faster than the printer can print it, all you have to do is set the computer and printer to transmit the data at a slower baud rate. Most printers can be set for 300 or 110 baud, and the computer can be set to send data at these baud rates. This should be slow enough for the printer to keep up.

ACTUALLY PRINTING SOMETHING

For BASIC programs, the way to print something out is by using the PRINT# command. First, open a channel to the printer using the OPEN statement described in the OPEN COMMAND section. The command:

```
PRINT#2,"THIS IS A LINE OF TEXT FOR THE PRINTER TO PRINT"
```

will print the data in quotes out on the printer. If you have a BASIC program which is designed to print out to the Commodore type printer (OPEN?,4,?), it can be converted to work with your serial printer by changing the OPEN statement which opens the printer line to one which suits your printer, leaving the first number (the Logical File number) of the OPEN statement the same. That leaves all the PRINT# commands in your program with the right number. Then your program will work fine.

Another way to print to the printer is by using the CMD command. Once you have opened a channel to Device number 2 (The USER I/O port), a CMD2 command routes all data that was destined to print on your screen to the device you have specified in the CMD command. For example, if you wanted to list a program to the printer, you would load the program into memory, and from READY, type in:

```
OPEN2,2,3,CHR$(3)+CHR$(1) (Used as an example. You use yours.)
CMD2
LIST
```

At that point, the program whould start listing out to the printer, just as it would have previously gone to the screen. When the printing is done, you need to type:

```
PRINT#2
CLOSE2
```

That closes off the channel. Until that point, everything will still go out to the printer. If, instead of typing LIST, you had typed PRINT"THIS IS A LINE", the printer would have printed THIS IS A LINE. This method could be used within a program for printing to the printer, but then you would not see the data being printed on the screen.

TRANSFERING BASIC PROGRAMS

Several persons had asked us about how to transfer BASIC programs onto their Commodore from another computer which was not a Commodore. This can be done fairly easily. The following method assumes that you have a terminal program on the other computer which can load up a BASIC program as an ASCII file (not tokens) in it's buffer, and send it out the RS232 port. If you can do this with the other computer, then this is what you have to do on the Commodore end.

The following BASIC program should be typed into your Commodore and run just before sending over the BASIC program from the other computer. The program below gets each character of the BASIC program as it is sent over the line, prints it on the screen, then the pokes put carriage returns in the keyboard buffer, causing the commands printed in line 4 to be executed, and the transfered BASIC program line to be entered into the computer, just as if you were typing in a basic program at 300 baud. The BASIC program being transfered must be in ASCII, not tokens, and the line numbers of the BASIC program being transfered must not start below 7. Once the transmission of the program is complete, break the Commodore program, delete the program lines contained below (they will still be there), and save the newly transfered BASIC program.

```
0 OPEN2,2,0,CHR$(6):PRINTCHR$(147)
1 GET#2,A$:IFVAL(A$)=0THEN1
2 PRINTA$;
3 GET#2,A$:PRINTA$;:IFA$<>CHR$(13)THEN3
4 PRINT:PRINT"POKE152,1:GOTO6"
5 POKE163,19:POKE632,13:POKE633,13,POKE634,12:POKE198,4:END
6 PRINTCHR$(147);:GOTO3
```

CHECKING OUT THE RS232 TRANSMIT/RECEIVE LINES

Occasionally we have had callers wondering whether the RS232 was functioning properly. The transmit and receive lines can be easily tested by typing and running the following program.

```
10 OPEN 2,2,3,CHR$(8)
11 GETA$
12 IFA$=""THEN11
13 PRINT#2,A$;
14 GET#2,A$
15 PRINTA$;
16 GOTO11
```

Type this program in and run it. Take a clip lead and connect pins 2 and 3 together on the end of the RS232 cable. Pin 2 on the cable is transmit data and pin 3 is receive data. When you type on the keyboard, it gets the byte and sends it out the RS232 port. Because pins 2 and 3 are connected together, it sends it right back in to the computer. If both the transmit and receive lines on the RS232 work, and your computer RS232 port is functioning correctly, line 14 will get the newly received byte from the RS232 buffer, and then it will be printed on the screen.

THE COMPLETE SERIAL PRINTER DRIVER

Omnitronix offers a Complete Serial Printer Driver for use with the VIC-20 and C64. This is a disk of various machine language routines, any of which can load into the computer before using other software or printing to the printer. It intercepts the data being sent out to the printer and handles it in various ways. A compact driver is supplied which allows you to use many professional programs, even if that software does not allow you to select using a serial printer. A Complete Graphics driver is supplied which allows the your printer to work like a Commodore type printer. The Graphics Driver allows you to print PET graphics and control code data to the printer. You can select to have graphics and control codes printed in brackets i.e. (BLK), (CLR), (198). If you have a serial printer which can do bit map graphics, the software can be configured to print actual graphics characters. A CBM Emulation mode is available which allows you to used Commodore printer type commands such tabbing or switching character sets. The driver corrects a difficulty with the Commodore ROM software which sometimes leaves the last bit of data you send out unprinted. We only recommend this software for use with BASIC programs. This is because the RS232 routines in the Commodore automatically use the top 512 bytes of the BASIC programs space for the RS232 input/output buffers, and if you are using a machine language program which also uses this space, it will mess things up. However, it would work fine with any machine language which did not use this space. The Complete Serial Printer Driver contains both VIC-20 and C64 drivers, and is available on disk only. The price is $29.95.

GUARANTEE

The Omnitronix Deluxe RS232 Interface is guaranteed to be free of defects in parts and labor for a period of one (1) year under normal usage. During that time, if any repair is needed, it will be done and shipped back within the US or Canada free of charge. After 1 year we will repair the unit for $10.00.

RETURN/REPAIRS

In the event of return for any reason, the a copy of the original invoice puchasing the product being returned must be included with the item. No COD returns will be accepted.