ALL ABOUT INTERFACING PART II

JEFF HOLTZMAN

This article, begun last month, is concluded here.

RS-232

In 1969 the EIA (Electronics Industries Association) revised the standards defining timing sequences, voltage levels, and pin designations for serial data transmission. That standard is known as RS-232-C. We will refer to it here as RS-232. The specifications are complex, and many companies alter them—seemingly at random—to suit their own needs, without paying attention to the details. We will discuss some of the uses of RS-232 with microcomputers and common peripherals, rather than the technical specifications.

RS-232 signals are bi-polar; that is, a logical "low" is a voltage below ground, and a "high" is a voltage above ground. These voltages must be equal in magnitude and opposite in polarity. They may range from ±3 to ±25 volts, and ±12 volts is very commonly used in microcomputers. The inactive state of an RS-232 line is low. A low signal is called a space, and a high signal, a mark.

You can implement an RS-232 interface with only 2 wires: a signal and a ground, but the sending software might need to provide delays after characters that cause operations like returning the carriage to the left side of the page, feeding the paper up a line, or feeding a whole page through the printer. Such schemes are used when the timing sequences are well defined, but most consumer equipment makes use of one or more of the hardware or software transmission regulation schmes discussed last time.

Many RS-232 interfaces include other signals with names like CARRIER DETECT, DATA TERMINAL READY, etc. (See Fig. 7.) Many of these signals were developed for use with MODEMs, and were meant to indicate presence on the telephone line of the ringing voltage, special timing signals, secondary transmit and receive lines used for diagnostic functions, etc. Many manufacturers didn't follow the "standards," and started putting all kinds of things on the connector, including power for peripheral devices. For interface devices like printers, most of those signals aren't used, though some may have to be accounted for.

The "Busy" line is commonly (but not always) available at pin 20 of a 25 pin "D" connector, which is the DTR line. DTR is an acronym for "Data Terminal Ready," and on some machines that signal indicates that the machine has been powered up, whether it is able to receive data or not. The Busy signal also may come on pins 4, 6, 8, 11 or 19. That is usually documented—but not always, and you may have to experiment to find out which is the busy line. The only things you can be certain of are that pin 7 is circuit ground, pin 1 is

Figure 7
RS-232-C DTE Pin Designations

Pin	Acronym	Description
1	PG	Protective Ground
2	TD	Transmitted Data
3	RD	Received Data
4	RTS	Request To Send
5	CTS	Clear To Send
6	DSR	Data Set Ready
7	GND	Ground
8	RLSD	Received Line Signal Detect
9		Test Pin
10		Test Pin
11		Unassigned (sometimes used for Busy line)
12	SCF	Secondary Received Line Signal Detect
13		SCB Secondary Clear To Send
14	SBA	Secondary Transmitted Data
15	DB	Transmitter Timing
16	SBB	Secondary Received Data
17	DD	Receiver Timing
18		Unassigned
19	SCA	Secondary Request To Send
20	DTR	Data Terminal Ready
21	CG	Signal Quality
22	CE	Ring Indicator
23	CH/CI	Data Rate Selector
24	DA	Transmit Timing
25		Unassigned

FIG. 7—EIA DESIGNATIONS for a standard RS-232 port.

frame ground, and pins 2 and 3 will be input and output pins.

Part of the EIA standard are the two acronyms, DTE and DCE. The former stands for Data Terminal Equipment, and the latter for Data Communications Equipment.The technical specifications are complex, but their meaning is that input and output pins are reversed on the two kinds of equipment. Connecting cables could be wired straight across. Computers usually use pin 2 as the data output pin, and pin 3 as the data input pin. MODEMs reverse those pins so pin 2 on one device may be wired directly to pin 2 on the other; likewise with pin 3.

Printers are (usually) connected as DTE devices, so to connect a DTE computer port to a printer may require that pins 2 and 3 be cross connected (2 to 3 and 3 to 2). To use the same computer port at different times with both a printer and a MODEM usually requires some sort of wiring adapter on one device to reverse the transmit and receive data pins.

A cable which cross-connects inputs and outputs is commonly called a "null-MODEM," named because it replaces two MODEMs connected back to back. If we connected each signal line straight across, inputs would go to inputs, and outputs would go to outputs. That wouldn't do, so it was necessary to cross-connect inputs and outputs. To account for all the MODEM control signals we would have to wire a cable as shown in Fig. 8a or Fig. 8b. Often we can cross the TD or RD lines as shown in Fig. 8c, and ignore the rest. It is often desirable to connect the serial ports of two personal computers together and use a special program to transfer files between them. Such communications programs frequently use software "Busy" protocols and simply ignore "Busy" pins. If so, the 3-wire configuration shown in Fig. 8c would suffice.

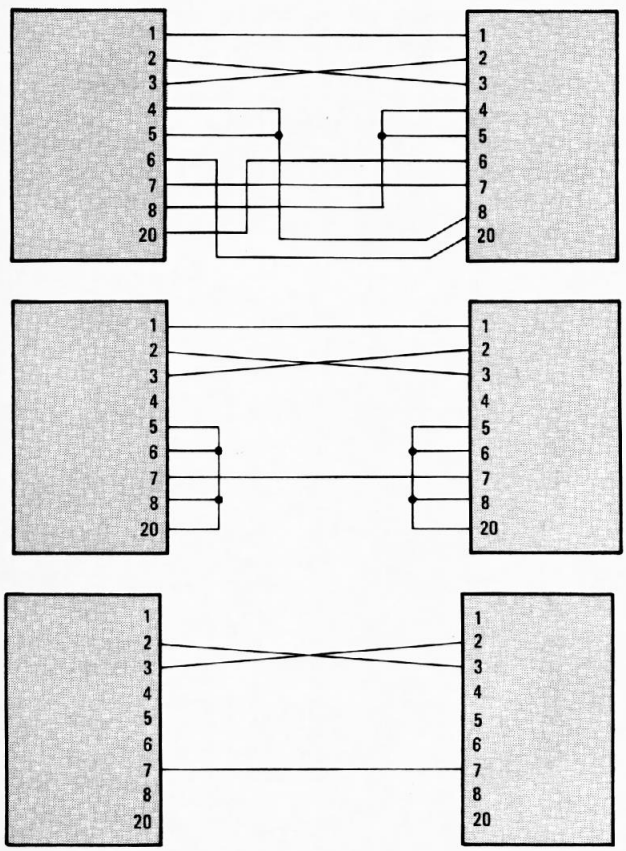

FIG. 8—THREE DIFFERENT NULL-MODEM cables are shown here. Those at *a* and *b* use the MODEM control lines, and the one at *c* is a simple computer-to-computer hook-up. What they have in common is a reversal of the main data lines, pins two and three.

Don't get hung up on the terminology. To wire up interface cables for microcomputers and peripherals, think functions, and ignore the EIA's confusing designations. Many microcomputer interfaces can get by using only TD, RD, GND and BUSY lines between the two devices. Determine which is which on each device using the manufacturer's documentation, or with test equipment. Other lines may often simply be tied high or ignored. Unless you have a reason for including the other lines, ignore them until you have a reason not to.

Hints for debugging serial interfaces

Suppose now that you have just bought a brand new printer from the Chopstick Printer Company of Hong Kong, and all it comes with is a diagram of the output connector—maybe not even that. How would you go about hooking it up to your personal computer?

First get all documentation for both pieces of equipment. Use those documents to draw a diagram (like that shown in Fig. 5 last time) with similarly labeled signals laid out opposite each other. Draw in lines connecting the important signals: RD, TD, GND and BUSY, showing pin numbers at both ends. Check everything over—there is nothing more disheartening than frying a brand-new printer—then wire up a cable using a "break-out" box and the appropriate connectors. Connect the two pieces of equipment.

What if you have no documentation, or have hooked things up according to the manufacturers suggestions—and it doesn't work? Build a test instrument like that shown in Fig. 9. Use a "Tri-color" LED, as it will allow you to see highs, lows and streams of data in different colors. Attach a tiny alligator clip to the resistor and a sharp probe to the LED. Connect the clip to pin seven (ground) of the computer, and turn the power on.

Examine pins two and three. One should be an input and one an output. The output will cause the LED to glow, and the color it glows indicates a low. Record which pin caused the LED to glow, and the color it

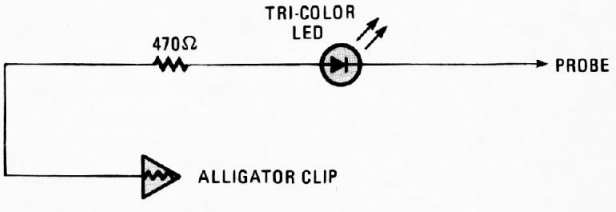

FIG. 9—THIS CIRCUIT comes in handy for debugging both parallel and serial interfaces. Use a tri-color LED for best results.

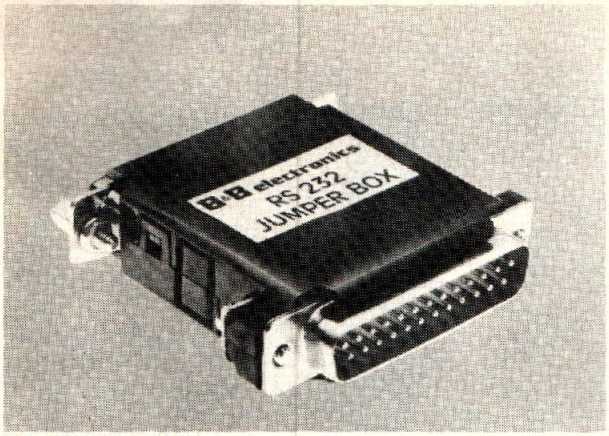

FIG. 10—This "D" connector shell is handy for wiring up RS-232 interface adapters. See the text for more information.

glowed. Repeat the process on the printer, recording everything. You can assign some pin numbers. The output of the computer should connect to the input of the printer, and vice versa. Don't forget to connect pins 7 on both devices together.

You'll have to decide on the "Busy" protocol you'll be using, i.e., hardware or software. If you have no reason for either, choose software, as it's more universal, so the same port could be used with both a MODEM and a printer, without having to re-configure.

If you go the hardware route you'll have to find the busy signal. With a printer, that's not too difficult. Connect your probe to pins 7 and 20. Note the color of the LED, then press the printer's "ON-LINE" switch. If the LED changes color, there's a chance you've found the Busy line. If the LED didn't change colors, try pins 4, 5, 6, 8, 11 or 19. Failing that, try all other pins one by one. Failing that, you'll have to contact your supplier or the manufacturer—and good luck with the latter!

You won't be able to use the LED probe to find the Busy input to your computer because, since it is an input, it won't make the LED glow. If you don't have it in your system documents, try pin 20. But first make sure that pin is not an output. If it is an output, try working with the other pins mentioned. Do so systematically, writing down your findings at each step. As frustration increases, we are increasingly likely to forget what we've already done. Remember, outputs go to inputs, and vice versa.

After all wires are connected, (at least tentatively), you'll need to make sure your computer knows that you have hooked the printer up. We can't stress the importance of this step; if we had a dollar for every interfacing problem we helped debug that eventually turned out to be a case of the computer's simply not knowing the device had been attached—we'd be writing this from a villa on the Riviera! Anyway, for CP/M systems you'll use the STAT command or a custom utility supplied by the manufacturer of your machine. MS-DOS machines will use the MODE command, or, again, a custom utility.

Let's do this a step at a time. First let's see if we can get just one character to the printer. Use the quickest method you have for getting a character to the device. CP/M and MS-DOS owners can type a CNTL-P followed by some data (assuming the printer is set up as the normal LST: or PRN device.) The printer should print everything you type at the keyboard until your next CNTL-P. Apple, Commodore and Sinclair owners will find using BASIC the simplest way of experimenting. Just use LPRINT statements like the following (don't forget the semicolon):

10 LPRINT "A";

Let's assume you sent one character to the printer. If it printed correctly, try sending another. If that works, try sending a number of characters, and follow them with a CRLF. In BASIC use:

10 LPRINT "EVERY GOOD BOY DOES FINE"

If nothing at all is printed, but the computer doesn't lock up (or, in MS-DOS, you don't get the "Abort, Retry, Ignore?" message), chances are the computer is sending data out some other port, and doesn't realize you're hooked up. If you end up having to re-boot (Reset) your machine a few times while experimenting, don't forget to initialize the port each time.

If garbage is printed, you probably have a Baud rate or protocol problem. Unless you have a specific reason not to, set both your computer and your printer up to operate with one start bit, eight data bits, one stop bit, and no parity. And if you are using a software protocol, make sure both devices use the same protocol. You can't run the computer with ETX/ACK and the printer with X-ON/X-OFF!

If your printer locks up after the first (sometimes second) character, you most likely have a Busy line problem. After sending each character the computer looks for some acknowledgement that it is safe to continue, and it may just loop forever waiting for a "high" at its busy input. To solve that problem, find a signal that appears to remain high and try connecting it to the pins you know are inputs (to the computer port), one by one, again writing down what happens at each

Table I
Debugging a Microcomputer Interface

1. Get all relevant documents for both machines
2. Decide on hardware or software "Busy" protocol
3. Assign and configure computer port (1 start and 1 stop bit, 8 data bits, no parity, unless specifically need other.)
4. Configure peripheral port exactly the same as printer port.
5. Make wiring diagram.
6. Assign pin numbers:
   A. Use documents, if available, otherwise
   B. Use LED tester to determine all outputs on both machines, especially TD, DTR, RTS.
7. Wire up most likely configuration using breakout box.
8. Get one character to transmit without locking up computer. If incorrect data prints, baud rate probably incorrect. If nothing prints, possible wiring error or port incorrectly configured. Use spare video terminal, if available, to verify data is being transmitted.
9. Get a second character to transmit. If computer locks up, probably busy line is not properly connected. Don't forget to re-initialize port.
   A. Busy line should toggle each time "ON-LINE" switch is pressed.
   B. Problem may be with MODEM lines (4, 5, 6, 8). Try tying high, singly and in combination, if necessary.
10. Print a couple of lines. If that works, print a several-page document. If data is lost, "Busy" line not working.

step. After making each connection, check if the computer has "unlocked," that is, whether you can type something at the keyboard.

If tying single lines high doesn't solve the problem, you may have to tie several lines high. MODEM control pins 4, 5, 6 and 8 are particularly likely candidates for that treatment. Try connecting them to pin 20.

After you've gotten the "Busy" line working, try printing a document several pages long. It may be that, even though the computer didn't "lock up," the "Busy" line wasn't doing its job. If, after successfully printing a part of your document, other parts seem to be lost, the printer is missing data from the computer because the computer is not reading the "Busy" line. The information presented in this section has been summarized in Table 1. Cut it out and paste it up near your workbench so you can refer to it when you need it.

If you need more help, human resources are the best. Try local computer clubs, or computer stores. Many electronic bulletin boards have special interest groups for people with various equipment. You might try leaving a "HELP!" message on a bulletin board with appropriate special interest groups.

For more technical information on the RS-232 standard, a good place to start is a book called *RS-232 Made Easy* by Martin D. Seyer (Prentice Hall: 1984). Seyer gives a good discussion of the MODEM control lines, summarizes the EIA standards, and lists, in several appendices, output connectors and interconnections for several hundred popular computers and printers. No MODEMs are included in the charts, however.

A good source for RS-232 adapter cables and test equipment is B & B Electronics, P.O. Box 1008H, Ottowa, IL 61350. They manufacture a very useful, yet quite inexpensive ($1.75) "D" connector shell (Model 232CS). As shown in Fig. 10, one or two 25-pin "D" connectors may be mounted in the 232CS, and that allows very neat do-it-yourself adapters to be made.◀Φ▶