Matchmaker!

Matchmaker!

Interfacing with Parallel and Serial Ports

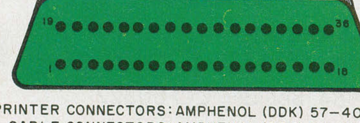
By Alex Marx

COMPUTER interfacing can be very confusing and frustrating. Although the two "standard" interfaces, "Centronics" *parallel* and "RS-232" *serial,* are supposed to allow simple plug-in connections, in reality this is often not the case. Frequently, a manufacturer adds features that don't fit the definitions of a given standard. The interfaces are really "conventions" that have come to be frequently used rather than "standards," which means that you can do almost anything you want and still call it "compatible."

In an effort to eliminate some confusion, several manufacturers have designed their equipment with the flexibility to deal with the plethora of "standards." Others have taken a different approach and provided a *real* standard interface, along with the peripherals to work with it. I am thinking particularly of Commodore, which has provided an IEEE-488 (Hewlett-Packard GPIB) interface since the introduction of its first computer back in the 1970s. This interface is rigidly defined right down to the type of wire to be used.

PIN 1	$\overline{\text{STROBE}}$	19	$\overline{\text{STROBE}}$	RETURN
2	DATA 1	20	DATA 1	RETURN
3	DATA 2	21	DATA 2	RETURN
4	DATA 3	22	DATA 3	RETURN
5	DATA 4	23	DATA 4	RETURN
6	DATA 5	24	DATA 5	RETURN
7	DATA 6	25	DATA 6	RETURN
8	DATA 7	26	DATA 7	RETURN
9	DATA 8	27	DATA 8	RETURN
10	$\overline{\text{ACK}}$	28	$\overline{\text{ACK}}$	RETURN
11	BUSY	29	BUSY	RETURN
12	PE	30	INIT	RETURN
13	SLCT	31	$\overline{\text{INIT}}$	
14	±0V	32	FAULT	
15	OSCXT*	33	NC	
16	±0V	34	LINE COUNT PULSE*	
17	CHASSIS GND	35	RETURN	
18	+5V	36	NC	

*GENERALLY NOT USED TODAY



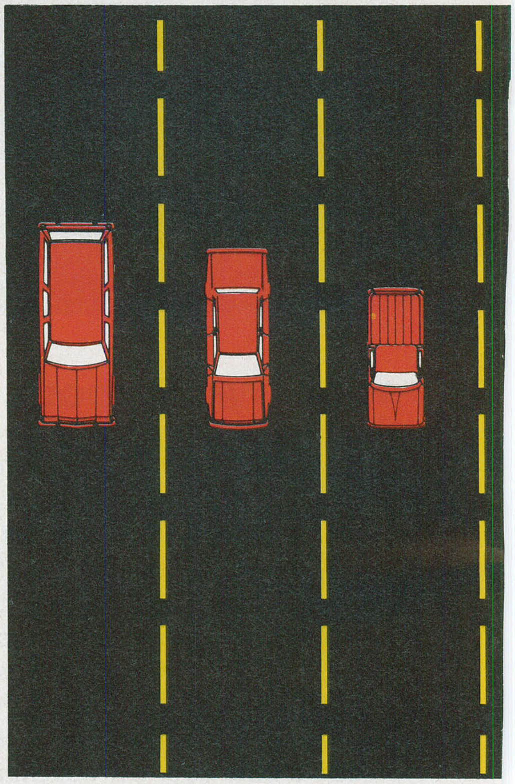
PRINTER CONNECTORS: AMPHENOL (DDK) 57-40360
CABLE CONNECTORS: AMPHENOL (DDK) 57-30360

Fig. 1. Pinout of a Centronics parallel connector.

Unfortunately, it is a costly standard. The connectors and cables are expensive (up to $70 for a six-foot cable) and very few other computer or peripheral manufacturers have adopted the system. The IEEE-488 interface was designed primarily to connect test and lab equipment to dedicated controllers, and it is in this sphere that it has been most successful. Since there have been few personal-computer peripherals to adopt this system, several companies have sprung up to adapt the IEEE-488 to the more prevalent Centronics and RS-232 interfaces.

The Parallel Interface. The parallel printer interface is often called a Centronics interface, after the company that developed it for its line of printers. The specifications for this interface call for eight unidirectional data lines (or wires), three handshake lines, several control and miscellaneous signal lines, and signal-return and ground lines.

The "standard" Centronics connector is a 36-pin D connector such as the Amphenol (DDK) 57-30360. The mating female connector on the printer is an Amphenol (DDK) 57-40360 type. These connectors are produced by many manufacturers and are available for ribbon cable as well as for multi-conductor cable.



The parallel-communications concept can be described in terms of a short eight-lane highway, supporting only one-way travel. At the start is a roadblock to control access and at the far end is a toll booth. To avoid congestion on the highway, only eight vehicles are allowed to cross at a time. All the vehicles must travel abreast of one another, or in parallel. To control the traffic flow, there are several signals used. At both ends of the highway there are bells: the roadblock end can ring the toll booth's bell and vice versa. At the roadblock end, there is also a red-green traffic signal across all eight lanes. This light is controlled from the toll booth, or sending, end. At each end there is a man in charge of the bells and lights.

To start the whole thing going, the receiving end makes sure it's ready and then turns the traffic light green. Eight vehicles enter the highway and, when they reach the toll booths, the man at the roadblock end pulls his cord to ring the bell at the booths. This causes the toll keepers to look up from their newspapers and collect the tolls. At the same time, the controller at the toll booth turns the roadblock's traffic light red and also rings the bell at that end. This acknowledges the receipt of eight tolls. When the eight vehicles have left the booths, he turns the light green and eight more vehicles roar across to the other side and the cycle repeats. If one of the toll clerks runs out of change and has to fetch more, the light is kept red to tell the sending side that the clerks are busy and traffic should be held up.

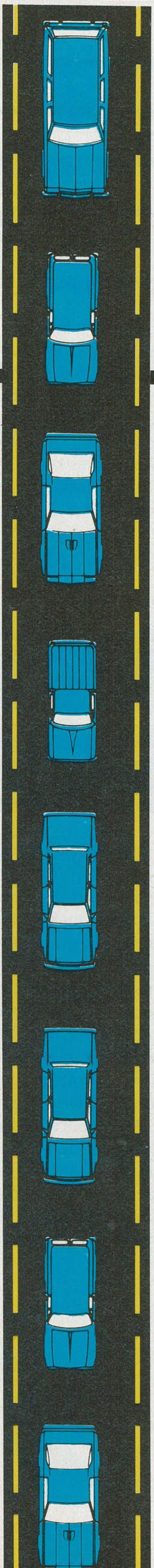
The idea of lights and bells is called *handshaking*. It is simply a method for one device to tell the other what it is doing. In the foregoing analogy, it showed when the toll booths were ready to receive more vehicles. In a printer, it would be used to tell the computer when it was ready to receive another character to print. The reason handshaking is necessary is that the printer, being a mechanical device, is much slower than the computer. You might reason that you could have the computer send out characters slowly enough for the printer to keep up. But, not only would this be inefficient, you couldn't take into account the fact that a printer takes longer to do a carriage return and line feed than it does to print a character. Also, you could not tell if the printer had run out of paper or if it were even turned on! Handshaking allows the computer to send characters as fast as the printer can accept them, as well as keep track of problems like running out of paper.

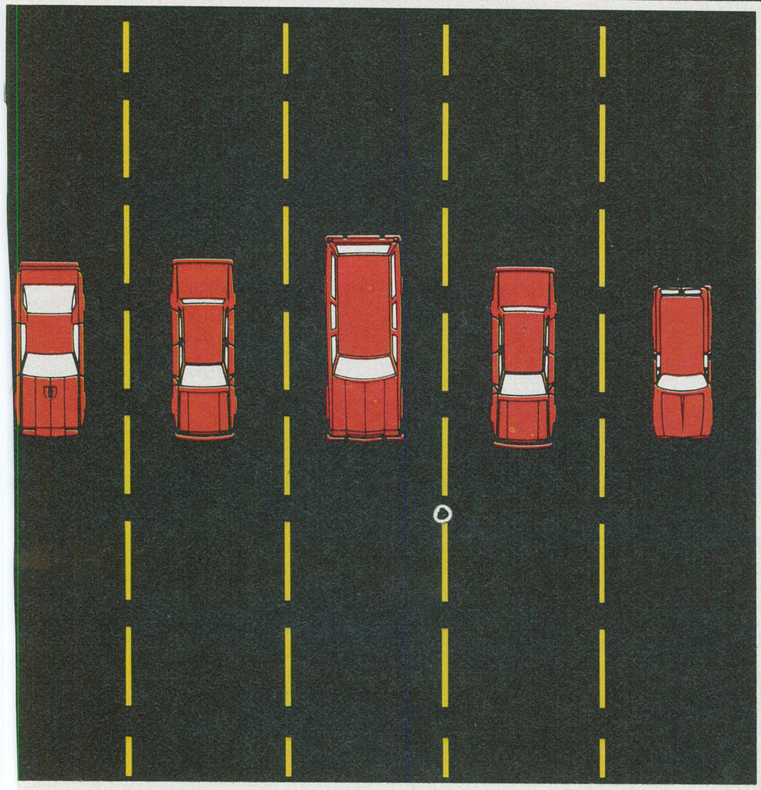
In the Centronics interface, the signal represented by the traffic light is called BUSY and is controlled by the printer. The printer uses it to tell the computer when it is busy, as when it is in the middle of printing a character or tied up doing a carriage return. The bell at the roadblock end is called Acknowledge or ACK. It is used to indicate when the printer has accepted the last character sent. The bell at the toll booth is called Strobe or STB. It is sent by the computer to tell the printer that there is a character ready to read on the cable. Without this signal, the printer cannot tell when a new character has been sent.

The true Centronics interface defines several more control signals, many of which are little used today. SELECT is used to tell the computer that the printer is on-line, or able to accept data through the interface (as opposed to accepting data from a panel switch or keyboard). Error conditions are also signaled by quite a few lines. These include Printer Error, or PE. It is used to indicate when the printer has malfunctioned in some way, as in the case of ribbon jam. FAULT or ERROR indicates some type of abnormal condition like a PE or the cover being open. The only reason I can see for all these error lines is to allow for intelligent interface software to tell the user, via the computer's display, the exact error condition. However, in most implementations today the BUSY signal is used to indicate all fault conditions. A fault condition is usually accompanied by the computer's "hanging," while waiting for the BUSY signal to disappear. A complete pin-out of a Centronics connector is shown in Fig. 1.

The voltage levels in the Centronics interface that define the high and low (or 1 and 0, yes and no, or on and off) control levels are the same as those used in the integrated circuits in microcomputers. In engineering jargon these are called *TTL levels*. A *high* is defined as being between 2.5 and 5.0 volts and a *low* is between 0 and 0.7 volts. As an aid to the user, timing charts are usually drawn to show the various signal levels and the timing relationships among them. Such a chart is shown in Fig. 2. The vertical axis shows time, and the horizontal axis voltage.

Note that the STROBE signal is true when it is low. This means that the


▶ **The serial interface is analogous to a single-lane road with each car following another at rigid time intervals**


▲ **The parallel concept can be thought of as a short strip of eight-lane highway with eight cars starting at the same time**

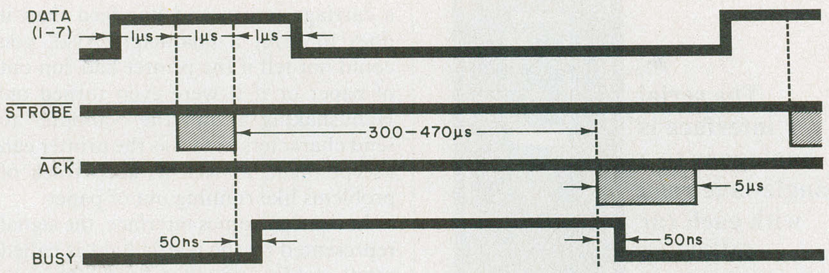


Fig. 2. Timing chart for a Centronics parallel interface.

printer reads data when the STROBE line changes from high to low. On the other hand, the BUSY signal indicates a busy condition when it goes from low to high. This is called *positive true.* The STROBE signal is an example of *negative true.* To indicate a negative true a bar is placed above the signal name, i.e. $\overline{\text{STROBE}}$ and $\overline{\text{ACK}}$.

The Data lines are represented by only one line in Fig. 2, as the timing of the data is more important than what data are sent. The time increments are in millionths of a second. The action is as follows. The computer puts data on the data lines and approximately 1 μs later causes the $\overline{\text{STROBE}}$ line to go low. The printer then makes the BUSY line true. When the printer has processed the character it takes the $\overline{\text{ACK}}$ line low, and a short time later releases the BUSY line. Shortly after that the $\overline{\text{ACK}}$ line is also released. Note how this compares to the toll booth and bell analogy.

Computer people are always looking for ways to simplify things, and it did not take them long to figure out that you do not have to use the $\overline{\text{ACK}}$ signal. Instead, if the computer just watches the BUSY line, it can determine when the printer is ready for another character. If the BUSY line is low, the printer can accept data. In fact, while printing, most computers just sit in a software loop doing nothing but watching the BUSY line. And since BUSY also indicates when the printer is off line, and error conditions, it eliminates hooking up the other signal lines! Most parallel interfaces today only use the $\overline{\text{STROBE}}$ and BUSY lines for handshaking. This is especially true in the Apple computers.

In the S-100 marketplace both methods are used. In fact, my system uses an old Processor Technology 3P+S board that has parallel outputs with generalized handshaking. To make it work requires setting it up to generate $\overline{\text{STROBE}}$ and accept BUSY. That requires not only the 3P+S but support software as well. A friend's S-100 system has a Godbout Interfacer 4 board, which has a Centronics type interface that supports full handshaking including a number of error and signal lines. Because the hardware supports all these functions, the software is simpler.

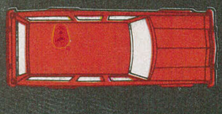
Do It Yourself. Armed with the information presented, it should be a simple matter for you to connect a parallel-interfaced printer and computer. Every day that task becomes simpler as more and more manufacturers strive for compatibility. However, problems can still arise. The minimum equipment you will need to get everything running is the documentation for both the printer's connections and the computer's connections and an ohmmeter or other sort of continuity testing device.

Most parallel-interfaced devices are limited to having about six feet of cable between them. The limitation is due to several factors, but it can be stretched. I have successfully run cables 15 feet or more. The best bet, though, is to keep the cable six feet or shorter. If you are trying to get a system running with long cables, make sure you get it working with a short cable first and then try the longer length. Many hours can be saved with this method. (Listen to the voice of experience!)

Using the data sheets, verify with the continuity tester that the wires go to the correct pins on both ends of the cable. Also check that none of the adjacent pins (side to side as well as front to back) are shorted together. This is a common problem with the newer clamp-on connectors for ribbon cable.

Unless one end of the interface specifically requires it, do not connect any pin with power on it to any other pin. Many printers have a +5-volt pin; I never connect anything to this pin. One bad connection or an accidental short can wreak havoc.

Printers generally have many pins labeled RETURN or 0 VOLT, and GROUND (GND), FRAME or PROTECTIVE GROUND. Always connect at least one RETURN or 0 VOLT and one GROUND, FRAME or PROTECTIVE GROUND. The RETURNS and 0 VOLTS are the return connections for the low-voltage data and signal wires. You can never connect too many of these together—in fact, the more the better. The GROUND, FRAME or PROTECTIVE GROUND is a connection to the metal chassis of the printer. It is used for safety purposes, both yours and the equipment's, to make sure that both the printer and computer are at the same electrical potential in case of some sort of error or accident at the 110-volt ac power connections. *Do not* use these connections for the low voltage signal returns. A missing ground connection or similar problem can severely damage



"I prefer the Centronics interface because it

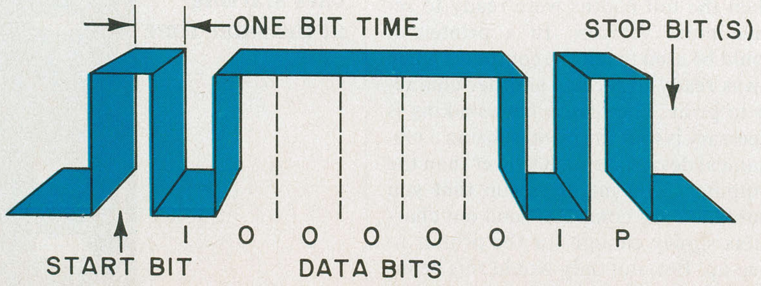


Fig. 3. Diagram of a typical serial data transmission.

your equipment, as well as present a shock hazard.

If all your connections seem proper but nothing works, it's time to check signal polarities. Though most printers today use positive true DATA, negative true $\overline{\text{STROBE}}$, positive true BUSY and negative true $\overline{\text{ACK}}$, don't take it for granted. Some printers and computers have switches or jumper wires to allow the polarities to be changed. Consult the manuals for as much detail as you can find and keep a record of your work. Make sure you cycle the printer off and on between trials since it could be "hung up" from an incorrect setup. Look for clues in the operation of the printer. For example, if the printer prints but starts dropping characters after a while, it is a sure sign that the handshaking is fouled up. Check the BUSY and $\overline{\text{ACK}}$ lines.

With a little patience and some sleuthing you should be able to get results. If all else fails, make sure the printer and computer interfaces are working. Other problems can include timing (does the $\overline{\text{STROBE}}$ signal last long enough?) as well as software bugs. Many computers send seven bits of data rather then eight. Simple printers may only have seven data lines while others may use the eighth bit for special functions like graphics. If the computer's software does not take into account the eighth bit, "interesting" effects can occur. Often just connecting the eighth bit on the printer side to a RETURN or 0 VOLT line may cure the problem.

Although all this sounds complicated, it really is not. I prefer the Centronics interface because it works the first time more than 90% of the time. And compared to the intricacies of the RS-232 serial interface, the parallel is almost child's play. However, if you need longer cable lengths than a parallel interface allows, or if you are working with a modem or similar piece of equipment, then RS-232 is inevitable.

The RS-232 Serial Interface. If I sound a little hostile towards RS-232 it's because it has been so corrupted over the years that a clear understanding of its subtleties is impossible. Although I have worked with it for years and it can be simple and efficient, I still prefer the Centronics interface because it is basically unchanged.

The RS-232 serial communications protocol was originally developed to connect a data terminal to a modem. Because computing was limited to mainframes and therefore expensive, users were generally connected via a terminal and modem to the often remote mainframe. The connection between the modem and the computer was frequently handled via dedicated lines, though telephone lines were a popular choice for all but the shortest distances. The RS-232 protocol was designed to connect Data Terminal Equipment, (DTE), which was a terminal, to Data Communications Equipment (DCE), which was a modem.

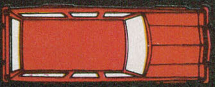
A modem converts digital signals into tones that can be sent along relatively inexpensive telephone lines. The price you pay for using the cheap lines is speed. Modems typically can send 30 characters per second (cps) as opposed to over 100,000 cps for a simple parallel interface. However, modems are not limited by distance, with communications possible over many thousands of miles.

Serial interfaces work by sending each bit of the transmitted character one after the other, or serially. The process is analogous to a single-lane road where each car must follow the other. The bits are sent at rigid time intervals. The transmission speed is called the baud rate and is equivalent to bits per second. Standard baud rates include 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600, and 19,200. Modems generally work at 300 baud but some also work at 1200 baud.

A high bit, or 1, is represented by a voltage between −5 and −25 V. A low bit, or 0, is represented by a voltage between +5 and +25 V. This negative true logic only applies to the data line; the handshaking lines which use the same voltage levels are positive true. The higher voltages and slower speeds of the serial interface permit longer cables—up to 50 feet for most appplications.

When no data is being sent, the

works the first time ,over 90% of the time."



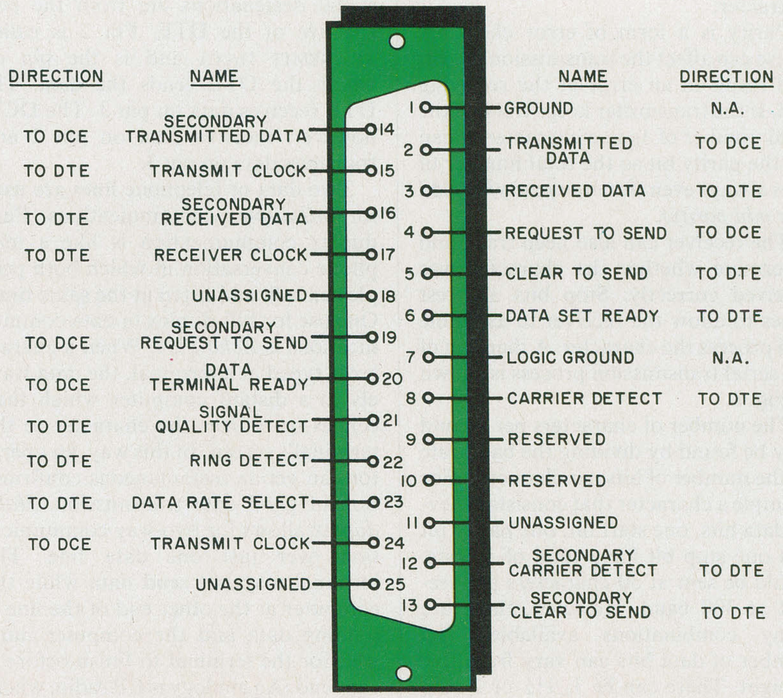


Fig. 4. Pinout of an RS-232 serial connector.

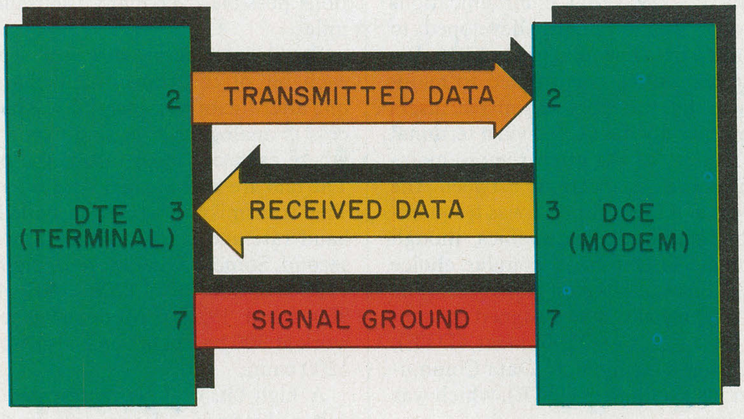


Fig. 5. The basic RS-232 connection uses only three lines.

dataline sits at a 1 condition. To announce the start of a character transmission a *start bit* is sent. This brings the data line to 0 and tells the receiving end that there is information following. After all the data bits have been transmitted, a *parity bit* and one or more *stop bits* are sent to mark the end of the character.

Parity is a form of error checking. Noise can affect the transmission of a bit and result in an error at the receiving end. If the transmitter keeps track of the total number of 1s in a character, it can set the parity bit so the total number of 1s is always even (for *even parity)* or odd (for *odd parity)*.

The receiver can also keep track and determine whether the character was received correctly. Stop bits are rest times to allow the receiver to assemble and process the character. A diagram of the serial transmission process is shown in Fig. 3.

The number of characters per second may be found by dividing the baud rate by the number of bits per character. For example a character that consists of seven data bits, one start bit, one parity bit and one stop bit for a total of 10 bits, would be sent at 30 characters per second at 300 baud. However, there are many combinations available. The number of data bits can vary from five to eight. There can be 1, 1½ or 2 stop bits, and the parity bit may or may not be used. All these factors are important in setting up the interface. The transmitting and receiving ends must be set up the same way; otherwise errors will result.

A diagram of an RS-232 connector is shown in Fig. 4. The connector is a standard DB-25, which is available in many styles, including clamp-on for ribbon cable. The DTE and DCE are both equipped with female connectors. All signal designations are from the perspective of the DTE. Pin 2 is called TRANSMIT DATA and is the pin on which the DTE sends the data. The DTE receives data on pin 3. The DCE, however, receives data on pin 2 and transmits data on pin 3.

Two data or telephone lines are used for *full-duplex* communications. Full-duplex communication is like a telephone conversation in which both people can talk and listen at the same time. One use for full-duplex in data communications is *Echo-Plex*. When a character is typed at a terminal, the data travels to a distant computer which then echoes or returns the character to the terminal's screen. In this way, an operator can get an instantaneous confirmation of good data transmission. *Half-duplex* allows for two-way communication over just one data line. The terminal may not send data while the computer at the other end of the line is sending data and the computer must wait for the terminal to finish before it can send. An analogy is CB radio, where only one person can talk at a time.

You will note from Fig. 4 that there are a number of signal and handshaking lines. What all those lines do is beyond the scope of this article. However, the lines of interest to the average computer user are TRANSMITTED DATA, RECEIVED DATA, EARTH GROUND, LOGIC GROUND (RETURN), REQUEST TO SEND (RTS), CLEAR TO SEND (CTS), DATA SET READY (DSR), CARRIER DETECT (CD) and DATA TERMINAL READY (DTR). Although this may seem like a handful, only a few of these lines are used in actual practice.

The simplest RS-232 connection uses only three lines, as illustrated in Fig. 5. This is fine for low-speed applications, or where both devices can keep up with each other.

More sophisticated applications can use this system with software handshaking, perhaps using the XON/XOFF protocol. The two devices watch each other's transmissions for two special control characters: XON which is ASCII 16 (Control-Q) and XOFF which is ASCII 18 (Control-S). When the transmitter gets an XOFF from the receiver, it stops and waits until it receives the XON character to continue. This form of handshaking requires intelligence on both ends to process the XON/XOFF characters. If it is available it should be used because it simplifies the number of connections required and eliminates having to worry about which handshaking lines to use.

The RS-232 interface was never meant for use with printers or computers. Computer interfaces and printers can be configured as DTE or DCE at the discretion of the manufacturer. If your computer's interface is configured as DCE (a common occurence) and you try to hook it up to a modem (also DCE), they won't talk to each other. The solution to this problem is to make a null modem adaptor. This is a short cable with two connectors: one mates with the existing cable and the other plugs into the modem. The two connectors are cross wired so that the proper signals connect: that is, pin 2 to pin 3, pin 3 to pin 2, etc. A diagram of a null modem cable is shown in Fig. 6.

Printers present a special problem because the RS-232 handshaking lines are meant to deal with terminals and modems. The problem gets really serious at speeds higher than 300 baud. Most printers can handle 30 characters per second, which is roughly equivalent

"If you need longer cable lengths than a parallel

to 300 baud, without losing characters. This is especially true of printers with internal buffers that can store the additional characters that would back up when the printer does a carriage return. That might lead you to believe that handshaking wouldn't be necessary at slow baud rates; and it wouldn't if all you were doing were printing. But if you need to stop the printer to readjust the paper, change a ribbon or answer the phone, the computer will still send characters, blissfully unaware that the printer stopped five minutes ago.

With all the handshake lines available in the RS-232 protocol, you would think that there would be a solution to this problem. Unfortunately, there is not a single line set aside for the purpose because this problem does not exist between a terminal and modem. If we examine the signals carefully, we find two that look like they might do the job: pin 4, REQUEST TO SEND (RTS) and pin 5, CLEAR TO SEND (CTS). They certainly sound like they are a perfect handshaking pair. The problem is that they are really only half a handshake.

The DTE asserts (makes true) the REQUEST TO SEND line when it has data to send. The DCE acknowledges this, when it is ready, by asserting the CTS line. When the DTE sees the CTS it starts to send data. However, the DCE may not un-assert the CTS line any time it wishes; it must wait for the RTS line to go false first. Since only the DTE can control the data flow, we are still at square one.

Many printer manufacturers use pins 4 and 5 anyway. Since a printer is not defined in the specification of the protocol, what difference does it make if you don't follow the strict definitions? Other manufacturers try using different pins such as DTR or DSR, or a combination of several signals. There is no guarantee that a particular scheme will work with any interface that expects true RS-232 handshaking. The situation is getting better, however, because many manufacturers are letting their systems work outside the definitions of the protocol.

You can see why I prefer to use the Centronics parallel interface. There are times, of course, when the Centronics solution is not the best way to go, such as when there are distances over six feet involved, or when the equipment in question only is available with a serial interface. For those times here are some handy tips.

Doing It Yourself. Get all the information you can about the equipment. You will need to know how to modify baud rates, parity, and data word and stop bit lengths for both pieces of equipment. Also try to find out what kind of handshaking is required and whether the equipment is configured to be DCE or DTE. A DTE will send data on pin 2 and receive on pin 3, while a DCE sends on pin 3 and receives on pin 2. Be aware that in both cases pin 2 is called TRANSMIT DATA and pin 3 is RECEIVE DATA because all signal definitions are from the perspective of the DTE. Many manuals have arrows denoting data flow; for example, if the arrow points away from pin 2, then the system is DTE.

The first step is to connect pin 1 to pin 1 and pin 7 to pin 7 on both connectors. These are FRAME GROUND and SIGNAL GROUND (RETURN), respectively. For safety's sake make sure both are connected. The next set of pins is the data lines, pins 2 and 3. In many cases these connections will have to be crossed, pin 2 to pin 3 and pin 3 to pin 2. If your documentation spells out the connections, then so much the better. This is a good point to stop and try out what you have done so far.

Set both pieces of equipment to a slow baud rate—300 baud is fine. Most default setups call for seven data bits, no parity, one start bit and two stop bits. Connect the equipment and try sending data. If the transmitter seems to be sending but there is no response from the receiver, try reversing the connections to pins 2 and 3 at one end. If the transmitter locks up or if the receiver loses characters, then handshaking is at fault. Here you are going to have to rely on the manuals and experimentation. Some pairs of devices may have several handshaking lines at one end but only one at the other. In this case you can fool the multi-handshake line by having it handshake with itself by connecting complementary signals on the same connector. Two common examples of this are to connect the RTS (pin 4) back to the adjacent CTS (pin 5), and to connect DTR (pin 20) to DSR (pin 6) and sometimes, additionally, to CD (pin 8).

With perseverance and determination you should be successful. Once the interface is working, you can bring it up to full speed. Most printer interfaces will go to 1200 baud, and most terminals to 9600 or 19,200 baud.

Interfacing Aids. Several aids exist to help you get RS-232 interfaces working. The most common is a "breakout box," which is a small board or box with two DB-25 connectors and several switched or wire-jumper interconnect areas in between. The interconnect area allows fast changes to be made between the data lines and among the handshaking lines. Once the right combination has been found, a cable can be prepared with that configuration, or the breakout box can be left in place.

Conclusion. We have only really scratched the surface of the Centronics parallel and RS-232 serial interfaces. The information presented here should be enough to clear up most of the fog surrounding interfacing. Hopefully, you can now approach your next interconnecting task with confidence. ◇

interface allows, the RS-232 is inevitable."

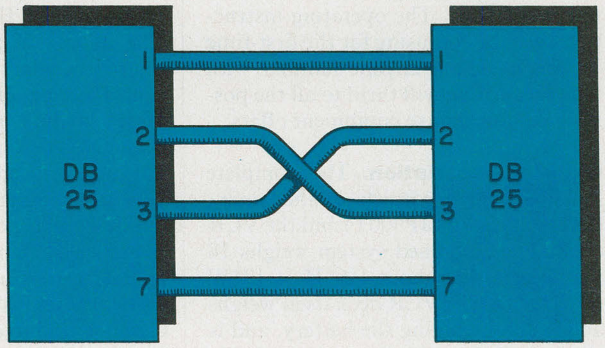


Fig. 6. A null modem cable reverses the leads to pins 2 and 3.