The RS232 STANDARD

A Tutorial with Signal Names and Definitions

(renamed the "EIA232 Standard" in the early 1990's)

Written by Christopher E. Strangio

Send Us Your Comments . . .

Contents

What is EIA232?
Next Topic | TOC

In the early 1960s, a standards committee, today known as the Electronic Industries Association, developed a common interface standard for data communications equipment. At that time, data communications was thought to mean digital data exchange between a centrally located mainframe computer and a remote computer terminal, or possibly between two terminals without a computer involved. These devices were linked by telephone voice lines, and consequently required a modem at each end for signal translation. While simple in concept, the many opportunities for data error that occur when transmitting data through an analog channel require a relatively complex design. It was thought that a standard was needed first to ensure reliable communication, and second to enable the

interconnection of equipment produced by different manufacturers, thereby fostering the benefits of mass production and competition. From these ideas, the RS232 standard was born. It specified signal voltages, signal timing, signal function, a protocol for information exchange, and mechanical connectors.

Over the 40+ years since this standard was developed, the Electronic Industries Association published three modifications, the most recent being the EIA232E standard introduced in 1991. Besides changing the name from RS232 to EIA232, some signal lines were renamed and various new ones were defined, including a shield conductor.

Likely Problems when Using an EIA232 Interface
Next Topic | Previous Topic | TOC

During this 40-year-long, rapidly evolving period in electronics, manufacturers adopted simplified versions of this interface for applications that were impossible to envision in the 1960s. Today, virtually all contemporary serial interfaces are EIA232-like in their signal voltages, protocols, and connectors, whether or not a modem is involved. Because no single "simplified" standard was agreed upon, however, many slightly different protocols and cables were created that obligingly mate with any EIA232 connector, but are incompatible with each other. Most of the difficulties you will encounter in EIA232 interfacing include at least one of the following:

1 - The absence or misconnection of flow control (handshaking) signals, resulting in buffer overflow or communications lock-up.

2 - Incorrect communications function (DTE versus DCE) for the cable in use, resulting in the reversal of the Transmit and Receive data lines as well as one or more handshaking lines.

3 - Incorrect connector gender or pin configuration, preventing cable connectors from mating properly.

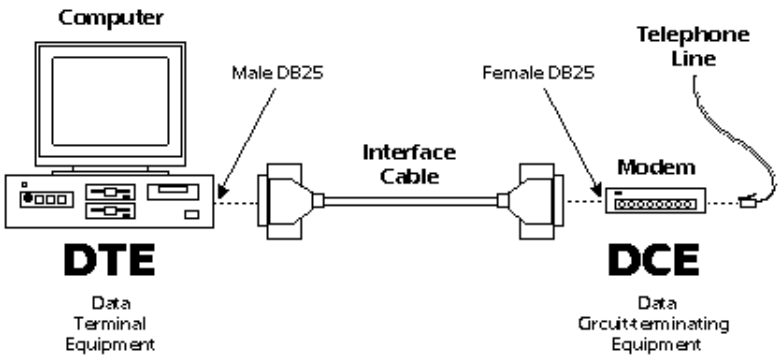
Fortunately, EIA232 driver circuitry is highly tolerant of misconnections, and will usually survive a drive signal being connected to ground, or two drive signals connected to each other. In any case, if the serial interface between two devices is not operating correctly, disconnect the cable joining this equipment until the problem is isolated.

Pin Assignments
Next Topic | Previous Topic | TOC

Go to DTE Pinout (looking into the computer’s serial connector)
Go to DCE Pinout (looking into the modem’s serial connector)

If the full EIA232 standard is implemented as defined, the equipment at the far end of the connection is

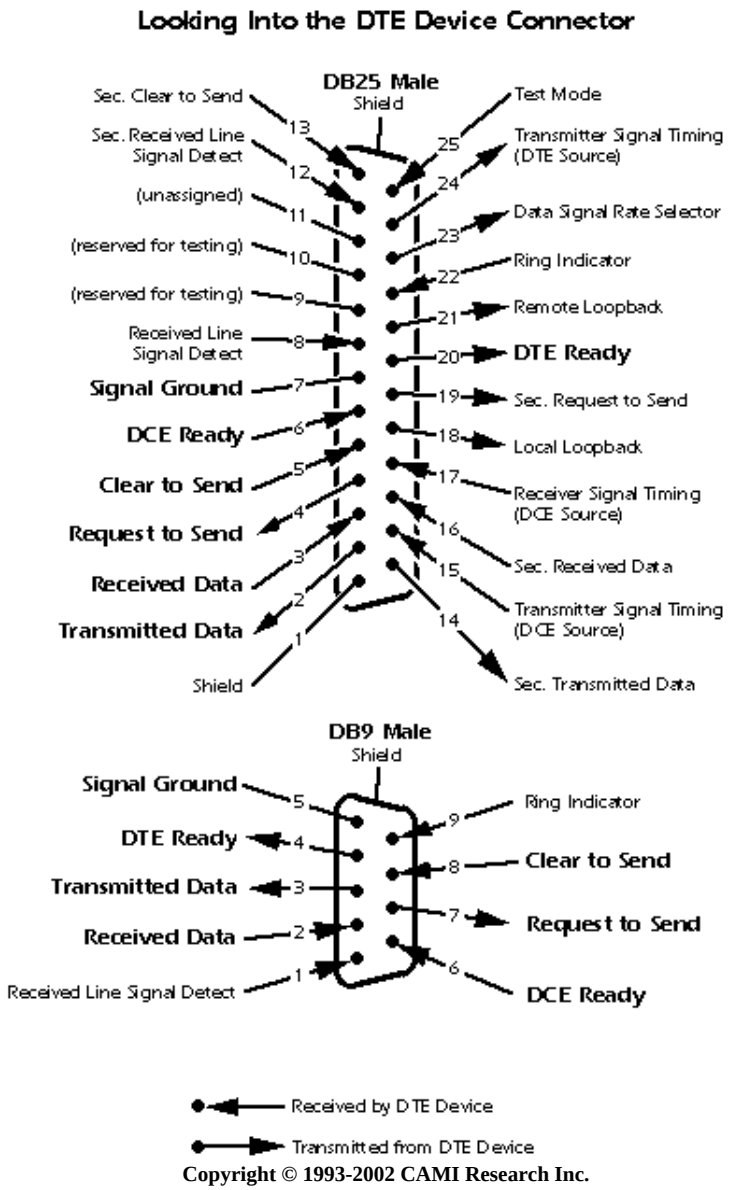
named the DTE device (Data Terminal Equipment, usually a computer or terminal), has a male DB25 connector, and utilizes 22 of the 25 available pins for signals or ground. Equipment at the near end of the connection (the telephone line interface) is named the DCE device (Data Circuit-terminating Equipment, usually a modem), has a female DB25 connector, and utilizes the same 22 available pins for signals and ground. The cable linking DTE and DCE devices is a parallel straight-through cable with no cross-overs or self-connects in the connector hoods. If all devices exactly followed this standard, all cables would be identical, and there would be no chance that an incorrectly wired cable could be used. This drawing shows the orientation and connector types for DTE and DCE devices:



EIA232 communication function and connector types for a personal computer and modem. DCE devices are sometimes called "Data Communications Equipment" instead of Data Circuit-terminating Equipment.

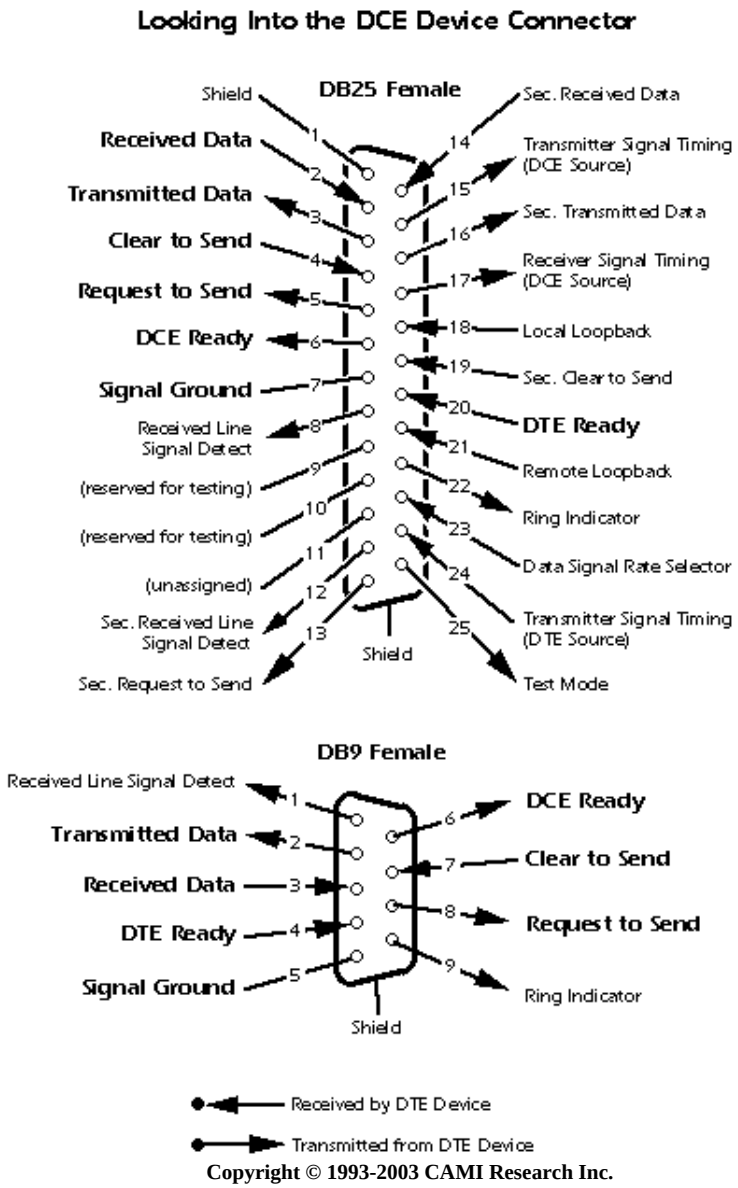
Here is the full EIA232 signal definition for the DTE device (usually the PC). The most commonly used signals are shown in bold.

[back to Pin Assignments description]



This shows the full EIA232 signal definition for the DCE device (usually the modem). The most commonly used signals are shown in bold.

[back to Pin Assignments description]



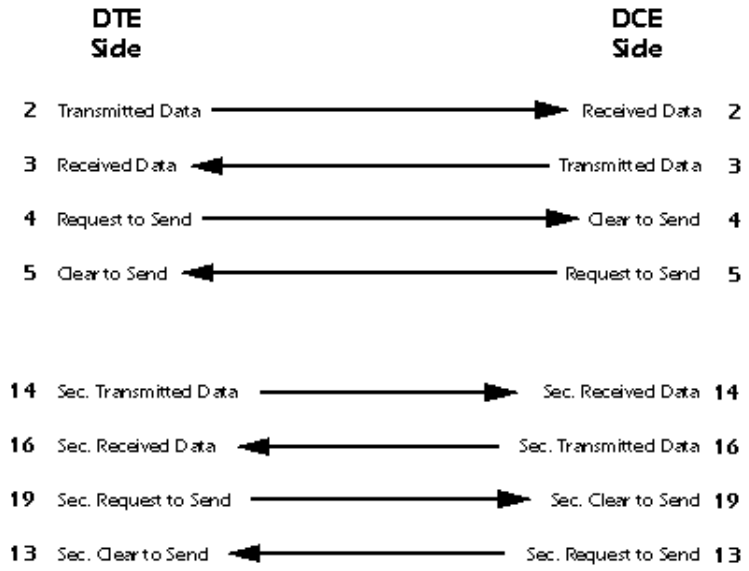
Many of the 22 signal lines in the EIA232 standard pertain to connections where the DCE device is a modem, and then are used only when the software protocol employs them. For any DCE device that is not a modem, or when two DTE devices are directly linked, far fewer signal lines are necessary.

You may have noticed in the pinout drawings that there is a secondary channel which includes a duplicate set of flow-control signals. This secondary channel provides for management of the remote modem, enabling baud rates to be changed on the fly, retransmission to be requested if a parity error is detected, and other control functions. This secondary channel, when used, is typically set to operate at a very low baud rate in comparison with the primary channel to ensure reliability in the control path. In addition, it may operate as either a simplex, half-duplex, or full-duplex channel, depending on the capabilities of the modem.

Transmitter and receiver timing signals (pins 15, 17, and 24) are used only for a synchronous

transmission protocol. For the standard asynchronous 8-bit protocol, external timing signals are unnecessary.

> ***IMPORTANT:*** Signal names that imply a direction, such as Transmit Data and Receive Data, are named from the point of view of the DTE device. If the EIA232 standard were strictly followed, these signals would have the same name for the same pin number on the DCE side as well. Unfortunately, this is not done in practice by most engineers, probably because no one can keep straight which side is DTE and which is DCE. As a result, direction-sensitive signal names are changed at the DCE side to reflect their *drive direction* at DCE. The following list gives the conventional usage of signal names:



Cable Wiring Examples
Next Topic | Previous Topic | TOC

The following wiring diagrams come from actual cables scanned by the CableEye® PC-Based Cable Test System. CableEye's software automatically draws schematics whenever it tests a cable. Click here to learn more about CableEye.

1 - DB9 All-Line Direct Extension
Next Cable | (no previous cable) || Next Topic

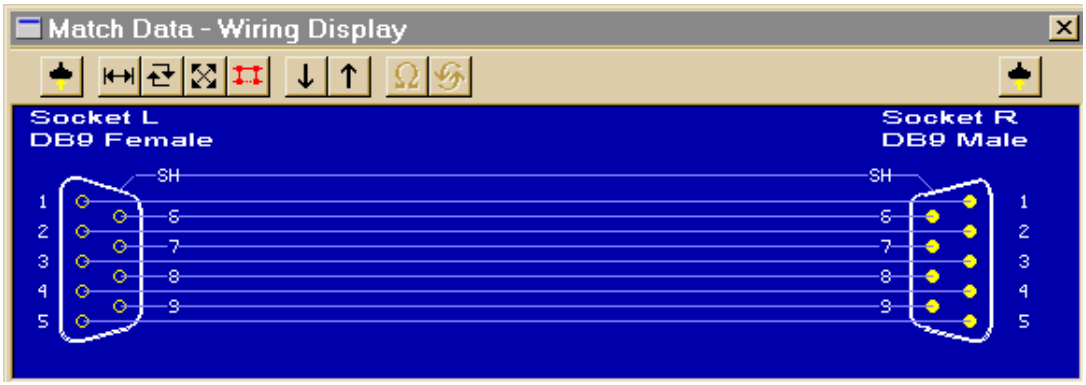
80K

This shows a 9-pin DTE-to-DCE serial cable that would result if the EIA232 standard were strictly followed. All 9 pins plus shield are directly extended from DB9 Female to DB9 Male. There are no crossovers or self-connects present. Use this cable to connect modems, printers, or any device that uses a DB9 connector to a PC's serial port.

This cable may also serve as an extension cable to increase the distance between a computer and serial device. *Caution:* do not exceed 25 feet separation between devices without a signal booster!

Left Side: Connect to **DTE** (computer)

Right Side: Connect to **DCE** (modem or other serial device)



Cable image created by CableEye®

---

2 - DB9 Loopback Connector

Next Cable | Previous Cable || Next Topic

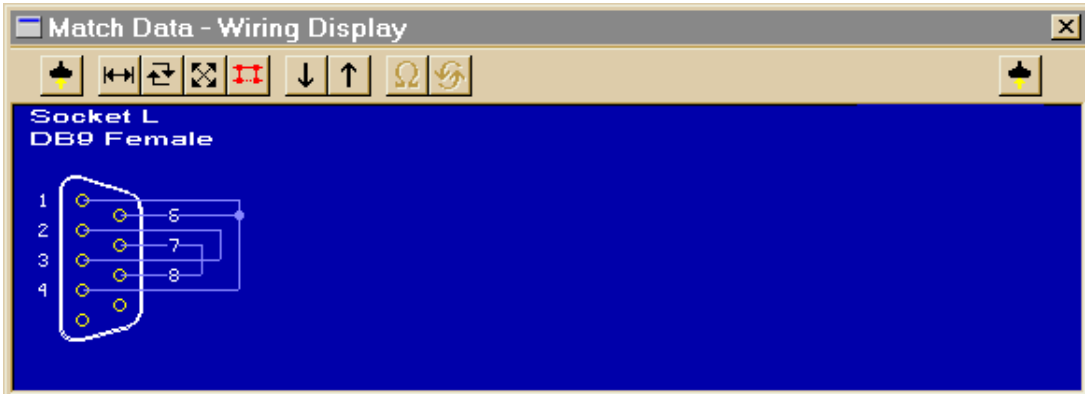
80K

A loopback connector usually consists of a connector without a cable and includes internal wiring to reroute signals back to the sender. This DB9 female connector would attach to a DTE device such as a personal computer. 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. Use loopback connectors to confirm proper operation of the computer's serial port. Once confirmed, insert the serial cable you plan to use and attach the loopback to the end of the serial cable to verify the cable.

In this case, Transmit Data joins to Received Data, Request-to-Send joins to Clear-to-Send, and DTE-Ready joins to DCE-Ready and Received Line Signal Detect.

Left Side: Connect to **DTE** (computer)

Right Side: (none)



Cable image created by CableEye®

---

3 - DB9 Null Modem Cable
Next Cable | Previous Cable || Next Topic

Use this female-to-female cable in any application where you wish to connect two DTE devices (for example, two computers). A male-to-male equivalent of this cable would be used to connect two DCE devices.

80K

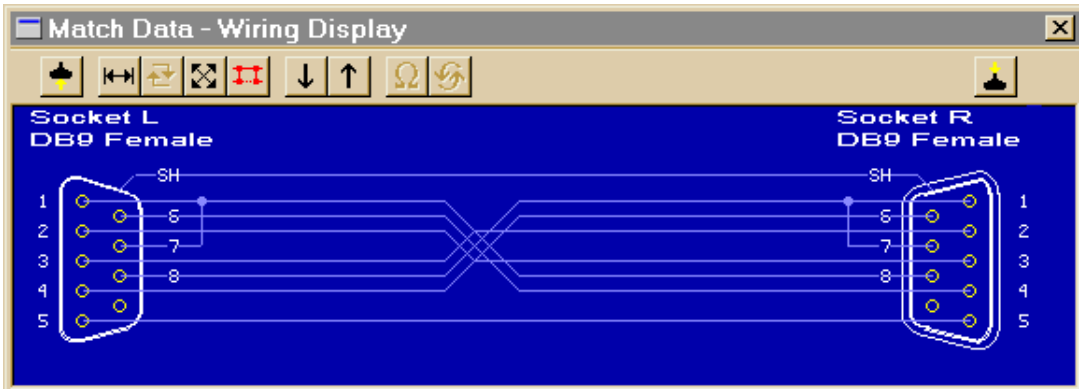
The cable shown below is intended for RS232 asynchronous communications (most PC-based systems). If you are using synchronous communications, the null modem will have additional connections for timing signals, and a DB25 connector would be necessary.

NOTE: Not all null modem cables connect handshaking lines the same way. In this cable, Request-to-Send (RTS, pin 7) asserts the Carrier Detect (pin 1) on the same side and the Clear-to-Send (CTS, pin 8) on the other side of the cable.

This device may also be available in the form of an adapter.

Left Side: Connect to 9-pin **DTE** (computer)

Right Side: Connect to 9-pin **DTE** (computer)



Cable image created by CableEye®

---

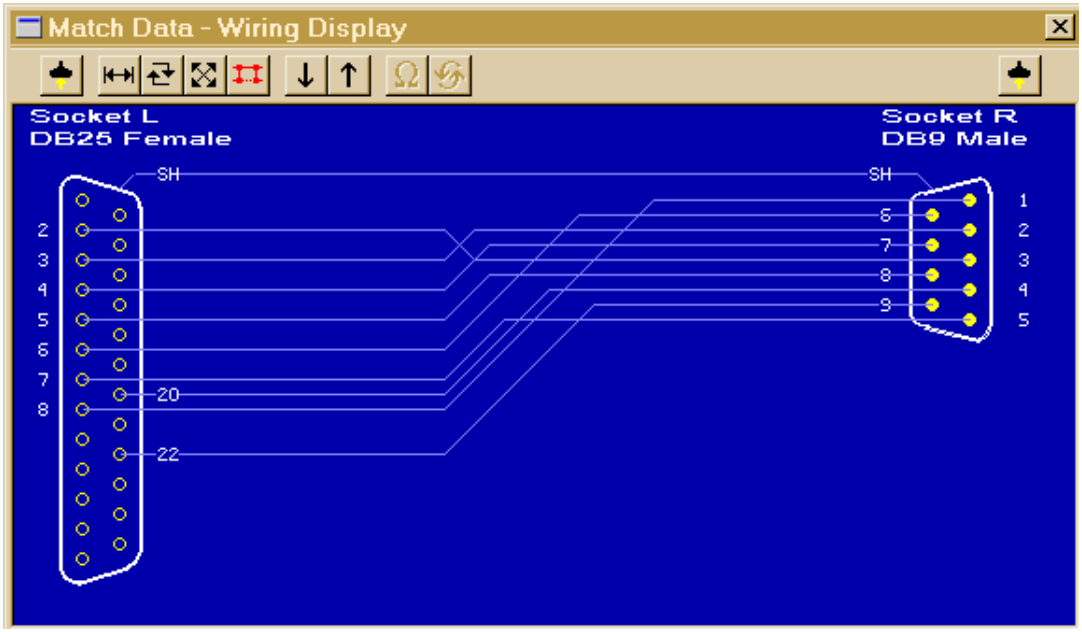
4 - DB25 to DB9 Adapter
Next Cable | Previous Cable || Next Topic

Signals on the DB25 DTE side are directly mapped to the DB9 assignments for a DTE device. Use this to adapt a 25-pin COM connector on the back of a computer to mate with a 9-pin serial DCE device, such as a 9-pin serial mouse or modem. This adapter may also be in the form of a cable.

80K

Left Side: Connect to 25-pin **DTE** (computer)

Right Side: Connect to 9-pin **DCE** (modem)



Cable image created by CableEye®

---

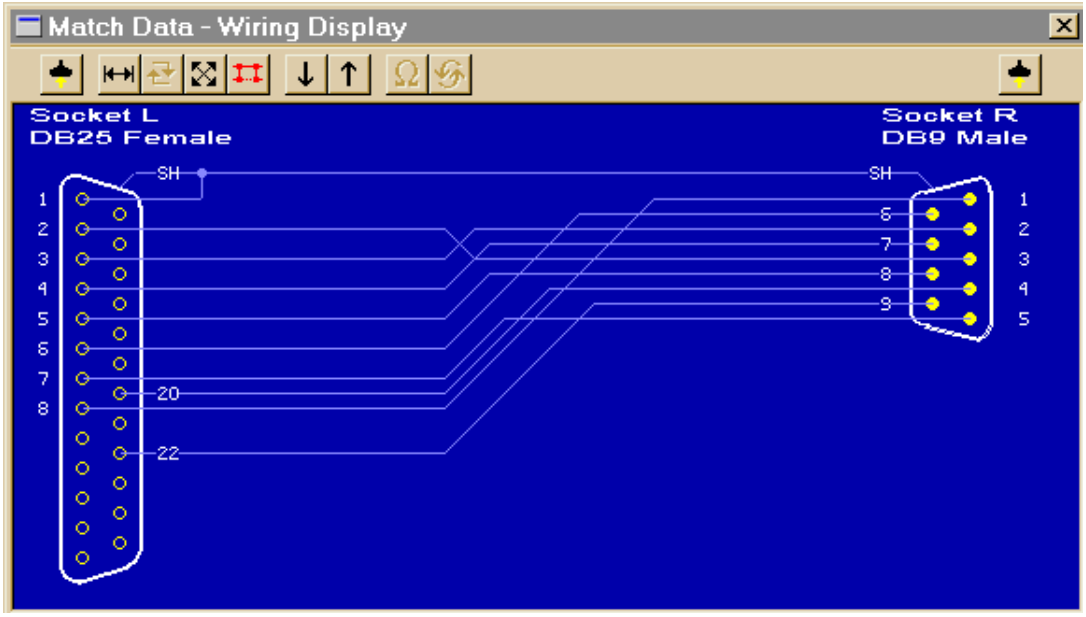
5 - DB25 to DB9 Adapter (pin 1 connected to shield)
Next Cable | Previous Cable || Next Topic

84K

This adapter has the same wiring as the previous cable (#4) except that pin 1 is wired to the connector shell (shield). Note that the cable's shield is usually a foil blanket surrounding all conductors running the length of the cable and joining the connector shells. Pin 1 of the EIA232 specification, called out as "shield", may be separate from the earth ground usually associated with the connector shells.

Left Side: Connect to 25-pin **DTE** (computer)

Right Side: Connect to 9-pin **DCE** (modem)



Cable image created by CableEye®

---

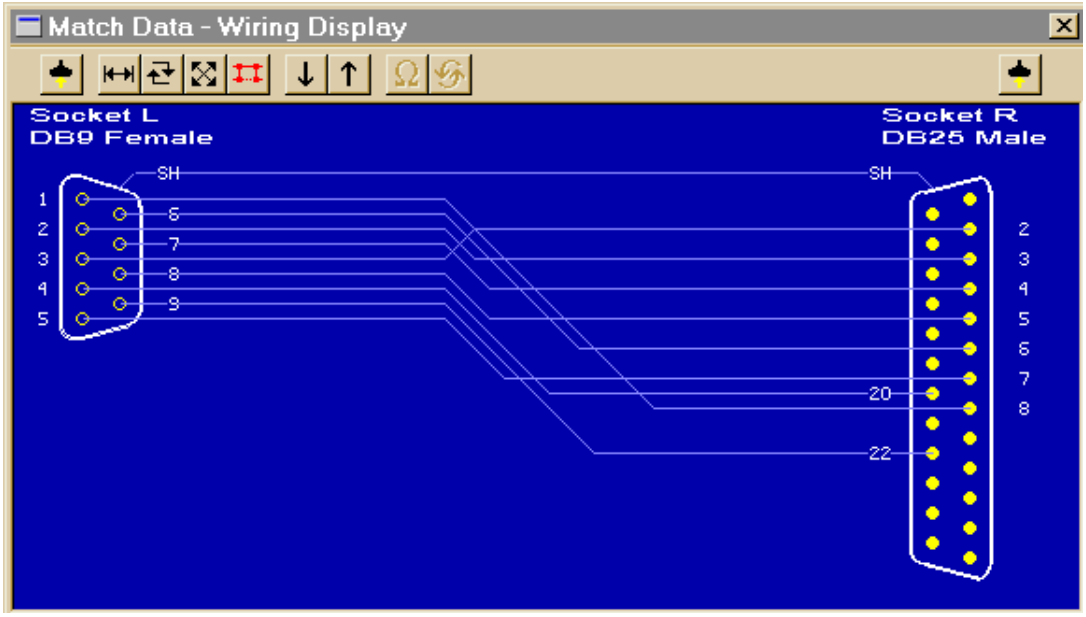
6 - DB9 to DB25 Adapter
Next Cable | Previous Cable || Next Topic

Signals on the DB9 DTE side are directly mapped to the DB25 assignments for a DTE device. Use this to adapt a 9-pin COM connector on the back of a computer to mate with a 25-pin serial DCE devices, such as a modem. This adapter may also be in the form of a cable.

80K

Left Side: Connect to 9-pin **DTE** (computer)

Right Side: Connect to 25-pin **DCE** (modem)



Cable image created by CableEye®

---

7 - DB25 All-Line Direct Extension
Next Cable | Previous Cable || Next Topic

84K

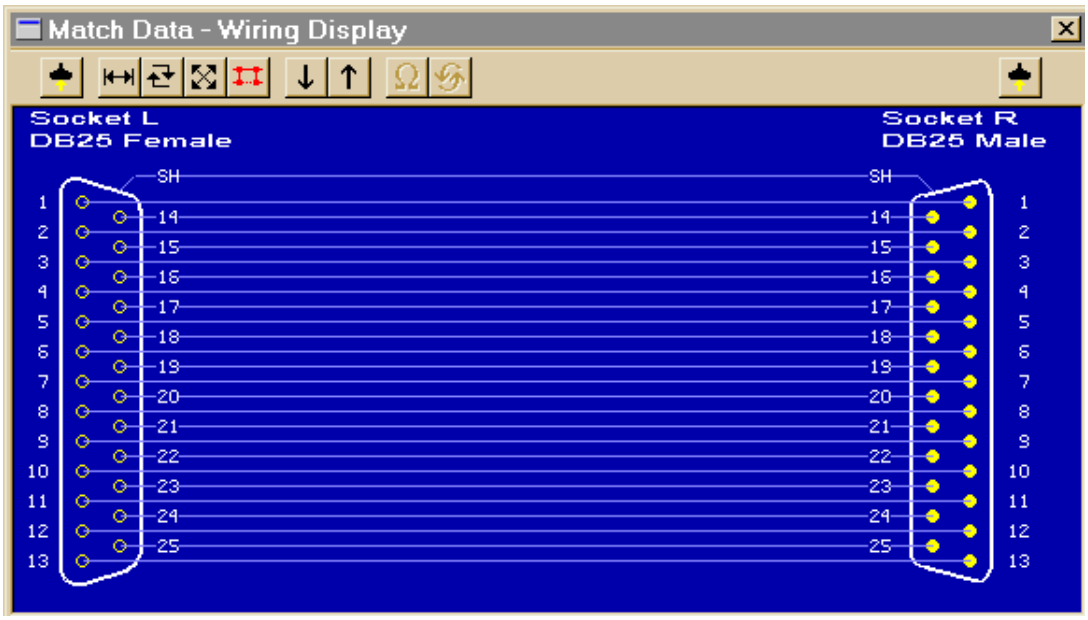
This shows a 25-pin DTE-to-DCE serial cable that would result if the EIA232 standard were strictly followed. All 25 pins plus shield are directly extended from DB25 Female to DB25 Male. There are no crossovers or self-connects present. Use this cable to connect modems, printers, or any serial device that uses a DB25 connector to a PC's serial port.

This cable may also serve as an extension cable to increase the distance between computer and serial device. *Caution:* do not exceed 25 feet separation between devices without a signal booster!

Caution: the male end of this cable (right) also fits a PC's parallel printer port. You may use this cable to extend the length of a printer cable, but DO NOT attach a serial device to the computer's parallel port. Doing so may cause damage to both devices.

Left Side: Connect to 25-pin **DTE** (computer)

Right Side: Connect to 25-pin **DCE** (modem)



Cable image created by CableEye®

---

8 - DB25 Loopback Connector
Next Cable | Previous Cable || Next Topic

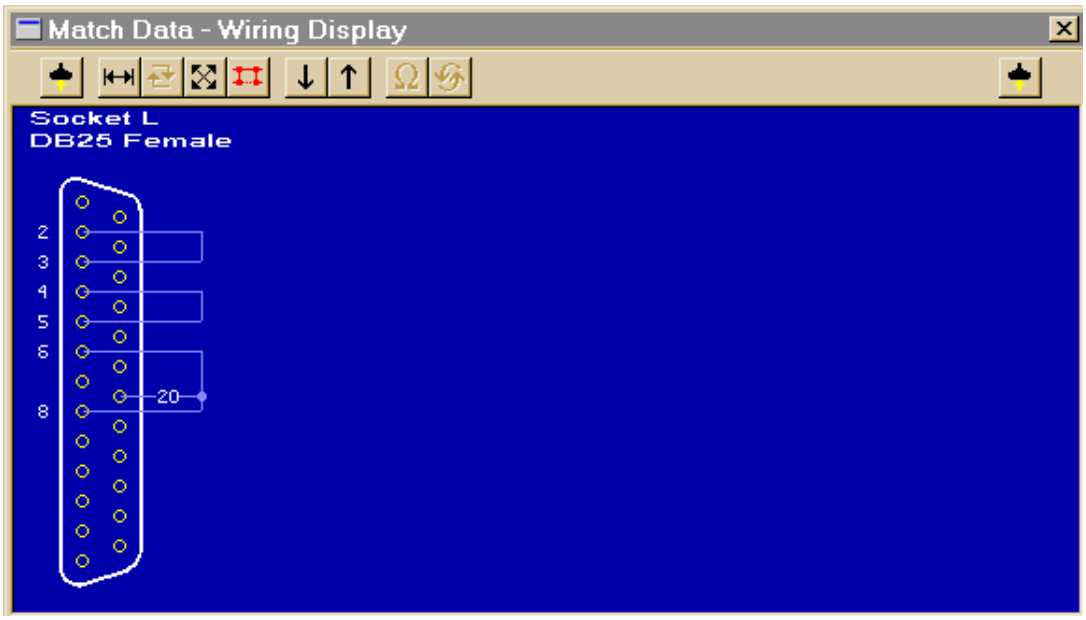
80K

A loopback connector usually consists of a connector without a cable and includes internal wiring to reroute signals back to the sender. This DB25 female connector would attach to a DTE device such as a personal computer. 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. Use loopback connectors to confirm proper operation of the computer's serial port. Once confirmed, insert the serial cable you plan to use and attach the loopback to the end of the serial cable the verify the cable.

In this case, Transmit Data joins to Received Data, Request-to-Send joins to Clear-to-Send, and DTE-Ready joins to DCE-Ready and Received Line Signal Detect.

Left Side: Connect to 25-pin **DTE** (computer)

Right Side: (none)



Cable image created by CableEye®

---

9 - DB25 Null Modem (no handshaking)
Next Cable | Previous Cable || Next Topic

Use this female-to-female cable in any application where you wish to connect two DTE devices (for example, two computers). A male-to-male equivalent of this cable would be used to connect two DCE devices.

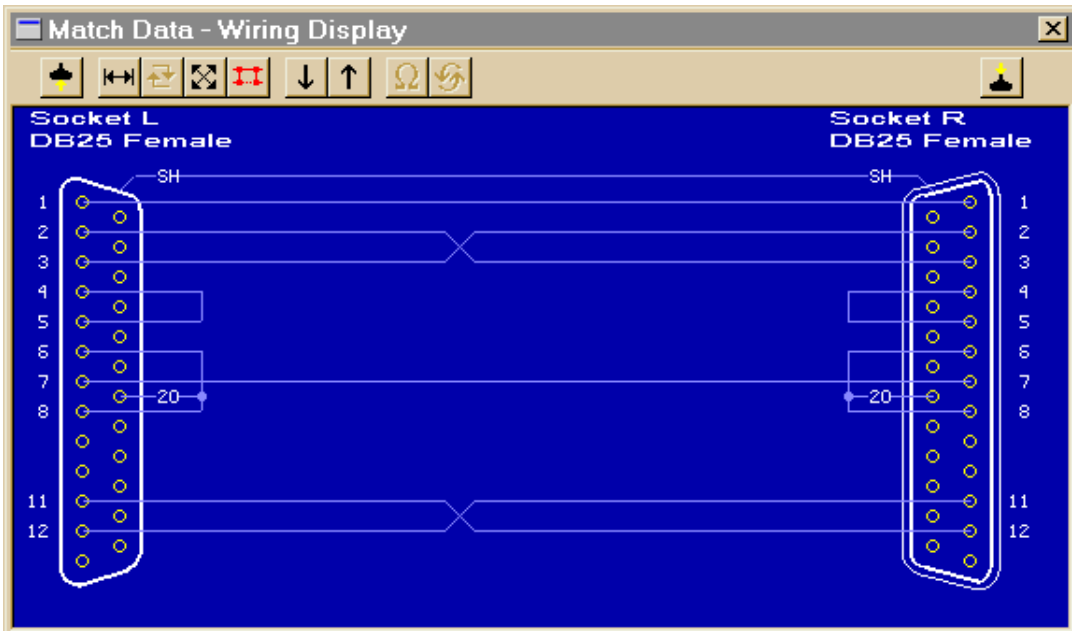
84K

Note that Pins 11 and 12 are not necessary for this null modem cable to work. As is often the case, the manufacturer of equipment that uses this cable had a proprietary application in mind. We show it here to emphasize that custom serial cables may include connections for which no purpose is clear.

IMPORTANT: This cable employs NO handshaking lines between devices. The handshake signals on each side are artificially made to appear asserted by the use of self-connects on each side of the cable (for example, between pins 4 and 5). Without hardware handshaking, you risk buffer overflow at one or both ends of the transmission unless STX and ETX commands are inserted in the dataflow by software.

Left Side: Connect to 25-pin **DTE** (computer)

Right Side: Connect to 25-pin **DTE** (computer)



Cable image created by CableEye®

---

10 - DB25 Null Modem (standard handshaking)
Next Cable | Previous Cable || Next Topic

Use this female-to-female cable in any application where you wish to connect two DTE devices (for example, two computers). A male-to-male equivalent of this cable would be used to connect two DCE devices.

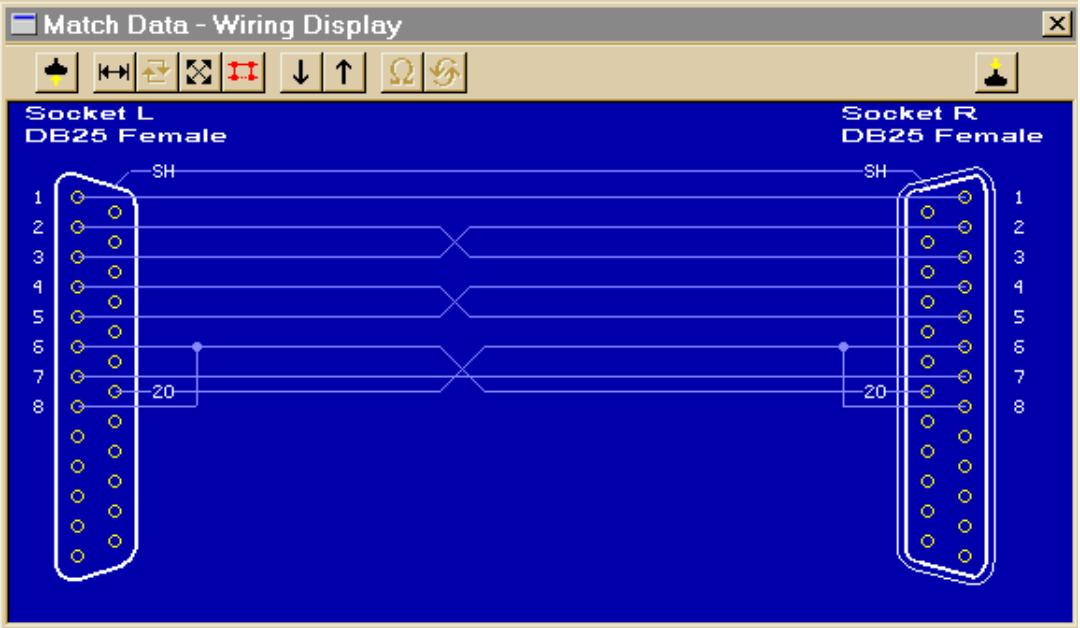
84K

The cable shown below is intended for EIA232 asynchronous communications (most PC-based systems). If you are using synchronous communications, the null modem will have additional connections for timing signals not shown here.

NOTE: Not all null modem cables connect handshaking lines the same way. Refer to the manual for your equipment if you experience problems. In this cable, the DTE Ready (pin 20) on one side asserts the DCE Ready (pin 6) and the Request to Send (pin 5) on the other side.

Left Side: Connect to 25-pin **DTE** (computer)

Right Side: Connect to 25-pin **DTE** (computer)



Cable image created by CableEye®

---

11 - DB25 Null Modem (unusual handshaking)
Next Cable | Previous Cable || Next Topic

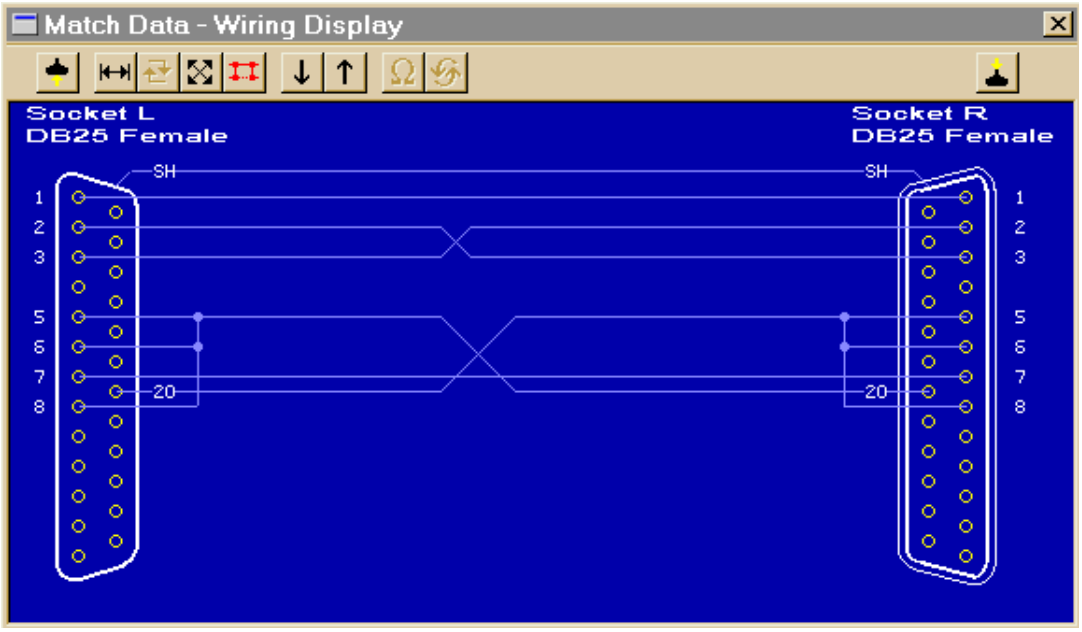
Use this female-to-female cable in any application where you wish to connect two DTE devices (for example, two computers). A male-to-male equivalent of this cable would be used to connect two DCE devices.

84K

NOTE: Not all null modem cables connect handshaking lines the same way. Refer to the manual for your equipment if you experience problems. In this cable, the DTE Ready (pin 20) on one side asserts the Clear to Send (pin 5), DCE Ready (pin 6), and Carrier Detect (pin 8) on the other side.

Left Side: Connect to 25-pin **DTE** (computer)

Right Side: Connect to 25-pin **DTE** (computer)



Cable image created by CableEye®

---

12 - DB25 Null Modem (unusual handshaking)
Next Cable | Previous Cable || Next Topic

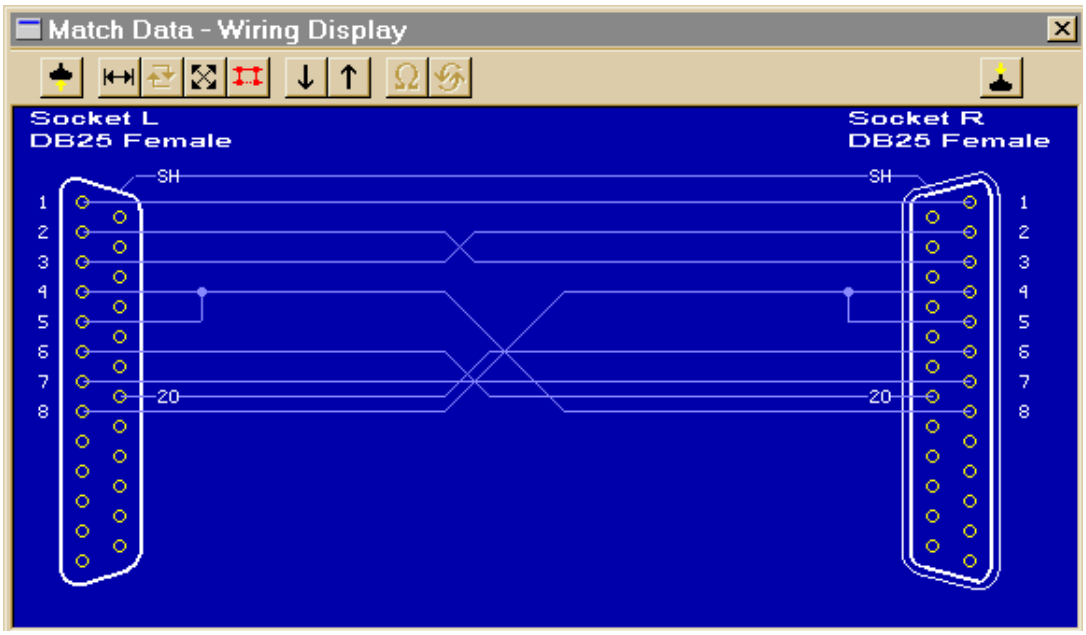
Use this female-to-female cable in any application where you wish to connect two DTE devices (for example, two computers). A male-to-male equivalent of this cable would be used to connect two DCE devices.

84K

NOTE: Not all null modem cables connect handshaking lines the same way. Refer to the manual for your equipment if you experience problems. In this cable, the Request-to-Send (pin 4) on one side asserts the Clear-to-Send (pin 5) on the SAME side (self-connect) and the Carrier Detect (pin 8) on the other side. The other handshaking signals are employed in a conventional manner.

Left Side: Connect to 25-pin **DTE** (computer)

Right Side: Connect to 25-pin **DTE** (computer)



Cable image created by CableEye®

---

13 - DB25 Null Modem (unusual handshaking)
Next Cable | Previous Cable || Next Topic

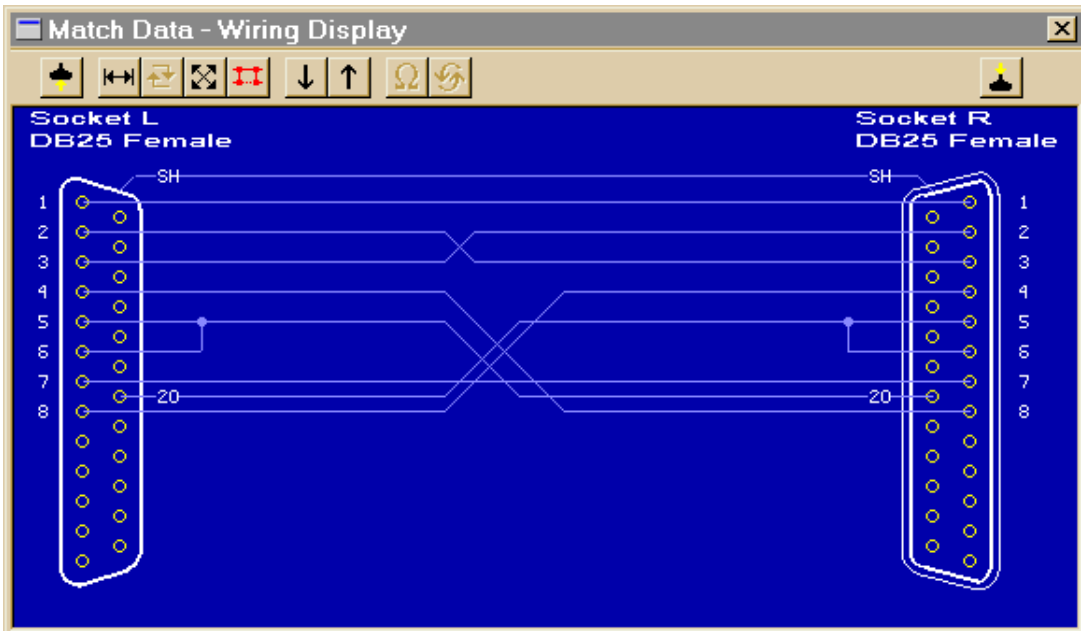
Use this female-to-female cable in any application where you wish to connect two DTE devices (for example, two computers). A male-to-male equivalent of this cable would be used to connect two DCE devices.

84K

NOTE: Not all null modem cables connect handshaking lines the same way. Refer to the manual for your equipment if you experience problems. In this cable, the DTE Ready (pin 20) on one side asserts the Clear-to-Send (pin 5) and the DCE Ready (pin 6) on the other side. Request-to-Send (pin 4) on one side asserts Received Line Signal Detect (pin 8) on the other side.

Left Side: Connect to 25-pin **DTE** (computer)

Right Side: Connect to 25-pin **DTE** (computer)



Cable image created by CableEye®

---

14 - DB25 Null Modem (unusual handshaking)
Next Cable | Previous Cable || Next Topic

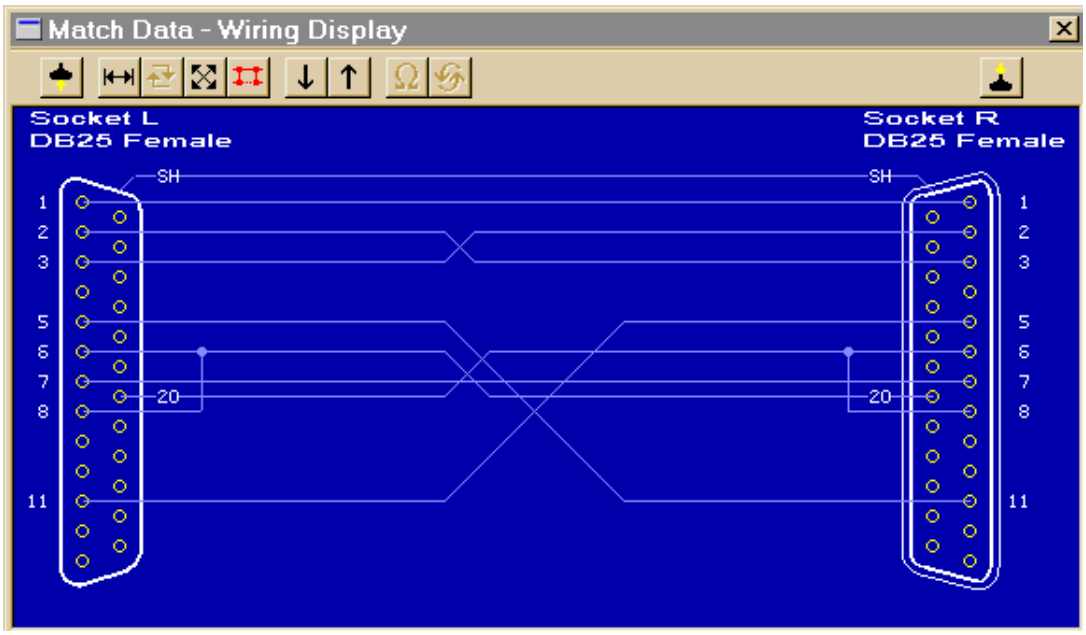
Use this female-to-female cable in any application where you wish to connect two DTE devices (for example, two computers). A male-to-male equivalent of this cable would be used to connect two DCE devices.

84K

NOTE: Not all null modem cables connect handshaking lines the same way. Refer to the manual for your equipment if you experience problems. In this cable, the DTE Ready (pin 20) on one side asserts the DCE Ready (pin 6), and Carrier Detect (pin 8) on the other side. Request to Send (pin 4) is unused, and Clear-to-Send (pin 5) is driven by a proprietary signal (pin 11) determined by the designer of this cable.

Left Side: Connect to 25-pin **DTE** (computer)

Right Side: Connect to 25-pin **DTE** (computer)



Cable image created by CableEye®

---

15 - DB25 Null Modem Cable (synchronous communications)
Next Cable | Previous Cable || Next Topic

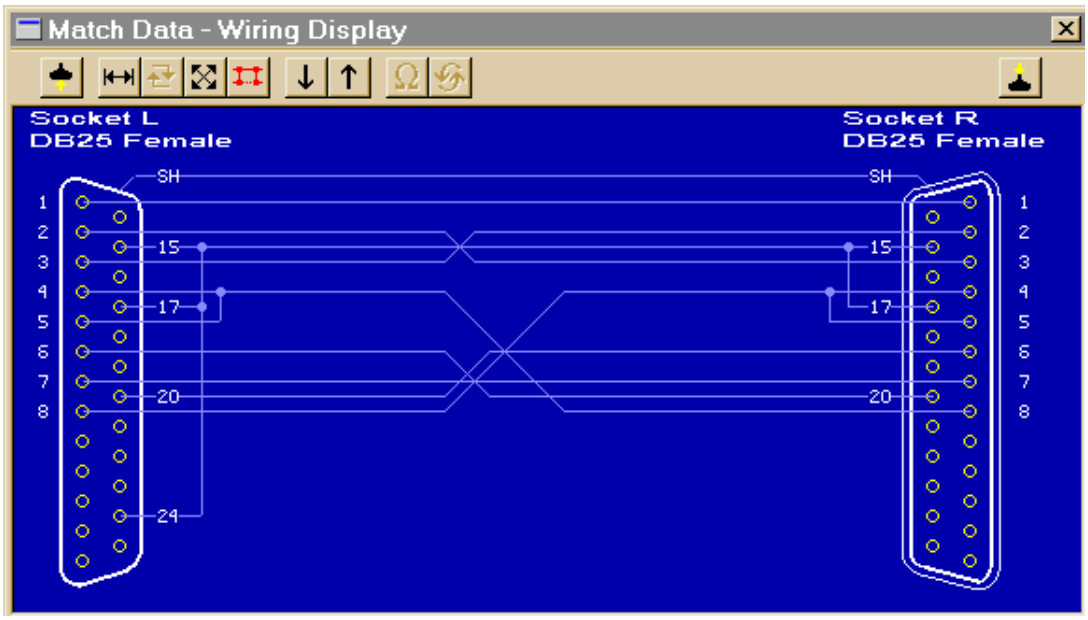
84K

This female-to-female cable is intended for **synchronous** EIA232 connections, and is designed to connect two DTE devices. It contains the standard connections of an asynchronous null modem cable, plus additional connections on pins 15, 17, and 24 for synchronous timing signals. To connect two DCE devices, use a male-to-male equivalent of this cable.

For synchronous communications, the null modem cable includes an additional conductor for timing signals, and joins pins 15, 17, and 24 on one side to pins 15 and 17 on the other. Pin 24 on the right side should connect to the timing signal source.

Left Side: Connect to 25-pin **DTE** (computer)

Right Side: Connect to 25-pin **DTE** (computer)



Cable image created by CableEye®

---

16 - DB25 Null Modem Cable (unconventional, may pose risk)
(no more) | Previous Cable || Next Topic

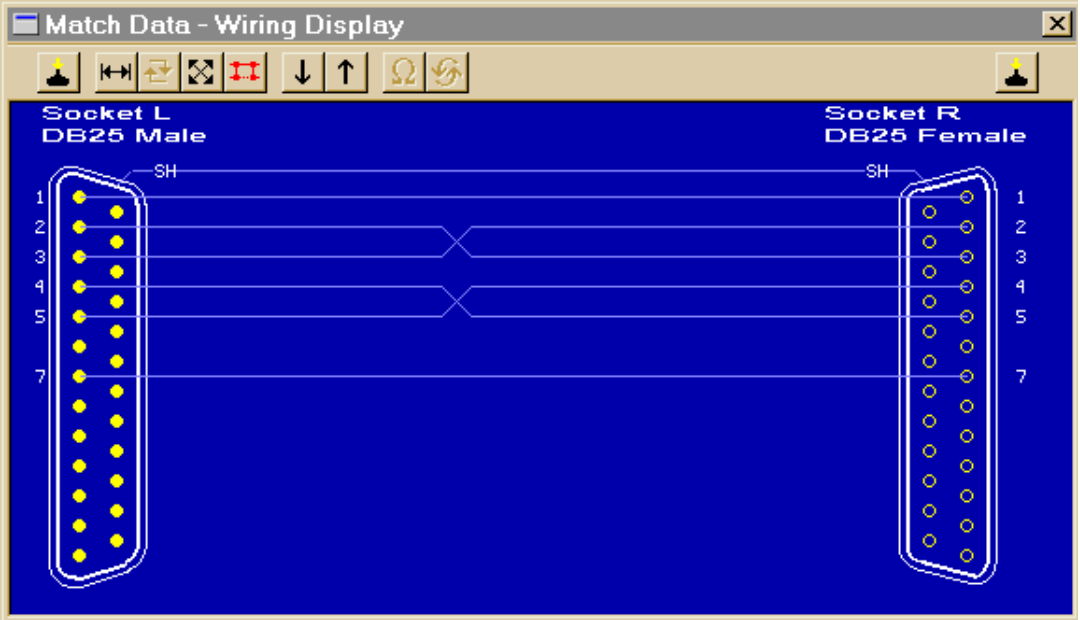
This simplified null modem cable uses only Request-to-Send (pin 4) and Clear-to-Send (pin 5) as handshaking lines; DTE Ready, DCE Ready, and Carrier Detect are not employed, so this cable should not be used with modems.

80K

CAUTION! Normally, null modem cables have the same gender on each connector (either both male for two DTE devices, or both female for two DCE devices). This cable would be used when the gender on one of the devices does not conform to the standard. However, the opposite genders imply usage as a straight through cable, and if used in that manner will not function. Further, if used as a standard null-modem between two computers, the opposite gender allows you to connect one end to the parallel port, an impermissible situation that may cause hardware damage.

Left Side: Connect to 25-pin **DTE** (computer) with Gender Changer

Right Side: Connect to 25-pin **DTE** (computer)



Cable image created by CableEye®

Signal Definitions
Next Topic | Previous Topic | TOC

Signal functions in the EIA232 standard can be subdivided into six categories. These categories are summarized below, after which each signal described.

1 - Signal ground and shield.

2 - Primary communications channel. This is used for data interchange, and includes flow control signals.

3 - Secondary communications channel. When implemented, this is used for control of the remote modem, requests for retransmission when errors occur, and governance over the setup of the primary channel.

4 - Modem status and control signals. These signals indicate modem status and provide intermediate checkpoints as the telephone voice channel is established.

5 - Transmitter and receiver timing signals. If a synchronous protocol is used, these signals provide timing information for the transmitter and receiver, which may operate at different baud rates.

6 - Channel test signals. Before data is exchanged, the channel may be tested for its integrity, and the baud rate automatically adjusted to the maximum rate that the channel can support.

Signal Ground and Shield

Pin 7, Pin 1, and the **shell** are included in this category. Cables provide separate paths for each, but internal wiring often connects pin 1 and the cable shell/shield to signal ground on pin 7.

Pin 7 - Ground All signals are referenced to a common ground, as defined by the voltage on pin 7. This conductor may or may not be connected to protective ground inside the DCE device. The existence of a defined ground potential within the cable makes the EIA232 standard different from a balanced differential voltage standard, such as EIA530, which provides far greater noise immunity.

Primary Communications Channel

Pin 2 - Transmitted Data (TxD) This signal is active when data is transmitted from the DTE device to the DCE device. When no data is transmitted, the signal is held in the mark condition (logic ’1’, negative voltage).

> *NOTE:* Pin 2 on the DCE device is commonly labeled "Received Data", although by the EIA232 standard it should still be called Transmitted Data because the data is thought to be destined for a remote DTE device.

Pin 3 - Received Data (RxD) This signal is active when the DTE device receives data from the DCE device. When no data is transmitted, the signal is held in the mark condition (logic ’1’, negative voltage).

> *NOTE:* Pin 3 on the DCE device is commonly labeled "Transmitted Data", although by the EIA232 standard it should still be called Received Data because the data is thought to arrive from a remote DTE device.

Pin 4 - Request to Send (RTS) This signal is asserted (logic ’0’, positive voltage) to prepare the DCE device for accepting transmitted data from the DTE device. Such preparation might include enabling the receive circuits, or setting up the channel direction in half-duplex applications. When the DCE is ready, it acknowledges by asserting Clear to Send.

> *NOTE:* Pin 4 on the DCE device is commonly labeled "Clear to Send", although by the EIA232 standard it should still be called Request to Send because the request is thought to be destined for a remote DTE device.

Pin 5 - Clear to Send (CTS) This signal is asserted (logic ’0’, positive voltage) by the DCE device to inform the DTE device that transmission may begin. RTS and CTS are commonly used as handshaking signals to moderate the flow of data into the DCE device.

> *NOTE:* Pin 5 on the DCE device is commonly labeled "Request to Send", although by the

EIA232 standard it should still be called Clear to Send because the signal is thought to originate from a remote DTE device.

Secondary Communications Channel
Next Topic | Previous Topic | TOC

Pin 14 - Secondary Transmitted Data (STxD)

Pin 16 - Secondary Received Data (SRxD)

Pin 19 - Secondary Request to Send (SRTS)

Pin 13 - Secondary Clear to Send (SCTS)

These signals are equivalent to the corresponding signals in the primary communications channel. The baud rate, however, is typically much slower in the secondary channel for increased reliability.

Modem Status and Control Signals
Next Topic | Previous Topic | TOC

Pin 6 - DCE Ready (DSR) When originating from a modem, this signal is asserted (logic ’0’, positive voltage) when the following three conditions are all satisfied:

1 - The modem is connected to an active telephone line that is "off-hook";

2 - The modem is in data mode, not voice or dialing mode; and

3 - The modem has completed dialing or call setup functions and is generating an answer tone.

If the line goes "off-hook", a fault condition is detected, or a voice connection is established, the DCE Ready signal is deasserted (logic ’1’, negative voltage).

IMPORTANT: If DCE Ready originates from a device other than a modem, it may be asserted to indicate that the device is turned on and ready to function, or it may not be used at all. If unused, DCE Ready should be permanently asserted (logic ’0’, positive voltage) within the DCE device or by use of a self-connect jumper in the cable. Alternatively, the DTE device may be programmed to ignore this signal.

Pin 20 - DTE Ready (DTR) This signal is asserted (logic ’0’, positive voltage) by the DTE device when it wishes to open a communications channel. If the DCE device is a modem, the assertion of DTE Ready prepares the modem to be connected to the telephone circuit, and, once connected, maintains the

connection. When DTE Ready is deasserted (logic ’1’, negative voltage), the modem is switched to "on-hook" to terminate the connection.

> *IMPORTANT:* If the DCE device is not a modem, it may require DTE Ready to be asserted before the device can be used, or it may ignore DTE Ready altogether. If the DCE device (for example, a printer) is not responding, confirm that DTE Ready is asserted before you search for other explanations.

Pin 8 - Received Line Signal Detector (CD) (also called carrier detect) This signal is relevant when the DCE device is a modem. It is asserted (logic ’0’, positive voltage) by the modem when the telephone line is "off-hook", a connection has been established, and an answer tone is being received from the remote modem. The signal is deasserted when no answer tone is being received, or when the answer tone is of inadequate quality to meet the local modem’s requirements (perhaps due to a noisy channel).

Pin 12 - Secondary Received Line Signal Detector (SCD) This signal is equivalent to the Received Line Signal Detector (pin 8), but refers to the secondary channel.

Pin 22 - Ring Indicator (RI) This signal is relevant when the DCE device is a modem, and is asserted (logic ’0’, positive voltage) when a ringing signal is being received from the telephone line. The assertion time of this signal will approximately equal the duration of the ring signal, and it will be deasserted between rings or when no ringing is present.

Pin 23 - Data Signal Rate Selector This signal may originate either in the DTE or DCE devices (but not both), and is used to select one of two prearranged baud rates. The asserted condition (logic ’0’, positive voltage) selects the higher baud rate.

Transmitter and Receiver Timing Signals
Next Topic | Previous Topic | TOC

Pin 15 - Transmitter Signal Element Timing (TC) (also called Transmitter Clock) This signal is relevant only when the DCE device is a modem and is operating with a synchronous protocol. The modem generates this clock signal to control exactly the rate at which data is sent on Transmitted Data (pin 2) from the DTE device to the DCE device. The logic ’1’ to logic ’0’ (negative voltage to positive voltage) transition on this line causes a corresponding transition to the next data element on the Transmitted Data line. The modem generates this signal continuously, except when it is performing internal diagnostic functions.

Pin 17 - Receiver Signal Element Timing (RC) (also called Receiver Clock) This signal is similar to TC described above, except that it provides timing information for the DTE receiver.

Pin 24 - Transmitter Signal Element Timing (ETC) (also called External Transmitter Clock) Timing signals are provided by the DTE device for use by a modem. This signal is used only when TC and RC (pins 15 and 17) are not in use. The logic ’1’ to logic ’0’ transition (negative voltage to positive voltage) indicates the time-center of the data element. Timing signals will be provided whenever the DTE is turned on, regardless of other signal conditions.

Channel Test Signals

Next Topic | Previous Topic | TOC

Pin 18 - Local Loopback (LL) This signal is generated by the DTE device and is used to place the modem into a test state. When Local Loopback is asserted (logic ’0’, positive voltage), the modem redirects its modulated output signal, which is normally fed into the telephone line, back into its receive circuitry. This enables data generated by the DTE to be echoed back through the local modem to check the condition of the modem circuitry. The modem asserts its Test Mode signal on Pin 25 to acknowledge that it has been placed in local loopback condition.

Pin 21 - Remote Loopback (RL) This signal is generated by the DTE device and is used to place the remote modem into a test state. When Remote Loopback is asserted (logic ’0’, positive voltage), the remote modem redirects its received data back to its transmitted data input, thereby remodulating the received data and returning it to its source. When the DTE initiates such a test, transmitted data is passed through the local modem, the telephone line, the remote modem, and back, to exercise the channel and confirm its integrity. The remote modem signals the local modem to assert Test Mode on pin 25 when the remote loopback test is underway.

Pin 25 - Test Mode (TM) This signal is relevant only when the DCE device is a modem. When asserted (logic ’0’, positive voltage), it indicates that the modem is in a Local Loopback or Remote Loopback condition. Other internal self-test conditions may also cause Test Mode to be asserted, and depend on the modem and the network to which it is attached.

Electrical Standards

Next Topic | Previous Topic | TOC

The EIA232 standard uses negative, bipolar logic in which a negative voltage signal represents logic ’1’, and positive voltage represents logic ’0’. This probably originated with the pre-RS232 current loop standard used in 1950s-vintage teletype machines in which a flowing current (and hence a low voltage) represents logic ’1’. Be aware that the negative logic assignment of EIA232 is the reverse of that found in most modern digital circuit designs. See the inside rear cover of the CableEye manual for a comparison.

Common Signal Ground

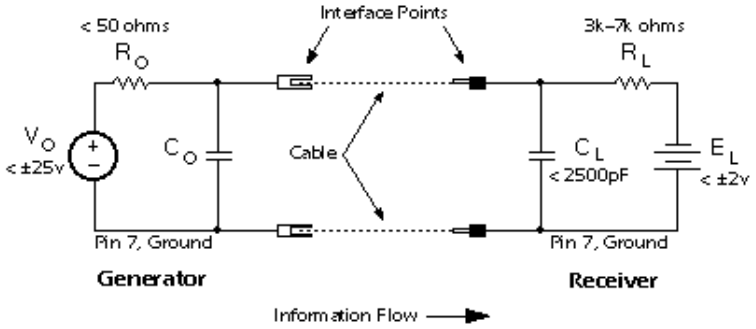
Next Topic | Previous Topic | TOC

The EIA232 standard includes a common ground reference on Pin 7, and is frequently joined to Pin 1 and a circular shield that surrounds all 25 cable conductors. Data, timing, and control signal voltages are measured with respect to this common ground. EIA232 cannot be used in applications where the equipment on opposite ends of the connection must be electrically isolated.

NOTE: optical isolators may be used to achieve ground isolation, however, this option is not mentioned or included in the EIA232 specification.

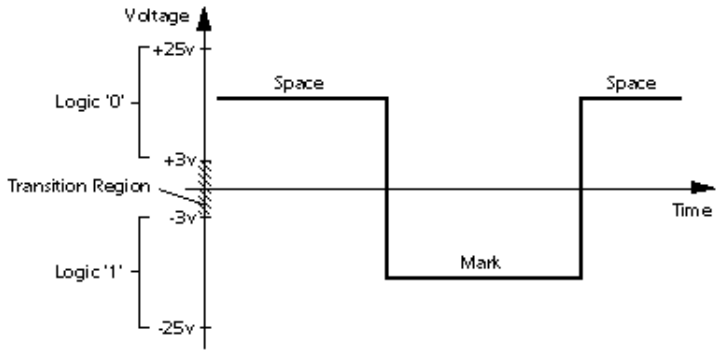
Signal Characteristics
Next Topic | Previous Topic | TOC

Equivalent Circuit - All signal lines, regardless of whether they provide data, timing, or control information, may be represented by the electrical equivalent circuit shown here:



This is the equivalent circuit for an EIA232 signal line and applies to signals originating at either the DTE or DCE side of the connection. "Co" is not specified in the standard, but is assumed to be small and to consist of parasitic elements only. "Ro" and "Vo" are chosen so that the short-circuit current does not exceed 500ma. The cable length is not specified in the standard; acceptable operation is experienced with cables that are less than 25 feet in length.

Signal State Voltage Assignments - Voltages of -3v to -25v with respect to signal ground (pin 7) are considered logic ’1’ (the marking condition), whereas voltages of +3v to +25v are considered logic ’0’ (the spacing condition). The range of voltages between -3v and +3v is considered a transition region for which a signal state is not assigned.



Logic states are assigned to the voltage ranges shown here. Note that this is a "negative logic" convention, which is the reverse of that used in most modern

digital designs.

Most contemporary applications will show an open-circuit signal voltage of -8 to -14 volts for logic ’1’ (mark), and +8 to +14 volts for logic ’0’ (space). Voltage magnitudes will be slightly less when the generator and receiver are connected (when the DTE and DCE devices are connected with a cable).

> *IMPORTANT:* If you insert an LED signal tester in an EIA232 circuit to view signal states, the signal voltage may drop in magnitude to very near the minimum values of -3v for logic ’1’, and +3v for logic ’0’. Also note that some inexpensive EIA232 peripherals are powered directly from the signal lines to avoid using a power supply of their own. Although this usually works without problems, keep the cable short, and be aware that noise immunity will be reduced.

Short-Circuit Tolerance - The generator is designed to withstand an open-circuit (unconnected) condition, or short-circuit condition between its signal conductor and any other signal conductor, including ground, without sustaining damage to itself or causing damage to any associated circuitry. The receiver is also designed to accept any signal voltage within the range of ±25 volts without sustaining damage.

> *CAUTION:* Inductive loads or magnetically induced voltages resulting from long cables may cause the received voltage to exceed the ±25-volt range momentarily during turn-on transients or other abnormal conditions, possibly causing damage to the generator, receiver, or both. Keep the cable length as short as possible, and avoid running the cable near high-current switching loads like electric motors or relays.

Fail-Safe Signals - Four signals are intended to be fail-safe in that during power-off or cable-disconnected conditions, they default to logic ’1’ (negative voltage). They are:

Request to Send - Default condition is deasserted.

Sec. Request to Send - Default condition is deasserted.

DTE Ready - Default condition is DTE not ready.

DCE Ready - Default condition is DCE not ready.
Note specifically that if the cable is connected but the power is off in the generator side, or if the cable is disconnected, there should be adequate bias voltage in the receiver to keep the signal above +3v (logic ’0’) to ensure that the fail-safe requirement is met.

Schmitt triggers or other hysteresis devices may be used to enhance noise immunity in some designs, but should never be adjusted to compromise the fail-safe requirement.

The EIA232 standard is applicable to data rates of up to 20,000 bits per second (the usual upper limit is 19,200 baud). Fixed baud rates are not set by the EIA232 standard. However, the commonly used values are 300, 1200, 2400, 9600, and 19,200 baud. Other accepted values that are not often used are 110 (mechanical teletype machines), 600, and 4800 baud.

Changes in signal state from logic ’1’ to logic ’0’ or vice versa must abide by several requirements, as follows:

1 - Signals that enter the transition region during a change of state must move through the transition region to the opposite signal state without reversing direction or reentering.

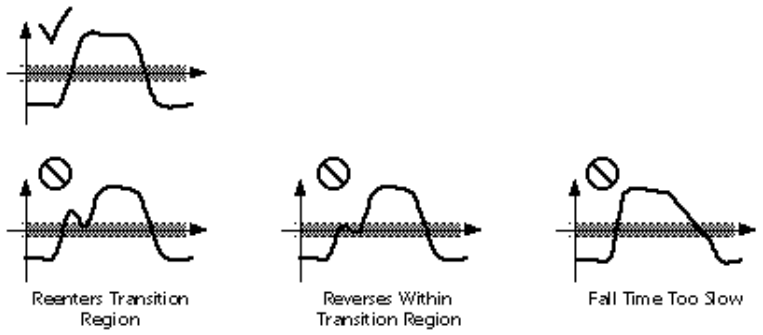
2 - For control signals, the transit time through the transition region should be less than 1ms.

3 - For Data and Timing signals, the transit time through the transition region should be

a - less than 1ms for bit periods greater than 25ms,

b - 4% of the bit period for bit periods between 25ms and 125µs,

c - less than 5µs for bit periods less than 125µs.
The rise and fall times of data and timing signals ideally should be equal, but in any case vary by no more than a factor of three.



An acceptable pulse (top) moves through the transition region quickly and without hesitation or reversal. Defective pulses (bottom) could cause data errors.

---

4 - The slope of the rising and falling edges of a transition should not exceed 30v/µS. Rates higher than this may induce crosstalk in adjacent conductors of a cable.

Note that neither the ASCII alphabet nor the asynchronous serial protocol that defines the start bit, number of data bits, parity bit, and stop bit, is part of the EIA232 specification. For your reference, it is discussed in the Data Communications Basics section of this web site.

Accepted Simplifications of the Standard
Previous Topic | TOC

The EIA232 document published by the Electronic Industries Association describes 14 permissible configurations of the original 22-signal standard. Each configuration uses a subset of the 22 defined signals, and serves a more limited communications requirement than that suggested by using all the available 22-signals. Applications for transmit-only, receive-only, half-duplex operation, and similar variations, are described. Unfortunately, connection to DCE devices other than modems is not considered. Because many current serial interface applications involve direct device-to-device connections, manufacturers do not have a standard reference when producing printers, plotters, print spoolers, or other common peripherals. Consequently, you must acquire the service manual for each peripheral device purchased to determine exactly which signals are utilized in its serial interface.

END

Return to TOC

The RS232 STANDARD

A Tutorial with Signal Names and Definitions

(renamed the "EIA232 Standard" in the early 1990's)

Written by Christopher E. Strangio

Send Us Your Comments . . .

Contents

What is EIA232?
Next Topic | TOC

In the early 1960s, a standards committee, today known as the Electronic Industries Association, developed a common interface standard for data communications equipment. At that time, data communications was thought to mean digital data exchange between a centrally located mainframe computer and a remote computer terminal, or possibly between two terminals without a computer involved. These devices were linked by telephone voice lines, and consequently required a modem at each end for signal translation. While simple in concept, the many opportunities for data error that occur when transmitting data through an analog channel require a relatively complex design. It was thought that a standard was needed first to ensure reliable communication, and second to enable the interconnection of equipment produced by different manufacturers, thereby fostering the benefits of

mass production and competition. From these ideas, the RS232 standard was born. It specified signal voltages, signal timing, signal function, a protocol for information exchange, and mechanical connectors.

Over the 30+ years since this standard was developed, the Electronic Industries Association published three modifications, the most recent being the EIA232E standard introduced in 1991. Besides changing the name from RS232 to EIA232, some signal lines were renamed and various new ones were defined, including a shield conductor.

Likely Problems when Using an EIA232 Interface
Next Topic | Previous Topic | TOC

During this 30-year-long, rapidly evolving period in electronics, manufacturers adopted simplified versions of this interface for applications that were impossible to envision in the 1960s. Today, virtually all contemporary serial interfaces are EIA232-like in their signal voltages, protocols, and connectors, whether or not a modem is involved. Because no single "simplified" standard was agreed upon, however, many slightly different protocols and cables were created that obligingly mate with any EIA232 connector, but are incompatible with each other. Most of the difficulties you will encounter in EIA232 interfacing include at least one of the following:

1 - The absence or misconnection of flow control (handshaking) signals, resulting in buffer overflow or communications lock-up.

2 - Incorrect communications function (DTE versus DCE) for the cable in use, resulting in the reversal of the Transmit and Receive data lines as well as one or more handshaking lines.

3 - Incorrect connector gender or pin configuration, preventing cable connectors from mating properly.

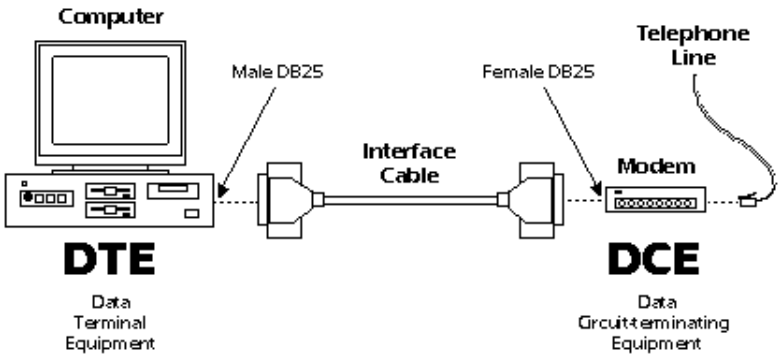
Fortunately, EIA232 driver circuitry is highly tolerant of misconnections, and will usually survive a drive signal being connected to ground, or two drive signals connected to each other. In any case, if the serial interface between two devices is not operating correctly, disconnect the cable joining this equipment until the problem is isolated.

Pin Assignments
Next Topic | Previous Topic | TOC

Go to DTE Pinout (looking into the computer’s serial connector)
Go to DCE Pinout (looking into the modem’s serial connector)

If the full EIA232 standard is implemented as defined, the equipment at the far end of the connection is named the DTE device (Data Terminal Equipment, usually a computer or terminal), has a male DB25

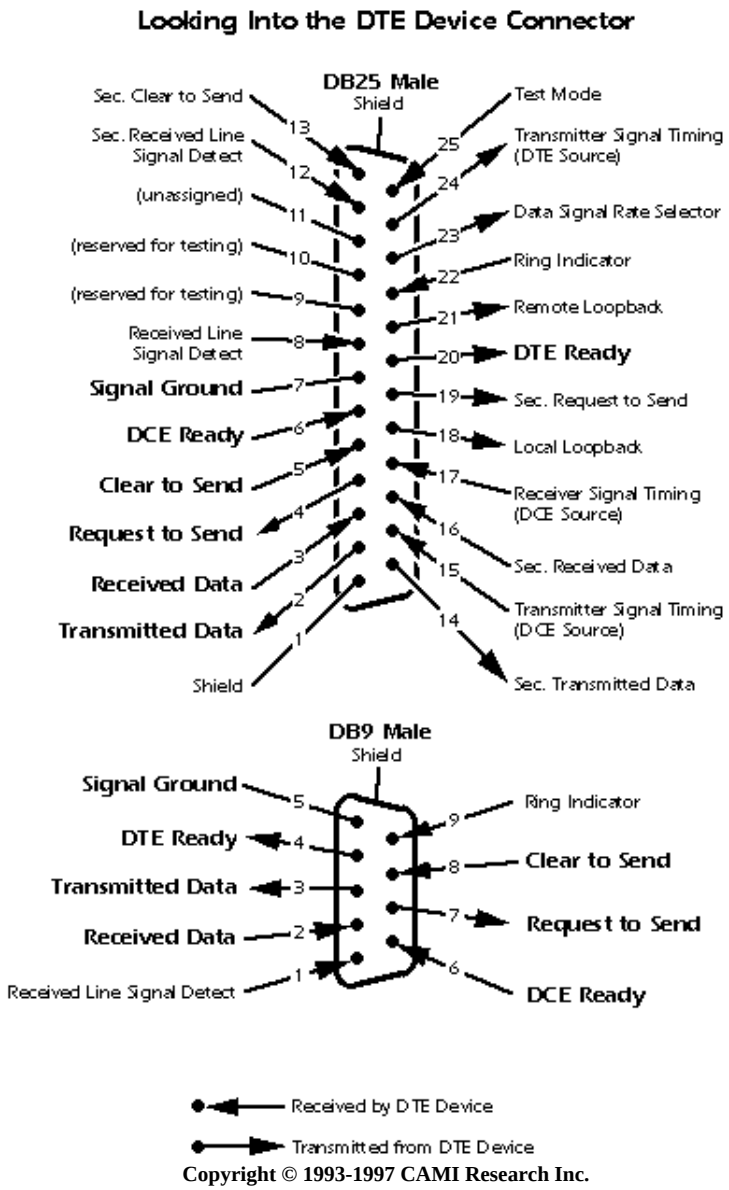
connector, and utilizes 22 of the 25 available pins for signals or ground. Equipment at the near end of the connection (the telephone line interface) is named the DCE device (Data Circuit-terminating Equipment, usually a modem), has a female DB25 connector, and utilizes the same 22 available pins for signals and ground. The cable linking DTE and DCE devices is a parallel straight-through cable with no cross-overs or self-connects in the connector hoods. If all devices exactly followed this standard, all cables would be identical, and there would be no chance that an incorrectly wired cable could be used. This drawing shows the orientation and connector types for DTE and DCE devices:



EIA232 communication function and connector types for a personal computer and modem. DCE devices are sometimes called "Data Communications Equipment" instead of Data Circuit-terminating Equipment.

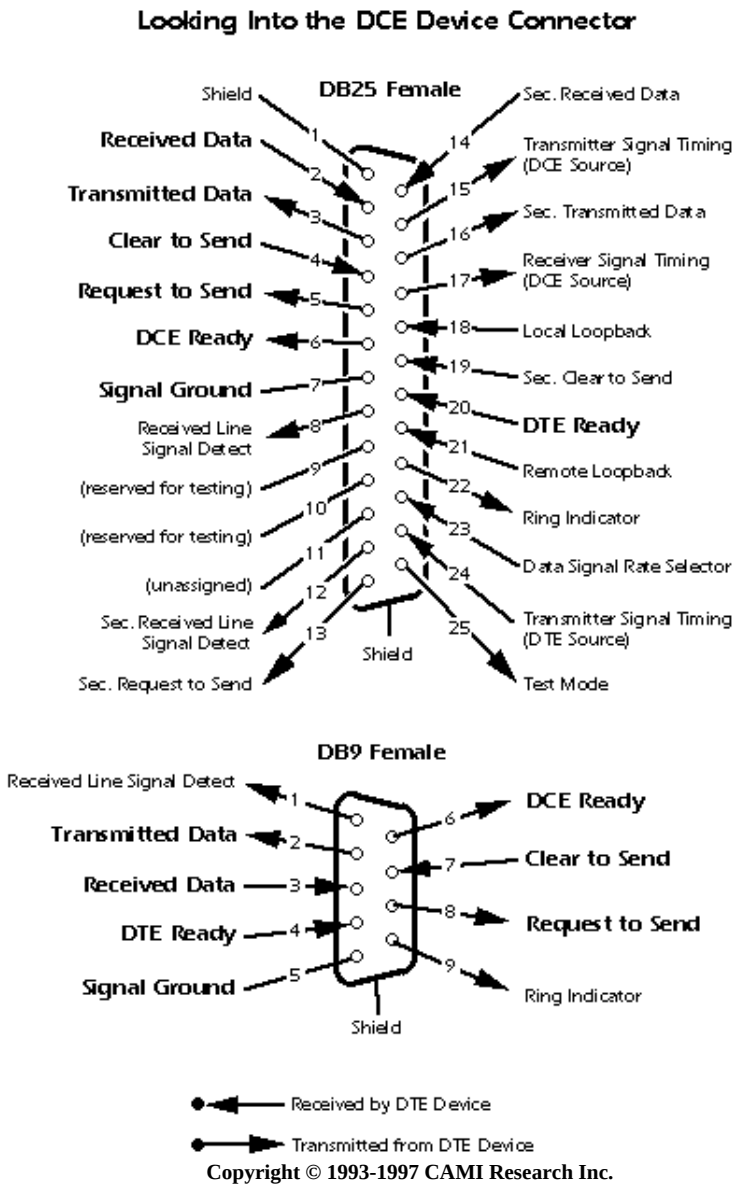
Here is the full EIA232 signal definition for the DTE device (usually the PC). The most commonly used signals are shown in bold.

[back to Pin Assignments description]



This shows the full EIA232 signal definition for the DCE device (usually the modem). The most commonly used signals are shown in bold.

[back to Pin Assignments description]



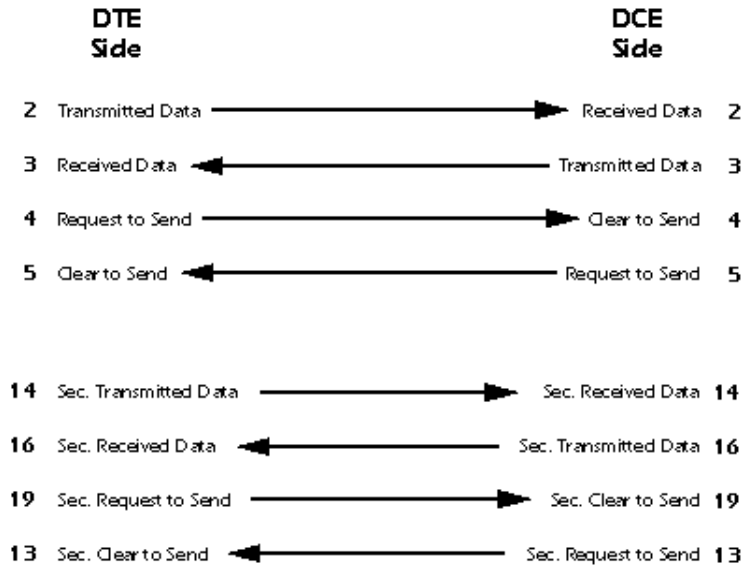
Many of the 22 signal lines in the EIA232 standard pertain to connections where the DCE device is a modem, and then are used only when the software protocol employs them. For any DCE device that is not a modem, or when two DTE devices are directly linked, far fewer signal lines are necessary.

You may have noticed in the pinout drawings that there is a secondary channel which includes a duplicate set of flow-control signals. This secondary channel provides for management of the remote modem, enabling baud rates to be changed on the fly, retransmission to be requested if a parity error is detected, and other control functions. This secondary channel, when used, is typically set to operate at a very low baud rate in comparison with the primary channel to ensure reliability in the control path. In addition, it may operate as either a simplex, half-duplex, or full-duplex channel, depending on the capabilities of the modem.

Transmitter and receiver timing signals (pins 15, 17, and 24) are used only for a synchronous

transmission protocol. For the standard asynchronous 8-bit protocol, external timing signals are unnecessary.

> ***IMPORTANT:*** Signal names that imply a direction, such as Transmit Data and Receive Data, are named from the point of view of the DTE device. If the EIA232 standard were strictly followed, these signals would have the same name for the same pin number on the DCE side as well. Unfortunately, this is not done in practice by most engineers, probably because no one can keep straight which side is DTE and which is DCE. As a result, direction-sensitive signal names are changed at the DCE side to reflect their *drive direction* at DCE. The following list gives the conventional usage of signal names:



Signal Definitions
Next Topic | Previous Topic | TOC

Signal functions in the EIA232 standard can be subdivided into six categories. These categories are summarized below, after which each signal described.

1 - Signal ground and shield.

2 - Primary communications channel. This is used for data interchange, and includes flow control signals.

3 - Secondary communications channel. When implemented, this is used for control of the remote modem, requests for retransmission when errors occur, and governance over the setup of the primary channel.

4 - Modem status and control signals. These signals indicate modem status and provide intermediate checkpoints as the telephone voice channel is established.

5 - Transmitter and receiver timing signals. If a synchronous protocol is used, these signals

provide timing information for the transmitter and receiver, which may operate at different baud rates.

6 - Channel test signals. Before data is exchanged, the channel may be tested for its integrity, and the baud rate automatically adjusted to the maximum rate that the channel can support.

Signal Ground and Shield
Next Topic | Previous Topic | TOC

Pin 7, Pin 1, and the **shell** are included in this category. Cables provide separate paths for each, but internal wiring often connects pin 1 and the cable shell/shield to signal ground on pin 7.

Pin 7 - Ground All signals are referenced to a common ground, as defined by the voltage on pin 7. This conductor may or may not be connected to protective ground inside the DCE device. The existence of a defined ground potential within the cable makes the EIA232 standard different from a balanced differential voltage standard, such as EIA530, which provides far greater noise immunity.

Primary Communications Channel
Next Topic | Previous Topic | TOC

Pin 2 - Transmitted Data (TxD) This signal is active when data is transmitted from the DTE device to the DCE device. When no data is transmitted, the signal is held in the mark condition (logic ’1’, negative voltage).

NOTE: Pin 2 on the DCE device is commonly labeled "Received Data", although by the EIA232 standard it should still be called Transmitted Data because the data is thought to be destined for a remote DTE device.

Pin 3 - Received Data (RxD) This signal is active when the DTE device receives data from the DCE device. When no data is transmitted, the signal is held in the mark condition (logic ’1’, negative voltage).

NOTE: Pin 3 on the DCE device is commonly labeled "Transmitted Data", although by the EIA232 standard it should still be called Received Data because the data is thought to arrive from a remote DTE device.

Pin 4 - Request to Send (RTS) This signal is asserted (logic ’0’, positive voltage) to prepare the DCE device for accepting transmitted data from the DTE device. Such preparation might include enabling the receive circuits, or setting up the channel direction in half-duplex applications. When the DCE is ready, it acknowledges by asserting Clear to Send.

NOTE: Pin 4 on the DCE device is commonly labeled "Clear to Send", although by the

EIA232 standard it should still be called Request to Send because the request is thought to be destined for a remote DTE device.

Pin 5 - Clear to Send (CTS) This signal is asserted (logic ’0’, positive voltage) by the DCE device to inform the DTE device that transmission may begin. RTS and CTS are commonly used as handshaking signals to moderate the flow of data into the DCE device.

NOTE: Pin 5 on the DCE device is commonly labeled "Request to Send", although by the EIA232 standard it should still be called Clear to Send because the signal is thought to originate from a remote DTE device.

Secondary Communications Channel
Next Topic | Previous Topic | TOC

Pin 14 - Secondary Transmitted Data (STxD)

Pin 16 - Secondary Received Data (SRxD)

Pin 19 - Secondary Request to Send (SRTS)

Pin 13 - Secondary Clear to Send (SCTS)

These signals are equivalent to the corresponding signals in the primary communications channel. The baud rate, however, is typically much slower in the secondary channel for increased reliability.

Modem Status and Control Signals
Next Topic | Previous Topic | TOC

Pin 6 - DCE Ready (DSR) When originating from a modem, this signal is asserted (logic ’0’, positive voltage) when the following three conditions are all satisfied:

1 - The modem is connected to an active telephone line that is "off-hook";

2 - The modem is in data mode, not voice or dialing mode; and

3 - The modem has completed dialing or call setup functions and is generating an answer tone.

If the line goes "off-hook", a fault condition is detected, or a voice connection is established, the DCE Ready signal is deasserted (logic ’1’, negative voltage).

IMPORTANT: If DCE Ready originates from a device other than a modem, it may be

asserted to indicate that the device is turned on and ready to function, or it may not be used at all. If unused, DCE Ready should be permanently asserted (logic ’0’, positive voltage) within the DCE device or by use of a self-connect jumper in the cable. Alternatively, the DTE device may be programmed to ignore this signal.

Pin 20 - DTE Ready (DTR) This signal is asserted (logic ’0’, positive voltage) by the DTE device when it wishes to open a communications channel. If the DCE device is a modem, the assertion of DTE Ready prepares the modem to be connected to the telephone circuit, and, once connected, maintains the connection. When DTE Ready is deasserted (logic ’1’, negative voltage), the modem is switched to "on-hook" to terminate the connection.

IMPORTANT: If the DCE device is not a modem, it may require DTE Ready to be asserted before the device can be used, or it may ignore DTE Ready altogether. If the DCE device (for example, a printer) is not responding, confirm that DTE Ready is asserted before you search for other explanations.

Pin 8 - Received Line Signal Detector (CD) (also called carrier detect) This signal is relevant when the DCE device is a modem. It is asserted (logic ’0’, positive voltage) by the modem when the telephone line is "off-hook", a connection has been established, and an answer tone is being received from the remote modem. The signal is deasserted when no answer tone is being received, or when the answer tone is of inadequate quality to meet the local modem’s requirements (perhaps due to a noisy channel).

Pin 12 - Secondary Received Line Signal Detector (SCD) This signal is equivalent to the Received Line Signal Detector (pin 8), but refers to the secondary channel.

Pin 22 - Ring Indicator (RI) This signal is relevant when the DCE device is a modem, and is asserted (logic ’0’, positive voltage) when a ringing signal is being received from the telephone line. The assertion time of this signal will approximately equal the duration of the ring signal, and it will be deasserted between rings or when no ringing is present.

Pin 23 - Data Signal Rate Selector This signal may originate either in the DTE or DCE devices (but not both), and is used to select one of two prearranged baud rates. The asserted condition (logic ’0’, positive voltage) selects the higher baud rate.

Transmitter and Receiver Timing Signals
Next Topic | Previous Topic | TOC

Pin 15 - Transmitter Signal Element Timing (TC) (also called Transmitter Clock) This signal is relevant only when the DCE device is a modem and is operating with a synchronous protocol. The modem generates this clock signal to control exactly the rate at which data is sent on Transmitted Data (pin 2) from the DTE device to the DCE device. The logic ’1’ to logic ’0’ (negative voltage to positive voltage) transition on this line causes a corresponding transition to the next data element on the Transmitted Data line. The modem generates this signal continuously, except when it is performing internal diagnostic functions.

Pin 17 - Receiver Signal Element Timing (RC) (also called Receiver Clock) This signal is similar to TC described above, except that it provides timing information for the DTE receiver.

Pin 24 - Transmitter Signal Element Timing (ETC) (also called External Transmitter Clock) Timing signals are provided by the DTE device for use by a modem. This signal is used only when TC and RC (pins 15 and 17) are not in use. The logic '1' to logic '0' transition (negative voltage to positive voltage) indicates the time-center of the data element. Timing signals will be provided whenever the DTE is turned on, regardless of other signal conditions.

Channel Test Signals

Next Topic | Previous Topic | TOC

Pin 18 - Local Loopback (LL) This signal is generated by the DTE device and is used to place the modem into a test state. When Local Loopback is asserted (logic '0', positive voltage), the modem redirects its modulated output signal, which is normally fed into the telephone line, back into its receive circuitry. This enables data generated by the DTE to be echoed back through the local modem to check the condition of the modem circuitry. The modem asserts its Test Mode signal on Pin 25 to acknowledge that it has been placed in local loopback condition.

Pin 21 - Remote Loopback (RL) This signal is generated by the DTE device and is used to place the remote modem into a test state. When Remote Loopback is asserted (logic '0', positive voltage), the remote modem redirects its received data back to its transmitted data input, thereby remodulating the received data and returning it to its source. When the DTE initiates such a test, transmitted data is passed through the local modem, the telephone line, the remote modem, and back, to exercise the channel and confirm its integrity. The remote modem signals the local modem to assert Test Mode on pin 25 when the remote loopback test is underway.

Pin 25 - Test Mode (TM) This signal is relevant only when the DCE device is a modem. When asserted (logic '0', positive voltage), it indicates that the modem is in a Local Loopback or Remote Loopback condition. Other internal self-test conditions may also cause Test Mode to be asserted, and depend on the modem and the network to which it is attached.

Electrical Standards

Next Topic | Previous Topic | TOC

The EIA232 standard uses negative, bipolar logic in which a negative voltage signal represents logic '1', and positive voltage represents logic '0'. This probably originated with the pre-RS232 current loop standard used in 1950s-vintage teletype machines in which a flowing current (and hence a low voltage) represents logic '1'. Be aware that the negative logic assignment of EIA232 is the reverse of that found in most modern digital circuit designs. See the inside rear cover of the CableEye manual for a comparison.

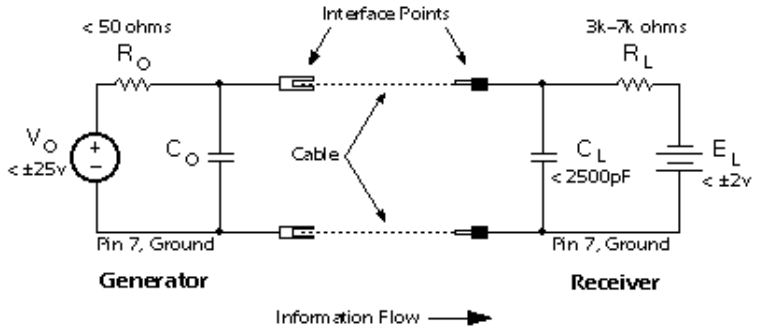
Common Signal Ground
Next Topic | Previous Topic | TOC

The EIA232 standard includes a common ground reference on Pin 7, and is frequently joined to Pin 1 and a circular shield that surrounds all 25 cable conductors. Data, timing, and control signal voltages are measured with respect to this common ground. EIA232 cannot be used in applications where the equipment on opposite ends of the connection must be electrically isolated.

> *NOTE:* optical isolators may be used to achieve ground isolation, however, this option is not mentioned or included in the EIA232 specification.

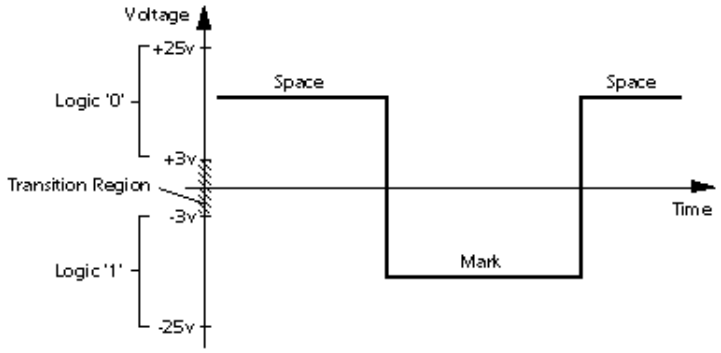
Signal Characteristics
Next Topic | Previous Topic | TOC

Equivalent Circuit - All signal lines, regardless of whether they provide data, timing, or control information, may be represented by the electrical equivalent circuit shown here:



This is the equivalent circuit for an EIA232 signal line and applies to signals originating at either the DTE or DCE side of the connection. "Co" is not specified in the standard, but is assumed to be small and to consist of parasitic elements only. "Ro" and "Vo" are chosen so that the short-circuit current does not exceed 500ma. The cable length is not specified in the standard; acceptable operation is experienced with cables that are less than 25 feet in length.

Signal State Voltage Assignments - Voltages of -3v to -25v with respect to signal ground (pin 7) are considered logic ’1’ (the marking condition), whereas voltages of +3v to +25v are considered logic ’0’ (the spacing condition). The range of voltages between -3v and +3v is considered a transition region for which a signal state is not assigned.



Logic states are assigned to the voltage ranges shown here. Note that this is a "negative logic" convention, which is the reverse of that used in most modern digital designs.

Most contemporary applications will show an open-circuit signal voltage of -8 to -14 volts for logic ’1’ (mark), and +8 to +14 volts for logic ’0’ (space). Voltage magnitudes will be slightly less when the generator and receiver are connected (when the DTE and DCE devices are connected with a cable).

> *IMPORTANT:* If you insert an LED signal tester in an EIA232 circuit to view signal states, the signal voltage may drop in magnitude to very near the minimum values of &shyp;3v for logic ’1’, and +3v for logic ’0’. Also note that some inexpensive EIA232 peripherals are powered directly from the signal lines to avoid using a power supply of their own. Although this usually works without problems, keep the cable short, and be aware that noise immunity will be reduced.

Short-Circuit Tolerance - The generator is designed to withstand an open-circuit (unconnected) condition, or short-circuit condition between its signal conductor and any other signal conductor, including ground, without sustaining damage to itself or causing damage to any associated circuitry. The receiver is also designed to accept any signal voltage within the range of ±25 volts without sustaining damage.

> *CAUTION:* Inductive loads or magnetically induced voltages resulting from long cables may cause the received voltage to exceed the ±25-volt range momentarily during turn-on transients or other abnormal conditions, possibly causing damage to the generator, receiver, or both. Keep the cable length as short as possible, and avoid running the cable near high-current switching loads like electric motors or relays.

Fail-Safe Signals - Four signals are intended to be fail-safe in that during power-off or cable-disconnected conditions, they default to logic ’1’ (negative voltage). They are:

Request to Send - Default condition is deasserted.

Sec. Request to Send - Default condition is deasserted.

DTE Ready - Default condition is DTE not ready.

DCE Ready - Default condition is DCE not ready.
Note specifically that if the cable is connected but the power is off in the generator side, or if the cable is disconnected, there should be adequate bias voltage in the receiver to keep the signal above +3v (logic ’0’) to ensure that the fail-safe requirement is met.

Schmitt triggers or other hysteresis devices may be used to enhance noise immunity in some designs, but should never be adjusted to compromise the fail-safe requirement.

Signal Timing
Next Topic | Previous Topic | TOC

The EIA232 standard is applicable to data rates of up to 20,000 bits per second (the usual upper limit is 19,200 baud). Fixed baud rates are not set by the EIA232 standard. However, the commonly used values are 300, 1200, 2400, 9600, and 19,200 baud. Other accepted values that are not often used are 110 (mechanical teletype machines), 600, and 4800 baud.

Changes in signal state from logic ’1’ to logic ’0’ or vice versa must abide by several requirements, as follows:

1 - Signals that enter the transition region during a change of state must move through the transition region to the opposite signal state without reversing direction or reentering.

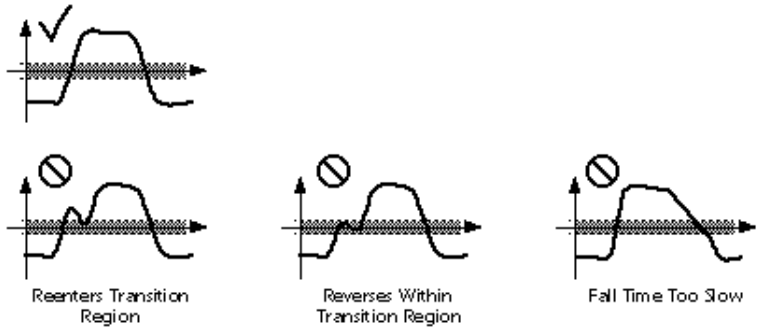
2 - For control signals, the transit time through the transition region should be less than 1ms.

3 - For Data and Timing signals, the transit time through the transition region should be

a - less than 1ms for bit periods greater than 25ms,

b - 4% of the bit period for bit periods between 25ms and 125µs,

c - less than 5µs for bit periods less than 125µs.
The rise and fall times of data and timing signals ideally should be equal, but in any case vary by no more than a factor of three.



An acceptable pulse (top) moves through the transition region quickly and without hesitation or reversal. Defective pulses (bottom) could cause data errors.

4 - The slope of the rising and falling edges of a transition should not exceed 30v/µS. Rates higher than this may induce crosstalk in adjacent conductors of a cable.

Note that neither the ASCII alphabet nor the asynchronous serial protocol that defines the start bit, number of data bits, parity bit, and stop bit, is part of the EIA232 specification. For your reference, it is discussed in the Data Communications Basics section of this web site.

Accepted Simplifications of the Standard
Previous Topic | TOC

The EIA232 document published by the Electronic Industries Association describes 14 permissible configurations of the original 22-signal standard. Each configuration uses a subset of the 22 defined signals, and serves a more limited communications requirement than that suggested by using all the available 22-signals. Applications for transmit-only, receive-only, half-duplex operation, and similar variations, are described. Unfortunately, connection to DCE devices other than modems is not considered. Because many current serial interface applications involve direct device-to-device connections, manufacturers do not have a standard reference when producing printers, plotters, print spoolers, or other common peripherals. Consequently, you must acquire the service manual for each peripheral device purchased to determine exactly which signals are utilized in its serial interface.

END

Return to TOC

SDTP, PPP Serial Data Transport Protocol

Description:

Protocol suite: PPP.
Type: PPP network layer protocol.
PPP protocol: 0x0049
Working groups: pppext, Point-to-Point Protocol Extensions.

Serial Data Transport Protocol (SDTP) is used for synchronous serial data compression over a PPP link.

Before any SDTP packets may be communicated, PPP must reach the Network-Layer Protocol phase, and the SDTP Control Protocol must reach the Opened state.

The maximum length of the SDTP datagram transmitted over a PPP link is limited only by the negotiated Maximum-Frame-Size and the maximum length of the Information field of a PPP encapsulated packet. Note that if compression is used on the PPP link, this the maximum length of the SDTP datagram may be larger or smaller than the maximum length of the Information field of a PPP encapsulated packet, depending on the particular compression algorithm and protocol used.

RFC 1963, pages 1 - 3:

> This document describes a new Network level protocol (from the PPP point of view), PPP Serial Data Transport Protocol, that provides encapsulation and an associated Serial Data Control Protocol (SDCP) for transporting serial data streams over a PPP link. This protocol was developed for the purpose of using PPP's many features to provide a standard method for synchronous data compression. The encapsulation uses a header structure based on that of the ITU-T Recommendation V.120.
>
> This document is a product of the TR30.1 ad hoc committee on compression of synchronous data. It represents a component of a proposal to use PPP to provide compression of synchronous data in DSU/CSUs.
>
> In addition to providing support for multi-protocol datagrams, the Point-to-Point Protocol (PPP) has defined an effective and robust negotiating mechanism that can be used on point to point links. When used in conjunction with the PPP Compression Control Protocol and one of the PPP Compression Protocols, PPP provides an interoperable method of employing data compression on a point-to- point link.
>
> This document provides a PPP encapsulation for serial data, specifying a transport protocol, PPP Serial Data Transport Protocol (PPP-SDTP), and an associated control protocol, PPP Serial Data Control Protocol (PPP-SDCP). When these protocols are added to above mentioned PPP protocols, PPP can be used to provide compression of serial data on a point-to-point link.

This first edition of PPP-SDTP/SDCP covers HDLC-like synchronous serial data and asynchronous serial data. It does this by using a terminal adaption header based on that of ITU-T Recommendation V.120. Support may be added in the future for other synchronous protocols as the marketplace demands.

The V.120 terminal adaption header allows transported data frames to be split over several packets, supports the transport of DTE port idle and error information, and optionally supports the transport of DTE control state information.

In addition to the V.120 Header, fields can be added to the packet format through negotiation to provide support for features not included in the V.120 header. The extra fields are: a Length Field, which is used to distinguish packets in compound frames, and a Port field, which is used to provide multi-port multiplexing capability. The protocol also allows reserved bits in the V.120 header to be used to transport non-octet aligned frames and to provide a flow control mechanism.

To provide these features, PPP-SDTP permits a single frame format to be selected from several possible formats by using PPP-SDCP negotiation. The terminal adaption header can be either fixed length or variable length, to allow either simplicity or flexibility.

The default frame format places the terminal adaption header at the end of the packet. This permits optimal transmitter timelines when user frames are segmented and compression is also used in conjunction with this protocol.

Packet format:

Glossary:

V.120.
CCITT Recommendation V.120 (09/92), "Support by an ISDN of Data Terminal Equipment with V-Series Type Interfaces with Provision for Statistical Multiplexing", 1993.

RFCs:

[RFC 1963] PPP Serial Data Transport Protocol (SDTP).

Disclaimer: This description is completely unofficial. Most of the information presented here is discovered by me, Eugene Crosser, while snooping the serial line and by trial and error. I never had an official protocol description, have never seen any related software source code, and have never done reverse engineering of any related software. This description may be incomplete, inaccurate or completely wrong. You are warned.

Some information is taken from 'camediaplay' package by Jun-ichiro Itoh <itojun@itojun.org>, from the findings of Thierry Bousch <bousch%linotte.uucp@topo.math.u-psud.fr> TsuruZoh Tachibanaya <tsuruzoh@butaman.ne.jp> and from other (open) sources and not checked by me.

Serial Protocol of Some Digital Cameras

Several models of digital cameras, namely Epson, Sanyo, Agfa and Olympus cameras, seem to use the same protocol for communication with the host. Follows the description of the high-level protocol they use over the serial line.

Protocol Basics

The host and the camera exchange with data packets and individual bytes. Serial line paramaters used are: 8bit, no parity. No flow control is used. All arithmetic data is transmitted least significant byte first ("little endian").

Protocol Elements

The elementary units of the protocol are:

Initialization Byte	NUL	0x00
Action Complete Notification	ENQ	0x05
Positive Achnowledgement	ACK	0x06
Unable to Execute Command	DC1	0x11
Negative Acknowledgement, also Camera Signature	NAK	0x15
Packet	Variable length sequence of bytes	
Termination Byte		0xff

Packet structure

The packet has the following structure:

Offset	Length	Meaning
0	1	Packet type
1	1	Packet subtype/sequence
2	2	Length of data
4	variable	Data
-2	2	checksum

Known packet types are:

Type	Description
0x02	Data packet that is not last in sequence
0x03	Data packet that is last in sequence
0x1b	Command packet

Data packets that are sent in responce to a single command are numbered starting from zero. If all requested data fits in one packet, it has type 0x03 and sequence 0.

Command packet has subtype 0x43 or 0x53. Only the first command packet in a session has subtype 0x53.

Maximum length of data field in a packet is 2048 bytes, which yields in 2054 total packet length.

Checksum is a simple 16 bit arithmetic sum of all bytes in the data field. As already mentioned above, length and checksum values are transmitted least significant byte first.

Flow of Control

A communication session flow is as follows:

Host	Camera
Port speed set to 19200 baud	
Host sends init byte 0x00	
	Camera responds with signature 0x15
Host sends command packet with subtype 0x53 and "set speed" command	
	Camera sends ACK 0x06
Port speed set to the new value	
Host sends command	
	Camera responds with either ACK plus optionally "action taken" notifier or data packet sequence
Host sends ACK to every data packet	
... Command - reply cycle repeated ...	
	Camera sends 0xff and resets after a few seconds (value is model-dependant) of inactivity

If the camera does not respond to a command in reasonable time, or responds with a NAK, the command can be resent. If the camera does not provide a complete data packet in reasonable time, or the data packet is corrupt (checksum does not match), the host can request resending of the packet by sending NAK instead of ACK.

Command format and codes

Command is a sequence of bytes sent in the data field of a command packet. Command format is as follows:

Offset	Length	Description
0	1	Command code
1	1	Register number or subcode
2	variable	Optional argument

Five command codes are known:

Code	Argument	Description
0	int32	Set value of integer register
1	none	Read value of integer register
2	vdata	Take action unrelated to registers
3	vdata	Set value of vdata register
4	none	Read value of vdata register

Commands 0 and 3 are replied with a single ACK 0x06. Command 2 is replied with an ACK 0x06 followed by an "action complete" notifier 0x05. Commands 1 and 4 are replied with a sequence of data

packets, each of them must be ACK’ed by the host.

Command 0 must be issued with a 4 byte argument containg the new value for the register (bytes in "LSB first" order). Command 2 typically is issued with a single zero byte as an argument. Command 3 is issued with an argument of variable number of bytes. If this is a printable string, it should **not** include the trailing zero byte.

Camera replies to the command 1 with a single data packet containing 4 bytes of a 32bit integer (in "LSB first" order). Camera replies to the command 4 with a sequence of data packets with variable number of data bytes. Note that if a printable string is returned, it **is** terminated with a zero byte, and thus may be safely printed or otherwise treated as a normal C language character string.

Registers

The following registers are known (read/writablity info is inaccurate):

No.	Type	R/W	Description
1	int32	R/W	Resolution: 1 - Std, 2 - Hi, 3 - Ext, other values possible
2	int32	R/W	Clock in UNIX time_t format
3	int32	R/W	Shutter speed (microseconds), 0 - auto
4	int32	W	Current frame number (or animation number if hi order byte is 0xff)
5	int32	R/W	Aperture: 0 - Auto, 1 - Low, 2 - Med, 3 - ?, 4 - Hi
6	int32	R/W	Color mode: 1 - Color, 2 - B/W
7	int32	R/W	Flash mode: 0 - Auto, 1 - Force, 2 - Off, 3 - Anti RedEye, 4 - Slow sync
8	int32	R/W	Unknown (128)
9	int32	R/W	Unknown (128)
10	int32	R	No. of frames in current folder
11	int32	R	No. of frames left
12	int32	R	Length of current frame *
13	int32	R	Length of current thumbnail *
14	vdata	R	Current frame data *
15	vdata	R	Current thumbnail data *
16	int32	R	Battery capacity percentage
17	int32	R/W	Communication speed 1 - 9600 .. 5 - 115200, 6 - 230400, 256 - 9600 .. 264 - 911600 (sync?)
18	int32	R	Unknown (1)
19	int32	R/W	Bright/Contrast: 0 - Normal, 1 - Contrast+, 2 - Contrast-, 3 - Brightnes+, 4 - Brightnes-
20	int32	R/W	White balance: 0 - Auto, 1 - Sunny, 2 - Incandescent, 3 - Fluorescent, 5 - Flash, 6- White preset, 255 - Cloudy
21	vdata	R/W	Unused

22	vdata	R/W	Camera I.D.
23	int32	R/W	Autoshut on host timer (seconds)
24	int32	R/W	Autoshut in field timer (seconds)
25	vdata	R/W	Serial No. (string)
26	vdata	R	Version
27	vdata	R/W	Model
28	int32	R	Available memory left
29	vdata	R/W	Upload image data to this register
30	int32	W	LED: 0 - Off, 1 - On, 2 - Blink
31	vdata	R	Unknown ("\0")
32	int32	R	Put "magic spell" 0x0FEC000E here before uploading image data
33	int32	R/W	Focus mode: 1 - Macro, 2 - Normal, 3 - Infinity/fisheye
34	int32	R	Operation mode: 1 - Off, 2 - Record, 3-Play, 6-Thumbnail
35	int32	R/W	LCD brightness 1 to 7
36	int32	R	Unknown (3)
37	vdata	R	Unknown ("\0")
38	int32	R	LCD autoshut timer (seconds)
39	int32	R	Protection state of current frame *
40	int32	R	True No. of frames taken
41	int32	R/W	LCD date format: 1 - ’YY MM DD, 2 - DD MM ’HH
42	vdata	R	Unknown ("")
43	vdata	R	Audio data description block * 0: expanded .wav length 1: compressed .wav length 3: Unknown (0) 4: Unknown (0) 5: Unknown (0) 6: Unknown (0) 7: Unknown (0)
44	vdata	R	Audio data *
45	vdata	R	Unknown ("")
46	vdata	R	Camera summary data: 32 bytes with copies of 8 other registers 0: Reg 1 (Resolution) 1: Reg 35 (LCD brightness) or Reg 7 (Flash mode) 2: Reg 10 (Frames taken) or Unknown 3: Unknown (0) or Unknown 4: Unknown (0) or Reg 16 (Battery capacity) 5: Unknown (0) or Reg 10 (Frames taken) 6: Unknown (0) or Reg 11 (Frames left) 7: Number of animations taken

47	vdata	R	Picture summary data: 32 bytes or 8 int32's * 0: Hi order byte: unknown, next 3 bytes: Length of current image 1: Length of current thumbnail 2: Audio data length (expanded) 3: Resolution 4: Protection state 5: TimeDate 6: Unknown (0) 7: Animation type: 1 - 10ms, 2 - 20ms
48	vdata	R	Manufacturer
49	vdata	R	Unknown ("")
50	int32	R	Unknown (0)
51	int32	R/W	Card detected: 1 - No, 2 - Yes
52	vdata	R/W	Unknown ("")
53	int32	R/W	Language: 3 - english, 4 - french, 5 - german, 6 - italian, 8 - spanish, 10 - dutch
54-59	vdata	R	Unknown ("")
60	int32	R	True No. of frames taken
61-68	vdata	R	Unknown ("")
69	vdata	R	Exposure Compensation 8 bytes 0: compensation value -20 to +20 1: 0 2: 0 3: 0 4: 10 5: 0 6: 0 7: 0
70	int32	R/W	Exp. meter: 2 - Center weighted, 3 - Spot, 5 - Multi element matrix
71	vdata	R/W	Effective zoom in tenths of millimeters: 8 bytes 0: LSB 1: MSB 2: 0 3: 0 4: 10 5: 0 6: 0 7: 0
72	int32	R/W	Bitmap: 1 - AEL/WBL, 2 - Fisheye, 4 - Wide, 8 - Manual zoom, 16 - B/W, 256 - 1.25x, 512 - 1.6x, 768 - 2.0x, 1024 - 2.5x, 1280 - off
73-76	vdata	R	Unknown ("")
77	int32	W	Size of data packet from camera (default 0x800)
78	vdata	R	Unknown ("")

79	vdata	R	Filename of current frame *
80-81	vdata	R	Unknown ("")
82	int32	W	Unknown (enable folder features? Write 60 here)
83	int32	R/W	Folder navigation When read, return number of folders on the card. When written **without** data, reset folder system (?) Or select current folder by its number
84	vdata	R/W	Current folder name (may read or set)
85	vdata	R	Unknown ("")
86	int32	R/W	Digital zoom; 0 - 1X, otherwise zoom factor x 100 (i.e. in percent)
87-90	vdata	R	Unknown ("")
91	vdata	R	Current folder I.D. and name

*** Note:** Marked registers only become useful for reading after setting register 4. If value of 0 assigned to register 4 after doing action 5, subsequent retrieval of picture data gives the "live preview".

For command 2, the second byte is action code not register number. The following action codes are known:

Code	Argument	Description
0	single zero byte	Erase last picure
1	single zero byte	Erase all picures (but not animations)
2	single zero byte	Take picture
4	single zero byte	Finish session immediately
5	single zero byte	Take preview snapshot (retreivable as frame zero)
6	single byte	Calibration / testing. Arg value: 1 Calibrate autofocus 3 Test zoom/exposure 4-6 Store 0 in Reg 32 9 Load LCD Brightness (0-31) from Reg 32 10 Load LCD size (25 for Nikon Coolpix 950) from Reg 32 11 LCD Saturation (0-32) from Reg 32 13 LCD Red-Green (0-32) from Reg 32 14 LCD Blue (0-32) from Reg 32 15 Store -1 in Reg 32 16 Multi shot (locks up if lcd is on) 17 Take picture 18 Store -1 in Reg 32 20-23 locks up if lcd is on 24-255 Store -1 in Reg 32
7	single zero byte	Erase current frame *

8	single byte	Switch LCD mode. Arg value: 1 - Off 2 - Record 3 - Play (show current frame fullscreen) 4 - preview thumbnails (?) 5 - Thumbnail view (smaller?) 6 - Thumbnail view (larger?) 7 - Next 8 - Previous
9	single byte	Set protection state of current frame to the value of parameter (binary 0 or 1)*
11	single zero byte	Store freshly uploaded image into NVRAM (see appendix A)
12	single byte	LCD test. Arg value: 0 - white 1 - gray 2 - black 3 - red 4 - green 5 - blue 6 - test pattern

*** Note:** actions 7 and 9 only useful after setting register 0x04.

Appendix A

```
Date: Sun, 14 Jul 2002 01:28:39 +0200 (CEST)
From: =?iso-8859-1?Q?Peter_=C5strand?= <astrand(at)lysator.liu.se>
To: allyn(at)fratkin.com, <wolfgang(at)charlotte.wsrcc.com>, <crosser(at)average.org
Subject: Upload on Olympus C-860L


FYI.

Tonight, I've been struggling with uploading arbitrary pictures to my
Olympus C-860L. I've finally found out that for the camera to accept the
picture, to two conditions must be met:

1) The subsampling must be 2x1, 1x1, 1x1
2) The EXIF info must be just like the pictures the camera itself
produces.

So, I've made a small script to fix this. Feel free to include it in FAQs
and/or photopc dists.

/more/data/pics/olympus-reference-pic.jpg is just some picture taken with
the camera.

photopc-upload-all:

#!/bin/sh

TMPFILE='mktemp /tmp/photopc-upload.XXXXXX' || exit 1
```

```
for file in $@; do
    echo Converting $file...
    djpeg $file | cjpeg -sample 2x1 > $TMPFILE
    jhead -te /more/data/pics/olympus-reference-pic.jpg $TMPFILE
    echo Uploading $file...
    photopc upload $TMPFILE
    sleep 2;
done

rm -f $TMPFILE


--
/Peter Åstrand <astrand(at)lysator.liu.se>
```

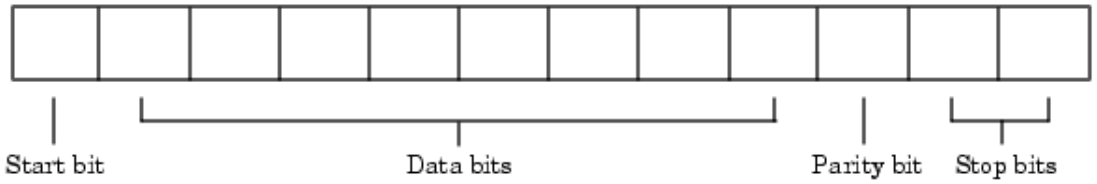
Appendix B

Some Nikon models support an extension to the protocol described above, specifically designed for remote control units. This protocol allows to control zoom, emulate half-depress of the shutter release button, bulb operation and possibly more. <*vladimir.vyskocil(at)wanadoo.fr*> compiled a partial description of this protocol, available here.

Please mail your corrections/additions to <*crosser at average dot org*>
See http://photopc.sourceforge.net/ for possible updates.

Serial Data Format

The serial data format includes one start bit, between five and eight data bits, and one stop bit. A parity bit and an additional stop bit might be included in the format as well. The diagram below illustrates the serial data format.



The format for serial port data is often expressed using the following notation

- number of data bits - parity type - number of stop bits

For example, 8-N-1 is interpreted as eight data bits, no parity bit, and one stop bit, while 7-E-2 is interpreted as seven data bits, even parity, and two stop bits.

The data bits are often referred to as a *character* because these bits usually represent an ASCII character. The remaining bits are called *framing bits* because they frame the data bits.

Bytes Versus Values

The collection of bits that comprise the serial data format is called a *byte*. At first, this term might seem inaccurate because a byte is 8 bits and the serial data format can range between 7 bits and 12 bits. However, when serial data is stored on your computer, the framing bits are stripped away, and only the data bits are retained. Moreover, eight data bits are always used regardless of the number of data bits specified for transmission, with the unused bits assigned a value of 0.

When reading or writing data, you might need to specify a *value*, which can consist of one or more bytes. For example, if you read one value from a device using the `int32` format, then that value consists of four bytes. For more information about reading and writing values, refer to Writing and Reading Data.

Synchronous and Asynchronous Communication

The RS-232 standard supports two types of communication protocols: synchronous and asynchronous.

Using the synchronous protocol, all transmitted bits are synchronized to a common clock signal. The two devices initially synchronize themselves to each other, and then continually send characters to stay synchronized. Even when actual data is not really being sent, a constant flow of bits allows each device to know where the other is at any given time. That is, each bit that is sent is either actual data or an idle character. Synchronous communications allows faster data transfer rates than asynchronous methods, because additional bits to mark the beginning and end of each data byte are not required.

Using the asynchronous protocol, each device uses its own internal clock resulting in bytes that are transferred at arbitrary times. So, instead of using time as a way to synchronize the bits, the data format is used.

In particular, the data transmission is synchronized using the start bit of the word, while one or more stop bits indicate the end of the word. The requirement to send these additional bits causes asynchronous communications to be slightly slower than synchronous. However, it has the advantage that the processor does not have to deal with the additional idle characters. Most serial ports operate asynchronously.

> **Note** When used in this guide, the terms "synchronous" and "asynchronous" refer to whether read or write operations block access to the MATLAB command line. Refer to Controlling Access to the MATLAB Command Line for more information.

How Are the Bits Transmitted?

By definition, serial data is transmitted one bit at a time. The order in which the bits are transmitted is given below:

1. The start bit is transmitted with a value of 0.
2. The data bits are transmitted. The first data bit corresponds to the least significant bit (LSB), while the last data bit corresponds to the most significant bit (MSB).
3. The parity bit (if defined) is transmitted.
4. One or two stop bits are transmitted, each with a value of 1.

The number of bits transferred per second is given by the *baud rate*. The transferred bits include the start bit, the data bits, the parity bit (if defined), and the stop bits.

Start and Stop Bits

As described in Synchronous and Asynchronous Communication, most serial ports operate asynchronously. This means that the transmitted byte must be identified by start and stop bits. The start bit indicates when the data byte is about to begin and the stop bit(s) indicates when the data byte has been transferred. The process of identifying bytes with the serial data format follows these steps:

1. When a serial port pin is idle (not transmitting data), then it is in an "on" state.
2. When data is about to be transmitted, the serial port pin switches to an "off" state due to the start bit.
3. The serial port pin switches back to an "on" state due to the stop bit(s). This indicates the end of the byte.

Data Bits

The data bits transferred through a serial port might represent device commands, sensor readings, error messages, and so on. The data can be transferred as either binary data or ASCII data.

Most serial ports use between five and eight data bits. Binary data is typically transmitted as eight bits. Text-based data is transmitted as either seven bits or eight bits. If the data is based on the ASCII character set, then a minimum of seven bits is required because there are 2^7 or 128 distinct characters. If an eighth bit is used, it must have a value of 0. If the data is based on the extended ASCII character set, then eight bits must be used because there are 2^8 or 256 distinct characters.

The Parity Bit

The parity bit provides simple error (parity) checking for the transmitted data. The types of parity checking are given below.

Table 9-2: Parity Types

Parity Type	Description
Even	The data bits plus the parity bit result in an even number of 1's.
Mark	The parity bit is always 1.
Odd	The data bits plus the parity bit result in an odd number of 1's.
Space	The parity bit is always 0.

Mark and space parity checking are seldom used because they offer minimal error detection. You might choose to not use parity checking at all.

The parity checking process follows these steps:

1. The transmitting device sets the parity bit to 0 or to 1 depending on the data bit values and the type of parity checking selected.
2. The receiving device checks if the parity bit is consistent with the transmitted data. If it is, then the data bits are accepted. If it is not, then an error is returned.

> **Note** Parity checking can detect only 1-bit errors. Multiple-bit errors can appear as valid data.

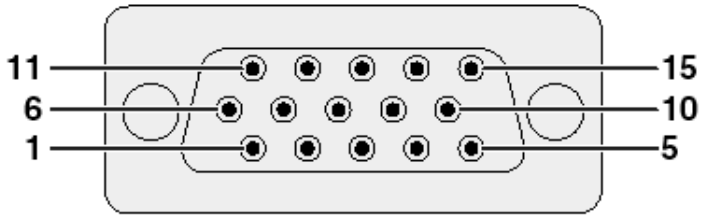
For example, suppose the data bits 01110001 are transmitted to your computer. If even parity is selected, then the parity bit is set to 0 by the transmitting device to produce an even number of 1's. If odd parity is selected, then the parity bit is set to 1 by the transmitting device to produce an odd number of 1's.

Serial Port Signals and Pin Assignments Finding Serial Port Information for Your Platform

Connectors

RGB Signal Input Port:

15-pin Mini D-sub female connector



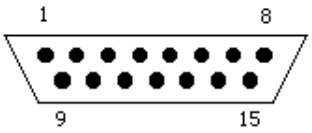
RGB Input

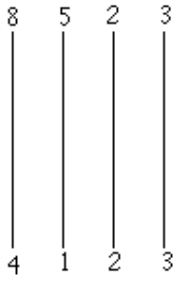
Analog

1. Video input (red)
2. Video input (green/sync on green)
3. Video input (blue)
4. Not connected
5. Composite sync
6. Earth (red)
7. Earth (green/sync on green)
8. Earth (blue)
9. Not connected
10. GND
11. GND
12. Bi-directional data
13. Horizontal sync signal
14. Vertical sync signal
15. Data clock

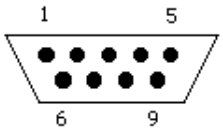
Computer - Pioneer DVD9300 player cable

DVD player DB-15 Male





Computer DB-9 female

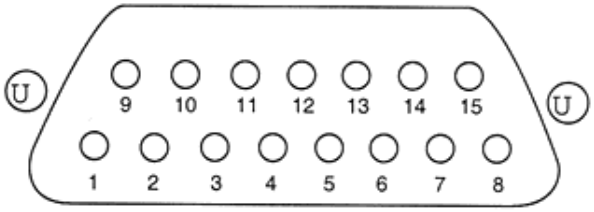


Pioneer DVD 7300 (7400)

2.1 Interface Connector

A computer may be connected to the DVD-V7400 using a 15-pin D-Sub connector (e.g., a JAE DALC-J15SAF connector with suitable plug such as the JAE DA-15PF-N) to the RS-232C serial port or to the parallel port.

The pins are identified below:

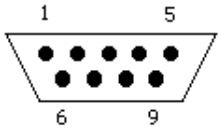


2.2 Serial Interface Pin Specification

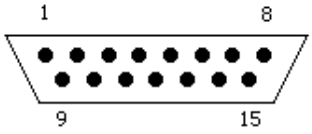
Pin #	Terminal	Input/Output	Function
1	GND	--	ground
2	TxD	Output	send data
3	RxD	Input	receive data
4	DTR	Output	enable data receiving
5	POWER	Output	external power control
6	SW1	Input	
7	SW2	Input	
8	SW3	Input	
9	SW4	Input	
10	SW5	Input	
11	SW6	Input	
12	SW7	Input	
13	SW8	Input	
14	DLTST	Input	used only for servicing the unit – do not connect
15	V +8V	Output	used only for servicing the unit – do not connect

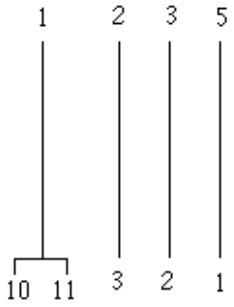
Serial controller - DVD player cable

Computer DB-9 male



DVD player DB-15 Male

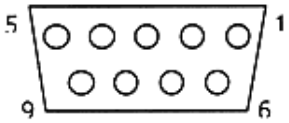




HT200/HT250

RS-232C Control Port:

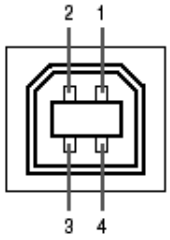
D-SUB 9-pin
(female)



Pin No	Signal	Definition
1	N/A	Not used
2	TD	Transmit data
3	RD	Receive data
4	N/A	Not used
5	GND	Ground
6	N/A	Not used
7	N/A	Not used
8	N/A	Not used
9	N/A	Not used

Parameter	Value
Transfer Rate	19200 b.p.s.
Data Length	8 bits
Parity Bit	None
Stop Bit	1 bit
Flow Control	None

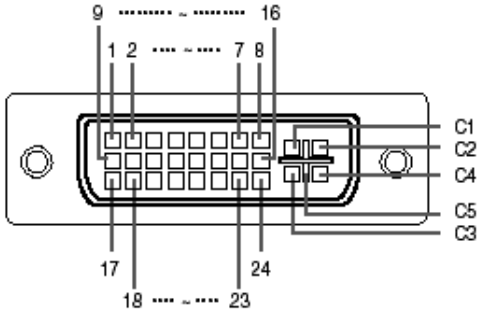
4-pin USB connector



- **USB connector: 4 pin B-type USB connector**

Pin no.	Signal	Name
1	VCC	USB power
2	USB–	USB data–
3	USB+	USB data+
4	SG	Signal Ground

DVI Digital / Analog INPUT 1 port : 29 pin connector



- **DVI Digital INPUT**

Pin No.	Signal	Pin No.	Signal
1	T.M.D.S data 2–	16	Hot plug detection
2	T.M.D.S data 2+	17	T.M.D.S data 0–
3	T.M.D.S data 2 shield	18	T.M.D.S data 0+
4	Not connected	19	T.M.D.S data 0 shield
5	Not connected	20	Not connected
6	DDC clock	21	Not connected
7	DDC data	22	T.M.D.S clock shield
8	Not connected	23	T.M.D.S clock+
9	T.M.D.S data 1–	24	T.M.D.S clock–
10	T.M.D.S data 1+	C1	Not connected
11	T.M.D.S data 1 shield	C2	Not connected
12	Not connected	C3	Not connected
13	Not connected	C4	Not connected
14	+5V power	C5	Ground
15	Ground		

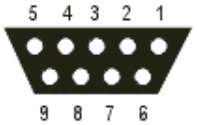
- **DVI Analog RGB Input**

Pin No.	Signal	Pin No.	Signal
1	Not connected	16	Hot plug detection
2	Not connected	17	Not connected
3	Not connected	18	Not connected
4	Not connected	19	Not connected
5	Not connected	20	Not connected
6	DDC clock	21	Not connected
7	DDC data	22	Not connected
8	Vertical sync	23	Not connected
9	Not connected	24	Not connected
10	Not connected	C1	Analog input Red
11	Not connected	C2	Analog input Green
12	Not connected	C3	Analog input Blue
13	Not connected	C4	Horizontal sync
14	+5V power	C5	Ground
15	Ground		

- **DVI Analog Component Input**

Pin No.	Signal	Pin No.	Signal
1	Not connected	16	Not connected
2	Not connected	17	Not connected
3	Not connected	18	Not connected
4	Not connected	19	Not connected
5	Not connected	20	Not connected
6	Not connected	21	Not connected
7	Not connected	22	Not connected
8	Not connected	23	Not connected
9	Not connected	24	Not connected
10	Not connected	C1	Analog input Pr/Cr
11	Not connected	C2	Analog input Y
12	Not connected	C3	Analog input Pb/Cb
13	Not connected	C4	Not connected
14	Not connected	C5	Ground
15	Ground		

Serial mouse



Pin	Description	
1	DCD	Data carried detect
2	RD	Receive data
3	TD	Transmit data
4	DTR	Data terminal ready
5	SG	Signal ground
6	DSR	Data set ready
7	RTS	Request to send
8	CTS	Clear to send
9	Ring	

Image Dimensions in Common Usage

Collated by Paul Bourke
May 2000

Dimensions	Ratio	Comments
8x8	1	
16x16	1	Macintosh cursor size Supported by Windows icon format
32x32	1	Macintosh icon size Supported by Windows icon format
64x64	1	Supported by Windows icon format
88x31	2.8387	WWW micro banner
128x96	1 1/3	
160x120	1 1/3	NTSC 13" 1/16th CuSeeMe small image size Considered a "small" QuickTime movie
160x144	1.1111	
176x144	1.2222	
180x132	1.3636	
180x135	1 1/3	
192x144	1 1/3	PAL 1/16
234x60	3.9	WWW small banner
256x192	1 1/3	10"/12" 1/4
320x200	1.6	Called CGA, IBM PS/2
320x240	1 1/3	NTSC 13" 1/4 CuSeeMe image size Considered a "large" QuickTime movie
320x288	1.1111	
320x400	0.8	Amigo

352x288	1.2222	PAL Video CD
352x240	1.46666	NTSC Video CD
384x256	1.5	Photo CD
384x288	1 1/3	PAL 1/4
392x72	5.444444	WWW banner
400x300	1 1/3	
460x55	8.36363636	WWW banner
468x32	14.625	WWW banner
468x60	7.8	WWW banner
512x342	1.497	Original Macintosh screen
512x384	1 1/3	
544x372	1.4623	WebTV image size
640x350	1.82857	Called EGA, IBM
640x480	1 1/3	NTSC full Called PGA, IBM VGA
640x576	1.1111	
704x576	1.22222	Full image size from ASUS video capture with TNT2
720x350	2.05714286	Called MDA, IBM / VESA
720x400	1.8	Called MCGA, IBM/VESA
720x480	1.5	Format 480p. NTSC video format as used by DPS PVR video hardware. NTSC DV SDTV
720x483	1.49068323	SDTV
720x484	1.48760331	Alternative Media-100 NTSC, eg: Media-100
720x486	40/27	CCIR 601 NTSC
720x540	1 1/3	CCIR 601 NTSC Sq
720x576	1.25	CCIR 601 DV PAL and DV SECAM

729x348	2.094828	Hercules graphics
768x576 Subtract 75 from left/right and 55 top/bottom for the PAL "safe" area	1 1/3	CCIR 601 PAL full
800x600	1 1/3	Called SVGA resolution. Used in the first generation of LCD projectors (their native resolution).
832x624	1 1/3	Macintosh
856x480	1.78333333	
896x600	1.49333333	
960x720	1 1/3	Non standard digital TV format supported by some suppliers.
1024x768	1 1/3	XGA. Native resolution of many LCD projectors. Used by VisionStation and VisionStation 3.
1080x720	1.5	HDTV
1152x768	1.5	
1152x864	1 1/3	VESA
1152x870	1.3241	Macintosh
1152x900	1.28	Sun / SGI
1280x720	16/9	Format 720p. HDTV WXGA
1280x800	1.6	
1280x854	1.498829	Mac G4 15" laptop
1280x960	1 1/3	SXVGA
1280x992	1.29	Optimised on the PowerStorm 350 OpenGL card/drivers from Compaq
1280x1024	1.25	SXGA
1360x766	1.77545692	

1365x768	16:9 (nearly, 1.77734375)	NEC 61" plasma
1365x1024	1 1/3	VisionStation 3 with upgrade and VisionStation 5.
1400x1050	1.3671875	DELL Laptop
1440x900	1.6	Apple 17" G4 laptop
1520x856	1.77570093	
1600x900	16:9	
1600x1024	1.5625	
1600x1200	1 1/3	VESA UXGA
1792x1120	1.6	
1792x1344	1 1/3	
1824x1128	1.61702128	
1824x1368	1 1/3	
1856x1392	1 1/3	
1920x1080	16/9	1080i format. HDTV, known as 1K
1920x1200	1.6	
1920x1440	1 1/3	
2000x1280	1.5625	QXGA
2048x1152	16:9	
2048x1536	1 1/3	Feature film, known as 2K Also used by DOME display controllers
2048x2048	1	Tiger high resolution display
2500x1340	1.86567	
3072x2252	1.3641	Sometimes used for IMAX (?)
3600x2613	1.37772675	Sometimes used for IMAX (?)
4096x3072	1 1/3	Image size for IMAX 3D rendering, known as 4K
4096x3840	1.0666666	NCSA tiled wall (2001)

Serial Mouse Data Formats

The Microsoft Serial Mouse format is the defacto standard for serial mice. The Microsoft mouse format allows for only two buttons. Three button mice working in Microsoft mode ignore the middle button.

The data packets are sent at 1200 baud with 1 stop bit and no parity. Each packet consists of 3 bytes. It is sent to the computer every time the mouse changes state (ie. the mouse is moved or the buttons are pressed/released).

	D6	D5	D4	D3	D2	D1	D0
1st byte	1	LB	RB	Y7	Y6	X7	X6
2nd byte	0	X5	X4	X3	X2	X1	X0
3rd byte	0	Y5	Y4	Y3	Y2	Y1	Y0

```
LB is the state of the left  button, 1 = pressed, 0 = released.
RB is the state of the right button, 1 = pressed, 0 = released
X0-7 is movement of the mouse in the X direction since the
     last packet.  Positive movement is toward the right.
Y0-7 is movement of the mouse in the Y direction since the
     last packet.  Positive movement is back, toward the user.
```

The mouse driver software collects the X and Y movement bits from the different bytes in the packet. All moves are sent as two’s complement binary numbers.

Although the Microsoft format only requires 7 data bits per byte, most mice actually send 8-bit data with the most significant bit set to 1. Since the most-significant-bit (D7) is last in the serial data stream, this is the same as sending two stop bits instead of one. The Joymouse sends data packets as shown below.

	D7	D6	D5	D4	D3	D2	D1	D0
1st byte	1	1	LB	RB	Y7	Y6	X7	X6
2nd byte	1	0	X5	X4	X3	X2	X1	X0
3rd byte	1	0	Y5	Y4	Y3	Y2	Y1	Y0

[Back to Data & Documentation] [Home]

Next: Motorola 68HC11 SCI Interface **Up:** Serial Communication **Previous:** Asynchronous Serial Communication (SCI)

RS-232 Serial Protocol

The RS-232 serial communication protocol is a standard protocol used in asynchronous serial communication. It is the primary protocol used over modem lines. It is the protocol used by the MicroStamp11 when it communicates with a host PC.

Figure 23 shows the relationship between the various components in a serial ink. These components are the *UART*, the serial channel, and the interface logic. An interface chip known as the **universal asynchronous receiver/transmitter** or **UART** is used to implement serial data transmission. The UART sits between the host computer and the *serial channel*. The serial channel is the collection of wires over which the bits are transmitted. The output from the UART is a standard TTL/CMOS logic level of 0 or 5 volts. In order to improve bandwidth, remove noise, and increase range, this TTL logical level is converted to an RS-232 logic level of -12 or $+12$ volts before being sent out on the serial channel. This conversion is done by the interface logic shown in figure 23. In your system the interface logic is implemented by the comm stamp.

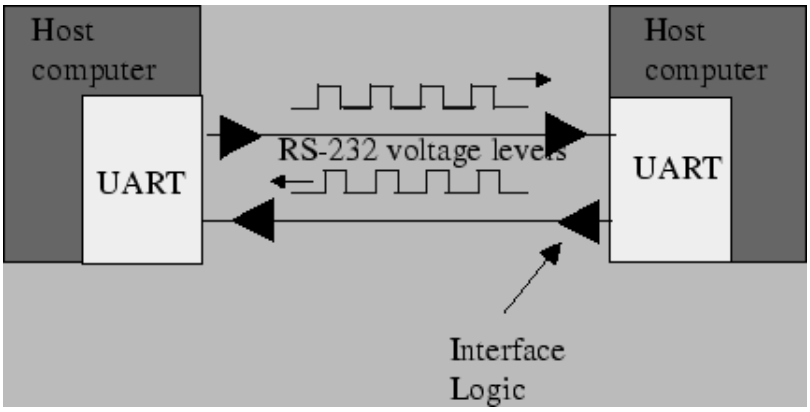


Figure 23: Asynchronous (RS-232) serial link

A **frame** is a complete and nondivisible packet of bits. A frame includes both *information* (e.g., data and characters) and *overhead* (e.g., start bit, error checking and stop bits). In asynchronous serial protocols such as RS-232, the frame consists of one start bit, seven or eight data bits, parity bits, and stop bits. A timing diagram for an RS-232 frame consisting of one start bit, 7 data bits, one parity bits and two stop bits is shown below in figure 24. Note that the exact structure of the frame must be agreed upon by both transmitter and receiver before the comm-link must be opened.

Figure 24: RS-232 Frame (1 start bit, 7 data bits, 1 parity bits,

and 2 stop bits)

Most of the bits in a frame are self-explanatory. The start bit is used to signal the beginning of a frame and the stop bit is used to signal the end of a frame. The only bit that probably needs a bit of explanation is the *parity* bit. Parity is used to detect transmission errors. For *even parity* checking, the number of 1's in the data plus the parity bit must equal an even number. For *odd parity*, this sum must be an odd number. Parity bits are used to detect errors in transmitted data. Before sending out a frame, the transmitter sets the parity bit so that the frame has either even or odd parity. The receiver and transmitter have already agreed upon which type of parity check (even or odd) is being used. When the frame is received, then the receiver checks the parity of the received frame. If the parity is wrong, then the receiver knows an error occurred in transmission and the receiver can request that the transmitter re-send the frame.

In cases where the probability of error is extremely small, then it is customary to ignore the parity bit. For communication between the MicroStamp11 and the host computer, this is usually the case and so we ignore the parity bit.

The *bit time* is the basic unit of time used in serial communication. It is the time between each bit. The transmitter outputs a bit, waits one bit time and then outputs the next bit. The start bit is used to synchronize the transmitter and receiver. After the receiver senses the true-false transition in the start bit, it waits one half bit time and then starts reading the serial line once every bit time after that. The *baud rate* is the total number of bits (information, overhead, and idle) per time that is transmitted over the serial link. So we can compute the baud rate as the reciprocal of the bit time.

Next: Motorola 68HC11 SCI Interface **Up:** Serial Communication **Previous:** Asynchronous Serial Communication (SCI)
Bill Goodwine 2002-09-29


Next: LCD display for the **Up:** Serial Communication **Previous:** RS-232 Serial Protocol

Motorola 68HC11 SCI Interface

The Motorola 68HC11 supports one SCI. We'll discuss both transmitting and receiving ends of the SCI. The programmer controls the operation of the SCI interface through a set of hardware registers that are memory mapped into the processor's address space. There are 5 control registers shown below in figure 25. This figure also shows the logical names for individual bits in the registers. The `BAUD` register is used to set the serial link's baud rate. There are two control registers `SCCR1` and `SCCR2` that specify how the SCI should work. There is a status register, `SCSR` that the programmer can use to check whether the transmission/reception of a frame has been completed. Finally there is the data register `SCDR` that holds the transmitted or received information bits.

BAUD

TCLR	0	SCP1	SCP0	RCKB	SCR2	SCR1	SCR0

SCCR1

R8	T8	0	M	WAK	0	0	0

SCCR2

TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK

SCSR

TDRE	TC	RDRF	IDLE	OR	NF	FE	0

SCDR

R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0

Figure 25: 68HC11 SCI Registers

From the number of control bits in the `SCCR1` and `SCCR2` registers you can see that the programmer has quite a bit of control over the SCI interface. Most of the situations we'll be using, however, have standard set-ups so we won't need to discuss the control register bits in detail. Instead, we'll provide standard functions that encapsulate the user's interface to SCI devices such as a personal computers and simply discuss how these functions work. The proper setup of the SCI subsystem is usually done in your program's `init()` initialization routine.

To understand how the SCI subsystem works, let's examine figure 26. Figure 26 shows how the programmer interacts with the SCI transmit and receive buffers. In order to transmit, the programmer first loads the 8-bit data register `SCDR` with the data to be sent. The SCI module automatically fills in the start bit, stop bit, and the extra `T8` bit from control register `SCCR1`. The `T8` bit can be used as a parity bit.

Once the SCDR register is loaded, the subsystem loads this data into the SCI module's transmit buffer. The SCI transmit buffer then holds a single frame and this frame is then clocked out of the transmit register one bit at a time at the rate specified in the BAUD register. Once all of the bits have been clocked out of the transmit buffer, the SCI module sets the TDRE (transmit data register empty) bit in the SCI's status register SCSR. This single bit can then be used to check whether or not the frame has been successfully transmitted.

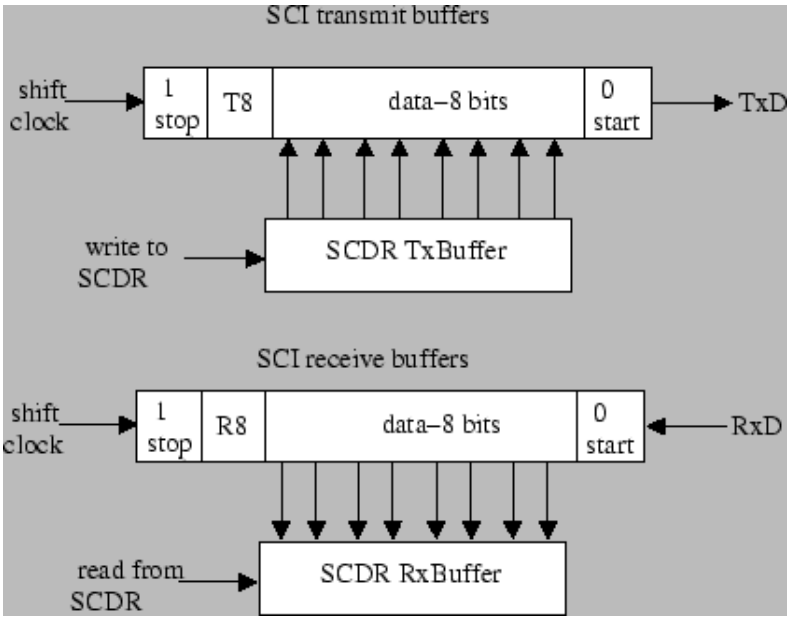


Figure 26: SCI transmit and receive buffers

A similar set of steps can be used to check and see if the receive data register has been filled. Once initialized, the receive data component of the SCI subsystem will wait for the true-to-false transition on the input line signalling a start bit. After the start bit has been detected, the receive subsystem will shift in 10 or 11 bits into the receive data register. The start and stop bits are removed and 8 bits of data are loaded into the SCI data register SCDR. The ninth parity bit R8 is put in the SCCR1 control register. When the receive data register is full, then the SCI subsystem sets the RDRF (receive data register full) flag in the status register SCSR. The programmer can then check this status flag to see if a full frame has been received.

The following code segment from an init() function can be used to initialize the SCI module to transmit and receive at 38 kbaud. This setup was used in our earlier kernel.c functions to send characters back and forth between the MicroStamp11 and the PC.

```
void init(void){
  asm(" sei");
    CONFIG = 0x04;
    BAUD=BAUD38K;
    SCCR1 = 0x00;
    SCCR2 = 0x0C;
  asm(" cli");
}
```

The instructions in this function do the following. The first instruction CONFIG = 0x04 turns off the

micro-controller's watchdog timer. The second instruction `BAUD=BAUD38K` sets the SCI subsystem's baud rate to 38 kilo-baud. The variable `BAUD38K` is a logical name whose actual value will be found in `hc11.h`. The next two lines set up the SCI subsystem's parameters. By zeroing `SCCR1`, we are ignoring the parity bit and creating a 10-bit frame. By setting `SCCR2=0x0C`, we've disabled all transmit and receive interrupts and we've enabled the transmit and receive modules in the SCI subsystem.

Note that the SCI module has hardware interrupts associated with the hardware events

- Transmission Complete (TC): which is set when the transmit shift register is empty.
- Transmit Data Register Empty (TDRE): which is set when the transmit data register is empty.
- Receive Data Register Full (RDRF): which is set when the receive data register is full.
- Idle line (ILIE): which is when the receive module detects an idle line.

These hardware interrupts can be used to do parity-bit processing on a transmitted or received frame. In our particular examples, however, we assume no parity checking so these interrupts have been disabled.

As specific examples of how the SCI interface can be used, we consider the two function `InChar()` and `OutChar()`. These functions are used to receive and transmit, respectively, a single byte (frame) of data.

```
void OutChar(char data){
  while((SCSR & TDRE) == 0);
  SCDR=data;
}

void InChar(void){
  while((SCSR & RDRF) == 0);
  return(SCDR);
}
```

The first function, `OutChar()` transmits a single byte. The function simply waits in a loop until the `TDRE` bit (transmit data register empty) is set. Once this is done, we know that any previously loaded bytes have been successfully shifted out of the transmit data register. The function then reloads the SCI's data register `SCDR` with the new data. The other function `InChar()` receives a single byte. The function waits in a while loop until the `RDRF` bit (receive data register full) is set, thereby indicating that 8-bits of have been shifted into the receive data register. Once this is done, the function returns a pointer to the `SCDR` data register.

Next: LCD display for the **Up:** Serial Communication **Previous:** RS-232 Serial Protocol
Bill Goodwine 2002-09-29

Serial Data Format

The serial data that we are interested in sending to and from the terminal is byte-wide ASCII data. ASCII is a standard code for sending alphanumeric data and is actually only 7-bits wide. The 8th bit is often used to indicate the parity of the 7-bit data word and used for error detection. For our circuit, the high-order bit will always be 0, but you should always send it anyway. So, each packet that is sent will consist of 8 bits, 7-bits of ASCII and one 0 in the high-order bit. A table of the ASCII code is shown in Figure 1.

$B_3B_2B_1B_0$	$B_6B_5B_4$ 000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	‘	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	”	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	’	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	—
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

Figure 1: ASCII Character Codes

Once we agree to send ASCII, we should also agree on how that data should be send as a serial data stream. The protocol we will use defines that the bits in the byte are passed least-significant bit first in a serial stream. So, send the bits one at a time, starting with the least-significant. How long each bit is asserted in this serial stream depends on how fast your baud rate is. At 9600 baud, for example, each bit will be asserted for 1/9600 of a second, or about 104 μ sec. Now the problem is how to decide when to look at the data wire in order to see the bit. The DCE and DTE will not be synchronized to a common clock, so in order to decide when to look at the data line to see new data being passed, they must synchronize with each new byte that is being passed. This is why the UART is ‘‘asynchronous’’ in operation. The data is being passed at a known frequency, but the starting time of each new byte is unknown. So, the receiving circuit must resynchronize at the start of each new byte.

In practice this is quite easy. You only need some sort of protocol that tells you when to expect new data. For our system (and most asynchronous serial protocols in general) the data line must be held in a 1 state (+5v in our case) until a byte is ready to be passed. When a byte is to be sent, the data line drops to 0 (gnd in our case) for one bit time to signal that a byte will follow. This is the *start bit* and its purpose is to wake up the receiver and alert it to the byte that is about to be sent. The 8 data bits then follow, least significant bit first, each asserted for one bit time (which depends on the baud rate). Finally, one or two *stop bits* are sent to indicate the end of the byte. Stop bits are 1-bits that are asserted for one bit time each. In our system the receiver will assume that only one stop bit is sent, and the sender will send two stop bits. This is the most general and safest solution. For example, if the receiver expects a single stop bit and two are sent, nothing bad happens except that some extra time elapses between that byte and the next due to the extra stop bit. On the other hand, if the sender sends only a single stop bit but the external receiver expects two, a mistake might be made. So, it's safer to expect only a single stop bit, but always send two (note that in practice, most terminals and other communication devices have settings to control how many stop bits are sent or expected.) A picture of this data protocol is shown in Figure 2.

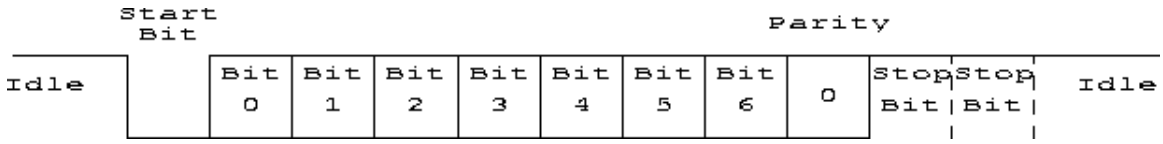


Figure 2: Data Byte Transmission Format

Erik Brunvand
Tue Apr 11 15:50:32 MDT 2000