Serial Interfaces

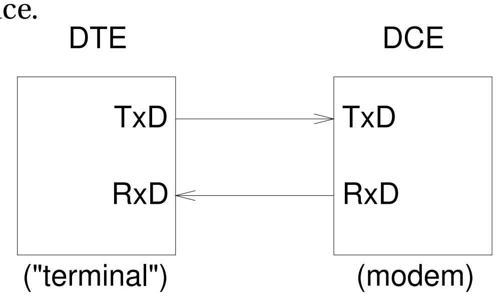
Serial interfaces are typically used to connect computer systems and low-speed external peripherals such as modems and printers. Serial interfaces reduce the cost of interconnecting devices because the bits in a byte are time-multiplexed onto a single conductor. Serial interfaces are covered in Chapter 9 of the text.
You should be able to describe the operation and format of data and handshaking signals on an RS-232 interface and the data format on an HDLC synchronous data stream.

RS-232 Interface

The most widely used peripheral interface is the "RS-232" serial interface. This interface is available on most general-purpose microcomputers.

DTE and DCE

The serial interface was originally designed to connect modems (Data Communications Equipment - DCE) to computer terminals (Data Terminal Equipment - DTE). In its simplest form the interface has two signal lines, Transmit Data (TxD or TD) and Receive Data (RxD or RD), and a ground reference. The TxD signal is an output on a DTE device and an input on a DCE device. Similarly, RxD is an output on a DCE device and input on a DTE device.

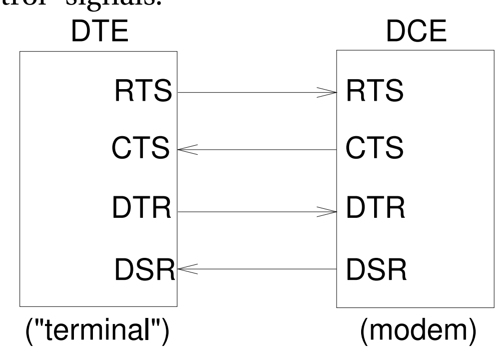


Exercise: Is the "Transmit Data" (TxD) signal an input or an output? How about "Receive Data" (RxD)? Is a computer a 'modem' or a 'terminal'?

The standard RS-232 connector is a 25-pin D-style connector called a "DB-25". Pin 2 is TxD, pin 3 is RxD and pin 7 is signal ground. When two serial devices are connected together they are connected pin-to-pin (RxD is connected to RxD and TxD is connected to TxD). This means that RxD must be an input on one device and an output on the other device. Thus the terms RxD and TxD *do not* say whether a pin is an input or output but are instead names for pins on the connector. Typically DTE connectors are male and DCE connectors are female.

In addition to the two data lines, most RS-232 devices implement additional handshaking pins. Of these, the most useful are called RTS (Request To Send) and CTS (Clear To Send). The CTS pin is a DCE output and is used by the DCE to indicate that it can accept data on the TxD line. The RTS line is an output on a DTE and is used to indicate that the DTE wants to send (RTS was originally used to control half-duplex modems – these are rarely seen today).

Since these signals are used to control the flow of data from the DTE (and optionally from the DCE) these pins are called [hardware] "flow control" signals.



The second set of control signals are DTR (Data Terminal Ready) and DSR (Data Set Ready). These signals indicate that the DTE and DCE devices respectively are connected and operational (typically, simply that the power is turned on). Some modems can use DTR to force a reset and DSR as a replacement for CD (see below).

In the original RS-232 specification there was no provision for hardware flow control of data from the DCE to the DTE. However many modems change the semantics of RTS so that it is used to flow control data from the modem to the terminal ("bidirectional RTS/CTS flow control") .

Exercise: Which data line (TxD or RxD) would RTS flow control in this case?

A number of other handshaking signals are available but are less widely used. Carrier Detect (CD) is asserted by modem-type DCEs when a carrier signal is present. This signal is typically used by system software to indicate the start and end of a dial-up session. Another signal pin is Ring Indicator (RI) which is used by modem-type DCEs to indicates that the attached phone is ringing. This signal is seldom used. The RS-232 specification defines a number of other signals (e.g. a secondary serial interface) but they are almost never used.

In addition to the standard DB-25 serial connector, there are a number of smaller connectors that are widely used. These connectors are physically smaller and carry a subset of the RS-232 pins. The most common are the DB-9 connectors popular on IBM PC-AT clones, the round DIN connectors (popular on Apple computers), and the inexpensive telephone-style "RJ-11" (6-pin) and "RJ-45" (8-pin) connectors (popular on devices with many serial interfaces).

Adapters are often used not only to convert between different styles of connectors but also to convert between male and female connectors (a "gender adapter" which allows two males or two females to be connected together) and to switch between DCE and DTE pinouts (a "null modem" which allows two DCEs or two DTEs to be connected together).

Interface Voltages

The serial interface voltage levels are bipolar with respect to ground. The table below summarizes the relationship between voltage level, logical meaning on handshaking lines and data bit value (TxD and RxD lines).

Signal Level	Logic Value	Line State	Data Value
negative	false	mark	1
positive	true	space	0

The received signal must be greater than +3 volts to be considered positive and less than -3 volts for negative. Intermediate values are considered invalid. This allows disconnected pins to be detected.

Note: *The data lines (TxD and RxD) are asserted when* **negative**. *The control lines (e.g. CTS) are asserted when* **postitive**.

Character Format

Data is transferred over the serial interface one bit at a time. A **positive** (zero) bit (the "start bit") is sent to indicate the start of the character being sent. This is followed by the bits in the character, from LS to MS bit. After sending the 7 (for plain ASCII) or 8 (for arbitrary bytes) bits, an optional parity bit (even or odd) can be sent, followed by a one "stop" bit.

Exercise: Draw the waveform used to send the ASCII character 'e' (hex 65) at 9600 bps with no parity.

The start bit allows a receiver to re-synchronize itself at the start of each character. This allows for small variations between transmitter and receiver timing.

Exercise: What happens if the receiver's clock is running faster than the transmitter clock?

The stop bit guarantees that there will be a transition at the start of each character. It also allows a receiver to re-synchronize to a character boundary in the middle of a continuous data stream. If the receiver does not see a 'one' stop bit (called a "framing error") it knows it is unsynchronized and treats that bit as a start bit. Eventually the receiver will synchronize to an actual start bit.

Exercise: What would happen if the receiver was expecting 8-bit characters and the transmitter was sending 7-bit characters? What about the reverse case?

There are a number of standard bit rates, typically powers of two times 1200 bps (1200, 2400, 4800 bps etc). The RS-232 standard specifies maximum bit rates, distances, etc but these are usually ignored in practical applications. For short distances it's possible to send in excess of 100 kbps.

The RS-422 serial interface specification uses a similar signaling scheme but uses differential signals (opposite voltages on two signal lines) to increase immunity to noise and increase maximum transmission distance. Data rates up to 1 Mbps are common.

Synchronous Interfaces

One problem with asynchronous interfaces is the 2-bit penalty imposed by the start and stop bits. In a synchronous interface the modem (DCE) supplies a clock signal output along with the data. The data source (DTE) must supply a clock along with data. The timing accuracy requirements for synchronous devices are much stricter.

Synchronous serial interfaces are often embedded in links between peripheral devices rather than between the computer and an external device. For example all modern modems use synchronous signaling between themselves.

A problem with synchronous interfaces is that some means must be provided to find the start of the desired data within a continuous bit stream. The HDLC (High-level Data Link Control) protocol, often used over synchronous bit stream channels, uses a procedure called bit stuffing. HDLC uses "flag" sequences of a zero, 6 consecutive '1' bits and another zero at the start and end of each frame to divide the data stream into frames. In order to prevent sequences of 6 '1' bits in the data from being misinterpreted as terminating a frame, a zero bit is inserted immediately after into any data sequence of 5 '1' bits. At the receiver a '0' received after 5 '1's is removed. The HDLC frame includes one-byte address and control fields at the start of the frame and a 16-bit check-sum (a CRC, actually) at the end.

Exercise: What is the probability of finding 6 consecutive '1' bits in a random data stream?

Serial Interface Chips (UARTs)

Except at the lowest speeds it is not possible for a microcomputer to generate the serial bit stream using software. A peripheral chip called a Universal Asynchronous Receiver Transmitter (UART) is used to do the conversion from bytes to serial bits and vice-versa. UARTs designed as microcomputer peripherals have control registers that are used to configure the chip for various baud rates, word lengths, number of stop bits, FIFO sizes, and interrupt options. UARTs also have status registers that can be used to indicate whether a character has been received or not, whether the previous transmitted character has been sent, and any error conditions.

Since the CPU may not be able to retrieve a received character as soon as it is read, most UARTS have FIFO (first in first out) buffers to store one or more received characters while other characters are being received.

Special chips are used to convert the logic voltages used by the UARTs to/from the RS-232 signal levels.

There are also chips (USARTs) that perform the same functions as UARTs but can also handle synchronous data streams including the HDLC framing, bit stuffing and error checking.

Serial Peripheral Interfaces

A third type of serial interface is used to connect peripheral chips such as A/D converters, serial EEPROMs, FPGAs or other microprocessors to a microprocessor. The purpose of these serial interfaces is to reduce the pin count. These interfaces are often found on microcontrollers and DSP microprocessors and include both a data line and a clock line. The clock can be driven by the processor (the CPU is called the "master") or by the source of the data (the CPU is the "slave").

Exercise: What type of sequential logic device(s) would be used to implement this type of serial interface?