of idle line 1s, which are typically transmitted when there is no information. Idle line 1s are interpreted by a receiver as continuous stop bits (i.e., no data). If a noise impulse occurs that causes the receive data to go low at the same time the receiver clock is active, the receiver will interpret the noise impulse as a start bit. If this happens, the receiver will misinterpret the logic condition present during the next clock as the first data bit \( b_0 \) and the following clock cycles as the remaining data bits \( b_1, b_2, \text{ and so on} \). The likelihood of misinterpreting noise hits as start bits can be reduced substantially by clocking the UART receiver at a rate higher than the incoming data. Figure 17b shows the same situation as shown in Figure 17a, except the receive clock pulse (RCP) is 16 times \( (16 \times) \) higher than the receive serial data input (RSI). Once a low is detected, the UART waits seven clock cycles before resampling the input data. Waiting seven clock cycles places the next sample very near the center of the start bit. If the next sample detects a low, it assumes that a valid start bit has been detected. If the data have reverted to the high condition, it is assumed that the high-to-low transition was simply a noise pulse and, therefore, is ignored. Once a valid start bit has been detected and verified (Figure 17c), the start bit verification circuit samples the incoming data once every 16 clock cycles, which essentially makes the sample rate equal to the receive data rate (i.e., \( 16 \text{ RCP}/16 = \text{ RCP} \)). The UART continues sampling the data once every 16 clock cycles until the stop bits are detected, at which time the start bit verification circuit begins searching for another valid start bit. UARTs are generally programmed for receive clock rates of 16, 32, or 64 times the receive data rate (i.e., \( 16 \times, 32 \times, \text{ and } 64 \times \)).

Another advantage of clocking a UART receiver at a rate higher than the actual receive data is to ensure that a high-to-low transition (valid start bit) is detected as soon as possible. This ensures that once the start bit is detected, subsequent samples will occur very near the center of each data bit. The difference in time between when a sample is taken (i.e., when a data bit is clocked into the receive shift register) and the actual center of a data bit is called the sampling error. Figure 18 shows a receive data stream sampled at a rate 16 times higher (16 RCP) than the actual data rate (RCP). As the figure shows, the start bit is not immediately detected. The difference in time between the beginning of a start bit and when it is detected is called the detection error. The maximum detection error is equal to the time of one receive clock cycle \( (t_{cl} = 1/R_{cl}) \). If the receive clock rate equaled the receive data rate, the maximum detection error would approach the time of one bit, which would mean that a start bit would not be detected until the very end of the bit time. Obviously, the higher the receive clock rate, the earlier a start bit would be detected.

**FIGURE 18** 16X receive clock rate
Because of the detection error, successive samples occur slightly off from the center of the data bit. This would not present a problem with synchronous clocks, as the sampling error would remain constant from one sample to the next. However, with asynchronous clocks, the magnitude of the sampling error for each successive sample would increase (the clock would slip over or slip under the data), eventually causing a data bit to be either sampled twice or not sampled at all, depending on whether the receive clock is higher or lower than the transmit clock.

Figure 19 illustrates how sampling at a higher rate reduces the sampling error. Figures 19a and b show data sampled at a rate eight times the data rate ($8\times$) and 16 times the data rate ($16\times$), respectively. It can be seen that increasing the sample rate moves the sample time closer to the center of the data bit, thus decreasing the sampling error.

Placing stop bits at the end of each data character also helps reduce the clock slip page (sometimes called clock skew) problem inherent when using asynchronous transmit and receive clocks. Start and stop bits force a high-to-low transition at the beginning of each character, which essentially allows the receiver to resynchronize to the start bit at the beginning of each data character. It should probably be mentioned that with UARTs the data rates do not have to be the same in each direction of propagation (e.g., you could transmit data at 1200 bps and receive at 600 bps). However, the rate at which data leave a transmitter must be the same as the rate at which data enter the receiver at the other end of the circuit. If you transmit at 1200 bps, it must be received at the other end at 1200 bps.

10-2 Universal Synchronous Receiver/Transmitter

A universal synchronous receiver/transmitter (USRT) is used for synchronous transmission of data between a DTE and a DCE. Synchronous data transmission means that a synchronous data format is used, and clocking information is generally transferred between the DTE and the DCE. A USRT performs the same basic functions as a UART, except for

![Diagram of sampling error with 8X RCP](a)

![Diagram of sampling error with 16X RCP](b)

**FIGURE 19** Sampling error: (a) 8X RCP; (b) 16X RCP
synchronous data (i.e., the start and stop bits are omitted and replaced by unique synchronizing characters). The primary functions performed by a USRT are the following:

1. Serial-to-parallel and parallel-to-serial data conversions
2. Error detection by inserting parity bits in the transmitter and checking parity bits in the receiver.
3. Insert and detect unique data synchronization (SYN) characters
4. Formatting data in the transmitter and receiver (i.e., combining items 1 through 3 in a meaningful sequence)
5. Provide transmit and receive status information to the CPU
6. Voltage-level conversion between the DTE and the serial interface and vice versa
7. Provide a means of achieving bit and character synchronization

11 SERIAL INTERFACES

To ensure an orderly flow of data between a DTE and a DCE, a standard serial interface is used to interconnect them. The serial interface coordinates the flow of data, control signals, and timing information between the DTE and the DCE.

Before serial interfaces were standardized, every company that manufactured data communications equipment used a different interface configuration. More specifically, the cable arrangement between the DTE and the DCE, the type and size of the connectors, and the voltage levels varied considerably from vendor to vendor. To interconnect equipment manufactured by different companies, special level converters, cables, and connectors had to be designed, constructed, and implemented for each application. A serial interface standard should provide the following:

1. A specific range of voltages for transmit and receive signal levels
2. Limitations for the electrical parameters of the transmission line, including source and load impedance, cable capacitance, and other electrical characteristics outlined later in this chapter
3. Standard cable and cable connectors
4. Functional description of each signal on the interface

In 1962, the Electronics Industries Association (EIA), in an effort to standardize interface equipment between data terminal equipment and data communications equipment, agreed on a set of standards called the RS-232 specifications (RS meaning “recommended standard”). The official name of the RS-232 interface is Interface Between Data Terminal Equipment and Data Communications Equipment Employing Serial Binary Data Interchange. In 1969, the third revision, RS-232C, was published and remained the industrial standard until 1987, when the RS-232D was introduced, which was followed by the RS-232E in the early 1990s. The RS-232D standard is sometimes referred to as the EIA-232 standard. Versions D and E of the RS-232 standard changed some of the pin designations. For example, data set ready was changed to DCE ready, and data terminal ready was changed to DTE ready.

The RS-232 specifications identify the mechanical, electrical, functional, and procedural descriptions for the interface between DTEs and DCEs. The RS-232 interface is similar to the combined ITU-T standards V.28 (electrical specifications) and V.24 (functional description) and is designed for serial transmission up to 20 kbps over a maximum distance of 50 feet (approximately 15 meters).

11-1 RS-232 Serial Interface Standard

The mechanical specification for the RS-232 interface specifies a cable with two connectors. The standard RS-232 cable is a sheath containing 25 wires with a DB25P-compatible
male connector (plug) on one end and a DB25S-compatible female connector (receptacle) on the other end. The DB25P-compatible and DB25S-compatible connectors are shown in Figures 20a and b, respectively. The cable must have a plug on one end that connects to the DTE and a receptacle on the other end that connects to the DCE. There is also a special PC nine-pin version of the RS-232 interface cable with a DB9P-compatible male connector on one end and a DB9S-compatible connector at the other end. The DB9P-compatible and DB9S-compatible connectors are shown in Figures 20c and d, respectively (note that there is no correlation between the pin assignments for the two connectors). The nine-pin version of the RS-232 interface is designed for transporting asynchronous data between a DTE and a DCE or between two DTEs, whereas the 25-pin version is designed for transporting either synchronous or asynchronous data between a DTE and a DCE. Figure 21 shows the eight-pin EIA-561 modular connector, which is used for transporting asynchronous data between a DTE and a DCE when the DCE is connected directly to a standard two-wire telephone line attached to the public switched telephone network. The EIA-561 modular connector is designed exclusively for dial-up telephone connections.

Although the RS-232 interface is simply a cable and two connectors, the standard also specifies limitations on the voltage levels that the DTE and DCE can output onto or receive from the cable. The DTE and DCE must provide circuits that convert their internal logic levels to RS-232-compatible values. For example, a DTE using TTL logic interfaced to a DCE using CMOS logic is not compatible. *Voltage-leveling circuits* convert the internal voltage levels from the DTE and DCE to RS-232 values. If both the DCE and the DTE output and accept RS-232 levels, they are electrically compatible regardless of which logic family they use internally. A voltage leveler is called a *driver* if it outputs signals onto the cable and a...
terminator if it accepts signals from the cable. In essence, a driver is a transmitter, and a terminator is a receiver. Table 7 lists the voltage limits for RS-232-compatible drivers and terminators. Note that the data and control lines use non-return to zero, level (NRZ-L) bipolar encoding. However, the data lines use negative logic, while the control lines use positive logic.

From examining Table 7, it can be seen that the voltage limits for a driver are more inclusive than the voltage limits for a terminator. The output voltage range for a driver is between $-5 \, \text{V}$ and $15 \, \text{V}$ or between $-5 \, \text{V}$ and $-15 \, \text{V}$, depending on the logic level. However, the voltage range in which a terminator will accept is between $3 \, \text{V}$ and $25 \, \text{V}$ or between $-3 \, \text{V}$ and $-25 \, \text{V}$. Voltages between $\pm 3 \, \text{V}$ are undefined and may be interpreted by a terminator as a high or a low. The difference in the voltage levels between the driver output and the terminator input is called noise margin (NM). The noise margin reduces the susceptibility to interface caused by noise transients induced into the cable. Figure 22a shows the relationship between the driver and terminator voltage ranges. As shown in Figure 22a, the noise margin for the minimum driver output voltage is $2 \, \text{V}$ ($5 - 3$), and the noise margin for the maximum driver output voltage is $10 \, \text{V}$ ($25 - 15$). (The minimum noise margin of $2 \, \text{V}$ is called the implied noise margin.) Noise margins will vary, of course, depending on what specific voltages are used for highs and lows. When the noise margin of a circuit is a high value, it is said to have high noise immunity, and when the noise margin is a low value, it has low noise immunity. Typical RS-232 voltage levels are $+10 \, \text{V}$ for a high and $-10 \, \text{V}$ for a low, which produces a noise margin of $7 \, \text{V}$ in one direction and $15 \, \text{V}$ in the other direction. The noise margin is generally stated as the minimum value. This relationship is shown in Figure 22b. Figure 22c illustrates the immunity of the RS-232 interface to noise signals for logic levels of $+10 \, \text{V}$ and $-10 \, \text{V}$.

The RS-232 interface specifies single-end (unbalanced) operation with a common ground between the DTE and DCE. A common ground is reasonable when a short cable is used. However, with longer cables and when the DTE and DCE are powered from different electrical buses, this may not be true.

**Example 8**

Determine the noise margins for an RS-232 interface with driver signal voltages of $\pm 6 \, \text{V}$.

**Solution**

The noise margin is the difference between the driver signal voltage and the terminator receive voltage, or

\[
\text{NM} = 6 - 3 = 3 \, \text{V} \quad \text{or} \quad \text{NM} = 25 - 6 = 19 \, \text{V}
\]

The minimum noise margin is $3 \, \text{V}$.

**11-1-1 RS-232 electrical equivalent circuit.** Figure 23 shows the equivalent electrical circuit for the RS-232 interface, including the driver and terminator. With these electrical specifications and for a bit rate of 20 kbps, the nominal maximum length of the RS-232 interface cable is approximately 50 feet.

**11-1-2 RS-232 functional description.** The pins on the RS-232 interface cable are functionally categorized as either ground (signal and chassis), data (transmit and receive)