chapter 3

The Transmission of Binary Data in Communication Systems

Since the mid-1970s, the proliferation of applications that send digital data over communication channels has resulted in the need for efficient methods of transmission, conversion, and reception of digital data. As with analog data, the amount of digital information that can be transmitted is proportional to the bandwidth of the communication channel and the time of transmission. This chapter further develops some of the basic concepts and equipment used in the transmission of digital data, including some mathematical principles that apply to transmission efficiency, and discusses the development of digital modulation, standards, error detection and correction methods, and spread spectrum techniques.

Objectives

After completing this chapter, you will be able to:

- Explain the difference between asynchronous and synchronous data transmission.
- State the relationship between communication channel bandwidth and data rate in bits per second.
- Name the four basic types of encoding used in the serial transmission of data.
- Describe the generation of FSK, PSK, QAM, and OFDM.
- Name three types of data modems and explain how they operate.
- Explain the need for and types of communication protocols.
- Compare and contrast redundancy, parity, block-check sequences, cyclical redundancy checks, and forward error correction.
- Explain the operation and benefits of spread spectrum systems.

3-1 Digital Codes

Data processed and stored by computers can be numerical (e.g., accounting records, spreadsheets, and stock quotations) or text (e.g., letters, memos, reports, and books). As previously discussed, the signals used to represent computerized data are digital, rather than analog. Even before the advent of computers, digital codes were used to represent data.

3-1-1 Early Digital Codes

Morse code

The first digital code was created by the inventor of the telegraph, Samuel Morse. The *Morse code* was originally designed for wired telegraph communication but was later adapted for radio communication. It consists of a series of "dots" and "dashes" that represent letters of the alphabet, numbers, and punctuation marks. This on/off code is shown in Fig. 3-1. A dot is a short burst of RF energy, and a dash is a burst of RF that is 3 times longer than a dot. The dot and dash on periods are separated by dot-length spaces or off periods. With special training, people can easily send and receive messages at speeds ranging from 15 to 20 words per minute to 70 to 80 words per minute.

The earliest radio communication was also carried out by using the Morse code of dots and dashes to send messages. A hand-operated telegraph key turned the carrier of a transmitter off and on to produce the dots and dashes. These were detected at the receiver and mentally converted by an operator back to the letters and numbers making up the message. This type of radio communication is known as *continuous-wave (CW) transmission.*

Continuous-wave (CW)

Baudot code

Another early binary data code was the *Baudot* (pronounced *baw-dough*) *code* used in the early teletype machine, a device for sending and receiving coded signals over a communication link. With teletype machines, it was no longer necessary for operators to learn Morse code. Whenever a key on the typewriter keyboard is pressed, a unique code is generated and transmitted to the receiving machine, which recognizes and then prints the corresponding letter, number, or symbol.

The 5-bit Baudot code is shown in Fig. 3-2. With 5 bits, $2^5 = 32$ different symbols can be represented. The different 5-bit combinations, together with two shift codes, can generate the 26 letters of the alphabet, 10 numbers, various punctuation marks, and control functions. If the message is preceded by the letter shift code (11011), all the following codes are interpreted as letters of the alphabet. Sending the figure shift code (11111) causes all the following characters to be interpreted as numbers or punctuation marks. The Baudot code is rarely used today, having been supplanted by codes that can represent more characters and symbols.

Figur 3-1 The Morse code. A dot (.) is a short click; a dash (—) is a long click.

A •—	N —•	, ——••——
B —•••	O ———	. •—•—•—
C —•—•	P •——•	1 •————
D —••	Q ——•—	2 ••———
E •	R •—•	3 •••——
F ••—•	S •••	4 ••••—
G ——•	T —	5 •••••
H ••••	U ••—	6 —••••
I ••	V •••—	7 ——•••
J •———	W •——	8 ———••
K —•—	X —••—	9 ————•
L •—••	Y —•——	0 —————
M ——	Z ——••	

Figur 3-2 The Baudot code.

Character Shift		Binary Code						Character Shift		Binary Code					
Letter	Figure	Bit:	4	3	2	1	0	Letter	Figure	Bit:	4	3	2	1	0
A	–		1	1	0	0	0	Q	1		1	1	1	0	1
B	?		1	0	0	1	1	R	4		0	1	0	1	0
C	:		0	1	1	1	0	S	bel		1	0	1	0	0
D	$		1	0	0	1	0	T	5		0	0	0	0	1
E	3		1	0	0	0	0	U	7		1	1	1	0	0
F	!		1	0	1	1	0	V	;		0	1	1	1	1
G	&		0	1	0	1	1	W	2		1	1	0	0	1
H	#		0	0	1	0	1	X	/		1	0	1	1	1
I	8		0	1	1	0	0	Y	6		1	0	1	0	1
J	′		1	1	0	1	0	Z	″		1	0	0	0	1
K	(		1	1	1	1	0	Figure shift			1	1	1	1	1
L	)		0	1	0	0	1	Letter shift			1	1	0	1	1
M	.		0	0	1	1	1	Space			0	0	1	0	0
N	,		0	0	1	1	0	Line feed (LF)			0	1	0	0	0
O	9		0	0	0	1	1	Blank (null)			0	0	0	0	0
P	0		0	1	1	0	1								

3-1-2 Modern Binary Codes

Binary code

For modern data communication, information is transmitted by using a system in which the numbers and letters to be represented are coded, usually by way of a keyboard, and the binary word representing each character is stored in a computer memory. The message can also be stored on magnetic tape or disk. The following sections describe some widely used codes for transmission of digitized data.

American Standard Code for Information Interchange. The most widely used data communication code is the 7-bit binary code known as the *American Standard Code for Information Interchange* (abbreviated ASCII and pronounced *ass-key*), which can represent 128 numbers, letters, punctuation marks, and other symbols (see Fig. 3-3). With ASCII, a sufficient number of code combinations is available to represent both uppercase and lowercase letters of the alphabet.

American Standard Code for Information Interchange (ASCII)

The first ASCII codes listed in Fig. 3-3 have two- and three-letter designations. These codes initiate operations or provide responses for inquiries. For example, BEL or 0000111 will ring a bell or a buzzer; CR is carriage return; SP is a space, such as that between words in a sentence; ACK means "acknowledge that a transmission was received"; STX and ETX are start and end of text, respectively; and SYN is a synchronization word that provides a way to get transmission and reception in step with each other. The meanings of all the letter codes are given at the bottom of the table.

Hexadecimal Values. Binary codes are often expressed by using their *hexadecimal,* rather than decimal, values. To convert a binary code to its hexadecimal equivalent, first divide the code into 4-bit groups, starting at the least significant bit on the right and working to the left. (Assume a leading 0 on each of the codes.) The hexadecimal equivalents for the binary codes for the decimal numbers 0 through 9 and the letters A through F are given in Fig. 3-4.

Hexadecimal

Two examples of ASCII-to-hexadecimal conversion are as follows:

1. The ASCII code for the number 4 is 0110100. Add a leading 0 to make 8 bits and then divide into 4-bit groups: 00110100 = 0011 0100 = hex 34.
2. The letter w in ASCII is 1110111. Add a leading 0 to get 01110111; 01110111 = 0111 0111 = hex 77.

GOOD TO KNOW

The transmitters used in garage door openers are modulated by binary-coded pulses.

Figur 3-3 The popular ASCII code.

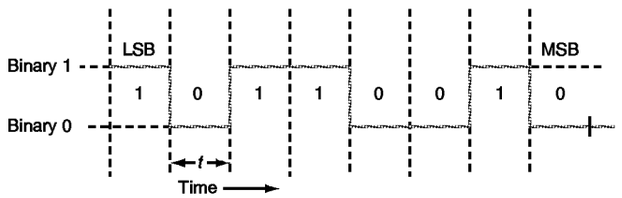
Bit:	6	5	4	3	2	1	0
NUL	0	0	0	0	0	0	0
SOH	0	0	0	0	0	0	1
STX	0	0	0	0	0	1	0
ETX	0	0	0	0	0	1	1
EOT	0	0	0	0	1	0	0
ENQ	0	0	0	0	1	0	1
ACK	0	0	0	0	1	1	0
BEL	0	0	0	0	1	1	1
BS	0	0	0	1	0	0	0
HT	0	0	0	1	0	0	1
NL	0	0	0	1	0	1	0
VT	0	0	0	1	0	1	1
FF	0	0	0	1	1	0	0
CR	0	0	0	1	1	0	1
SO	0	0	0	1	1	1	0
SI	0	0	0	1	1	1	1
DLE	0	0	1	0	0	0	0
DC1	0	0	1	0	0	0	1
DC2	0	0	1	0	0	1	0
DC3	0	0	1	0	0	1	1
DC4	0	0	1	0	1	0	0
NAK	0	0	1	0	1	0	1
SYN	0	0	1	0	1	1	0
ETB	0	0	1	0	1	1	1
CAN	0	0	1	1	0	0	0
EM	0	0	1	1	0	0	1
SUB	0	0	1	1	0	1	0
ESC	0	0	1	1	0	1	1
FS	0	0	1	1	1	0	0
GS	0	0	1	1	1	0	1
RS	0	0	1	1	1	1	0
US	0	0	1	1	1	1	1
SP	0	1	0	0	0	0	0
!	0	1	0	0	0	0	1
"	0	1	0	0	0	1	0
#	0	1	0	0	0	1	1
$	0	1	0	0	1	0	0
%	0	1	0	0	1	0	1
&	0	1	0	0	1	1	0
'	0	1	0	0	1	1	1
(	0	1	0	1	0	0	0
)	0	1	0	1	0	0	1
*	0	1	0	1	0	1	0

Bit:	6	5	4	3	2	1	0
+	0	1	0	1	0	1	1
,	0	1	0	1	1	0	0
-	0	1	0	1	1	0	1
.	0	1	0	1	1	1	0
/	0	1	0	1	1	1	1
0	0	1	1	0	0	0	0
1	0	1	1	0	0	0	1
2	0	1	1	0	0	1	0
3	0	1	1	0	0	1	1
4	0	1	1	0	1	0	0
5	0	1	1	0	1	0	1
6	0	1	1	0	1	1	0
7	0	1	1	0	1	1	1
8	0	1	1	1	0	0	0
9	0	1	1	1	0	0	1
:	0	1	1	1	0	1	0
;	0	1	1	1	0	1	1
<	0	1	1	1	1	0	0
=	0	1	1	1	1	0	1
>	0	1	1	1	1	1	0
?	0	1	1	1	1	1	1
@	1	0	0	0	0	0	0
A	1	0	0	0	0	0	1
B	1	0	0	0	0	1	0
C	1	0	0	0	0	1	1
D	1	0	0	0	1	0	0
E	1	0	0	0	1	0	1
F	1	0	0	0	1	1	0
G	1	0	0	0	1	1	1
H	1	0	0	1	0	0	0
I	1	0	0	1	0	0	1
J	1	0	0	1	0	1	0
K	1	0	0	1	0	1	1
L	1	0	0	1	1	0	0
M	1	0	0	1	1	0	1
N	1	0	0	1	1	1	0
O	1	0	0	1	1	1	1
P	1	0	1	0	0	0	0
Q	1	0	1	0	0	0	1
R	1	0	1	0	0	1	0
S	1	0	1	0	0	1	1
T	1	0	1	0	1	0	0
U	1	0	1	0	1	0	1

Bit:	6	5	4	3	2	1	0
V	1	0	1	0	1	1	0
W	1	0	1	0	1	1	1
X	1	0	1	1	0	0	0
Y	1	0	1	1	0	0	1
Z	1	0	1	1	0	1	0
[	1	0	1	1	0	1	1
E	1	0	1	1	1	0	0
]	1	0	1	1	1	0	1
^	1	0	1	1	1	1	0
–	1	0	1	1	1	1	1
`	1	1	0	0	0	0	0
a	1	1	0	0	0	0	1
b	1	1	0	0	0	1	0
c	1	1	0	0	0	1	1
d	1	1	0	0	1	0	0
e	1	1	0	0	1	0	1
f	1	1	0	0	1	1	0
g	1	1	0	0	1	1	1
h	1	1	0	1	0	0	0
i	1	1	0	1	0	0	1
j	1	1	0	1	0	1	0
k	1	1	0	1	0	1	1
l	1	1	0	1	1	0	0
m	1	1	0	1	1	0	1
n	1	1	0	1	1	1	0
o	1	1	0	1	1	1	1
p	1	1	1	0	0	0	0
q	1	1	1	0	0	0	1
r	1	1	1	0	0	1	0
s	1	1	1	0	0	1	1
t	1	1	1	0	1	0	0
u	1	1	1	0	1	0	1
v	1	1	1	0	1	1	0
w	1	1	1	0	1	1	1
x	1	1	1	1	0	0	0
y	1	1	1	1	0	0	1
z	1	1	1	1	0	1	0
{	1	1	1	1	0	1	1
¦	1	1	1	1	1	0	0
}	1	1	1	1	1	0	1
	1	1	1	1	1	1	0
DEL	1	1	1	1	1	1	1

NUL = null
SOH = start of heading
STX = start of text
ETX = end of text
EOT = end of transmission
ENQ = enquiry
ACK = acknowledge
BEL = bell
BS = back space
HT = horizontal tab
NL = new line
VT = vertical tab
FF = form feed
CR = carriage return
SO = shift-out
SI = shift-in
DLE = data link escape
DC1 = device control 1
DC2 = device control 2
DC3 = device control 3
DC4 = device control 4
NAK = negative acknowledge
SYN = synchronous
ETB = end of transmission block
CAN = cancel
SUB = substitute
ESC = escape
FS = field separator
GS = group separator
RS = record separator
US = unit separator
SP = space
DEL = delete

Figur 3-4 Serial transmission of the ASCII letter M.



Extended Binary Coded Decimal Interchange Code (EBCDIC)

Extended Binary Coded Decimal Interchange Code. The *Extended Binary Coded Decimal Interchange Code* (EBCDIC, pronounced *ebb-see-dick*), developed by IBM, is an 8-bit code similar to ASCII allowing a maximum of 256 characters to be represented. Its primary use is in IBM and IBM-compatible computing systems and equipment. It is not as widely used as ASCII.

3-2 Principles of Digital Transmission

Data can be transmitted in two ways: parallel and serial.

3-2-1 Serial Transmission

Parallel data transmission is not practical for long-distance communication. Data transfers in long-distance communication systems are made serially; each bit of a word is transmitted one after another (see Fig. 3-4). The figure shows the ASCII form for the letter M (1001101) being transmitted 1 bit at a time. The LSB is transmitted first, and the MSB last. The MSB is on the right, indicating that it was transmitted later in time. Each bit is transmitted for a fixed interval of time t. The voltage levels representing each bit appear on a single data line one after another until the entire word has been transmitted. For example, the bit interval may be 10 μs, which means that the voltage level for each bit in the word appears for 10 μs. It would therefore take 70 μs to transmit a 7-bit ASCII word.

Expressing the Serial Data Rate. The speed of data transfer is usually indicated as number of bits per second (bps or b/s). Some data rates take place at relatively slow speeds, usually a few hundred or several thousand bits per second. Personal computers, communicate at rates up to 53,000 bps with a conventional modem or up to 8 Mbps with special digital modems over the telephone lines. However, in some data communication systems such as local-area networks, bit rates as high as tens of billions bits per second are used.

The speed of serial transmission is, of course, related to the bit time of the serial data. The speed in bits per second, denoted by bps, is the reciprocal of the bit time t, or bps $= 1/t$. For example, assume a bit time of 104.17 μs. The speed is bps $= 1/104.17 \times 10^{-6} =$ 9600 bps.

If the speed in bits per second is known, the bit time can be found by rearranging the formula: $t = 1/\text{bps}$. For example, the bit time at 230.4 kbps (230,400 bps) is $t = 1/230{,}400 = 4.34 \times 10^{-6} = 4.34$ μs.

Baud rate

Another term used to express the data speed in digital communication systems is *baud rate.* Baud rate is the number of signaling elements or symbols that occur in a given unit of time, such as 1 s. A *signaling element* is simply some change in the binary signal transmitted. In many cases it is a binary logic voltage level change, either 0 or 1, in which case, the baud rate is equal to the data rate in bits per second. In summary, the baud rate is the reciprocal of the smallest signaling interval.

$$\text{Bit rate} = \text{baud rate} \times \text{bit per symbol}$$

$$\text{Bit rate} = \text{baud rate} \times \log_2 S$$

where S = number of states per symbol

Quadrature amplitude modulation (QAM)

The symbol or signaling element can also be one of several discrete signal amplitudes, frequencies, or phase shifts, each of which represents 2 data bits or more. Several unique modulation schemes have been developed so that each symbol or baud can represent multiple bits. The number of symbol changes per unit of time is no higher than the straight binary bit rate, but more bits per unit time are transmitted. Multiple symbol changes can be combined to further increase transmission speed. For example, a popular form of modulation known as *quadrature amplitude modulation (QAM)* combines

multiple amplitude levels with multiple phase shifts to produce many bits per baud. (QAM is discussed later in this chapter.) Each symbol is a unique amplitude level combined with a unique phase shift that corresponds to a group of bits. Multiple amplitude levels may also be combined with different frequencies in FSK to produce higher bit rates. As a result, higher bit rates can be transmitted over telephone lines or other severely bandwidth-limited communication channels that would not ordinarily be able to handle them. Several of these modulation methods are discussed later.

Assume, e.g., a system that represents 2 bits of data as different voltage levels. With 2 bits there are $2^2 = 4$ possible levels, and a discrete voltage is assigned to each.

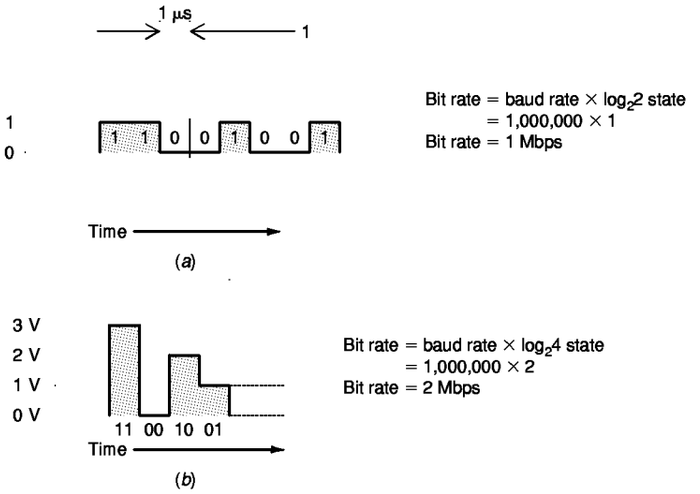
00	0	V
01	1	V
10	2	V
11	3	V

In this system, sometimes called pulse-amplitude modulation (PAM), each of the four symbols is one of four different voltage levels. Each level is a different symbol representing 2 bits. Assume, e.g., that it is desired to transmit the decimal number 201, which is binary 11001001. The number can be transmitted serially as a sequence of equally spaced pulses that are either on or off [see Fig. 3-5(*a*)]. If each bit interval is 1 μs, the bit rate is $1/1 \times 10^{-6} = 1{,}000{,}000$ bps (1 Mbps). The baud rate is also 1 million bps.

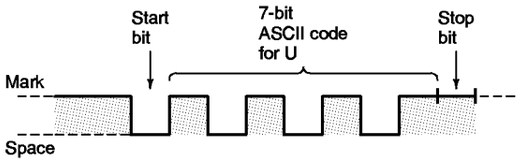
Using the four-level system, we could also divide the word to be transmitted into 2-bit groups and transmit the appropriate voltage level representing each. The number 11001001 would be divided into these groups: 11 00 10 01. Thus the transmitted signal would be voltage levels of 3, 0, 2 and 1 V, each occurring for a fixed interval of, say, 1 μs [see Fig. 3-5 (*b*)]. The baud rate is still 1 million because there is only one symbol or level per time interval (1 μs). However, the bit rate is 2 million bps—double the baud rate—because each symbol represents 2 bits. We are transmitting 2 bits per baud. You will sometimes see this referred to as bits per hertz or bits/Hz. The total transmission time is also shorter. It would take 8 μs to transmit the 8-bit binary word but only 4 μs to transmit the four-level signal. The bit rate is greater than the baud rate because multiple levels are used.

The PAM signal in Fig. 3-5(*b*) can be transmitted over a cable. In wireless applications, this signal would first modulate a carrier before being transmitted. Any type of modulation can be used, but PSK is the most common.

Figure 3-5 The bit rate can be higher than the baud rate when each symbol represents 2 or more bits. (*a*) Bit rate = baud rate = 1 bit/μs = 1,00,000 bps = 1 Mbps. (*b*) Baud rate = 1 symbol/μs = 1,000,000 baud rate; bit rate = 2 bits/Baud = 2,000,000 bps.



Figur 3-6 Asynchronous transmission with start and stop bits.



Because of the sequential nature of serial data transmission, naturally it takes longer to send data in this way than it does to transmit it by parallel means. However, with a high-speed logic circuit, even serial data transfers can take place at very high speeds. Currently that data rate is as high as 10 billion bits per second (10 Gbps) on copper wire cable and up to 100 billion bits per second (100 Gbps) on fiber-optic cable. So although serial data transfers are slower than parallel transfers, they are fast enough for most communication applications.

GOOD TO KNOW

With a high-speed logic circuit, even serial data transfers can take place at very high speeds. Data on copper wire cable can travel at speeds up to 1 billion bits per sec, and fiber-optic cable can transport data speeds up to 40 Gbps.

3-2-2 Asynchronous Transmission

Asynchronous transmission

Start and stop bits

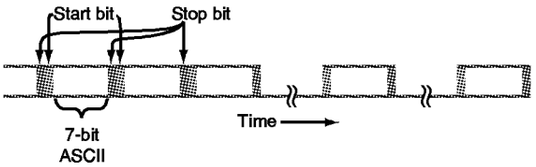
In *asynchronous transmission* each data word is accompanied by *start and stop bits* that indicate the beginning and ending of the word. Asynchronous transmission of an ASCII character is illustrated in Fig. 3-6. When no information is being transmitted, the communication line is usually high, or binary 1. In data communication terminology, this is referred to as a *mark*. To signal the beginning of a word, a start bit, a binary 0 or space, as shown in the figure, is transmitted. The start bit has the same duration as all other bits in the data word. The transmission from mark to space indicates the beginning of the word and allows the receiving circuits to prepare themselves for the reception of the remainder of the bits.

After the start bit, the individual bits of the word are transmitted. In this case, the 7-bit ASCII code for the letter U, 1010101, is transmitted. Once the last code bit has been transmitted, a stop bit is included. The stop bit may be the same duration as all other bits and again is a binary 1 or mark. In some systems, 2 stop bits are transmitted, one after the other, to signal the end of the word.

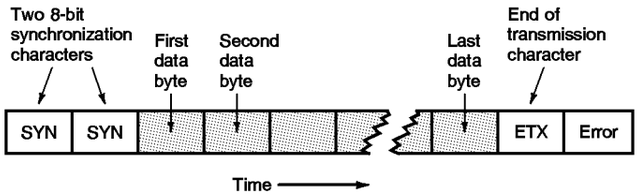
Most low-speed digital transmission (the 1200- to 56,000-bps range) is asynchronous. This technique is extremely reliable, and the start and stop bits ensure that the sending and receiving circuits remain in step with each other. The minimum separation between character words is 1 stop plus 1 start bit, as Fig. 3-7 shows. There can also be time gaps between characters or groups of characters, as the illustration shows, and thus the stop "bit" may be of some indefinite length.

The primary disadvantage of asynchronous communication is that the extra start and stop bits effectively slow down data transmission. This is not a problem in low-speed applications such as those involving certain printers and plotters. But when huge volumes of information must be transmitted, the start and stop bits represent a significant percentage of the bits transmitted. We call that percentage the *overhead* of transmission. A 7-bit ASCII character plus start and stop bits is 9 bits. Of the 9 bits, 2 bits are not data. This represents $2/9 = 0.222$, or 22.2 percent inefficiency or overhead. Removing the start and stop bits and stringing the ASCII characters end to end allow many more data words to be transmitted per second.

Figur 3-7 Sequential words transmitted asynchronously.



Figur 3-8 Synchronous data transmission.



3-2-3 Synchronous Transmission

Synchronous data transmission

The technique of transmitting each data word one after another without start and stop bits, usually in multiword blocks, is referred to as *synchronous data transmission*. To maintain synchronization between transmitter and receiver, a group of synchronization bits is placed at the beginning of the block and the end of the block. Figure 3-8 shows one arrangement. Each block of data can represent hundreds or even thousands of 1-byte characters. At the beginning of each block is a unique series of bits that identifies the beginning of the block. In Fig. 3-8, two 8-bit synchronous (SYN) codes signal the start of a transmission. Once the receiving equipment finds these characters, it begins to receive the continuous data, the block of sequential 8-bit words or bytes. At the end of the block, another special ASCII code character, ETX, signals the end of transmission. The receiving equipment looks for the ETX code; detection of this code is how the receiving circuit recognizes the end of the transmission. An error detection code usually appears at the very end of the transmission.

The special synchronization codes at the beginning and end of a block represent a very small percentage of the total number of bits being transmitted, especially in relation to the number of start and stop bits used in asynchronous transmission. Synchronous transmission is therefore much faster than asynchronous transmission because of the lower overhead.

An important consideration in synchronous transmission is how the receiving station keeps track of the individual bits and bytes, especially when the signal is noisy, since there is no clear separation between them. This is done by transmitting the data at a fixed, known, precise clock rate. Then the number of bits can be counted to keep track of the number of bytes or characters transmitted. For every 8 bits counted, 1 byte is received. The number of received bytes is also counted.

Synchronous transmission assumes that the receiver knows or has a clock frequency identical to that of the transmitter clock. Usually, the clock at the receiver is derived from the received signal, so that it is precisely the same frequency as, and in synchronization with, the transmitter clock.

Example 3-1

A block of 256 sequential 12-bit data words is transmitted serially in 0.016 s. Calculate (*a*) the time duration of 1 word, (*b*) the time duration of 1 bit, and (*c*) the speed of transmission in bits per second.

a. $t_{\text{word}} = \dfrac{0.016}{256} = 0.000625 = 625\ \mu\text{s}$

b. $t_{\text{bit}} = \dfrac{625\ \mu\text{s}}{12\ \text{bits}} = 52.0833\ \mu\text{s}$

c. $\text{bps} = \dfrac{1}{t} = \dfrac{1}{52.0833} \times 10^{-6} = 19{,}200\ \text{bps} \quad \text{or} \quad 19.2\ \text{kbps}$

3-2-4 Encoding Methods

Whether digital signals are being transmitted by baseband methods or broadband methods (see Sec. 1-4), before the data is put on the medium, it is usually encoded in some way to make it compatible with the medium or to facilitate some desired operation connected with the transmission. The primary encoding methods used in data communication are summarized below.

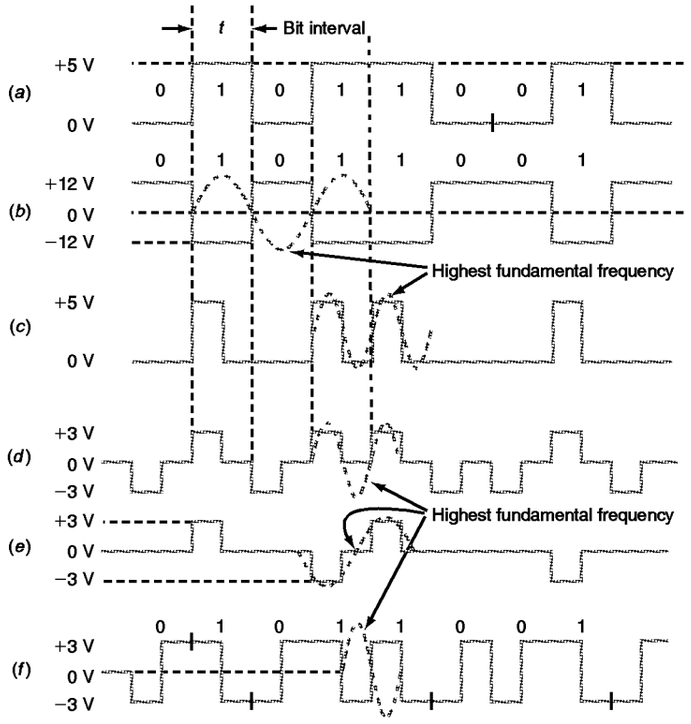
Nonreturn to zero (NRZ) encoding

Nonreturn to Zero. In the *nonreturn to zero (NRZ) method of encoding* the signal remains at the binary level assigned to it for the entire bit time. Figure 3-9(*a*), which shows unipolar NRZ, is a slightly different version of Fig. 3-4. The logic levels are 0 and +5 V. When a binary 1 is to be transmitted, the signal stays at +5 V for the entire bit interval. When a binary 0 is to be sent, the signal stays at 0 V for the total bit time. In other words, the voltage does not return to zero *during* the binary 1 interval.

In *unipolar* NRZ, the signal has only a positive polarity. A *bipolar* NRZ signal has two polarities, positive and negative, as shown in Fig. 3-9(*b*). The voltage levels are +12 and −12 V. The popular RS-232 serial computer interface uses bipolar NRZ, where a binary 1 is a negative voltage between −3 and −25 V and a binary 0 is a voltage between +3 and +25 V.

The NRZ method is normally generated inside computers, at low speeds, when asynchronous transmission is being used. It is not popular for synchronous transmission because there is no voltage or level change when there are long strings of sequential binary 1s or 0s. If there is no signal change, it is difficult for the receiver to determine just where one bit ends and the next one begins. If the clock is to be recovered from the transmitted data in a synchronous system, there must be more frequent changes, preferably one per bit. NRZ is usually converted to another format, such as RZ or Manchester, for synchronous transmissions.

Figure 3-9 Serial binary encoding methods. (*a*) Unipolar NRZ. (*b*) Bipolar NRZ. (*c*) Unipolar RZ. (*d*) Bipolar RZ. (*e*) Bipolar RZ-AMI. (*f*) Manchester or biphase.



Return to zero (RZ) encoding

Unipolar RZ

Return to Zero. In *return to zero (RZ) encoding* [see Fig. 3-9(*c*) and (*d*)] the voltage level assigned to a binary 1 level returns to zero during the bit period. *Unipolar RZ* is illustrated in Fig. 3-9(*c*). The binary 1 level occurs for 50 percent of the bit interval, and the remaining bit interval is zero. Only one polarity level is used. Pulses occur only when a binary 1 is transmitted; no pulse is transmitted for a binary 0.

Bipolar RZ

Bipolar RZ is illustrated in Fig. 3-9(*d*). A 50 percent bit interval +3-V pulse is transmitted during a binary 1, and a −3-V pulse is transmitted for a binary 0. Because there is one clearly discernible pulse per bit, it is extremely easy to derive the clock from the transmitted data. For that reason, bipolar RZ is preferred over unipolar RZ.

Alternative mark inversion (AMI)

A popular variation of the bipolar RZ format is called *alternative mark inversion (AMI)* [see Fig. 3-9(*e*)]. During the bit interval, binary 0s are transmitted as no pulse. Binary 1s, also called *marks,* are transmitted as alternating positive and negative pulses. One binary 1 is sent as a positive pulse, the next binary 1 as a negative pulse, the next as a positive pulse, and so on.

Manchester encoding

Manchester. *Manchester encoding,* also referred to as *biphase encoding,* can be unipolar or bipolar. It is widely used in LANs. In the Manchester system a binary 1 is transmitted first as a positive pulse, for one half of the bit interval, and then as a negative pulse, for the remaining part of the bit interval. A binary 0 is transmitted as a negative pulse for the first half of the bit interval and a positive pulse for the second half of the bit interval [see Fig. 3-9(*f*)]. The fact that there is a transition at the center of each 0 or 1 bit makes clock recovery very easy. However, because of the transition in the center of each bit, the frequency of a Manchester-encoded signal is 2 times an NRZ signal, doubling the bandwidth requirement.

Choosing a Coding Method. The choice of an encoding method depends on the application. For synchronous transmission, RZ and Manchester are preferred because the clock is easier to recover. Another consideration is average dc voltage buildup on the transmission line. When unipolar modes are used, a potentially undesirable average dc voltage builds up on the line because of the charging of the line capacitance. To eliminate this problem, bipolar methods are used, where the positive pulses cancel the negative pulses and the dc voltage is averaged to zero. Bipolar RZ or Manchester is preferred if dc buildup is a problem.

DC buildup is not always a problem. In some applications the average dc value is used for signaling purposes. An example is an Ethernet LAN, which uses the direct current to detect when two or more stations are trying to transmit at the same time.

Other encoding methods are also used. The encoding schemes used to encode the serial data recorded on magnetic floppy and hard disks are an example. Other schemes used in networking will be discussed later.

3-3 Transmission Efficiency

Transmission efficiency—i.e., the accuracy and speed with which information, whether it is voice or video, analog or digital, is sent and received over communication media—is the basic subject matter of a field known as *information theory.* Information theorists seek to determine mathematically the likelihood that a given amount of data being transmitted under a given set of circumstances (e.g., medium, bandwidth, speed of transmission, noise, and distortion) will be transmitted accurately.

3-3-1 Hartley's Law

The amount of information that can be sent in a given transmission is dependent on the bandwidth of the communication channel and the duration of transmission. A