COMMUNICATING WITH DISPLAY TERMINALS

Roger K. deBry

COMMUNICATING WITH DISPLAY TERMINALS

Roger K. deBry

McGraw-Hill Book Company

New York St. Louis San Francisco Auckland
Bogotá Hamburg Johannesburg London Madrid
Mexico Montreal New Delhi Panama Paris
São Paulo Singapore Sydney Tokyo Toronto

CONTENTS

1

INTRODUCTION

This book deals with communication in the context of an information-processing network, as exemplified in figure 1.1. Such a network can be characterized in a number of different ways. Physically, it is simply a connection between two end users of the network. Logically, a network is characterized by the set of functions which it provides. Network functions are most often described in terms of a set of architected *layers*. An example of this, shown in figure 1.2, is the Open Systems Interconnection (OSI) architecture model defined by the International Organization for Standardization (ISO).

The lower layers of the OSI model provide for the transmission of electrical signals, the control of data circuits, the transfer of bits, and the reliable transfer of blocks of information between adjacent nodes in the network. The intermediate layers provide for the routing of data through the network, optimization of network resources, and recovery. Although we will touch upon many of these subjects to add perspective, the focus of this book will be upon data streams, the flow of data formats and codes between presentation layers of the network.

There seem to be two significant forces which drive the definition of device data streams. The first of these is historical: that is, with what existing code standards, devices, or software must the data stream work? Much of the data-stream architecture in existence today reflects direc-

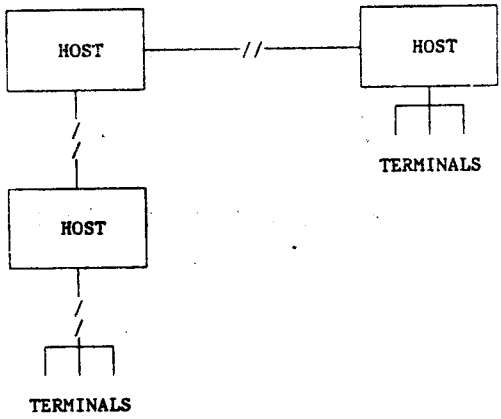


figure 1.1 An information-processing network.

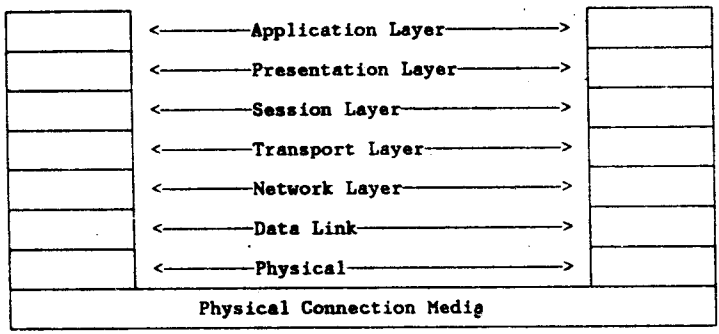


figure 1.2 Open Systems Interconnection model.

tions which were set in the beginnings of the data-processing industry. Thus it is important to understand these early beginnings and how they may affect the definition of new data streams.

The first information-processing network was probably created when primitive people used resonant logs to communicate over long distances. Later it was found that animal membranes stretched over the ends of the logs could produce a more distinctive sound. Some tribes used male (or low-pitched) drums in combination with female (or high-pitched) drums to encode their messages. Other primitive people used reed pipes, whistles made of bone, and rams' horns to communicate. These early communication systems had one thing in common: the use of "coded" information, where particular sound patterns carried predefined meanings to the listener.

The earliest attempts to transmit information through electrical means were made by Stephen Gray in 1727 and Charles Francis Dufay in 1733. The object of these experiments was really to determine the distance electricity could be transmitted, rather than to transmit information. The development of the Leyden jar gave impetus to such experiments, and in 1747 Sir William Watson succeeded in discharging a jar and measuring the effect over almost 2 miles of iron wire supported on poles.

Early experiments in telegraphy were based on the use of such high-voltage schemes and electrostatics to transmit information. For example, Morrison is credited with having suggested the first telegraph system in 1753. A separate circuit was used for each character to be signaled. A frictional generator was to be connected at the sending end, and small pieces of paper would be attracted by suitable electrodes at the receiving end.

The development of electromagnetic telegraphy came later, only after several other significant developments:

In 1800, Count Alessandro Volta produced the battery, making continuous current available for the first time.

In 1819, Oersted found that a magnetized needle would be deflected from its normal position when brought close to a wire that carried current.

André Marie Ampère, studying electromagnetism, proposed the use of magnetic needles and coils for the reception of signals. His first telegraph system employed a pair of line wires for each character to be sent, much like the earlier electrostatic systems.

In 1825, William Sturgeon developed the electromagnet. The magnetized bar rang a bell and transmitted intelligence by code.

In 1832, Samuel F. B. Morse was returning from Europe aboard a ship when he overheard a conversation describing the development of the electromagnet. This discussion excited his imagination, and upon his return home he began work on a telegraph system using this principle. By 1835 he had developed the first working model of his telegraph. By 1838 he had developed a code for use in transmitting messages, and in 1839 he transmitted the first message over his telegraph system. The first commercial telegraph system, set up by Morse in 1843, linked Baltimore and Washington.

Early telegraph systems had a signal speed of about 2 characters per second. In 1874, Jean-Maurice-Émile Baudot invented a scheme which allowed 6 communication channels to be combined on a single physical link. Two years later, Alexander Graham Bell spoke his first sentence over the telephone. Telephone lines were first constructed in the 1890s. In 1913 vacuum-tube repeaters were introduced into telephony. In 1918 the first

carrier system allowed several voice channels to be carried over a single wire pair.

Today, in the United States a single high-capacity coaxial cable or microwave link can transmit thousands of voice channels. In a hundred years, the capacity of communications systems has increased from 2 characters per second to over 100 million characters per second. The future will certainly bring laser communications systems capable of transmitting billions of characters per second.

It seems, then, that we have developed a tremendous ability to transmit great quantities of information and will be able to transmit much larger quantities in the future. Unfortunately, our capacity for absorbing all this information and making it useful to us is severely limited. In 1945, Dr. Vannevar Bush concluded in an article in the *Atlantic Monthly* that research was being bogged down because of the vast quantities of data through which researchers had to sort. Now, several decades later, the body of knowledge in the world is said to be doubling every 5 years. Without the aid of machines, computers—which have a far greater capacity to consume and digest information than we have—we cannot begin to cope with the mass of information we must understand.

The large-scale use of computers in information-processing networks began in the mid-1950s, when it was recognized that teletypewriters could be used for transmitting data to and from the computer. This allowed more people to interact with the computer and introduced the concept of timesharing—a technique that enables many people to use one computer simultaneously on different problems, whether the computer is in the same room or hundreds of miles away. The first data streams, then, were born in this environment and were based upon the communication codes developed for teletype equipment.

The second force driving data-stream architecture is the requirement to provide the richest possible set of data-presentation functions, consistent with available technology and permissible costs. The functions available on today's display terminals would not have been thought possible 20 years ago. The ability to mix graphics, color, and advanced presentation and processing functions in the terminal gives the software developer a powerful set of tools to use in presenting data to the terminal operator. The terminal's data stream is the vehicle which the application program uses to communicate this rich set of functions to the display terminal.

We have recently experienced a great change in the data-processing industry. Technology has allowed computer manufacturers to shrink the size and cost of the computer, and it is not difficult to foresee a time when every office and home will have a small computer or a computer terminal of some kind. These will provide access to multiple private and public databases, allow us to solve complex problems—displaying the solutions

in color and graphic pictures, perform word-processing tasks, keep our grocery lists, and balance our checkbooks. The terminal becomes a window into the information processed and stored in the computer. This window may be very opaque or very clear, depending upon the capabilities of the terminal and the way in which the data is presented. A key element of this interface is the definition of the terminal's data stream. A proper set of terminal functions, and proper programming of the interface between the computer and the terminal, will provide a way for us to look clearly through this window and gain maximum benefit from the information stored in the computer.

This book attempts to develop the concepts of display terminal data streams from these two perspectives. The reader will not simply learn the bits and bytes that make up the data streams but will also gain some appreciation of why data streams are defined as they are and of how application programs should be structured to take maximum advantage of the terminal's capabilities. Thus this book will serve as a useful tool not only to those who may develop terminal systems but also to those who will write application programs which use them.

2

REPRESENTING INFORMATION IN THE COMPUTER

If the computer terminal were literally a window through which we could look into the computer's memory, most of us would find it very difficult to understand what we saw (see figure 2.1). Modern computers store information using bistable storage elements; that is, they are capable of being in one of two states. Thus what we would see if we could look directly into the computer memory would be a series of these storage elements, some of which would be in the 1 state and others of which would be in the 0 state. Looking through such a window, could you understand the following message?

```
10010001001111101011101000010000001010010
10001010100000101100110011111101010111111
11000110100011110010101000001110101000 1011
01000010100000101001000010100110100001010 1
01001010010111010111111101010000101101010 0
11111111001111010000101000100100000111110 1
```

Probably not! In any communication system, the information to be exchanged must be in some form that is understood by both the sender and the receiver. If I speak German and my friend speaks French, either I must also speak French or my friend must also speak German in order for us to communicate. When we communicate with a computer, we face a similar problem. As you can see from the above example, if we had to

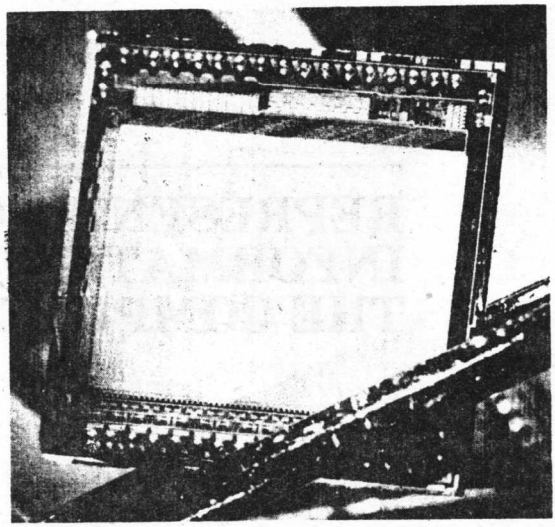
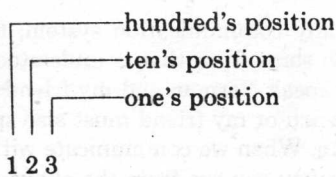


figure 2.1 Close-up view of a memory chip. *(Reproduced by permission of International Business Machines Corporation.)*

communicate with the computer using only the digits 1 and 0, we would have a difficult problem indeed. Fortunately, in modern computer systems, a computer program most often translates this internal representation of data into a form more understandable to the terminal operator.

DECIMAL COMPUTERS AND THE DECIMAL NUMBER SYSTEM

The number system most familiar to us is the decimal number system, so called because the *base* or *radix* of the number system is 10. The decimal number system has 10 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. Modern number systems use a positional notation, in which the meaning of a particular symbol, or digit, is modified by its position in the number. In the decimal number system, the digits are modified by powers of 10, according to their position, as shown:



In this example, the digit 3 is in the one's position, the digit 2 is in the ten's position, and the digit 1 is in the hundred's position. This means that there are 3 ones in the number, 2 tens, and 1 hundred. Let's verify this:

$$\begin{array}{r} 1 \times 100 = 100 \\ 2 \times 10 = 20 \\ 3 \times 1 = 3 \\ \hline 100 + 20 + 3 = 123 \end{array}$$

We can write this in a slightly different manner, to better reflect what we mean when we talk about a base 10 number. Remember that any number raised to the zero power is equal to 1!

$$\begin{aligned} 123 &= 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 \\ &= 1 \times 10 \times 10 + 2 \times 10 + 3 \times 1 \\ &= 100 + 20 + 3 \\ &= 123 \end{aligned}$$

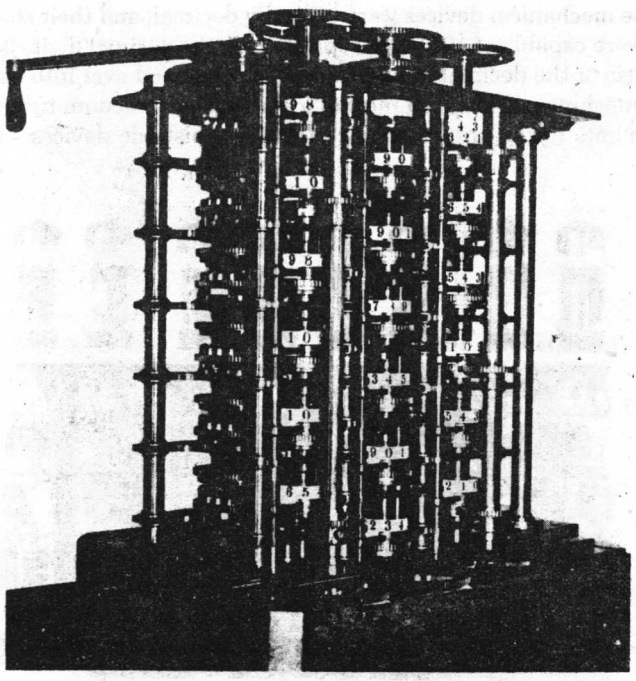


figure 2.2 Charles Babbage's Difference Engine. *(Reproduced by permission of International Business Machines Corporation.)*

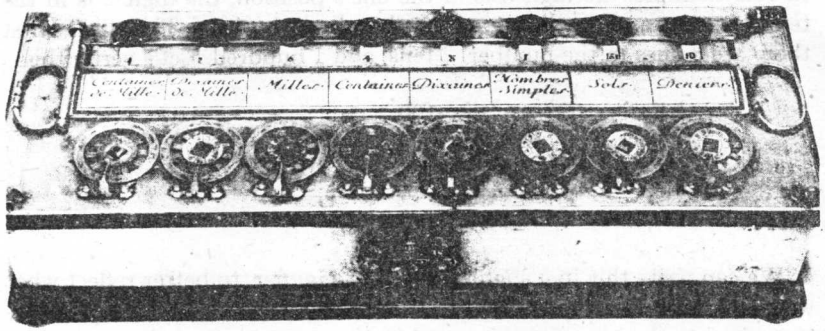


figure 2.3 Pascal's Pascaline. *(Reproduced by permission of International Business Machines Corporation.)*

Every calculating machine stores numbers by setting some storage element into one of several possible states. Early calculating machines depended upon mechanical storage elements such as wheels or shafts, as shown in figures 2.2 and 2.3.

These mechanical devices were typically decimal, and their storage elements were capable of 10 states, representing the decimal digits 0 through 9. The use of the decimal number system was carried over into early computing machines which used mechanical relays and vacuum tubes as storage elements (see figure 2.4). Since these are bistable devices—i.e., they

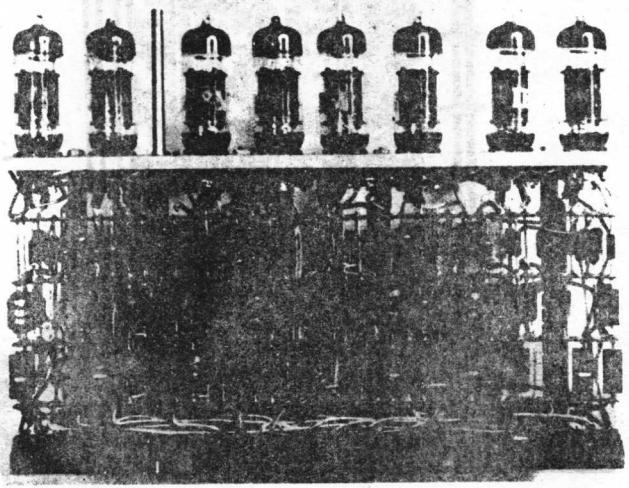


figure 2.4 Vacuum-tube storage elements. *(Reproduced by permission of International Business Machines Corporation.)*

may assume one of two stable states—the representation of a decimal number requires the use of 10 such storage elements per decimal digit. Thus the element representing the value of the required digit would be turned on, and the others would be turned off.

This is illustrated in figure 2.5, where a set of 10 switches represents a

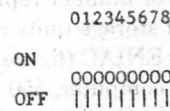


figure 2.5 Representing a decimal digit with 10 storage elements.

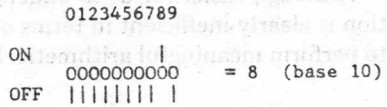


figure 2.6 Representing the digit 8 with 10 storage elements.

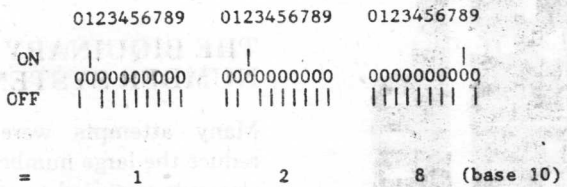


figure 2.7 Representing the decimal digit 128 with three sets of 10 storage elements.

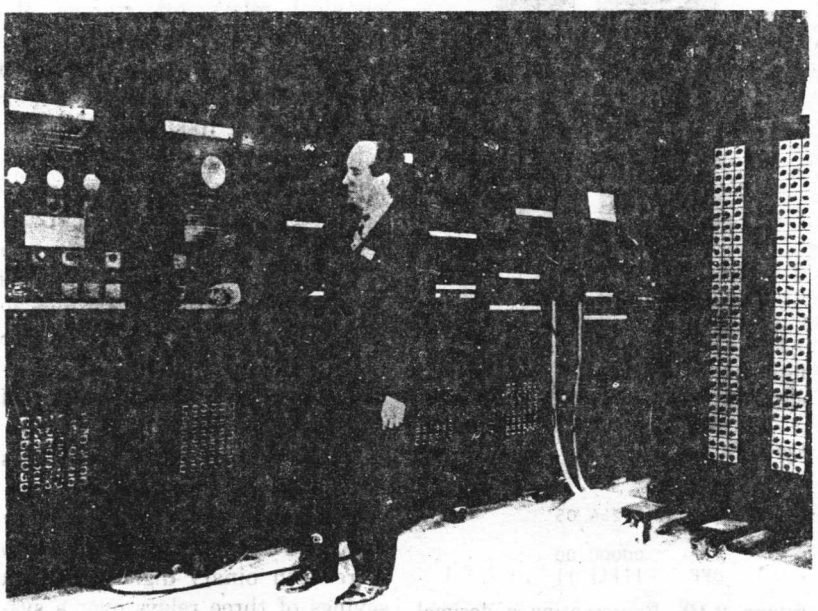


figure 2.8 ENIAC. *(Reproduced by permission of International Business Machines Corporation.)*