File Number S360-30
GC30-2007-1

IBM Systems Reference Library

IBM System/360

Introduction to Teleprocessing

This publication provides computer applications analysts and programmers with an introduction to Teleprocessing. Following a historical survey and some brief application descriptions is a review of equipment characteristics and programming techniques. Introductory material on two levels of IBM System/360 Teleprocessing programming support is then presented. A bibliography and a technical glossary conclude the publication.

PREFACE

This publication is designed to present concepts of Teleprocessing to persons familiar with computer applications in the business and scientific fields, but who have not yet used the "long distance" computing techniques and equipment. The history of Teleprocessing introduces terminology and developments such as real-time, multiprogramming, and time-sharing. Present difficulties and possibilities for the future are included.

Teleprocessing requires distinctive communication networks, codes, and procedures for error detection and for control of terminals and message transmission. The development of specialized programming techniques that can handle the new complex applications is discussed in the section on applications and in the final section, which deals with the two access methods available: BTAM (Basic Telecommunications Access Method), and QTAM (Queued Telecommunications Access Method). A bibliography and glossary conclude the publication.

Second Edition (July 1968)

Significant changes or additions to the contents of this publication will be reported in subsequent revisions or technical newsletters.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Corporation, Programming Documentation, Department 844, Research Triangle Park, North Carolina 27709.

Contents

Illustrations

This publication is an introduction to Teleprocessing concepts, system applications, equipment characteristics, and programming techniques for persons already familiar with conventional computer applications, equipment, and programming methods. Its purpose is to introduce the overall capabilities of the programming support for utilizing the IBM System/360 in Teleprocessing applications. There are two levels of support: the Basic Telecommunications Access Method (BTAM) and the more comprehensive Queued Telecommunications Access Method (QTAM). This publication provides background information for the detailed technical material in the corresponding Systems Reference Library (SRL) manuals concerning these two types of support.

HISTORICAL REVIEW

Although a majority of Teleprocessing (TP) computer systems have been developed relatively recently, the potential benefits of these systems were noted by early computer users. It is important to realize that present TP equipment and programming techniques are not innovations but evolutionary steps in the development of computing systems.

This evolutionary development has produced significant factors in present-day TP systems. One such factor is that much telecommunications equipment was developed long before the modern digital computer. As a consequence, facilities that were designed for convenient or economical data communication are not always ideally suited to a computer system. Teleprocessing demands new techniques not required for input/output devices designed specifically as peripheral computer equipment. Another factor is that data is often entered into a computer directly from a terminal unit. This direct man-machine communication requires data validation and programming procedures not encountered in off-line systems.

Finally, perhaps the most significant factor to computer personnel first encountering TP applications is the unfamiliar technical terminology. This makes TP concepts appear difficult and discourages many people from learning more about them. People who have little patience with TP terminology should consider the situation of a communications expert first encountering such common computer jargon as compiler, loader, dump, bug, and data set!

The attachment of communications equipment to digital computers involves much more than establishing appropriate electrical connections; it also involves a merging of the separate technologies of communications and computing. Substantial new capabilities, not previously possible, become practical; however, to capitalize on this potential, one must have a general understanding of TP devices and terminology and of associated computer equipment and programming techniques. This publication is designed to present this additional information to persons already familiar with computer systems.

Developments in Teleprocessing

It comes as a surprise to many that the first use of TP equipment with a computer occurred before the advent of stored program systems. As early as 1940, scientific calculations were telegraphed several hundred miles to an electromechanical calculator, and within a minute results were returned to the distant users. Such pioneering demonstrations indicate that there was an early awareness of the convenience of using a powerful calculator from distant locations. It is also worth noting that these early experiments were considered to be more than just connecting communications terminals to calculating equipment, to the extent that a patent was issued on one such system. Another interesting observation is that experimenters recognized that a calculator was fast enough to service several remote terminals concurrently, and, although the terms were not used then, the concept of sharing the central calculator was a precursor of recent advances in multiprogramming (running several interleaved programs concurrently) and time-sharing (distributing the resources of a computer system to a number of independent users).

During the early part of the decade following World War II, military planners realized that new approaches were needed to provide an adequate air-defense system. They recognized that vast amounts of data would have to be collected and processed, and results returned immediately to command personnel. These command personnel could interrogate the system and plan effective defensive action. Here then was a need for transmitting data from remote sensing devices (for example, radar stations) to a central computer, processing the data rapidly enough to influence the environment being monitored (that is, in "real-time"), and using a man-machine system to combine the judgment and experience of command personnel with the rapid, accurate processing of a computer. The equipment and techniques formulated for such systems as SAGE (Semi-Automatic Ground Environment) had a substantial influence on the evolution of similar systems for commercial applications.

The first commercial applications of communications-oriented computer systems were in industries that had requirements for real-time control or for rapid access to large data files. Hence, by the late 1950s, computers were installed for controlling both industrial processes and critical time-dependent business inventories. A number of airline reservation systems were developed. At first, these used special-purpose, fixed-program machines to maintain inventories of passenger-seat space available on future flights. Soon, however, general-purpose systems were programmed to provide more general reservation-accounting functions. The design of the largest of these, SABRE (for Semi-Automatic Business Research Environment), was stimulated, as the name implies, by the earlier military systems.

These general-purpose systems required special terminal units, separate channels to attach the communications equipment, modified processors to protect against inadvertent destruction of data, additional core storage and special drums to hold transactions awaiting processing, and large disk files to hold the reservation data. Not only was commercially available equipment extensively modified and augmented with special devices, but also complicated, special-purpose programs were developed to control and coordinate the equipment complex. Developers soon recognized that the programming of large Teleprocessing systems was vitally important to system performance and reliability. Designing and testing these complex programs proved to require extremely competent personnel, long development schedules, and extensive amounts of machine time for testing.

Reliability and efficiency became as important for programs as for equipment. The cost of programming approached, and in some cases exceeded, the cost of the equipment itself. Recognition of these factors indicated that widespread, economical use of such systems in commercial applications would require the development of:

- Computers incorporating TP features as standard equipment rather than as special attachments.
- General-purpose, pretested programs to service the Teleprocessing environment.

A third phase (after the military and business applications) in the evolution of Teleprocessing systems began in the early 1960s. Although the earliest application of computers was to scientific problems, the potential of TP systems in this area went relatively unnoticed. Even though computer equipment had become more economical and, with the introduction of operating systems, more convenient and efficient, user access to scientific computers was not improved. In fact, individual users often were experiencing more inconvenience and longer turnaround time (the interval between submitting a job and receiving results). The influence of turnaround time upon programmer productivity was most obvious where the results of computer processing were needed before other work could be performed. Several leading university computer centers, recognizing the impact of turnaround time on user productivity, developed the concept of providing many people with direct computer access via numerous remote terminal units sharing the central computer. Experienced people could apply their knowledge and judgment directly to the formulation and solution of complex scientific problems; inexperienced users could program and debug problems with less detailed computer knowledge.

Teleprocessing Today

The following trends summarize this historical introduction:

- TP and computing technologies have different heritages with attendant differences in equipment requirements and technical terminology.
- Present combinations of TP and computing technologies are the result of trends that have evolved over the past two decades.
- Special-purpose equipment and computer features are being supplanted by general-purpose, communications-oriented computer equipment.
- Special-purpose control programs are evolving into general-purpose, communications-oriented programming systems.

At the present stage of development, the following factors are important:

- Direct substitution of a TP system for a conventional computer system is no more sensible than rigid conversion of a manual accounting application to a computer. The best use of a new system is made not by doing an old job in the same manner on new equipment, but by rethinking the entire approach with consideration of new capabilities.
- It is not currently feasible to satisfy all requirements with standard products and programming support.
- However, many application requirements can be satisfied by standard, general-purpose computer equipment.
- Also, a large number of applications can be processed under available operating systems augmented with communications-oriented control programs.

System/360

The IBM System/360 offers a complete set of integrated communications-oriented features:

- A compatible family of central processing units with an interrupt facility and instruction repertoire designed specifically to meet requirements of Teleprocessing;
- A large, directly addressed, protected primary core storage for holding control and application programs and transaction queues;
- Secondary storage devices with a wide spectrum of speed, capacity, and price characteristics;
- Several types of transmission control units to attach communications lines to the computer;
- A broad range of terminal devices for data collection, message input/output, and graphic display;
- A range of facilities to control and coordinate the central processing equipment and the multiprogramming essential for TP applications;
- Two data-management access methods designed specifically for Teleprocessing.

Access Methods

The two access methods for IBM System/360 TP environments are:

1. Basic Telecommunications Access Method (BTAM),
2. Queued Telecommunications Access Method (QTAM).

Both of these access methods are available under either the full Operating System (OS) or the smaller Disk Operating System (DOS). The two different versions of programming support are necessary because of the considerable variation in TP system requirements. BTAM (Basic Telecommunications Access Method) provides the programmer with simple, efficient access to the communications environment so that he can program terminal units in a manner consistent with that used for conventional sequential input/output devices. BTAM controls data transmission but it does not provide for elaborate message queuing or for processing of the message itself.

The other access method, QTAM (Queued Telecommunications Access Method), provides all the capabilities just mentioned for BTAM. In addition, as the name implies, it includes facilities for queuing messages on direct-access storage devices. It also provides complete capabilities for data-collection and message-switching applications, among others. Like BTAM, QTAM insulates the programmer from most of the details of TP equipment.

BTAM and QTAM alleviate many of the problems traditionally associated with TP systems. In particular, they largely eliminate the need to train personnel in the details of TP equipment. With these tested packages, the programmer need not be as concerned with the diabolically elusive bugs normally encountered in real-time system testing. The net result is more reliable programs developed sooner and at lower cost than is usually the case with specialized TP control programs. BTAM and QTAM are not panaceas; they do not satisfy all application requirements. However, they do provide facilities for most applications, and, with modifications, they can be extended to meet other needs.

Readers desiring more detailed information on BTAM and QTAM may want to read the *IBM System/360 Telecommunications Access Method* section of this manual before studying the specific SRL manuals. Readers desiring general information on TP applications, equipment characteristics, and programming techniques should continue with the next section.

As already noted, the early use of TP systems in military command and control applications was necessary since there was no other realistic way to perform the job. In a commercial situation, whether with a business or a scientific orientation, the evaluation of a TP system must include consideration of several factors besides mere technical feasibility. Justification of a TP system involves most of the same factors that must be considered in installing a conventional computer system or converting a new application to an installed computer. Speed, accuracy, economy, and flexibility must be evaluated as always; however, other factors become more significant: organizational disruption, system reliability, and personnel training, for example. It is difficult to evaluate these factors, much less to assign quantitative measures to them. TP systems, however, are usually justified by such factors as:

- Customer convenience. Brokerage firms, for example, require rapid execution and confirmation of customer orders.
- Inventory control. Manufacturing applications are common, but there are other "inventories"—such as airline space availability—that can be effectively controlled by a TP system.
- Credit Control. Central data files can provide assurance that a customer has funds or credit approval.
- Management control. Immediate access to centralized data files provides more timely information for control of business operations.
- Industrial control. Computer control of key production factors increases productivity of capital equipment (for example, petroleum refineries).
- Equipment centralization. In collecting data from remote sources, either intermittently (as in production data collection) or periodically (as in central summarizing of statistical data from distant branch offices), one central computer may do the processing that would otherwise require a separate system at each remote site.
- Innovation. Some applications are just not possible without a TP system (for example, on-line debugging, text editing, and computer-assisted instruction).

A brief survey of some common TP applications provides the background prerequisite to a description of the equipment and programming characteristics of the TP environment. The remainder of this section describes applications from the user's viewpoint (that is, what the system does) rather than from the engineer's or programmer's viewpoint (that is, how the system operates). However, since the user must communicate with the system by means of a terminal device, commonly used terminal equipment is described and illustrated.

DATA COLLECTION

The most basic form of TP application, data entry, may involve several variations:

1. Data can be transcribed first into some medium (for example, punched cards or paper tape) readable by the remote terminal. The medium is then placed in the terminal and, after contact is established with the central computer, the terminal reads the data and transmits it to the computer, which stores it for later processing.

The IBM 1050 Data Communication System (see Figure 1) is often used for this purpose. The 1050 can be employed for off-line data preparation as well as for on-line data transmission. It is a modular unit accommodating (as can be seen in the picture) data on both punched cards and paper tape. This form of data collection is similar to the data-conversion operations (for example, card-to-tape) performed at most computer installations. The principal difference is that the reader is a terminal device located at a remote location rather than at the central computer site.

2. Another approach is to key the data directly on-line to the computer, bypassing transcription to a physical medium. This approach precludes manual data verification and permits less efficient use of the communications line since data can be read from a terminal faster than it can be keyed manually. This manual keying can be described as "on-line keypunching."

3. In another variation of data collection, instead of the terminal site contacting the computer, the computer initiates the connection. With appropriate equipment, the computer can dial a terminal, read a batch of data, turn the terminal off, and "hang-up." Computer-initiated dialing is known as *autocalling*, and terminal operation without an operator is known as *unattended operation*. This kind of data entry is

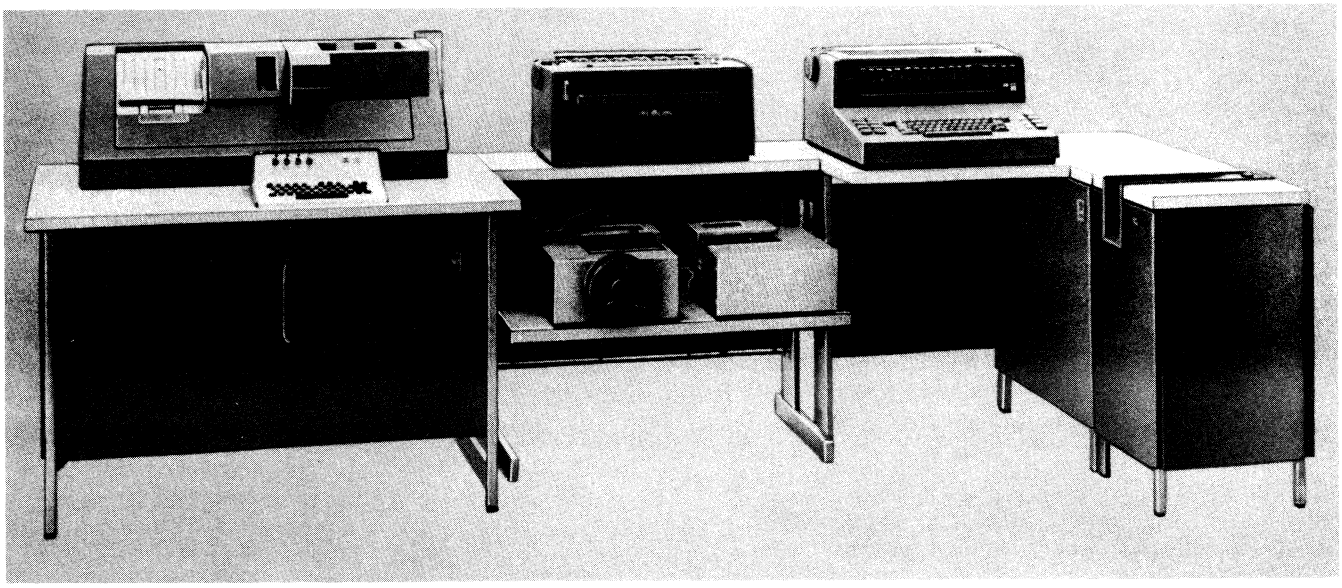


Figure 1. IBM 1050 Data Communication System

often used to gather daily operating data from remote sites. The data is placed in the terminals at the close of the business day, and during the night the computer "calls" the terminals and collects the data.

Often the same facilities that are used during the day for voice communication are used for data transmission after business hours, when they are free of voice traffic, or, in some instances, when lower tariffs (the rates charged by the communication companies) are in effect.

4. Data may be sent to the computer intermittently rather than in batches. This requires a different form of communication with the computer since it would be impractical to contact the computer every time data is ready. Instead, a group of terminals share a line that is always connected to the computer. This is termed a *multipoint* line since several terminals are connected to it simultaneously. This line is often "private," meaning that it is privately owned or leased from a common carrier (a company providing communications services). Since only one terminal can use the line at any one time, there are occasions when a terminal has to wait for the line to become free. Also, there must be some way to prevent several terminals from trying to use the line simultaneously. Two types of control are commonly used for multipoint lines: *contention*, in which the terminals contend for the line and an interlocking mechanism resolves "tie" situations, and *polling*, in which a control mechanism invites each terminal in turn to send any data that it has.

5. Two other varieties of data collection are often used in industrial applications. Sometimes a process is monitored by metering devices that automatically read out their current settings. Another form of production data collection uses an IBM 1030 Data Collection System (see Figure 2). Upon completion of a job, an employee enters into the 1030 such data as his man number, the number of units produced, and the job status. This terminal is designed to operate in industrial environments and has special features for reading identification badges, coded data cartridges, and numerical counts. The computer records the time of completion, stores the record, and later processes it for labor accounting and production control purposes. The IBM 1030 is a good example of a job-oriented terminal performing functions unsuited to conventional data communications equipment.

MESSAGE SWITCHING

Data collection, as described above, involves neither on-line processing of message data nor transmission of data from the computer to a terminal. Message switching extends TP system capabilities to encompass these two additional functions.

A message-switching configuration includes a number of terminals connected to a central computer. A message may be transmitted from a sending terminal to one or more receiving terminals by noting in the beginning portion of the

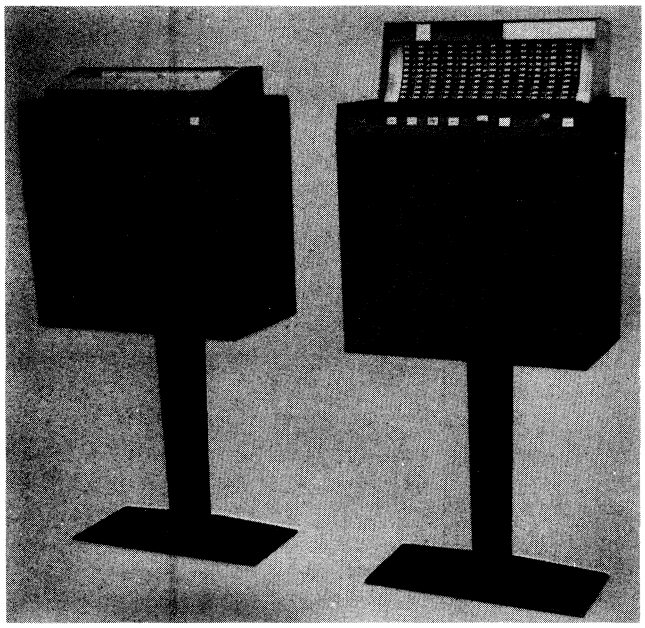


Figure 2. IBM 1030 Data Collection System

message, called the *message header*, one or more symbolic addresses of the terminals to which the remaining part of the message, called the *message text*, is to be sent. The computer program analyzes the header, converts symbolic destinations to actual terminal locations, *addresses* the receiving terminal (that is, contacts it and determines if it can receive messages), and transmits the message. If an addressed terminal is busy sending or receiving other information or is otherwise unavailable, the computer program may store the message until the terminal is available, then send it. Because of this function, message switching is often called *store and forward* switching.

Notice that, in addition to both sending and receiving data, the computer must also analyze the message header to identify the receiving terminals. These operations, as well as storing, or *queuing* (that is, placing in a waiting line), messages until the receiving terminals become available, are functions required for message switching but not needed in the data-collection applications described earlier. The programmer must inform the TP program of the control characters that delimit destination locations in the header and of means for determining the end of the header and beginning of the text. Tables relating symbolic locations to actual terminals must be established. In addition, the system must provide for checking for lost messages, retransmission in case of line errors, retrieval of earlier messages, addressing of groups of terminals, detection of invalid addresses, control of the allocation of space for storing messages, recognition of important messages and assignment of priority, and a number of similar operations.

INQUIRY APPLICATIONS

In this kind of application, an inquiry entered from a terminal is processed by the computer, data is retrieved from a file, and a reply is returned to the originating terminal. It differs from message switching in that the complete message text (in contrast to only the message header) is usually analyzed, a central data file is referenced and maintained, and a new message is composed for return to the original (in contrast to some other) terminal.

This mode of terminal-computer communication is often termed *conversational* since sending and receiving operations alternate. The time interval between the completion of an inquiry and the beginning of a computer response is called the system *response time*, typically a period of a few seconds. The duration of conversations may vary considerably since a computer response may stimulate the terminal user to pose another inquiry. For example, if one particular airline flight is fully booked, the customer may be interested in space available on flights at other times.

Inquiry applications differ greatly according to the type of terminal employed as well as the type of central data file employed. A few examples will illustrate this point.

Audio Response

Certainly the most widely available terminal device imaginable is the ordinary telephone. Although we all use it for communication with other people, it can also be employed for computer inquiry. A simple example is an inventory application by which a salesman wishes to determine the delivery period for an item.

Even while in the customer's office, the salesman can dial a central computer and then, using touch buttons, enter the item number of the product. Equipment at the computer site decodes the stock number, retrieves the associated inventory record from a central file, and determines the delivery time. Then the computer constructs a reply to the user and issues a series of codes to an audio-response device. This equipment obtains from its recorded vocabulary the appropriate spoken words and sends them to the inquiring telephone. This audio response informs the salesman of the expected delivery time. The salesman can then discuss this with the customer and either enter the order (again via the telephone) or make another inquiry, perhaps for some related item.

Figure 3 shows an IBM 7770 Audio Response Unit. A similar device, the IBM 7772, has a larger vocabulary, but cannot service as many communication lines as the 7770.

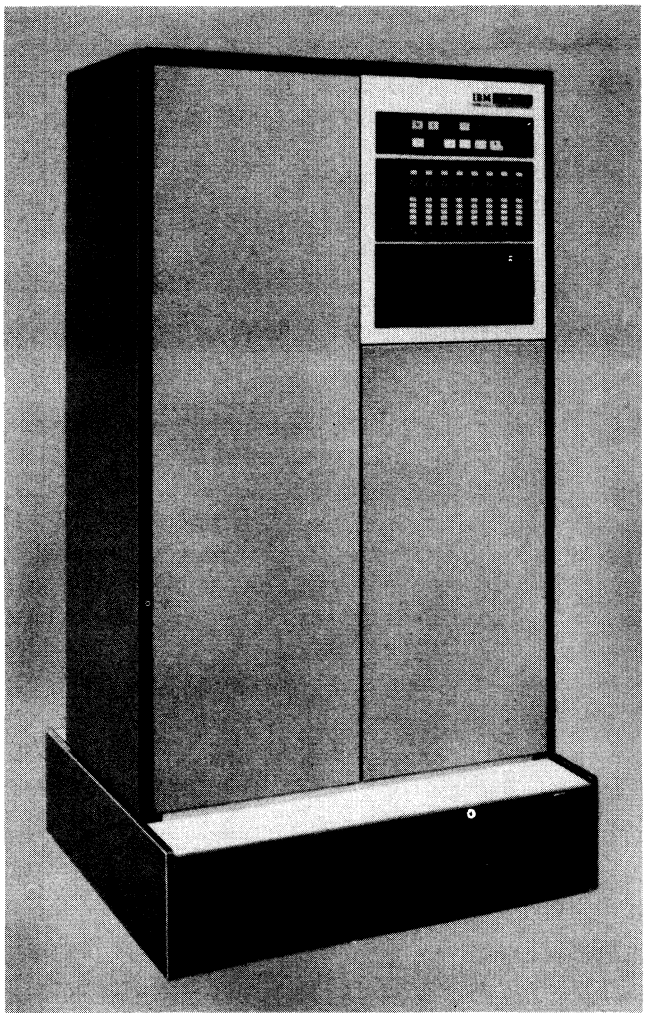


Figure 3. IBM 7770 Audio Response Unit

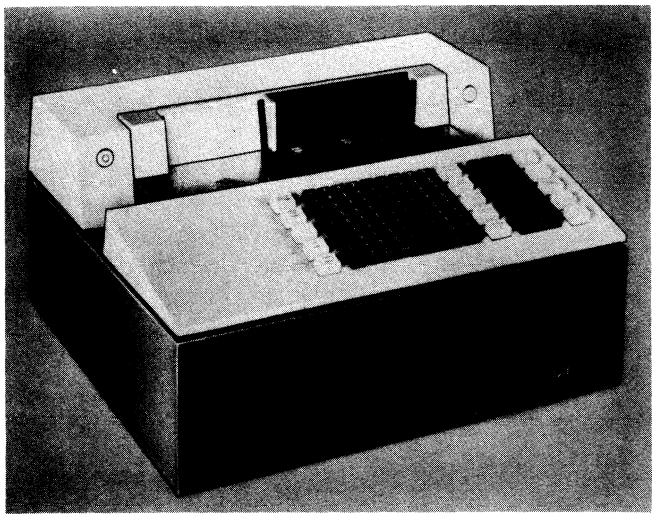


Figure 4. IBM 1060 Data Communication System

Figure 5. IBM 2260 Display Station

Deposit Accounting

Another common inquiry system is used in savings bank applications. A terminal specifically designed for this purpose, the IBM 1060 Data Communication System, is shown in Figure 4. Its keyboard is arranged for banking use; it can print directly upon a customer's savings passbook. A typical operation involves keying in the passbook number together with the dollar amount to be deposited or withdrawn. This data is used by the computer to check a customer record and update it with the new balance. Concurrently, a message is returned to print the transaction, date, and new balance in the passbook. Many other functions, such as checking for overdrafts, stopping payment on lost books, and computing interest payments, are also provided.

Information Retrieval

In retrieving information from a large centralized data file, a user is often interested in examining an amount of data rather than in obtaining direct responses to specific inquiries. Consequently, there is a requirement for considerably faster output than in the other inquiry applications described. A visual display satisfies this requirement, and the IBM 2260 Display Station (pictured in Figure 5) is often used in retrieval applications. The user enters a key word or a series

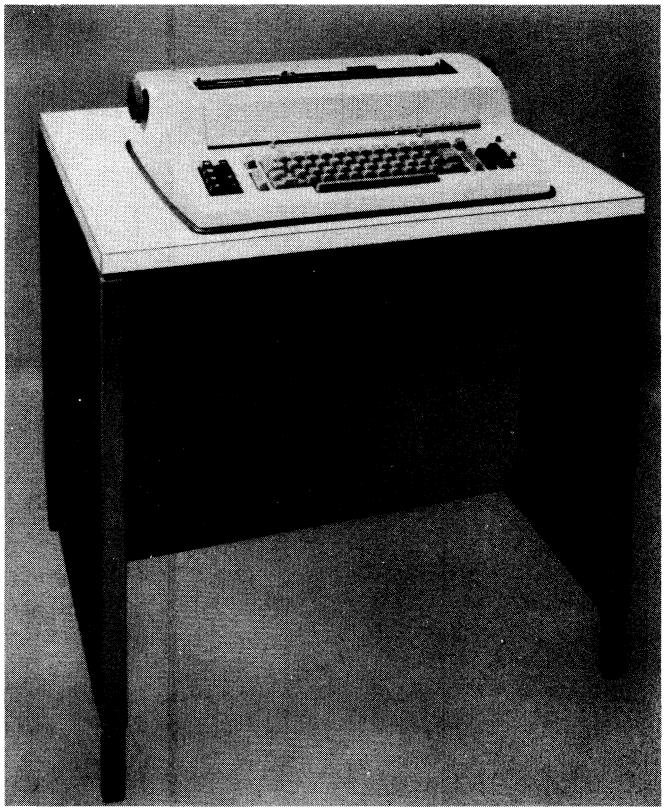


Figure 6. IBM 2740 Communications Terminal

of terms describing the area of interest. After analyzing these, the computer retrieves abstracts of related information and returns them to the display station.

Displaying data on a 2260 screen is faster than printing the same data on a terminal printer. As a consequence, the user can scan a page and, by entering new keywords, either indicate a desire to receive another page or obtain more detailed information on topics currently displayed. In either case, this display is ideally suited to applications in which a considerable volume of information must be visually scanned. A new page of several hundred characters can be transmitted from computer to terminal in a few seconds.

REMOTE PROCESSING APPLICATIONS

In this type of application, data entered from a terminal is processed by system programs available under the operating system, and results are returned later by the computer. Unlike the applications already described, this requires no interaction between the separate terminals, and there is little sharing of common data files among a group of users. Instead, each terminal user has the impression that he is the sole user of the computer.

Text Editing

In this application, the user can enter, modify, and print textual material. The IBM 2740 Communications Terminal (Figure 6) is often used because it is identical in appearance and keyboard touch to an electric typewriter. Secretaries can use it off-line as a conventional office typewriter and on-line to retrieve form letters or manuscript drafts, modify them, and then direct the system to print them.

Remote Computing

Here, the user enters computer programs (in contrast to data) for later input to any of the language processors available under the operating system. The user may build up a library of programs, request compilations, receive results or diagnostic data, and make program modifications.

Communication with the computer may be by any of a number of terminals, including the IBM 1050 and 2740 already described. AT&T TWX (Teletypewriter Exchange) service may be employed for remote computing applications, with the Model 33 or Model 35 Teletypewriter serving as the terminal unit. TWX is essentially a form of dial-up teletypewriter service. In the past, this service was used mostly for point-to-point communication between TWX terminals. More recently, however, TWX has been used to access time-shared remote computing centers.

* * * * *

Many TP applications do not fit nicely into the foregoing categories of data collection, message switching, inquiry, or remote processing. These classifications are intended to represent general approaches to TP applications, to illustrate the wide range of capabilities in TP systems, and to describe certain problems. This background material has introduced some communications terminology together with a survey of terminal equipment. The next sections discuss more TP equipment, programming techniques, and access methods.

More detailed information on terminal devices is contained in technical reference manuals listed in the bibliography.

SPECIAL CHARACTERISTICS OF TP SYSTEMS

The reader has probably already noticed two characteristics of the input/output data formats that are not very common in conventional computer applications. The first is that the header portion of a message is in "free form"; that is, its fields are not delimited by specific character positions, as is usually the case with unit records. The reason for this is that Teleprocessing systems must be designed to accommodate the people using them, even at the expense of added programming complexity or decreased system efficiency. Since it is easier for a person to enter a series of symbols

than it would be to enter a fixed number of actual destination codes, the message header is designed for user, as opposed to programmer, convenience and efficiency.

A second characteristic of TP systems is that message lengths are highly variable. In message switching, for example, short messages containing just a few words must be processed with long messages containing thousands of characters of text. The program design must accommodate such variations with flexibility and efficiency. This is in contrast to the more rigid record sizes found in a majority of off-line computer systems.

Unlike conventional batch-job applications, many TP applications never actually end. At the end of each day's operating period, messages still are queued for transmission, and the job is never completed. Instead, a "closedown" is performed to temporarily discontinue processing until a "startup" is initiated the next day.

Another characteristic of message switching and most other TP applications is the fact that, because the job is never actually completed, it is not possible simply to do a rerun in case of a system error. Instead, vital system status data is recorded at "checkpoints" so that job steps may be restarted in case of error. Checkpoint/restart procedures are complicated by the fact that a TP system is seldom totally idle and, as a consequence, it is difficult to record the exact status of the many concurrent tasks in operation at any instant. Yet the importance of checkpoint/restart procedures is apparent since users are heavily dependent on the reliability of the central system.

Another point demonstrated by such a relatively simple application as message switching is that the loads placed on the system are unpredictable. At some times the system load is light; at other times the demand for processing time and storage space is substantially greater. The loading is determined by the external operating environment and cannot be dictated by the computer. Instead the programming system must have considerable agility to be able to acquire system resources when needed, yet not continually monopolize valuable computer facilities in anticipation of worst-case contingencies.

One final TP characteristic that message switching demonstrates is the problems of debugging the user-written control programs. How can peak load conditions be simulated? How are bugs associated with complex timing or queuing conditions detected? When the system is "down" because of equipment malfunction or program error, how are the users informed? When the system is restarted, what data, if any, has been destroyed? Who is affected by the loss? Questions like these give some idea of the problems associated with the designing and testing of TP programs.

This section describes the basics of TP equipment. It is organized around the primary requirements of a TP system:

Communication networks
Character codes
Error detection
Transmission control
Terminal control

COMMUNICATIONS NETWORKS

Communications networks are composed of *channels* (also called *circuits* or *lines*). There are basically two types of networks: switched and nonswitched.

A switched network is one in which connection between a terminal and a computer is made through common-carrier exchange equipment. Dialing establishes a connection, and the connection is maintained only while transmitting data. If the computer is equipped with an *automatic calling* facility, it can, under program control, issue a sequence of dialing digits. Otherwise, manual dialing is used at either the terminal or the computer. Another available facility is *auto-answering* whereby the answering location (the called location) automatically responds to the originating location (the calling location). There is also a provision at each location to change the mode of transmission between *talk mode* (for normal voice communication) and *data mode* (for transmitting data).

A nonswitched network is one in which the lines connecting the computer and terminals are permanently established. These do not require dialing and are available for use at any time. They are also known as *private* or *leased* lines since they are reserved for the exclusive, private use of one customer and are leased from the common carrier on a contract basis.

A communications line can be further classified according to the direction in which it communicates data:

1. A line that can transmit data in only one direction is a *simplex* line. A terminal that only sends input data to a computer is termed *send only*, and a terminal that only receives computer output is termed *receive only*.
2. A line that can transmit data in either direction, but not simultaneously, is called *half-duplex*. A terminal on such a line is in either *send mode* or *receive mode* (but not both).
3. A line that can transmit data in both directions at the same time is called *duplex*. A terminal on such a line can send and receive at the same time.

Transmission Speed

A line can also be classified by its speed; that is, the maximum rate at which it can accommodate data transmission. This quality is sometimes expressed in terms of *bandwidth:* the range of frequencies the line can accommodate. Directly related to the bandwidth of a line is its speed capability in *bits per second.* A character is represented on a communication line by a series of bits, the number of which depends on the transmission code and transmission technique used. Each bit requires a specific amount of time on the line, and the bit rate, expressed in bits per second, is the reciprocal of this amount of time.

Another term frequently used to express speed capability is the *baud.* To properly define this term would require a more detailed discussion of transmission techniques than is warranted in a publication of this scope. Because the numerical value of a line's speed in bauds and in bits per second is sometimes the same, the two terms have come to be used interchangeably. This is, however, incorrect, as the two terms are not synonymous. The term *baud* should be avoided, and the more useful *bits per second* used instead.

Many different types of channels are available, in a variety of speed capabilities. For data communications purposes they may be classed as follows:

1. *Narrow-band*, or low-speed, lines have a bit rate of up to 300 bits per second (bps). Included in this category are *telegraph-grade* and *sub-voice grade* lines, which refer to the lower speeds in the narrow-band range.
2. *Voice-band*, or *voice-grade*, lines operate at medium speeds—over 300 bps. The term voice-grade is used because this range can be accommodated by the circuits used for ordinary voice communication in the audio range (frequencies that can be heard by the human ear).
3. *Wideband*, or high-speed, lines operate at bit rates of about 18,000 bps and higher. These lines often are used for transmitting data directly from one computer to another at very high speeds.

Generally, higher line speeds require more sophisticated and expensive common-carrier facilities, and the tariffs for their use are correspondingly higher.

Two other measures of a line's speed are useful: The *character rate* equals the bit rate divided by the number of bits per character, and is expressed in characters per second. To express speed in *words per minute* requires that *word* be defined. In communications usage a word is usually considered to be six characters: five symbols followed by a space character. Using this convention,

$$\text{words per minute} = \frac{\text{bits per second}}{\text{bits per character}} \times 10$$

Note that the foregoing discussion pertains to the rated capacity of a communication line. Actual data rates will be lower; they depend on the sending speed of the terminals attached to the line, and, for keyboard-entered data, upon the keying speed of the terminal operator.

Equipment Connections

Data processing equipment is connected to a communications network by a device that goes by many names, including modulator and demodulator unit, mod/demod, modem, subset, and data set. Certain trade names such as Data-Phone (*an AT&T trademark*) also are used. Whatever its name, this device provides an *interface* or common boundary between data processing equipment and communications equipment. In this publication, the term *modem* is used.

Modems vary considerably according to the types of networks, data rates, and forms of signalling employed. However, they all are designed to convert the binary signals of business machines to the transmission frequencies of communications equipment and vice versa. For example, an IBM 1050 terminal sends a sequence of bits to a modem which converts (that is, modulates) it to an equivalent sequence of transmission signals. At the receiving computer site, another modem reconverts (that is, demodulates) this signal sequence into the same bit sequence originally sent by the terminal.

The electrical interface between data processing equipment and communications equipment is defined by industry standards. This fact, coupled with the fact that the modulation/demodulation process is in most cases "transparent" to the telecommunications equipment, means that in actual practice, there is little need for system designers and programmers to become intimately familiar with modulation techniques.

CHARACTER CODES

Teleprocessing systems use several different methods to represent data characters. Some of these were originally designed for communications equipment; others are derived from codes used for representing data on computer I/O equipment. The codes differ primarily in the number of bits used to represent the characters (called the *level* of the code) and in the particular patterns of bit settings used to represent the characters. A coded character may be classified as either a graphic character, representing a symbol, or a control character, controlling a terminal function. The number of different characters that can be coded is dependent on the level of the code and on the coding scheme employed.

In some codes, often termed *shifted codes*, certain control characters are used to specify the way in which following graphic characters are to be interpreted. With such a use of control characters, called a shift convention, each bit pattern can represent more than one symbol. Thus, in this type of code, the number of characters that can be represented is greater than the number of distinct bit combinations. However, these codes have the disadvantage of sometimes requiring two coded characters (a shift control character and a graphic character (to represent one symbol. Some five-level telegraphy codes have shift conventions for figures and letters, and some seven-level codes have such conventions for upper and lower case.

Baudot Code

An important shifted code is the widely used Baudot code, named in honor of a pioneer French telegrapher. It is a five-level code and can thus represent $2^5 = 32$ different combinations (see Figure 7). Since this is not sufficient to represent the alphanumeric data (the 26 letters of the alphabet, 10 digits, plus special characters such as +, -, $) occurring in most message texts, Baudot codes are interpreted two ways depending on the shift status of the printing mechanism.

When in letters shift (LTRS) the codes represent letters; when in figures (FIGS) the codes represent digits and special symbols. Four character codes are given the same interpretation independent of shift position: blank (no bits), delete (all bits), carriage return (CR), and line feed (LF); two characters are used to change shifts (LTRS and FIGS). The result, as can be seen in Figure 7, is that Baudot code can accommodate 2 (2^5) - 6 = 58 different characters. Baudot code does not have error checking, and terminal

Byte Positions 4,5,6,7 (2,3,4,5)	Hex	S/360 Main Storage Byte Positions 0,1,2,3,(0,0,S,1) 0000	0001	0010	0011
		0	1	2	3
0000	0	Blank	E	Blank	3
0001	1	T	Z	5	"
0010	2	CR	D	CR	$
0011	3	O	B	9	5/8
0100	4	Space	S	Space	Bell
0101	5	H	Y	◆	6
0110	6	N	F	7/8	1/4
0111	7	M	X	●	/
1000	8	LF	A	LF	-
1001	9	L	W	3/4	2
1010	A	R	J	4	,
1011	B	G	FIGS ↑	&	FIGS ↑
1100	C	I	U	8	7
1101	D	P	Q	0	1
1110	E	C	K	1/8	1/2
1111	F	V	LTRS ↓	3/8	LTRS ↓

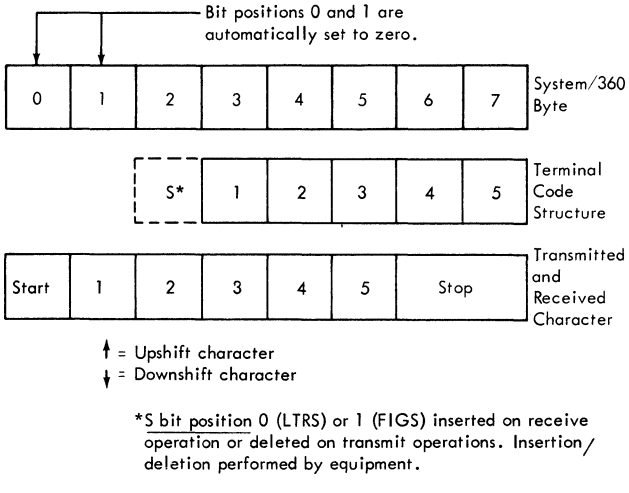


Figure 7. Baudot Code

control is achieved by special sequences of characters. Thus, for example, the sequence FIGS H LTRS is commonly used to indicate the end of a message (often abbreviated as EOM).

USASCII

The absence of checking and the limitation on the number of characters representable in Baudot code have led to increasing use of a code termed ASCII or USASCII (United States of America Standard Code for Information Interchange). This is a seven-level code, thus providing $2^7 = 128$ possible characters. An eighth parity bit, though not part of the standard, is often associated with the seven data bits. The USASCII character set consists of 34 control codes and 94 text characters including the letters of the alphabet in both upper and lower case, the 10 digits, and a number of special characters (see Figure 8).

EBCDIC

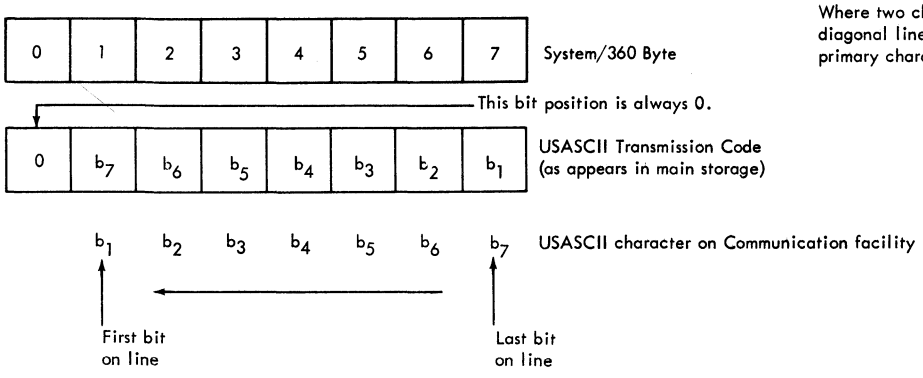
Another widely used code is EBCDIC (Extended Binary Coded Decimal Interchange Code). This is an eight-level code and, as the name implies, is widely used for exchanging data between computer systems. It has 256 possible combinations: 17 of these are used for control purposes, 96 are used for text characters, and the remaining code combinations are unassigned (see Figure 9).

In addition to the dissimilarity of code structure between the EBCDIC and USASCII codes, their collating sequences (ordering of the binary representations of the characters) differ. In the USASCII collating sequence, for example, digits precede letters; in EBCDIC, digits follow letters. Thus, the characters A, B, C, 1, 2, 3 are in sequential order (ascending) in EBCDIC, but not in USASCII.

Many TP systems accommodate several different codes by providing for code conversion, or translation, between different code representations. For example, data may be received in one transmission code, converted to a code suitable for computer processing, and then converted to still a third code suitable for transmission to an output terminal.

Translation tables are used to convert from one character code to another, and special computer instructions often are available to utilize such tables automatically. Another aid for conversion to and from 5-bit codes is incorporation of a shift bit along with the five data bits to form a 6-bit code, thus eliminating the need for explicit shift characters. This considerably simplifies internal processing since all characters then are represented by a single byte (8 bits) rather than sometimes requiring a pair of bytes.

Byte Positions 4,5,6,7 (b_4, b_3, b_2, b_1,)	Hex	S/360 Main Storage Byte Positions 0, 1, 2, 3, (0, b_7, b_6, b_5)															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	0	NUL	DLE	SP	0	@	P	`	p								
0001	1	SOH	DC1	! / ¦	1	A	Q	a	q								
0010	2	STX	DC2	"	2	B	R	b	r						·		
0011	3	ETX	DC3	#	3	C	S	c	s								
0100	4	EOT	DC4	$	4	D	T	d	t								
0101	5	ENQ	NAK	%	5	E	U	e	u								
0110	6	ACK	SYN	&	6	F	V	f	v								
0111	7	BEL	ETB	'	7	G	W	g	w								
1000	8	BS	CAN	(	8	H	X	h	x								
1001	9	HT	EM	)	9	I	Y	i	y								
1010	A	LF	SUB	*	:	J	Z	j	z								
1011	B	VT	ESC	+	;	K	[	k	{								
1100	C	FF	FS	'	<	L	\	l	¦								
1101	D	CR	GS	-	=	M	]	m	}								
1110	E	SO	RS	.	>	N	^ / ¬	n	~								
1111	F	SI	US	/	?	O	_	o	DEL								



Note:
Where two characters appear separated by a diagonal line, the character on the left is the primary character, the other is an alternate.

Figure 8. USASCII Code

	S/360 Main Storage Byte Positions 0, 1, 2, 3, (0, 1, 2, 3)															
Byte Positions 4,5,6,7 (4,5,6,7)	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL	DLE	DS		SP	&	-								0
0001	1	SOH	DC1	SOS				/		a	j			A	J		1
0010	2	STX	DC2	FS	SYN					b	k	s		B	K	S	2
0011	3	ETX	TM							c	l	t		C	L	T	3
0100	4	PF	RES	BYP	PN					d	m	u		D	M	U	4
0101	5	HT	NL	LF	RS					e	f	v		E	N	V	5
0110	6	LC	BS	ETB (EOB)	UC					f	o	w		F	O	W	6
0111	7	DEL	IL	ESC (PRE)	EOT					g	p	x		G	P	X	7
1000	8		CAN							h	q	y		H	Q	Y	8
1001	9		EM							i	r	z		I	R	Z	9
1010	A	SMM	CC	SM		¢	!		:								
1011	B	VT	CU1	CU2	CU3	.	$	,	#								
1100	C	FF	IFS		DC4	<	*	%	@								
1101	D	CR	IGS	ENQ	NAK	(	)	_	'								
1110	E	SO	IRS	ACK		+	;	>	"								
1111	F	SI	IUS	BEL	SUB	¦	¬	?	"								

0	1	2	3	4	5	6	7	System/360 Byte
0	1	2	3	4	5	6	7	EBCDIC Structure
0	1	2	3	4	5	6	7	Transmitted and received character

Figure 9. EBCDIC Code

ERROR DETECTION TECHNIQUES

Communications circuits are subject to a variety of environmental conditions not encountered in a computer system. Some of these conditions may create electrical "noise" in the circuit and cause unpredictable errors in transmitted data. There are many sources of noise, such as nearby high voltage circuits, cross-talk (accidental induction between circuits), and impulses caused by lightning strikes. Requisite to satisfactory system performance is the ability for such transmission errors to be detected and, where possible, corrected. As common-carrier facilities do not generally provide such a capability, the burden of error detection and correction falls upon the computer and terminal equipment at either end of the communication channel.

Error detection is usually performed by means of some redundant data in the transmitted message. Checking can be done at the character level and/or at the message level. Character checking is usually performed with the addition of an extra bit to the data bits comprising the character code so that the total number of 1 bits in any character is always odd (or always even). The receiving unit checks each character for proper *parity*. Such checking is identical to that performed in many computer systems, and, since the character is often depicted as a vertical set of bits, this method is called vertical redundancy checking (VRC).

While VRC techniques detect all single bit errors (picking up or dropping a single bit), it does not detect double errors. A form of coding called *fixed count coding* accomplishes more thorough checking at the expense of slightly less efficient transmission and somewhat more sophisticated checking equipment. The term *fixed count code* is derived from the fact that for any character a fixed number of bits are always one and the remaining bits are always zero. A common example of fixed count coding is the "4 of 8" code, in which every character has exactly 4 bits set to one and the remaining 4 bits set to zero. Two things should be noticed: more error conditions are detected, but there are only 70 valid character representations, whereas a VRC code using the same number of bits has 128 valid characters.

Checking at the character level detects many errors. However, it does not detect multiple bit errors or missing characters. To overcome these deficiencies, it is useful to accumulate parity longitudinally along a message as well as vertically across characters. In this technique, called *longitudinal redundancy checking* (LRC), a special check character is inserted at the end of a message or message block so that the total of the settings along the same bit position in all characters is always odd. LRC checking thus detects missing characters. Again, more sophisticated terminal equipment is necessary both to generate LRC codes when sending and to check LRC codes when receiving data.

A more thorough checking technique accumulates checking data both vertically across characters and longitudinally along messages. These *cyclic codes* require more elaborate equipment to generate and test them. A type of cyclic checking is used on most IBM direct access storage devices.

TRANSMISSION CONTROL

Transmission is controlled by devices called *transmission control units* (TCU 's). These provide a control interface between the computer and the communications network. Depending on the number and speed of lines to be serviced, several types of units are available. In the IBM System/360, for example, up to eight of the following types of transmission control units may be connected to a multiplexer channel:

2701 for up to 4 low speed lines
2702 for up to 31 low speed lines
2703 for up to 176 low speed lines
7770 for audio response on up to 48 lines
7772 for audio response on up to 8 lines

One of the most important functions performed by the TCU is coordination of the input and output of data to and from the communications network. Two modes of transmission are widely used. *Start-stop* mode, commonly used on low speed networks, delimits each character with special control bits denoting the beginning and the end of the sequence of bits forming a data character. *Synchronous* mode transmission, commonly used on medium and high speed lines, establishes a timing synchronization between sending and receiving equipment before sending a stream of characters. Start-stop mode is simple and inexpensive but requires extra bits to control each transmitted character. Synchronous mode requires more sophisticated electronics in the terminal equipment but more efficiently utilizes the communications facility.

A specific example will illustrate how a TCU operates in start-stop mode. First assume a number of low speed lines (say 30) attached to the same TCU . The TCU continuously apportions its time to each of the lines in turn. In the case of input data, the TCU continually monitors the line for an indication that a new character is about to start. This is indicated by a transition of the line from a *mark* (one or more 1 bits) status to *space* (a 0 bit) status. Then an oscillator operating at the line's bit rate causes the TCU to sample that line during each successive bit time. These line samples (either 0 or 1 bits) are assembled serially-by-bit in a shift register until the required number of samples and a stop bit have been accumulated into a character. The character is then stripped of start/stop bits and placed in another register to await transmission (in a parallel-by-bit manner) to the computer via the multiplexer channel. Meanwhile the TCU is looking for the start bit of the next character.

Output is performed in a similar fashion. A character is received from the computer and, after having start-stop bits added, is moved in parallel to a shift register which shifts the bits onto the line at the proper bit rate.

Depending on the particular types of lines involved, the TCU also has other functions. It may perform validity checks on incoming data, delete certain "idle" characters from the data stream, and detect shift codes and automatically control shift status (upper case or lower case) of a terminal. The TCU performs all these functions without program intervention, on a number of lines concurrently. As far as the programmer is concerned, the TCU presents much the same appearance as any other control unit (for example, a card reader) attached to the multiplexer channel.

The computer must assemble incoming characters into messages, translate codes, and edit messages prior to performing conventional processing operations. For output, it performs similar operations: converts codes, inserts terminal selection and control characters, and passes the outgoing message, character by character, to the TCU via the multiplexer channel.

TERMINAL CONTROL

Most computer input/output equipment uses separate paths (or lines) to transmit data and to indicate control functions. Communications equipment is connected to a computer with one or (in the case of duplex transmission) two lines. Consequently, conventions must be established to identify and distinguish between the data and the control information that must be transmitted over the same communications channels. Several factors are involved.

A first factor is that, unlike other I/O devices, a terminal often is not continuously connected to the system. The terminal user may dial the computer to submit input, or the computer may establish contact with the terminal (via the automatic calling facility described earlier) prior to transmitting output data. Although much of this is handled by automatic calling and answering, the system must still initiate and confirm the communication link to the terminal. This introductory communication is sometimes called "handshaking."

A second factor is that several terminals may be attached to the same private line. This reduces the cost of the communication facilities but requires additional control conventions to determine which of the several terminals is to send or receive data. Terminals designed to operate on multipoint lines (those connected to multiple terminals) have special control units or "stunt boxes" (a telegraphy term) that provide this capability, and operate in one of two modes: control or text. When in control mode, a terminal interprets received characters as special control codes; when in text mode, a terminal treats most characters as normal textual data.

In the case of input, the computer sends one or more control characters putting all terminals in the control mode and then, using polling characters, invites a specific terminal to send any messages it may have. If a polled terminal is ready and has a message to send, it reverts to text mode to transmit the message. Other terminals on the same line remain dormant. A special code or sequence of codes indicates the end of message (EOM) and returns the terminal to control mode. If a polled terminal is ready but does not have a message to send, it usually responds with a code indicating its ready status. If the computer does not receive any response (a text message or status code) within a brief time, then the terminal "times out" and is considered not ready (for example, is off-line).

Output from the computer to the terminal is controlled in a similar fashion. The computer first *addresses* (attempts to select) the receiving terminal by means of a control sequence or, in telegraphy terms, a *call directing code* (CDC). If the terminal is ready to receive, it responds to the computer with an *acknowledge* character, and goes into text mode. The computer then transmits the output message.

A final elaboration on the line control methods is concerned with error correction. With some types of terminals, transmission can be error checked as discussed earlier. Following input of a message from terminal to computer, the computer may respond with control codes indicating that transmission was correct and that the terminal should go ahead and transmit the next message. Or, if an error has been detected, the computer may respond with control codes to request retransmission of the message. For some terminals, retransmission requires manual intervention; for others, it is performed automatically.

For output from the computer to a terminal, a similar process is used. The terminal performs error detection, and, following receipt of a message, sends a control sequence to the computer indicating either correct transmission or detection of an error. In the former case, the computer will start sending the next message; in the latter, the computer usually retransmits the same message.

* * * * *

This section has introduced some of the more important characteristics of commonly used communications equipment. More detailed information may be found in the technical reference manuals listed in the bibliography.

Since programming techniques vary considerably according to specific equipment configuration and processing requirements, it is difficult to give a concise description of general TP programming techniques. This section describes some programming functions that have unique characteristics in TP systems:

- allocation and scheduling
- buffering
- queuing
- message control
- message processing.

ALLOCATION AND SCHEDULING

How the total resources (processing time, main storage, I/O paths, etc.) of a TP system are apportioned is termed *allocation.* When these resources are apportioned is termed *scheduling.* In early systems, the only allocation necessary was the assignment of storage areas for program routines, constants, working storage, and I/O areas. Such allocation was planned ahead of time by the programmer and then specified at the time a program was assembled. This allocation is termed *static* since, once specified, it cannot be changed without reassembling (or recompiling) the program. In later systems, the allocation of storage was deferred until the time the program was loaded. This enabled separately compiled programs to share common data and I/O areas, but once the program was loaded, no further allocation was possible. More recently, responsibility for the allocation of portions of storage and the assignment of I/O units was undertaken by operating systems, some of which have been extended to the point where they now control all storage, I/O units, and data channels. Not only are more resources allocated in a more flexible manner, but, wherever possible, allocation is deferred until the resources are actually needed.

In a similar manner, jobs that once were manually scheduled for execution later were scheduled by the operating system, but only one job was initiated at a time. Once started, a job ran to completion or until an error was detected or a time allotment expired. The next advance in the evolution of job scheduling techniques was initiation of several jobs concurrently and dynamic interleaving of their executions. This interleaved execution of several programs is termed *multiprogramming*, while the sharing of computer time is quite naturally called *time-sharing.*

Thus, not only has the range of system resources allocable been increased, so that now even central processing unit time can be allocated; the time range over which they can be allocated has been increased as well, so that some resources now can be allocated during program execution.

There are several reasons for this increased generality in scheduling and allocating resources:

- Efficiency is increased since resources that might otherwise be idle can be used by other tasks.
- Flexibility is improved since the system can readily adjust to varying demands for resources.
- Growth potential is enhanced since added storage or other facilities can be readily accommodated without extensive reprogramming of the scheduling and allocation routines.
- Response rates are improved since tasks can progress in in parallel rather than serially.

These factors, fundamental to any computer system, are even more vital to a TP system in which the demands for system resources vary dynamically and unpredictably. Designing a good allocation and scheduling system for the TP environment proves to be an extremely difficult undertaking. Some of the factors that must be considered include:

1. Which responsibilities are reserved for operating personnel, and which are handled automatically by the system? The man-machine tradeoffs are necessary to permit human intervention and control and yet not hinder the trivial administrative functions best performed automatically by the system. Too much human intervention leads to inefficiency and human error; too little human intervention leads to loss of control and of adaptability to changes in processing priorities.
2. Which allocations are static and which are dynamic? Some resources must be reserved for use by the operating system itself. In addition, it is often con-

venient to dedicate certain system units to specialized functions. On the other hand, some units must be statically assigned to a pool of resources from which the system may dynamically draw to satisfy processing needs.

3. How should resources be divided into useful units for purposes of allocation? The larger the number of units, the more flexibly and efficiently they may be utilized. Unfortunately, however, the greater attention that must be devoted to scheduling and accounting for the resource units may far outweigh the improved utilization of the resource itself. For example, core storage is often partitioned into a number of units, which are allocated only in whole multiples. The larger these units, the greater the possible "trim loss" (the portion of the unit that is unused, where only a portion of a unit is needed). The amount of trim loss must be weighed against the additional table space and supervisor overhead that would be entailed if storage were partitioned into more, but smaller, allocation units.
4. Finally, what is the proper relationship between allocation and scheduling? That is, should scheduling be a function of resource allocation, or should it be the other way around? There is no one correct answer to this question; in practice, the distinction becomes blurred. To achieve the theoretical maximum resource utilization would require almost continuous reallocation; the supervisor time required for this is prohibitive. The designer of a programming system must discover the optimum balance between efficient utilization and minimal supervisor time devoted to allocation.

The treatment of allocation-scheduling interaction has a profound influence on the power and efficiency of any modern operating system. Some allocation and scheduling functions are peculiar to the TP environment, and it would unnecessarily burden a general purpose operating system to accommodate them as a standard capability. As a consequence, specialized modules are often available to encompass those additional functions. The most important of these functions are the buffering, queuing, and terminal control functions which are discussed below in more detail.

BUFFERING

A *buffer* is a storage area that temporarily stores data during transmission from one device to another. Buffers are used to compensate for differences either in data rates or in times of occurrence of related events. An example of the first use occurs in a data entry application in which incoming characters are collected in core storage buffers. Often, as each buffer becomes filled with data, the data is written from this buffer to another buffer area in a disk file. The core buffer compensates for the difference in data rates between a communications terminal (typically 15 characters per second) and a computer data channel (typically over 150,000 characters per second).

An example of using a buffer to compensate for time differences occurs frequently in message switching applications. A message is received and collected in a disk buffer area until the destination terminal becomes available. The buffer is used to compensate for the difference in time between receiving a message and forwarding it to the destination terminal.

The programming techniques used to assign and control allocation of buffer storage, called *buffering*, are a critical factor in the design of TP systems since there are vast differences both in data rates and in the time intervals between events. As mentioned earlier, a third complicating factor is that in many TP applications (message switching being a prime example), both data volumes and message lengths may vary widely and at unpredictable times.

Buffer Allocation

Buffers generally must be maintained on several different storage devices, and a system design objective is to efficiently utilize each of these. With core and disk buffers the following questions arise: what should be retained in core, what should be moved to the disk, and when should this be done? If core buffers are not made available for reuse promptly, there may be no place left for arriving input data. Loss of data is possible in a TP system since much simultaneous data input may be in process at any instant and the program does not have the same close control over terminal input as it does over input from tape or disk units. Other consequences of having insufficient core buffers or of not making them available promptly are a reduction in system operating efficiency and a slowing of terminal responses while waiting for the buffer congestion to clear.

In many conventional computer systems, buffering is used to obtain overlap of I/O operations with computing. In the simplest case, data is moved from a work area to an output buffer; then data is moved from an input buffer to the work area; and, finally, while the computer is processing the record in the work area, data channels are concurrently transmitting the output buffer and refilling the input buffer. The amount of read/write/compute overlap depends on the balance between these three concurrent operations. Elaborations of this approach involve using multiple buffers, performing the processing in the buffer rather than in a work area, and using a group of buffers (called a *buffer pool*) to supply buffer areas dynamically rather than having fixed buffer assignments. Each of these techniques provides for greater flexibility and more efficient utilization of buffer storage.

Buffer pools have an added advantage in TP systems. Because record sizes (i.e., message sizes) are highly variable, it would be wasteful of storage space always to assign for each message, regardless of length, a single buffer area large enough to hold the largest expected message. Instead, each message is contained in a series of buffers that are allocated dynamically as needed. Requiring the buffers containing the segments of a message to be in contiguous storage locations would unduly restrict buffer utilization. Each buffer is therefore allocated without regard for the location of previously allocated buffers, but is linked to the next buffer by *chaining*: each buffer contains the storage address of the buffer containing the next segment of the message. (Some convention is used to identify the last buffer in a chain; often, a zero chaining address signifies this.) Thus, although the segments of a message are contained in buffers dispersed throughout a buffer pool, a computer or channel program can access the entire message by progressing through the chain sequentially. When the message contained in the buffer chain has been processed, the buffers are returned to the chain of available buffers. This is easy to do since only the first of the newly available buffers need be chained to the chain of available buffers; the remaining buffers are, by their association with the first buffer, automatically returned.

Buffering Considerations

The size of the buffer pool and the size of the individual buffers depend on a number of factors, including the average and peak amount of input/output data, the ratio of input volume to output volume, the data rates of the TP equipment, the amount of storage available, and the form of allocation used with secondary storage units.

The program must dynamically adjust its operation to meet unpredictable changes in the operating environment. To overdesign a program to handle the worst possible case as the typical situation leads to excessive waste of expensive system resources. Conversely, to design a program without recognition of every infrequent, yet possible, situation is equally undesirable. Programs must be designed to efficiently handle the average case and yet manage to cope with the worst situations. Some parameters must be changed dynamically; others must be established at system generation time to "tune" the system to the typical load profile of the specific TP environment.

QUEUING

A *queue* is a waiting line, and the term *queuing* is used to denote the programming techniques employed to control transactions awaiting servicing by some computer facility. Queues are a consequence of items arriving for service at an unpredictable rate. If each item is serviced before the next item arrives, then no queue develops. Conversely, if the arrival rate continually exceeds the servicing time, then an infinitely long queue develops. The most common queuing situation is that in which, although the average arrival rate is less than the service rate, intermittent surges in the arrival rate cause a queue to form.

Like buffering, queuing is a function which, though present in some conventional computing systems, assumes far greater importance in the TP environment. In some conventional systems, especially those with both multiprogramming and direct access storage devices, queues may develop when several programs make requests for data faster than the device can access the data. A queue of file requests is formed, and some form of queue management is used to determine which item in the queue is to be serviced next. A number of different techniques are possible.

Sequence Handling

Probably the most common queuing technique is to service the queued items in the sequence that they arrive. This method, sometimes called FIFO (first in, first out), is certainly the simplest to implement. However, other approaches may be more efficient. For example, some situations may require a LIFO (last in, first out) rule. This approach, often called a "pushdown" or "stack" in computer terminology, is frequently used in performing arithmetic operations. In the case of a disk storage request queue, access times might be reduced if the current position of the access arm is noted and the queue is scanned for the item requiring the shortest motion of the access mechanism. In this case, safeguards must be established to prevent some item in the queue from being delayed an intolerable time because it is continually bypassed in favor of other items closer to the current position of the access mechanism. Other methods of queue management would be to assign priorities to the queued items depending not on the servicing device, as in the previous example, but on the resources needed to process the queued item, or to give favor to the oldest items in the queue, or to give output priority over input operations.

The purpose of the foregoing discussion is to illustrate that although the concept of queuing is simple, the techniques used to process queues can have a significant influence on overall system performance. This is especially true of TP systems, in which queues are very common, because continually varying demands are being made for a number of facilities and because unpredictable utilization of communication lines inevitably causes temporary queuing of output messages.

Now, whereas buffering is frequently accommodated in primary storage, it is more common to relegate queues to secondary storage devices. Control information containing enough data for queue management is retained in primary storage. The queues then can be rapidly scanned to determine the next item to be serviced. When ready for servicing, the item can be retrieved from secondary storage. As with

buffering, the TP control program must provide for assigning and reclaiming storage areas allocated to queuing functions. In addition, provision must be made to recognize when queues are building up to critical lengths and to modify normal program execution to give top priority to reducing the backlog of queued items to a safer level.

Problems in Queuing

Queuing in the TP environment creates some special problems not encountered in conventional computer systems. One problem is that a message may belong to more than one queue at the same time. Consider the case of a multiple-addressed message in a store-and-forward switching application. It would be wasteful to duplicate the entire message in the queue for each destination. Instead, only one copy of the message is queued, but control entries referencing the message are put in the pertinent output queue control areas. The same message text can then be directed, at different times, to the various destinations.

Another queuing facility necessary in many TP systems also can be illustrated by an example from the message switching application. Suppose a user has a very important message to send, and the system is loaded to the level at which transmission delays of perhaps one-half hour are occurring. The user desires to have the system send this message to its destination ahead of other items queued for the same destination. He can do this by inserting a priority code in the message header. The queue management routine recognizes this code and gives the message priority treatment when scanning the output queue. An elaboration of this approach is to have a number of levels of priority, and, in fact, most TP systems do provide for multiple priority levels.

In summary, these examples illustrate that TP systems must have queuing capabilities far beyond those found in conventional systems. It is seen that the flexibility of the queuing techniques employed can enhance the use of the system. In addition, the efficiency and modularity of the queuing techniques can have a substantial impact on overall system efficiency.

MESSAGE CONTROL

This section considers the programming aspects of the TP equipment described in the section, *Teleprocessing Equipment Characteristics*. It assumes that all communication channels are half-duplex and that only terminals of the same type are connected to any one line.

Once the line configuration is determined, information regarding the detailed characteristics of different types of terminal units must be associated with specific lines. Each device has an I/O module containing model channel programs that are used when READ/WRITE routines are invoked from macro instructions embedded in the user problem program. In addition, associations between lines and control units must be established. This kind of TP configuration information is established at the time a system is initially generated and need be modified only when the configuration is changed.

Further information is provided through declarative macro instructions contained in a user program. These determine, at assembly time, the allocation of core storage space for each type of communication line to be used. For example, such items as buffer pool size, buffer size, and type of buffering are indicated together with information relating which lines are to share common control areas. Another kind of information, also provided through declarations, defines the structure of the lists used for such functions as polling, addressing, calling, and answering. For example, a polling list for each line indicates the sequence in which the terminals on that line are to be polled. Considerable flexibility is usually possible. For instance, the same polling code can appear several times in the list to cause the corresponding terminal to be polled more frequently than the others.

Information Handling

Once presented with the necessary declarative information, the TP control program accesses terminal units in such a manner that the programmer is relatively unaware of the details of the terminal operation. For example, upon receiving control from a READ macro instruction executed in a user program, the control program will:

1. Send a control character placing all terminals on the line in control mode;
2. Send the polling characters for the designated terminal;
3. Wait for a response indicating whether the polled terminal has input ready to send;
4. Receive the message, test each block of the message for errors, and request retransmission of any block containing errors;
5. Upon completion of input from the polled terminal, poll another terminal on the line.

This example is considerably simplified. In reality, the program performs many other functions, and numerous options are available to the problem programmer.

The previous section outlined the programming techniques used to allocate buffer storage and to read and write messages. If queuing is desired or if records are to be referenced at the logical level via GET/PUT-type access, further facilities must be added.

Message Routing

Routing of messages in accordance with data contained in their headers is a function common to many applications. Here the programmer must establish *terminal tables* relating each symbolic terminal identifier to the terminal it represents. This simplest case must usually be extended to accommodate *group addressing* and *distribution lists*, by which several terminals can be represented by the same symbolic identifier. For example, in the case of all those branch locations that provide a particular service, it is more convenient and less error prone to reference the entire group by a single symbolic name.

In many applications, messages are routed not to another terminal but to one of several processing programs. Here the TP routing data is used as a transaction code to cause control to pass to the proper processing routines. The programmer employs essentially the same technique used for message routing, the only difference being that the destination location is not a terminal address but the symbolic entry point of a routine that is to receive control when the message is ready for processing.

An example is an inquiry system: inquiry messages arrive in the transmission code of the type of terminal from which they were entered, and are usually translated to a character code (e.g., EBCDIC) suitable for processing. Similarly, reply messages generated in the processing code are converted to the transmission code of the destination terminal. Code translation is usually performed by a table look-up technique in which, for each character in succession, the original code representation of a character is applied as an index value to the beginning of the translation table, to find the corresponding new code representation.

In the IBM System/360, this translation from one character code to another is greatly facilitated by an instruction designed for this purpose. If new terminal types are added or different translations are desired, it is easy to accommodate them by adding another table or by modifying an existing table. The programmer need not be concerned with the control codes used by each terminal, since, as explained earlier, terminal control and error checking are automatically performed by message control routines.

Message Processing

Although the message control facilities described above encompass a number of functions common to a wide range of TP applications, they cannot accommodate all requirements of all applications for which they are used. Even if these requirements could be adequately defined, a general purpose program capable of fulfilling them would require a machine configuration far beyond that required for any specific application. Therefore, the key to generalizing TP programs is knowing the tradeoff between technical feasibility and general utility. Some functions must be provided in a basic programming package, additional functions may be provided in an extended package, and some functions of value to some users cannot be accommodated without penalizing the majority of users.

Message processing functions unique to a specific application are programmed by the user. In many applications, the programmer can, through use of message control routines, construct an interface to the communications environment such that the message processing routines may be designed, coded, and tested without consideration of the fact that they will be executed in a TP system. Since the application programmer is in effect insulated from the environment, he does not need a detailed knowledge of TP equipment, programming methods, and testing techniques. This fact can result in lowered development schedule time and costs.

* * * * *

This section has introduced some of the programming techniques that play important roles in TP systems. The best sources of additional information are to be found in the reference manuals describing specific telecommunications access methods. A list of these is contained in the bibliography of this publication.

IBM programming support for TP systems is provided in the form of access methods under the Data Management portion of both the S/360 Operating System and the S/360 Disk Operating System.

The principal function of a telecommunications access method is to control the transmission of information between a computer and remote TP equipment in much the same manner as other access methods support other types of input/output equipment. The programmer designs, writes, and tests his application routines in the usual manner, and he performs input/output operations by means of macro instructions supplied by the access method. The user may also develop his own macro instructions to replace, or augment those supplied by the access method.

There are two levels of telecommunications access methods: one, at the READ/WRITE level, is called Basic (BTAM); the other, at the GET/PUT level, is called Queued (QTAM).

BTAM is designed to provide the basic modules for constructing a TP program, including routines for controlling a variety of terminal units, communications lines, and transmission control units. With a minimum of system overhead, it not only provides the basic tools to build a sophisticated system but also is modified easily to support special configurations not supported by other programming packages. BTAM provides the basic capabilities to:

- Poll terminals and receive messages,
- Address terminals and send messages,
- Dynamically chain input buffers,
- Dial and answer,
- Detect and correct errors,
- Write output buffer chains,
- Perform code translation.

QTAM has two characteristics that distinguish it from other access methods:

- Scheduling and allocation functions are performed by a separate control program within the operating system.
- Operations to control and process communications data are specified by a unique macro language.

QTAM includes the BTAM capabilities mentioned above and, in addition, provides extensive queuing facilities. QTAM is directly applicable without modification to a number of common TP applications, for example, data collection and message switching. QTAM provides the basic capabilities for:

- Controlled and automatic terminal polling and message input,
- Controlled and automatic terminal addressing and message output,
- Input/output buffering,
- Error detection and checking,
- Message queuing, logging, and routing,
- Code translation.

Figure 10 shows the various types of terminals supported by BTAM and QTAM.

BTAM

As already stated, BTAM controls terminal input/output operations initiated by READ and WRITE macro instructions issued in the user's problem program. The primary purpose of BTAM is to provide input/output support at the message level under the operating system. As a consequence there are really two BTAMs: one for the full Operating System (OS) and one for the Disk Operating System (DOS). These two versions of BTAM have a similar appearance to to the user. The principal external differences are:

- Audio response equipment (IBM 7770 and 7772 Audio Response Units) is supported only under DOS.
- Operating systems incorporating BTAM have a minimum memory size of 32K bytes for DOS and 64K bytes for OS.

The principal internal difference is that buffers are allocated to accept a maximum size message under DOS, while READ/WRITE buffers are allocated dynamically under OS.

SUPPORTED DEVICES	OS		DOS	
	BTAM	QTAM	BTAM	QTAM
Start-Stop Device Support				
IBM 1030 Data Collection System	X	X	X	X
IBM 1050 Data Communication System	X	X	X	X
IBM 1060 Data Communications System	X	X	X	X
IBM 2260 - 2848 Display Complex (Remote)	X	X	X	X
IBM 2260 - 2848 Display Complex (Local)			X	
IBM 2740 Communications Terminal	X	X	X	X
IBM 7770 Audio Response Unit			X	X
IBM 7772 Audio Response Unit			X	X
AT&T 83B3 Selective Calling Stations	X	X	X	X
Western Union Plan 115A Outstations	X	X	X	X
AT&T Model 33/35 Teletypewriter Exchange Terminal	X	X	X	X
Binary Synchronous Communication Support				
IBM System/360 to IBM System/360	X		X	
IBM System/360 to IBM 1130	X		X	
IBM System/360 to IBM 2780	X		X	

Figure 10. BTAM and QTAM Device Support

The use of BTAM is recommended for those systems having one or more of the following characteristics:

- A small number (1-4) of communication lines.
- A requirement for a specialized TP control program.

Storage Requirements

The primary storage requirements of BTAM depend on the particular configuration of terminal equipment, the buffering requirements, and the macro instructions used. A typical configuration might consist of 4 lines, each with four 1050 terminals, attached to an IBM System/360 by means of an IBM 2701 TCU. Assuming one 140-byte buffer for each line, and polling and addressing lists each having one 3-byte entry for each of the 16 terminals, the total core storage needed is in the range of 3,000-4,000 bytes for BTAM under OS and about 4,500-5,500 bytes for BTAM under DOS.

Description

BTAM facilities must be incorporated into the operating system at system generation time.

Three facilities are made available through the macro generation capabilities of any OS/360 or DOS/360 assembler language translator. During assembly of a problem program, macro instructions coded by the user are expanded into:

- Linkages to the executable BTAM routines,
- Tables defining the lines, terminals, and options to be used,
- Buffer areas.

Initial communication between a user problem program and BTAM is established upon execution of an OPEN macro in the problem program. This also establishes communication between BTAM and I/O supervisor.

After an OPEN is executed, a message may be sent or received by the simple execution of a WRITE or READ macro instruction, causing a branch and a link transfer of control to the BTAM Read/Write routine. This routine first builds a channel program to perform the operation, then passes control to the I/O supervisor which starts executing the channel program just developed. At this point, control passes back to the problem program, and the channel program is executed concurrently with the problem program.

An important feature of BTAM is the ability to repeatedly restart channel programs in response to conditions occurring on the communications line. Thus, a single READ macro instruction can cause successive polling of a number of terminals without any intervening direction from the problem program, which is notified only when a message has been read. Similarly a single WRITE can signal a number of terminals to prepare to receive.

Another feature of OS BTAM is dynamic buffering, by which BTAM can interrupt the problem program to secure additional input buffer areas as needed.

BTAM Facilities

BTAM is easy to use. The programmer need only define control blocks and terminal lists (for polling and addressing) before performing the following simple functions:

- Open
- Read/Write messages
- REQBUF/RELBUF – to request and release buffers

BTAM is not a complete TP system. The BTAM user must:

- Comprehend its basic capabilities and limitations;
- Have a reasonable understanding of terminal and TCU equipment;
- Provide programs, initiate operations, test for exceptional conditions, and make decisions on control flow and the disposition of data;
- Provide routines for any necessary scheduling and allocation functions.

BTAM is an ideal tool for constructing TP programs. Since it is a general purpose interface for input/output with TP equipment, it can be used as a component in the development of more sophisticated TP systems. It can also be employed in supporting a few TP lines on a computer configuration with limited available memory space.

Installations considering extensive augmentation of BTAM functions should carefully weigh this task against the alternative of using the more comprehensive QTAM program.

QTAM

QTAM includes all the facilities previously described for BTAM. In fact, QTAM branches to a variation of BTAM for dynamic generation of channel programs to send and receive messages. However, QTAM is much more than an extension of BTAM capabilities. It not only controls message transmission between remote terminals and the central computer but also controls message queuing on a direct access secondary storage device. QTAM is a control program in its own right, and it provides synchronous operation for all programming based on completion of events, availability of resources, and processing priorities.

Storage Requirements

QTAM operates under both the Disk Operating System and the full Operating System. Although operating systems incorporating QTAM have a minimum storage size of 32K bytes for DOS and 64K bytes for OS, representative systems generally require 64K and 128K bytes respectively. The space taken by QTAM varies according to user-selected options, and there is no simple rule to provide a realistic storage estimate. However, the following approximations may be useful for obtaining a general awareness of QTAM storage requirements.

The subroutines for message control occupy about 8K-12K bytes for primary storage. Many of the macro instructions generate in-line linkages to functional subroutines. If the same macro is used more than once, space is needed for the linkage, but only one copy of the subroutine is provided. A typical macro might generate 10–15 bytes of in-line linkage, while the subroutine might occupy 100 or so bytes. Space is also needed for control blocks, tables, polling/addressing lists, and channel programs and related areas. In addition, since QTAM performs message processing functions, as well as message control functions, storage is needed for these processing routines.

The following example indicates the approximate storage needed by QTAM to process a typical large TP configuration. Assume 15 lines each with 6 IBM 1050 terminals and another 15 lines each with 6 AT&T 83B3 stations or Western Union Plan 115A stations. Assume further that two 100-character buffers are allocated for each line and that message processing has three representative routines operating on three process queues held on one direct access device. The QTAM storage requirements for this system would be in the range of 25,000 - 35,000 bytes.

Description

The overall organization of QTAM is shown in Figure 11. As can be seen, data sent by terminals is collected in core buffers and then moved to disk queues for later message processing. Results are then moved to another queue to await output, again via a core buffer, to a destination terminal.

This message flow, shown in more detail in Figure 12, can be considered in the following seven steps:

1. The input message, consisting of message header and text, is prepared at a remote terminal. The header portion normally contains codes denoting source and destination terminals as well as a message sequence number and priority information. When the source terminal is polled, the message is sent to the computer.
2. The message enters the computer and is placed in user-defined, fixed-length buffers. As many buffers are allocated as are necessary to contain the message. QTAM affixes a header prefix containing information for control and queuing purposes. Each of the remaining buffers has a text prefix, also containing control and queuing information, that precedes the text data. As soon as each buffer is filled, QTAM performs such user-selected functions as code translation, routing, time and date stamping, and sequence checking.

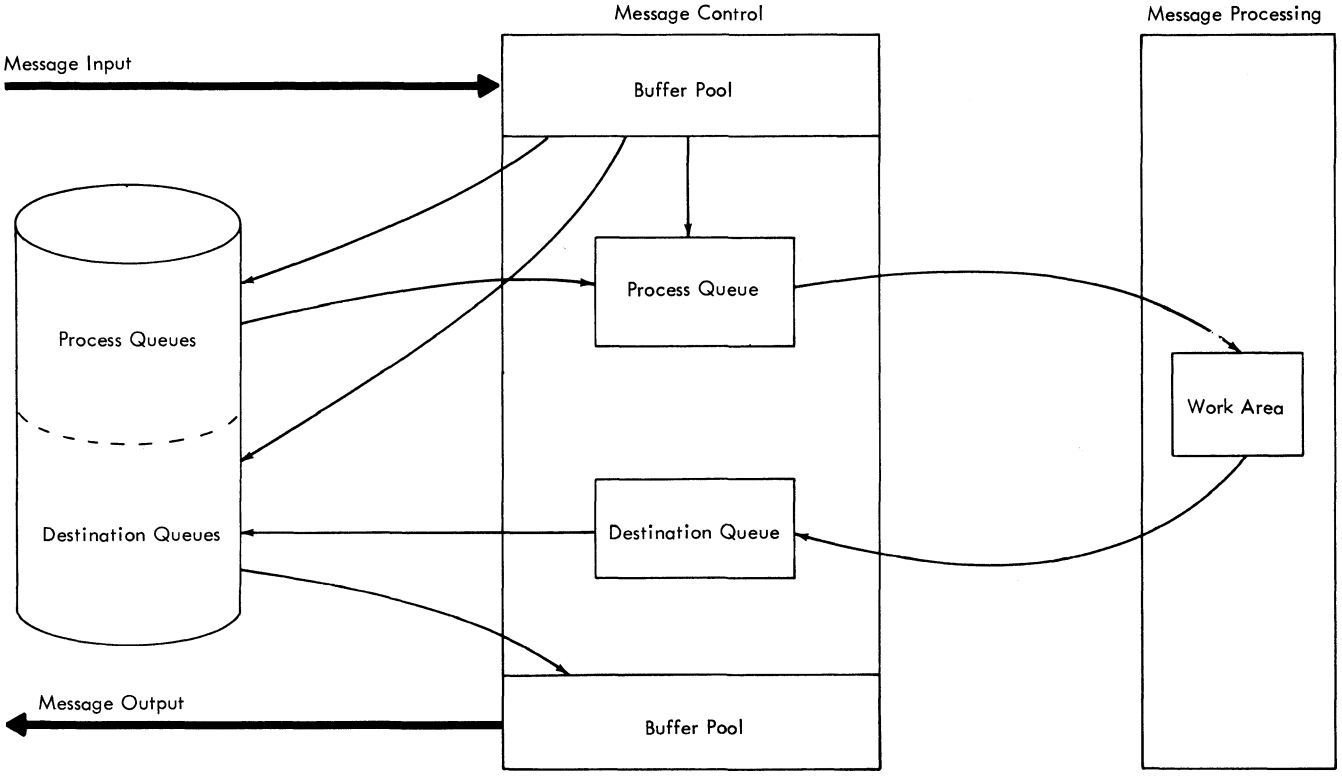


Figure 11. QTAM Organization

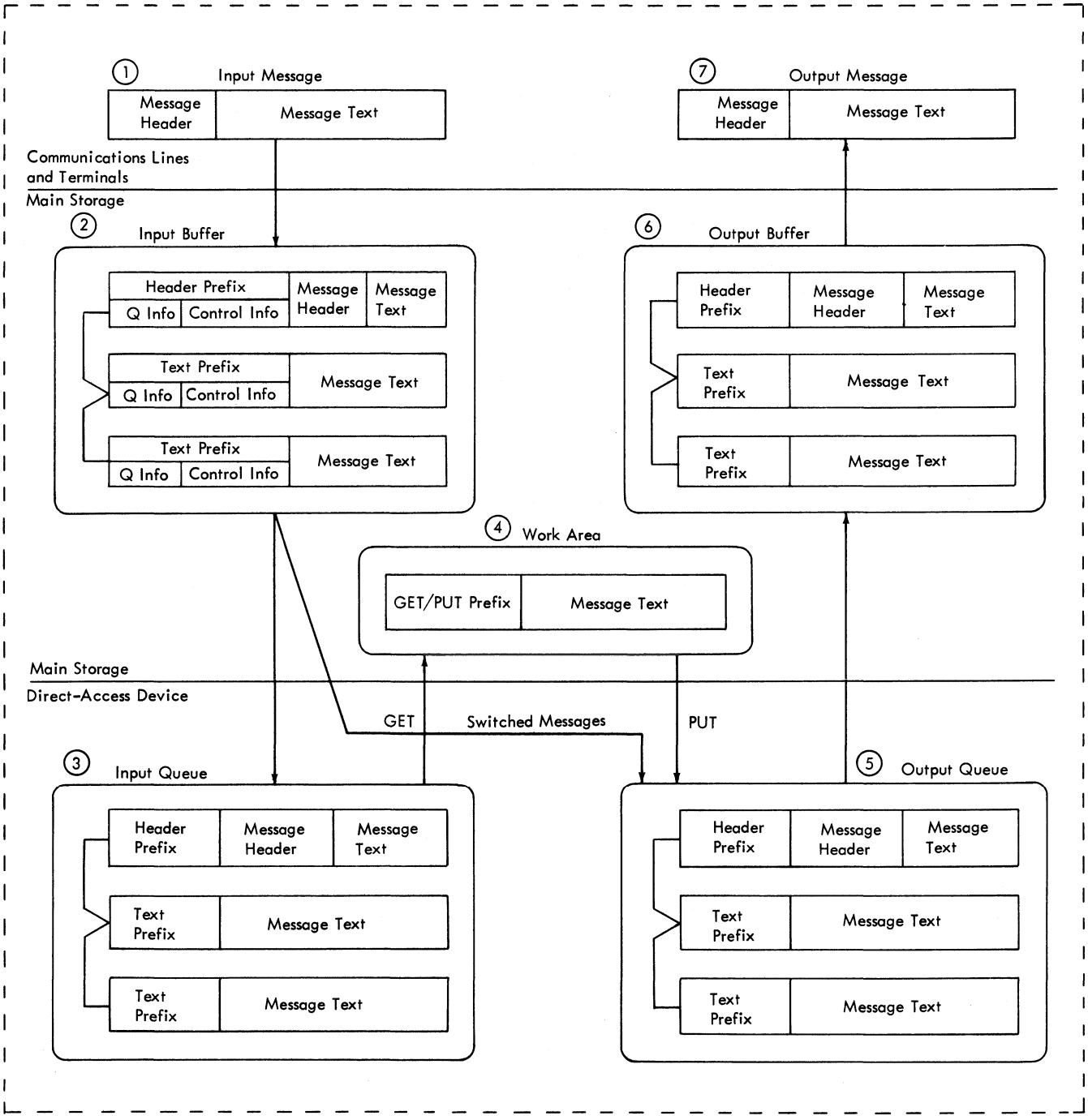


Figure 12. QTAM Message Flow

3. If the message requires additional processing, each segment is sent to a process queue (input queue), which may be either in main storage or on a direct access storage device.
4. User-written routines can GET messages, segments, or records from this queue and process them. Following this, the user can PUT the message into an output (destination) queue.
5. If no processing (steps 3 and 4 above) is required, the message may proceed directly to a destination queue.
6. Message segments are retrieved from the output queue on a FIFO basis within priority groups.
7. Message segments are stripped of header and text prefixes and sent to the destination terminal as one continuous message.

Facilities

Since QTAM employs many of the programming techniques described earlier (under *Teleprocessing Programming Techniques*), the remainder of this section, briefly describing some QTAM facilities, is organized around the same topics used earlier: allocation and scheduling, buffering, queuing, message control, and message processing.

The initial allocation of the storage areas is static and is specified by the user, while allocation of individual segments within these areas is done dynamically. Normally, messages move to a disk queue while awaiting processing, but there are programmer options available to expedite messages by bypassing the disk phase.

QTAM scheduling is designed to optimize use of the communication lines. The scheduling is a function of resource allocation in contrast to the opposite approach of first allocating resources and then scheduling by events.

Buffering is accomplished by dynamic assignment of storage segments. To conserve storage space, buffers are emptied as quickly as possible. The same buffer pool is used both for terminal input/output and for holding messages awaiting service by message processing programs.

Queues are specified for both processing tasks and destination terminals. Queues are organized to minimize the arm motion of the disk access mechanism and are managed by means of queue control blocks residing in core memory and containing all information necessary to schedule and locate the associated queued items on the disk.

An enumeration of other QTAM functions in addition to the allocation and scheduling, buffering, and queuing functions just described gives some idea of its message control capabilities:

- Controls all message traffic between central computer and remote terminals;
- Performs such message editing functions as code translation and header analysis;
- Routes messages to destination terminals or to processing routines;
- Optionally logs all or certain messages;
- Performs numerous error detection and correction procedures including intercepting, rerouting, or cancelling of messages in error

The user-written message processing programs operate as one or more individual tasks and communicate with QTAM to initiate, activate, and terminate the QTAM message control program. In addition, GET/PUT logic is used for passing messages through the message queue, which is the main connection between a system message control task and a user message processing task.

SUMMARY

BTAM is a general purpose input/output interface; QTAM is a complete TP system in its own right. Without modification, QTAM can perform some applications, message switching for example, in their entirety; in other cases, it provides most processing functions that are common to a variety of TP applications. Its design accommodates a majority of the system applications, equipment characteristics, and programming techniques introduced in this manual.

More detailed information on BTAM and QTAM is contained in the reference manuals listed in the bibliography.

acknowledge . . . to respond to polling or addressing, or to receipt of a message.

acknowledgment . . . the act of sending a response to polling or addressing, or to receipt of a message; also, the character or character sequence comprising the response. An acknowledgment to polling or addressing may indicate the status of the terminal that sends it; an acknowledgment to a message may indicate whether it was received without error.

address (n.) . . . the coded representation of the destination of a message.

address (v.) . . . to condition a terminal for receiving data.

allocate . . . to assign a system resource to a specific function.

answering station . . . the station responding to a dialed call; opposite of originating station.

audio (a.) . . . within the range of frequencies which can be heard by the human ear (usually in the range 15-20,000 Hertz [cycles per second]).

Auto Answer . . . the facility of an answering station to automatically respond to a call.

Auto Call . . . the facility of an originating station to automatically initiate a call.

band . . . the range of frequencies between two defined frequencies.

bandwidth . . . the difference, expressed in Hertz (cycles per second), between the two limiting frequencies of a band.

batch processing . . . processing of data after a number of similar input items have been accumulated and grouped together; contrast with in-line processing.

bit . . . contraction of *binary digit.*

bit rate . . . the speed at which bits travel over a communication channel, usually expressed in bits per second.

buffer . . . a storage device used to compensate for a difference in the rate of flow of information, or the time of occurrence of events.

call directing code (CDC) . . . a code used to address a terminal (a Western Union term).

channel . . . a path for electrical data transmission between two or more stations; also called circuit (not synonymous with *data channel* in computer usage).

channel, duplex . . . a channel providing simultaneous transmission in both directions.

channel, half-duplex . . . a channel capable of transmission in both directions, but only one direction at a time.

channel, simplex . . . a channel which permits transmission in one direction only.

channel, voice-grade . . . a channel suitable for transmission of speech.

character . . . the actual or coded representation of a digit, letter, special symbol, or control function.

character, check . . . a character used for validity checking purposes.

character, control . . . a character used for control purposes.

character, graphic . . . a character used for printing or display.

checkpoint . . . a point in a computer program at which sufficient information can be stored to permit restart of processing from that point.

circuit . . . a connection between two or more points; usually, a physical, metallic path.

coaxial cable . . . a cable consisting of two concentric conductors insulated from each other.

code . . . a system of symbols and rules for their use in representing information; also, the coded representation of a character.

code unit . . . the number of bits used to represent a transmission character.

code level . . . the number of bits used to represent a data character.

communication . . . the transfer of information from one point to another.

communication common carrier . . . a company recognized by an appropriate regulatory agency as having a vested interest in furnishing communication services.

contention (n.) . . . the condition on a multipoint communication channel when two or more locations try to transmit at the same time.

contention (a.) . . . relating to a communication system in which contention can occur.

conversational . . . a mode of communication involving the alternate sending and receiving of data.

cyclic checking . . . a method of error control employing a weighted sum of transmitted bits.

data collection . . . the process of bringing data from one or more remote points to a central point.

Data Phone . . . a term used by AT&T to describe any of a family of data set devices.

data set . . . a device containing the electrical circuitry necessary to connect data processing equipment to a communication channel; also, called subset, Data Phone, modulator/demodulator, modem. (Not to be confused with the IBM System/360 Operating System term.)

data transmission . . . the sending of data from one place to another or from one part of a system to another.

demodulation . . . the process used to convert communication signals to a form compatible with data processing equipment.

dial exchange . . . a common carrier exchange in which all subscribers originate their calls by dialing.

display unit . . . a terminal device that presents data visually, usually by means of a cathode ray tube.

dynamic allocation . . . the technique of assigning storage areas during processing; contrast with *static allocation.*

EBCDIC . . . abbreviation for extended binary coded decimal interchange code.

end of address . . . control character(s) separating message address(es) from message text; often abbreviated EOA.

end of message . . . control character(s) denoting the end of a message; often abbreviated EOM.

end of transmission . . . control character(s) denoting the conclusion of data transmission; often abbreviated EOT. It is usually sent by an originating station to signify that it is finished with the communication line.

exchange service . . . a service permiting interconnection of two customers' telephones through the use of switching equipment.

FIFO . . . abbreviation for first-in, first-out queuing, in which items are removed from a queue in the same order as entered; contrast with LIFO.

free form . . . formatting of data fields by embedded delimiter characters rather than organizing of data in fixed-length fields.

group addressing . . . a technique for addressing a group of terminals by use of a single address.

hard copy . . . a machine-printed document, as opposed to visually displayed data.

header . . . initial portion of a message containing any information, control codes, etc., that is not a part of the text. Usually includes information for routing the message to its destination(s).

in-line processing . . . processing of input data in random order, without preliminary editing, sorting, or batching; contrast with batch processing.

in-plant system . . . a system confined to one plant locality.

interface . . . a shared common boundary between two systems or two devices; for example, a physical connection or a programming convention.

LIFO . . . abbreviation for last-in, first-out queuing, in which the next item to be removed is the most recently entered item in the queue. Also called "stack" or "push-down;" contrast with FIFO.

LRC . . . abbreviation for longitudinal redundancy checking method, in which parity is checked longitudinally along all the characters comprising a transmitted record.

mark state . . . state of a communication line corresponding to an on, closed, or logical one condition; contrast with *space state.*

message . . . a finite sequence of transmitted words and/or symbols.

message routing . . . the function of selecting the route, or alternate route, by which a message will proceed to its destination. Sometimes used to mean "message switching."

message switching . . . the technique of receiving a message, storing it until the proper outgoing circuit is available, and then retransmitting it. Also called "store and forward switching."

modulation . . . process used to convert signals from data processing equipment to a form compatible with communication facilities.

modem . . . contraction of modulator-demodulator (see *data set*).

multiple address message . . . a message which is to be delivered to more than one destination.

multiplexing . . . the interleaved or simultaneous transmission of two or more messages on a single channel during a given time interval.

multipoint line . . . a communication line interconnecting several stations.

multiprogramming . . . the interleaved (that is, concurrent) execution of two or more programs by a single computer.

narrow-band (a.) . . . denoting a communication channel capable of a transmission rate of up to 300 bits per second.

network . . . a series of points interconnected by communication channels.

network, leased line or private wire . . . a network reserved for the exclusive use of one customer.

off-line . . . pertaining to devices not in direct communication with a computer.

on-line . . . pertaining to devices in direct communication with a computer.

out-plant system . . . a system not confined to one plant or locality.

parity check . . . a test to determine whether the number of ones (or zeros) in an array of binary digits is odd or even.

point-to-point transmission . . . transmission of data between two points.

polling . . . a flexible, systematic, centrally controlled method, for permitting stations on a multipoint circuit to transmit without contending for the line.

priority indicators . . . groups of characters in the header of a message, specifying the order of transmission of messages over a communication channel.

queue . . . a group of items awaiting processing by some facility.

real-time processing . . . processing data rapidly enough to provide results useful in directly controlling a physical process or guiding a human user.

record . . . a group of related data items treated as a unit.

response . . . equivalent to acknowledgment (which see).

response time . . . the interval between completion of an input message and receipt of an output response.

restart . . . to return to a previous point in a program and resume operation from that point; often associated with a checkpoint.

shutdown . . . temporary termination of computer processing to be resumed at some later time.

space state . . . state of a communication line corresponding to an off, open, or logical zero condition; contrast with *mark state.*

startup . . . initiation of computer processing or resumption of it from a point of temporary termination.

start-stop mode . . . a mode of data transmission in which each character is delimited by special control bits denoting the beginning and end of the sequence of data bits representing the character; contrast with *synchronous mode.*

static allocation . . . the technique of assigning fixed storage areas prior to processing; contrast with *dynamic allocation.*

store and forward switching . . . see *message switching.*

stunt box . . . a device to control non-printing functions of a teletypewriter terminal.

subset . . . a modulation/demodulation device designed to provide compatibility of signals between data processing equipment and communication facilities. Also called modem.

synchronous mode . . . a mode of data transmission in which character synchronism is controlled by timing signals generated at the sending and receiving stations; contrast with *start-stop mode.*

tariff . . . the published rate for a particular approved commercial service of a common carrier; also, a list of services provided and requirements for their use.

telecommunication . . . communication by electromagnetic systems; often used interchangeably with communication.

TCU . . . abbreviation for transmission control unit.

Teleprinter . . . trade name used by Western Union to refer specifically to telegraph page printers.

Teletype . . . trademark of the Teletype Corporation.

Teletypewriter . . . trade name used by AT&T to refer specifically to telegraph page printers.

Teletypewriter Exchange Service (TWX) . . . a semi-automatic switching service provided by AT&T for interconnecting public teletypewriter subscribers.

Telex . . . an automatic switching service provided by Western Union for interconnecting teleprinter subscribers.

Telpak . . . a tariff offered by AT&T for leasing or broadband channels.

terminal unit . . . equipment on a communication channel that may be used for either input or output, or both.

text . . . that part of a message which contains the information to be conveyed; contrast with *header.*

tie-line . . . a leased communication channel or circuit.

time-share . . . to interleave the use of a device or system for two or more purposes.

transmission . . . the electrical transfer of information from one location to another.

transmission control unit . . . a unit to interface communication lines with a computer. Sometimes abbreviated as TCU.

turnaround time . . . the interval of time between submission of a job for computer processing and receipt of results; the time interval required to reverse the direction of transmission over a communication line.

TWX . . . abbreviation of Teletypewriter Exchange Service.

unattended operation . . . use of a terminal unit without an attending operator.

USASCII . . . United States of America Standard Code for Information Interchange.

voice grade line . . . a channel suitable for transmission of speech.

VRC . . . abbreviation for vertical redundancy checking method, in which parity is checked vertically across each character in a record.

voice-band (a.) . . . denoting a communication channel capable of a transmission state exceeding 300 bits per second; a channel suitable for transmission of speech.

WATS . . . abbreviation for AT&T's Wide Area Telephone Service, providing a special line on which the subscriber may make unlimited calls to certain zones on a direct distance dialing basis for a flat monthly charge.

wide-band (a.) . . . denoting a communication channel capable of a transmission rate greater than about 18,000 bits per second.

Word . . . in telegraphy, a word consists of six code combinations.

BIBLIOGRAPHY

Systems Reference Library Manuals

IBM 1030 Data Collection System (A24-3018)
IBM 1050 Data Communication System (A24-3020)
IBM 1060 Data Communication System (A24-3034)
IBM 2260 Display Unit (A27-2700)
IBM 2701 Data Adapter Unit (A22-6864)
IBM 2702 Transmission Control Unit (A22-6846)
IBM 2703 Transmission Control Unit (A27-2703)
IBM 7770 Audio Response Unit (A22-6800)
IBM 7772 Audio Response Concepts and Vocabulary (A22-6847)
IBM System/360 Disk Operating System BTAM (C30-5001)
IBM System/360 Disk Operating System, QTAM Message Control Program (C30-5004)
IBM System/360 Disk Operating System, QTAM Message Processing Program Services (C30-5003)
IBM System/360 Operating System, BTAM (C30-2004)
IBM System/360 Operating System, QTAM Message Control Program (C30-2005)
IBM System/360 Operating System, QTAM Message Processing Program Services (C30-2003)

Books

Desmonde, W. H.: Real-Time Data Processing Systems, Prentice-Hall (1964).
Head, R. T.: Real-Time Business Systems, Holt-Reinhart-Winston (1964).
Martin, J. T.: Design of Real-Time Computer Systems, Prentice-Hall (1967).
Martin, J. T.: Programming Real-Time Computer Systems, Prentice-Hall (1965).
——: Data Communications in Business, American Telephone & Telegraph Co., New York (1965).

READER'S COMMENT FORM

IBM System/360—Introduction to Teleprocessing
SRL

GC30-2007-1

- How did you use this publication?

  As a reference source ☐
  As a classroom text ☐
  As a self-study text ☐

- Based on your own experience, rate this publication . . .

As a reference source:	Very Good	Good	Fair	Poor	Very Poor
As a text:	Very Good	Good	Fair	Poor	Very Poor

- What is your occupation?

- We would appreciate your other comments; please give specific page and line references where appropriate. If you wish a reply, be sure to include your name and address.

- Thank you for your cooperation. No postage necessary if mailed in the U. S. A.

Staple

GC30-2007-1

YOUR COMMENTS, PLEASE . . .

This publication is one of a series that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, help us produce better publications for your use. Each reply is carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in using your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Fold Fold

FIRST CLASS
PERMIT NO. 569
RESEARCH TRIANGLE PARK
NORTH CAROLINA

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
P. O. Box 12275
Research Triangle Park
North Carolina 27709

Attention: Programming Documentation, Dept. 844

Fold Fold

Cut Along Line

Printed in U.S.A. GC30-2007-1

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

Additional Comments:

READER'S COMMENT FORM

IBM System/360—Introduction to Teleprocessing
SRL

GC30-2007-1

- How did you use this publication?

  As a reference source ☐
  As a classroom text ☐
  As a self-study text ☐

- Based on your own experience, rate this publication . . .

As a reference source:	Very Good	Good	Fair	Poor	Very Poor
As a text:	Very Good	Good	Fair	Poor	Very Poor

- What is your occupation?

- We would appreciate your other comments; please give specific page and line references where appropriate. If you wish a reply, be sure to include your name and address.

- Thank you for your cooperation. No postage necessary if mailed in the U. S. A.

Staple

GC30-2007-1

YOUR COMMENTS, PLEASE . . .

This publication is one of a series that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, help us produce better publications for your use. Each reply is carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in using your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Fold Fold

FIRST CLASS
PERMIT NO. 569
RESEARCH TRIANGLE PARK
NORTH CAROLINA

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
P. O. Box 12275
Research Triangle Park
North Carolina 27709

Attention: Programming Documentation, Dept. 844

Fold Fold

Cut Along Line

Printed in U.S.A. GC30-2007-1

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

Additional Comments:

Printed in U.S.A. GC30-2007-1

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]