File No. S360-30
Form C28-6553-2

Systems Reference Library

IBM Operating System/360

Telecommunications:

Preliminary Specifications

This publication contains preliminary information on how to apply and use the IBM System/360 Operating System for remote message processing and how to use the control program for performing the input/output operations of a data communications system.

Guidance is provided for problem programming within the system. Descriptions of applicable macro-instructions, suggesting how, when, and where to use them, are also included.

PREFACE

This publication contains preliminary information on planning and implementing a system for remote message handling and/or processing under the System/360 Operating System.

Completion of a basic course in programming Computing System/360 is a prerequisite to using this publication. It is also required that the reader be familiar with the overall programming concepts and terminology introduced in the following publications:

IBM System/360 Operating System: Introduction, Form C28-6534

IBM System/360 Operating System: Concepts and Facilities, Form C28-6535

IBM System/360 Operating System: Job Control Language, Form C28-6539

IBM System/360 Operating System: Assembler Language, Form C28-6514

IBM System/360 Operating System: Data Management, Form C28-6537

IBM System/360 Operating System: Control Program Services, Form C28-6541

It is suggested that the reader be familiar with the following additional publications:

IBM 2701 Data Adapter Unit, Principles of Operation, Form A22-6864

IBM 2702 Transmission Control, Form A22-6846

Minor Revision (December 1965)

This publication is a minor revision of IBM Operating System/360, Telecommunications, Form C28-6553-1, and incorporates Technical Newsletter N27-1233 into the publication. Information in this publication is still classified as preliminary.

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print chain.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.

A form for reader's comments appears at the back of this publication. Address any additional comments concerning the contents of this publication to: IBM Corporation, Programming Documentation, Department 840, P.O. Box 9361, Raleigh, North Carolina 27603

CONTENTS (continued)

ILLUSTRATIONS

TABLES

System/360 is designed to provide a variety of programming and data processing services that can be requested and used from a local or a remote location, or both. Any of the services provided by System/360 Operating System for local use are also available for use from geographically dispersed locations via telecommunications networks. In addition, the operating system can also provide services that are peculiar to telecommunications, such as message switching and data collection. This publication describes how the operating system can be extended to remote locations in order to:

- Decrease the response time to individual requests for services.
- Share the extensive programming and data processing facilities of System/360 with other locations.
- Make the system an integral and more natural part of the activity it supports.

Depending on the specific application, the operating system can be used to process messages from remote locations exclusively, process messages and execute local jobs concurrently, or alternate between the processing of messages and the executing of local jobs.

To extend the operating system for telecommunications applications, a queued and a basic telecommunications access method are available for controlling the sending and receiving of messages to and from remote terminals. The queued telecommunications access method (QTAM) can be used in a variety of applications ranging from a message switching application to a high-volume inquiry or transaction processing application such as a banking or airline reservation system. The basic telecommunications access method (BTAM) is designed for limited applications, which do not require extensive message control facilities. Either access method (QTAM or BTAM) can be used to receive messages from and send messages to the following terminal devices:

- IBM 1030 Data Collection System.
- IBM 1050 Data Communications System (leased line or dial service).
- IBM 1060 Data Communications System.
- AT&T 83B2 Selective Calling Stations.
- Western Union Plan 115A Outstations.
- Common Carrier (8-level code) TWX Stations; e.g., AT&T Model 33 or 35 Teletypewriter Terminal (dial service).

These devices can be attached through a control unit (the IBM 2701 Data Adapter Unit or the IBM 2702 Transmission Control) to the multiplex channel of the System/360 Central Processing Unit (CPU) as illustrated in Figure 1.

QUEUED TELECOMMUNICATIONS ACCESS METHOD

The QTAM facilities include a comprehensive set of message control and editing routines that relieve the programmer assigned to the telecommunications application of the detailed, intricate, and specialized programming normally required for such an application. These routines can be assembled into an integral message control program, which is designed to meet the exact requirements of an installation.

To simplify and speed the construction of a message control program, a special message control language is provided by QTAM. This language is compiled by an System/360 Operating System assembler program and consists of a set of macro-instructions and parameters. The language is specifically designed for easy use in describing the line procedures, line configurations, buffer lengths, polling procedures, and types of message editing required for a particular application. Using the message control language, a complete message control program for a Teleprocessing application can be described, and compiled in days, rather than in months.

The message control program serves as an intermediary between the remote terminals and any message processing programs. It enables the terminals to be referred to indirectly, in much the same way as local input/output devices are referred to, using standard language statements, such as GET, PUT, OPEN, and CLOSE. Detailed functions, such as the actual sending or receiving of messages, buffer allocating, message routing, message code translating, message formatting, and error checking, are performed automatically by the message control program.

The message control program may be executed as a separate task independently of and concurrently with any message processing programs. As input messages are received, they are routed (after translating, checking, etc., and any editing pro-

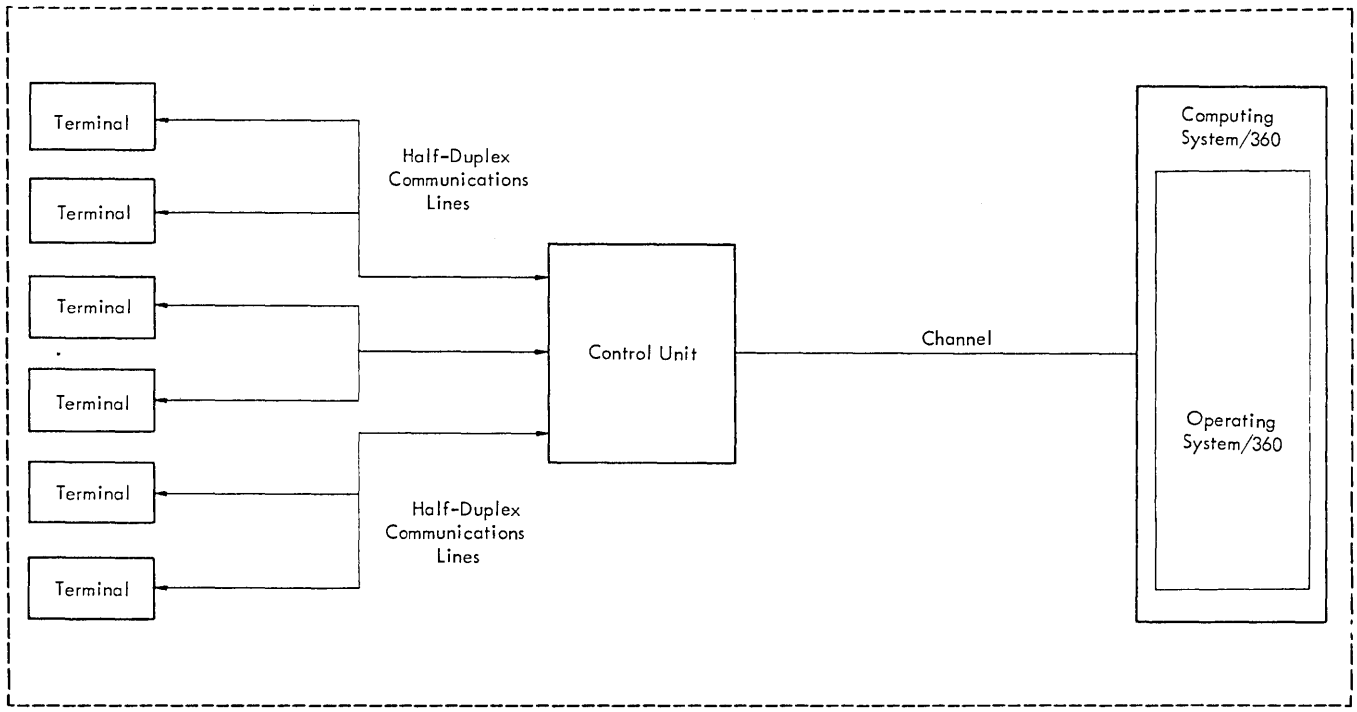


Figure 1. Telecommunications System - Conceptual Diagram

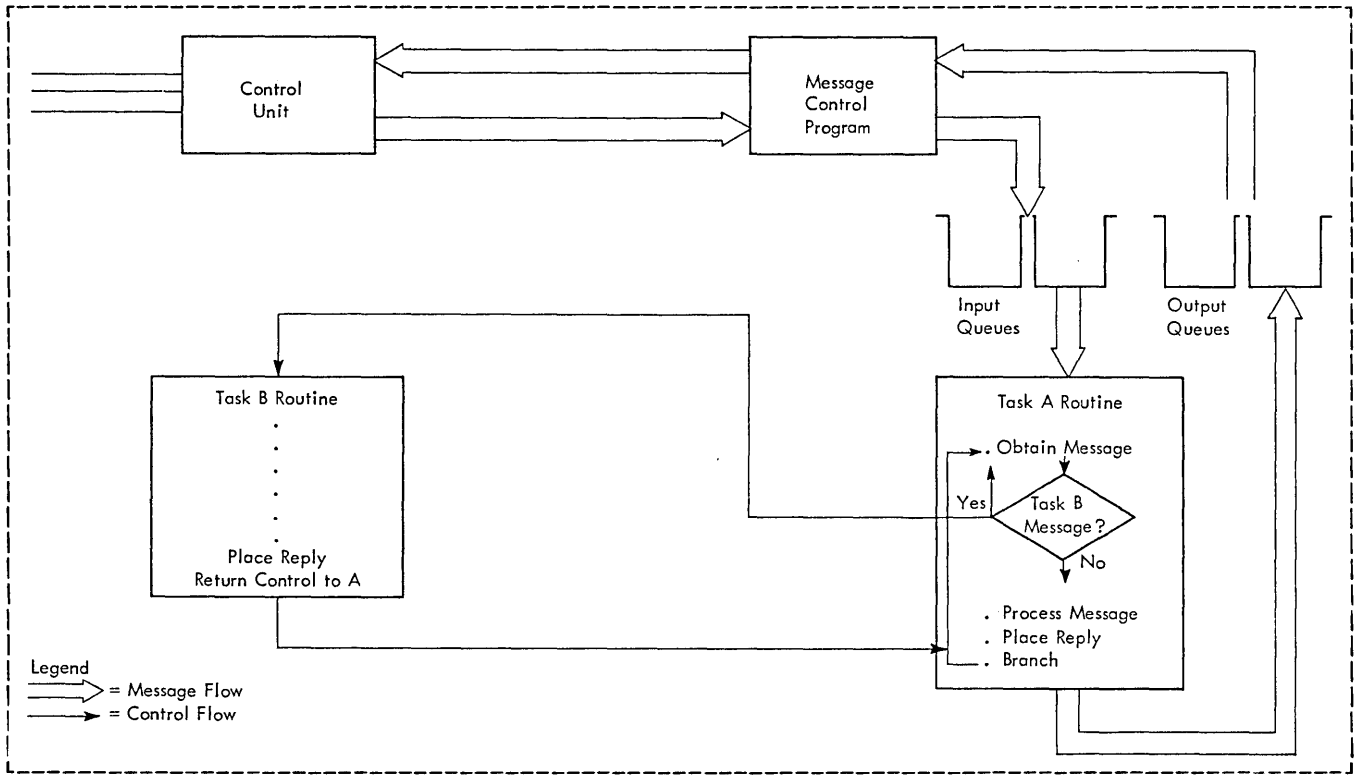


Figure 2. Inquiry Application - Simplified Block Diagram

vided by the user) to one or more message queues in main or direct-access storage. Message processing programs take them from there as in ordinary processing. When a message is to be sent to a terminal by a processing program, it is placed on an output queue in main or direct-access storage. The message is then sent by the message control program to its destination. In the case of message switching, a message processing program may not be required -- the message control program can route an inbound message directly to an appropriate output queue.

A telecommunications job can be entered into the system in the same way as any other job. The job scheduler of the operating system, therefore, can be used to allocate any input/output devices and direct-access storage space required for message logs and message queues, and to prepare and schedule the job for processing. A message control program and any message processing programs associated with it can be entered into the system as separate jobs or they can be combined and entered as a single job. With some operating system configurations, more than one job can be run concurrently. Therefore, other jobs can share the physical resources of the system with a telecommunications job, and thereby improve efficiency, particularly during periods when message traffic is low.

A message control program can be designed for one or more types of applications including inquiry (or transaction) processing, data collection, job processing, and message switching. Each of these is briefly described below followed by a brief explanation of the procedure required to construct a message control program for one or more of the applications.

INQUIRY PROCESSING

Inquiry processing is an application in which inquiries in the form of messages are received from a number of remote terminals and are routed by the message control program to one or more input queues. The messages are picked up from the input queue and processed by a message processing program. After the message is processed, a reply can be sent to the terminal from which the inquiry originated or to any other terminal. The reply is placed on an output queue by the message processing program and transmitted to the terminal by the message control program.

With this type of application, the system can directly participate in and control a variety of commercial and scientific activities as they are carried on. For example, the system may be used to service, from a central location, a geographically dispersed banking activity. In such a system, master files that contain account records for thousands of depositors are stored in direct-access storage. By entering pertinent data into the system, tellers at remote locations can check balances, update passbook records, and handle similar transactions, all within a few seconds.

The message processing program that responds to inquiries must be designed for the specific application. In designing the program, all of the facilities of the operating system are available including the language processors, the service programs, and the data and task management facilities.

The processing of messages can be performed sequentially as a single task or more than one message can be processed concurrently. Figure 2 is a simplified block diagram of an inquiry application in which the messages are processed as two separate tasks. As the routine for task B is processing one message, the routine for task A may be processing another message while the next message is being obtained from an input queue and a reply to a previous message is being placed on an output queue.

In many inquiry applications, a message processing program requires access to data and routines stored in local direct-access storage. In such an application, it would be possible to process several messages concurrently as separate tasks. Consequently, as the processing of one message is delayed while access is being gained to direct-access storage, another message could be processed. By processing several messages concurrently, the total message throughput of the system can be significantly increased. Since many of the messages in such an application would often require identical processing, a single program in main storage could be used to perform each of several concurrent tasks, and thereby save main storage space and program loading time.

Because the message control program and the message processing program are executed as separate tasks in an inquiry application, an System/360 Operating System control program is required that can control concurrently more than one data processing task. The general-purpose, multiple task-management facilities of the control program are particularly appropriate to this type of application. They enable the system to be used for many high-message-volume applications that would otherwise be

impractical without a specially designed control program.

DATA COLLECTION

Data collection is an application in which messages received from one or more terminals are collected and stored for later processing. Data collection can be performed completely by the message control program. Messages to be collected can be routed to an input queue on a direct-access storage device or logged on a sequential storage device, such as magnetic tape.

JOB PROCESSING

Job processing is an application in which jobs like those that are entered into the system locally are received from one or more remote terminals. The jobs may be entered at the remote location via input devices such as keyboards, punched card readers, and tape units. Output from a job can be directed to the terminal from which the job originated, transmitted to one or more other terminals, printed or otherwise processed by the operating system, or cataloged and stored (for later retrieval) in the operating system library. Data which is processed by the job can either be a part of the job itself or can be retrieved from the system library.

All of the operating system facilities, such as the language processors, the service programs, and the data, job, and task management facilities, that are available to the local programmer are also available to the programmer at the remote location. The operating system, in fact, is specifically designed for use at remote locations as well as for local use. The data cataloging and management facilities of the system, for example, enable individual programmers to compile, store, test, update, recompile, link, and execute programs within the confines of the operating system without resorting to the use of punched cards or without specific knowledge of the input/output configuration of the system.

Jobs that are received from remote locations are placed by the message control program on an input queue in direct-access storage in a format acceptable to the job scheduler. The actual processing of the jobs can be performed later, or the jobs on the input queue can be read and scheduled by the job scheduler as other jobs are being added to the input queue.

MESSAGE SWITCHING

Message switching is an application in which messages received from one remote terminal are sent to one or more other remote terminals. Like data collection, message switching can be performed completely by the message control program. When an incoming message is to be switched, it is routed to an output (destination) queue instead of an input queue. If necessary, the messages can be logged and functions, such as time and date stamping, sequence numbering and checking, and destination code validating, can be performed by the message control program before the message is transmitted.

CONSTRUCTING A MESSAGE CONTROL PROGRAM WITH THE QTAM LANGUAGE

To construct a message control program tailored to one or more specific applications, a programmer must provide three general types of information: data set definitions, control information, and line procedure specifications. Each type of information is described in the following sections.

Data Set Definition

A data set must be defined for each data set referred to by the message control program. This is done by means of the data control block (DCB) macro-instruction and data definition (DD) control statements that are entered into the system when the telecommunications job is scheduled for execution. A DCB macro-instruction must be provided in the message control program for the following:

- Each group of communications lines that are to be referred to by the message control program. Each line group can consist of any number of communications lines provided they have certain characteristics in common.
- The combined input and output queues that are to be established in direct-access storage. If the input and output queues are in main storage, a DCB macro-instruction is not required for the queues.
- Each message log, if any.

A DCB macro-instruction must also be provided in each message processing program, if any, for a queue from which messages are to be obtained and a queue to which messages are to be sent.

Control Information

In addition to the data set definitions, the programmer must supply, in the form of macro-instructions, control information that is used by the message control program to control the sending and receiving of messages. The control information consists of the following:

- The code name and address of each terminal together with related information, such as any special distribution lists for sending a message to more than one terminal.
- The code name of each queue to which incoming messages are to be sent. The messages can be either in main or direct-access storage. If extremely fast response is required, an option can be specified which causes messages for one or more input queues to be routed directly to the processing program, bypassing the direct-access storage queues.
- A polling list for each line that indicates the order in which the terminals are to be polled.
- The size and number of main storage buffers that are to be used for transmitting and receiving messages to and from the terminals, and for enqueueing and dequeueing messages. Buffers are automatically and dynamically allocated from a common buffer pool in accordance with immediate requirements.

Line Procedure Specifications

The procedure to be followed by a message control program, when operating upon messages, is defined by a series of macro-instructions called a line procedure specification (LPS). A line procedure specification is required for each communications line group unless similar procedures are to be followed in servicing the messages of more than one line group.

Two types of macro-instructions are used to specify a line procedure: functional and delimiter. The functional macro-instructions, in general, perform specific operations on messages. The delimiter macro-instructions perform initialization procedures, and classify and identify sequences of functional macro-instructions so that control can be directed to a required sequence based on whether a message is being sent or received and whether the complete message or just the message header is to be operated upon.

The operations that can be performed by the functional macro-instructions include the following:

- Performing message editing functions, such as code translating, time and date stamping, sequence checking, source and destination code validating, sequence numbering, and message length checking.
- Routing messages to specified queues based upon destination codes.
- Maintaining a log of messages on an auxiliary storage device.
- Checking for errors in message transmission.
- Initiating corrective action when an error is detected.
- Rerouting, canceling, or intercepting transmission of messages in error.

The functional macro-instructions in a line procedure specification are divided by delimiter macro-instructions into two major sections: a receive section consisting of macro-instructions that are to operate on incoming messages and a send section consisting of macro-instructions that are to operate on outgoing messages. Each major section is further subdivided into sections consisting of macro-instructions that are to operate on the header portion of a message and macro-instructions that are to operate on both the header and the text portions of a message. A third optional subsection may also be included in each major section, which consists of functional macro-instructions that perform certain end functions such as testing for errors and performing error procedures.

Functional macro-instructions in a message header subsection that operate on specific fields of the message header can be arranged in any order. Therefore, the programmer has complete freedom in defining the order of the message header fields.

Some of the functional macro-instructions are designed for use in either the send section or the receive section, or both. Most of the macro-instructions are optional. Many of these can operate either upon all messages unconditionally or selective messages, as indicated by a character code in the message itself. In addition, any optional portion of the line procedure can be bypassed for a particular message based on a character code in the message.

A programmer can, if he chooses, use the assembler language to insert his own in-line coding or his own macro-instruction into the line procedure specification, either as a substitute for an optional macro-instruction provided by IBM or in order to provide additional functions. The programmer also has the option of speci-

fying that his own code conversion tables be used to translate messages instead of the optional code conversion tables supplied by IBM.

An optional macro-instruction is provided that can be used to direct control, at different points in the line procedure specification, to an out-of-line subroutine supplied either by IBM or by the user. The transfer of control can be either unconditional or conditional depending on the presence of a character code in a message. IBM routines, which perform the following functions, are provided for use with the optional macro-instruction:

- Assigning priority to a message based on an alphameric character in the message header. The priority order of the characters that define priority correspond to the standard collating sequence.
- Holding the communications line open until a complete message has been received and a reply has been sent. This subroutine may be used for a "conversational" type of application.
- Routing message segments to their destination before the entire message has been received.

MESSAGE PROCESSING MACRO-INSTRUCTIONS

In addition to the macro-instructions that are provided for in the message control program, a set of macro-instructions is provided for use in a message processing program or other processing programs. The following macro-instructions are included in the set:

- The GET macro-instruction, which obtains the next sequential record, message segment, or message from the input queue.
- The PUT macro-instruction, which places a message, message segment, or record into the output queue.
- The RETRIEVE macro-instruction, which retrieves a specific message segment from a specific queue.
- The STOPLN and STRTLN macro-instructions, used to deactivate and activate, respectively, a communications line.
- Macro-instructions which can be used to copy a terminal table, a polling list, or a queue status report list into a work area of main storage, as well as macro-instructions which can be used to change entries in the terminal table and the polling list. These macro-instructions can be used to modify the status or operating characteristics of the message control program. They may be used in a program that receives and analyzes operator messages entered via a local terminal.

BASIC TELECOMMUNICATIONS ACCESS METHOD

The BTAM facilities are designed chiefly to provide the basic tools required to construct a telecommunications program. These include facilities for creating terminal lists and performing the following operations:

- Polling terminals.
- Answering.
- Receiving messages.
- Allocating buffers dynamically.
- Addressing terminals.
- Dialing.
- Writing buffer chains.
- Changing the status of terminal lists.

BTAM may be used instead of QTAM at installations that have few communications lines and limited requirements for inquiry processing or data collection. It also may be used at a large Tele-processing installation where the user chooses to construct his own message control program rather than use the one provided with QTAM.

QTAM is one of the data management facilities provided in System/360 Operating System. It is specified as a high-level macro-instruction language that enables operation of communications lines in a manner similar to directly attached input/output devices.

Selection of appropriate macro-instructions within QTAM provides a system that performs the following:

- Polls terminals (contacts remote terminals to request that they send messages).
- Receives messages from terminals.
- Addresses terminals (contacts terminals to request that they receive messages).
- Sends messages to terminals.
- Provides storage allocation for message buffering.
- Performs such message editing functions as code translating, time and date stamping, sequence checking, source and destination code validating, sequence numbering, and message length checking.
- Routes messages to specified queues based upon destination codes (a queue may be used for a processing task, a terminal, or a group of terminals).
- Queues messages either on a direct-access storage device (to provide for handling long messages without requiring large buffers) or in main storage.
- Maintains a log of messages on an auxiliary storage device.
- Checks for errors in message transmission.
- Initiates corrective action when an error is detected.
- Reroutes messages.
- Intercepts transmission of messages in error.
- Cancels messages that contain an error.

Message flow, as controlled by QTAM, is illustrated in Figure 3. It involves input/output buffering and message queueing. Buffering is dynamically handled by QTAM. A user need only specify the number of buffers and the length of each buffer to be used. (Refer to "Buffer Assignment.") If the supply of buffers is depleted during system operation, an error indication on one or more lines will result. For this reason, a user must specify sufficient buffer space to meet his system requirements.

Queues must be specified for various processing tasks and destination terminals. These queues are the primary connection between the user-supplied message processing tasks and the QTAM message control task. Message flow (Figure 3) can be considered in the following seven steps:

1. The input message, consisting of message header and text, is prepared at the remote source terminal. The header portion contains source terminal code, destination codes, message sequence number, and message priority information. When the source terminal is polled, the message is sent to the computer via a communications line.
2. The variable length message enters the computer and is placed in user-defined, fixed-size buffers. As many buffers as are necessary to handle the message are filled. QTAM attaches a header prefix that contains control information (24 bytes) and queueing information (8 bytes). The header prefix and the message header must all be contained within the first buffer. Each of the remaining buffers assigned to the message has a text prefix that contains queueing information (8 bytes) and control information (14 bytes). These buffers also contain the message text. As soon as each buffer is filled, QTAM performs such user-selected functions as code translating, routing, time and date stamping, and sequence checking.
3. When a message requires additional processing, each segment is sent to a process queue. This queue may be either in main storage or on a direct-access storage device.
4. The user's message processing program can issue a GET macro-instruction to obtain messages, segments, or records from the process queue. The message obtained by the processing program contains a modified prefix (4 bytes). (Refer to "GET Macro-Instruction.") This message can then be processed. A user in his processing program can send replies by forming a message with the 4-byte prefix and by issuing a PUT macro-instruction. The message is then placed on an output (destination) queue and handled as in message switching (Step 5).

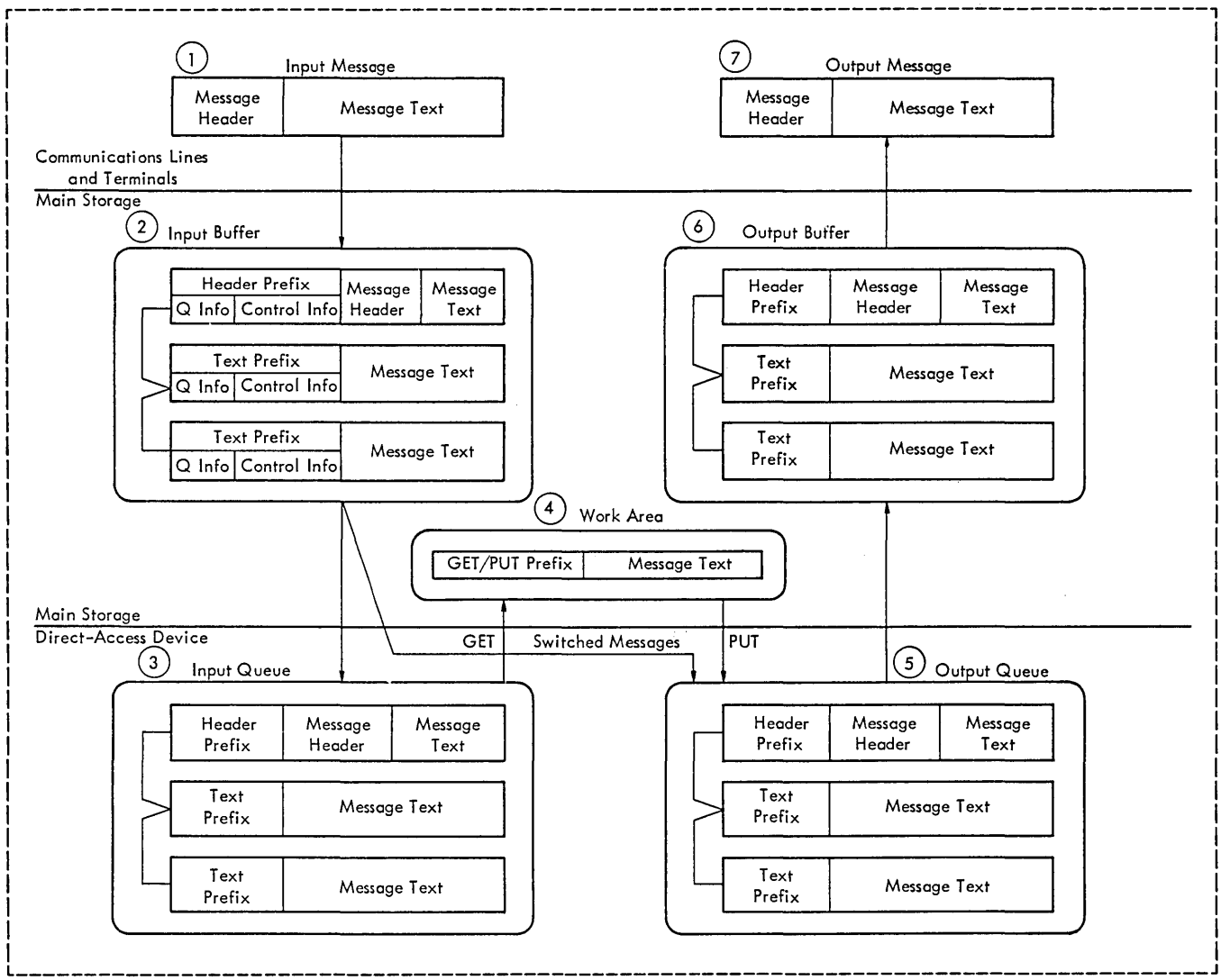


Figure 3. Telecommunications Access Method - Message Flow Diagram

5. In message switching, each message segment is routed to a destination queue. This queue may be either in main storage or on a direct-access storage device.
6. Message segments are retrieved from the output queue on a first-in-first-out basis within priority groups. These segments pass through QTAM for such functions as code translating, time and date stamping, and logging. The segment is then ready to be sent to the terminal.
7. Message segments are stripped of header and text prefixes and sent to the terminal as one continuous message.

In order to communicate with QTAM, message processing programs must use the GET, PUT, and RETRIEVE macro-instructions described under "Message Processing Tasks." Other functions of the user programs for message processing may be written using the Assembler Language (refer to IBM Operating System/360: Assembler Language) or a higher level language compiler that produces an equivalent object code.

There may be one or more individual message processing tasks. A message processing task can obtain a message from a queue, process the message, and place a message or result in an output queue. The programmer need not be concerned with the functions being performed by the message control task and the QTAM device handling routines. The message processing task, however, cannot be executed without the message control task. Problem programs may be written as if they were using sequential input/output devices.

Message processing programs that use directly attached input/output devices may use the macro-instruction language of the appropriate access method.

System status macro-instructions are provided to enable a user to obtain information that concerns the terminal polling sequence, the terminal table entries, and the present status of message queues. Macro-instructions can be used in a message processing task to extract system status information and to change the system status.

In specifying QTAM for his system, the user must provide information in the following areas:

- Data set definition - The user must use macro-instructions provided by System/360 Operating System and QTAM control statements to define the data sets used in his telecommunications system.
- Control information - The user must provide detailed information, in the form of tables and lists, concerning the terminals in the system, the desired sequence of polling for each line, the queueing technique, and the message buffers provided.
- Message control task - The user must select and place into order macro-instructions provided by QTAM to create the message control task. These macro-instructions perform the message code translating, the editing, the format checking, the logging, and the routing functions.
- Message processing tasks - The user must write the programs necessary to meet his requirements. These programs operate as one or more individual tasks.

Wherever macro-instructions are described in this publication, the following conventions are used to illustrate the coding format:

- Upper case letters and punctuation marks (except for the brackets and braces described) represent information that must be coded exactly as shown.
- Lower case letters and words are generic terms that represent information to be supplied; i.e., a substitution must be made when coding a parameter or option so represented.
- Information within brackets [ ] represents an option that can be either included or omitted entirely, depending upon program requirements.
- When options are enclosed in braces { } one of the enclosed options must be included, and the others omitted according to the programmer's discretion. One of several options enclosed in the braces may be underlined, indicating that the operating system automatically assumes this option if the parameter is omitted.
- An ellipsis (...) indicates that a variable number of items may be listed.

DATA SET DEFINITION

Before a telecommunications system can use QTAM, the data sets of the system must be specified. A description of the four types of data sets that can be associated with QTAM and the means by which they may be defined follows.

Four types of data sets are supported directly by the QTAM message control task:

- Data sets that consist of messages transmitted via communications lines. One or more data sets of this type are required.
- Data sets that consist of message queues on direct-access storage devices (IBM 2302 Disk Storage, IBM 2311 Disk Storage Drive, and IBM 2314 Direct Access Storage Facility). The requirement for this type of data set depends on the requirements of the user.
- Data sets that consist of message logs on a secondary storage device. The requirement for this type of data set depends on the requirements of the user.
- Data sets that consist of queues of messages. This type of data set is used in the message processing tasks.

Available sources for information concerning a data set include the data control block (DCB) macro-instruction, the data definition (DD) statement, and the data set label (DSL). The DCB macro-instruction provides for assembly time definition of a data control block. This data control block is the primary source of data set characteristics and control information for the QTAM routines. The DCB macro-instruction reserves space for the data control block and assigns values to those fields for which the user provides parameters.

Note: Information provided in the DCB macro-instruction takes precedence over information provided in the DD statement or the DSL. For general information concerning the DD statement and the DSL, refer to the publications IBM Operating System/360: Job Control Language and IBM Operating System/360: Data Management.

A data control block provides control information for a communications line group, where the line group consists of any number of lines with the following characteristics:

- All lines are associated with the same type of terminal devices. (For this purpose, IBM 1050 and IBM 1060 Data Communications Systems may be treated as the same device.)
- All lines require the same number of buffers to be requested before they are actually needed.
- All lines share the same line procedure specifications (LPS).
- All lines operate under the same relative priority specification.
- All lines operate with the same polling interval.
- The relative address of the device-address field is the same for all terminal table entries associated with the lines.
- None of the lines is defined in another line group.

Certain information that describes a data set must be supplied through DD statements. Each communications line group, direct-access queue and message log requires device allocation information. In addition, the direct-access queue requires space allocation on a direct-access storage device.

Each communications line group requires one or more DD statements that specify which communications lines are to constitute that line group. This is specified in the UNIT operand of the statement. It may be specified in either of two ways:

1. //ddname DD UNIT=(name,n)

   where name is the name of a list that was created at system generation time and n is the number of lines in the list. This list contains a description of each line. The order in which the lines appear in the list determines the relative line number for each.

2. //ddname DD UNIT=($name_1$,n_1)
   DD UNIT=($name_2$,n_2)
   .
   .
   .
   DD UNIT=(name ,n)

   where x lists are named and are concatenated in the order in which they are specified, and where n is the number of lines defined in each list.

If direct-access queueing is to be used, the user must acquire space for the queue (using the DD statement). The user must also execute a utility program so that the acquired space may be used for a fixed record length (equal to the buffer size to be used by QTAM). This data set must then be cataloged and identified to the message control task on the DD statement associated with the direct-access queue.

One input data control block must be created for each queue from which messages are to be obtained by the message processing tasks. To obtain a message from a queue, the message processing task refers to the data control block associated with that queue.

At least one output data control block must be created for each message processing task from which messages are to be sent. Since all destinations may be referred to by using one data control block, the need for additional data control blocks for output is eliminated.

Data Control Block (DCB) Macro-Instruction

The DCB macro-instruction causes the creation of a data control block. A data control block must be created for each communications line group, direct-access message queue, processing program message queue, and message log.

Name	Operation	Operand
dcbname	DCB	keyword parameters

keyword parameters
Indicate the parameters listed in Tables 1 through 5. When an alternate source code is shown with a parameter, it indicates the other sources of information that concern the parameter: C indicates the parameter that may be specified in a DD statement; D indicates the parameter that may be supplied from a DSL; and E indicates the parameter whose value may be supplied at any time up to and including the DCB exit provided at open time. (Refer to the publication IBM Operating System/360: Control Program Services.)

Table 1. Communications Line Group DCB Macro-Instruction Keyword Parameters

Keyword Parameter	Alternate Source Code	Parameter Description
DDNAME=name		name Specifies the name that appears in the DD statement associated with the data control block.
DSORG=CX		CX Identifies the data set organization as that of a communications line group.
MACRF=(G,P)		(G,P) Specifies that access to the line group is to be gained with the GET and PUT macro-instructions.
CPOLL=(d_1, d_2,...,d_n)		(d_1,d_2,...,d_n) Specifies n symbolic addresses of polling lists, where n is the number of lines in the line group. Each symbolic address is the same as the name in the POLL macro-instruction used to define the list for that line. The symbolic addresses must be specified in the same order as the lines are specified in the DD statement (refer to IBM Operating System/360: Job Control Language). If a line is used for output only, the address of a polling list with no terminal entries must be specified. Any number of output-only lines may reference this address.
BUFRQ=n	C,E	n Specifies the number of buffers to be requested before they are actually needed to receive data from a line, where $2 \leq n \leq 255$. The number should be specified such that buffer requests may be scheduled sufficiently in advance to ensure that a buffer will be available when it is actually needed for data. The primary factors to be considered in determining the value of this number are the line speed, the size of the buffer pool versus the average number of buffers that are active at any one time, the size of each buffer versus the average size of a transmitted block, and the total system loading. Note: BUFRQ=2 will be assumed if no value is provided through the DCB macro-instruction or an alternate source, or if a value less than 2 is provided.
INTVL=t	C,E	t Specifies the number of seconds of intentional delay between passes through a polling list, where $t \leq 255$. When all of the terminals on a polling list for a given line have been polled, there will be a delay of t seconds before polling is restarted at the beginning of the list. The primary purpose of the polling interval is to limit nonproductive polling. It may also be used to limit the frequency of message transmissions from a highly active line group. Note: INTVL=0 will be assumed if no value is provided through the DCB macro-instruction or an alternate source.

(continued)

Table 1. Communications Line Group DCB Macro-Instruction Keyword Parameters (continued)

Keyword Parameter	Alternate Source Code	Parameter Description
[{CPRI=R / CPRI=E / CPRI=S}]	C,E	R, E, or S Indicates the relative priority to be given to sending and receiving operations, as follows: R - receiving has priority over sending. Output message will be sent on a given line only during a polling interval. E - receiving and sending have equal priority. After each full polling sequence on a given line, all output messages queued for that line will be transmitted. S - sending has priority over receiving. After polling each individual terminal on a line, the line is made available for outbound messages and the next terminal is polled only when there are no output messages in queue for that line. Note: CPRI=S will be assumed if no value is provided through the DCB macro-instruction or an alternate source.
ACLOC=n	E	n Specifies the relative address of the device-address field in a terminal table entry relative to the first byte of the terminal table entry (defined by the TERM macro-instruction), where n≤255. If no value is provided through the DCB macro-instruction or an alternate source, the job will be terminated.
CLPS=addr	E	addr Specifies the symbolic address of the start of the line procedure specifications. If no value is provided through the DCB macro-instruction or an alternate source, the job will be terminated.
[EXLST=addr]		addr Specifies the symbolic address of the exit list. (Refer to Appendix D of the publication IBM Operating System/360: Control Program Services.)

Table 2. Direct-Access Message Queue DCB Macro-Instruction Keyword Parameters

Keyword Parameter	Alternate Source Code	Parameter Description
DDNAME=name		name Specifies the name that appears in the DD statement associated with the data control block.
DSORG=CQ		CQ Identifies the data set organization as that of a direct-access message queue for communications.
MACRF=(G,P)		(G,P) Specifies that access to the queue is to be gained with the GET and PUT macro-instructions.

Table 3. Message Log DCB Macro-Instruction Keyword Parameters

Keyword Parameters	Alternate Source Code	Parameter Description
DDNAME=name		name Specifies the name that appears in the DD statement associated with the data control block.
DSORG=PS		PS Specifies a sequential data set organization.
MACRF=(PM)		(PM) Specifies the use of the PUT macro-instruction with the move mode.
{RECFM=V RECFM=VB}	C,D	V Specifies variable length records. VB Specifies variable length blocked records.
BLKSIZE=n	C,D	n Specifies the maximum physical block length. It should be greater than or equal to the maximum segment length.
BFTEK=SMP	C,D	SMP indicates simple buffering.
BUFNO=n	C	n Specifies the number of buffers to be obtained by QSAM at open time, where n≤255. The parameter n should be specified greater than or equal to two. If it is specified as one, the message control task may have to wait for the buffer. Note: For additional information concerning the specification of this DCB macro-instruction, refer to "Queued Sequential Access Method (QSAM)" in the publication IBM Operating System/360: Control Program Services.
BUFL=n	C	n Specifies the length in bytes of the buffers to be obtained by QSAM at open time, where n≤32,767. The parameter should be greater than or equal to the maximum segment length.
{DEVD=DA DEVD=DT DEVD=TA DEVD=PC DEVD=PR}		DA Specifies that the logging device is a direct-access device. DT Specifies that the logging device is a paper tape. TA Specifies that the logging device is a magnetic tape. PC Specifies that the logging device is a card punch. PR Specifies that the logging device is a printer.
[{DEN=0 DEN=1 DEN=2}]	C,E	0 Specifies that the tape density is low. 1 Specifies that the tape density is medium. 2 Specifies that the tape density is high. Note: This parameter is required only for tape devices. If it is not specified, high density is assumed.

Table 4. Message Processing Task Input DCB Macro-Instruction Keyword Parameters

Keyword Parameters	Alternate Source Code	Parameter Description
DDNAME=name		name Specifies the name that appears in the DD statement associated with the data control block. This name is also the name used in the terminal table to identify the process queue.
DSORG=MQ		MQ Identifies the data set organization as that of a processing program message queue for telecommunications.
MACRF=G		G Specifies that access to the queue is to be gained with the GET macro-instruction.
BUFRQ=n	C,E	n Specifies the number of buffers to be read in advance (from GET macro-instruction), where n≤255. Note: BUFRQ=0 will be assumed if a value is not provided through the DCB macro-instruction or an alternate source.
SOWA=n	C,E	n Specifies the size in bytes of the user-provided input workarea, where n≤32,767. If this parameter is not provided through the DCB macro-instruction or an alternate source, the job will be terminated.
[{RECFM=G / RECFM=S / RECFM=R}]	C,E	G, S, or R Specifies the work unit as follows: G - message (defined by the end-of-transmission character). S - segment (defined by the buffer size). R - record (defined by the carriage return, line feed, combined carriage return-line feed, or end-of-block character). Note: RECFM=S will be assumed if no value is provided through the DCB macro-instruction or an alternate source.
EODAD=addr		addr Specifies the symbolic address of a user-provided routine to be entered if no messages are available when a GET macro-instruction is issued. If no value is provided through the DCB macro-instruction or an alternate source, a WAIT macro-instruction will be implied.
TRMAD=addr	E	addr Specifies the symbolic address of a user-provided area to contain the terminal name. When a GET macro-instruction is issued, the source terminal name will be placed at the specified address by QTAM. If the parameter is not provided through the DCB macro-instruction or an alternate source, the job will be terminated.
SYNAD=addr	E	addr Specifies the symbolic address of a user-provided routine to be entered if a work unit is longer than the workarea provided for input. If the parameter is not provided through the DCB macro-instruction or an alternate source, the remainder of the work unit will be supplied when the next GET macro-instruction is issued.

Table 5. Message Processing Task Output DCB Macro-Instruction Keyword Parameters

Keyword Parameters	Alternate Source Code	Parameter Description
DDNAME=name		name Specifies the name that appears in the DD statement associated with the data control block. (Refer to the publication IBM System/360 Operating System: Job Control Language.)
DSORG=MQ		MQ Identifies the data set organization as that of a processing program message queue for telecommunications.
MACRF=P		P Specifies that the queue is to be referred to by the PUT macro-instruction.
SOWA=n	C,E	n Specifies the size in bytes of the user-provided output workarea, where n≤32,767. If this parameter is not provided through the DCB macro-instruction or an alternate source, the jcb will be terminated.
[{RECFM=G / RECFM=S / RECFM=R}]	C,E	G, S, or R Specifies the work unit as follows: G - message (defined by the end-of-transmission character). S - segment (defined by the buffer size). R - record (defined by the carriage return, line feed, combined carriage return-line feed, or end-of-block character). Note: RECFM=S will be assumed if no value is provided through the DCB macro-instruction or an alternate source.
TRMAD=addr	E	addr Specifies the symbolic address of a user-provided area to contain the terminal table entry name. If the parameter is not provided before the data set is opened, the job will be terminated. When a PUT macro-instruction is issued, the destination terminal name must be provided at the specified address by the user. The name must be defined in a terminal table entry within the message control task. (Refer to the TERM, LIST, and PROCESS macro-instructions described under "Terminal Table Specifications.")

Examples: The DCB macro-instruction that defines the parameters of a data control block associated with a communications line group (Table 1) could be:

```
Name      Operation  Operand
GROUPONE  DCB        DDNAME=DDGROUP1,
                     DSORG=CX,
                     CPOLL=(POLLINE1,
                      POLLINE2),
                     MACRF=(G,P),BUFRQ=3,
                     CPRI=E,ACLOC=25,
                     CLPS=LPS1
```

The DCB macro-instruction that defines the parameters of a data control block associated with a direct-access storage device to be used for message queueing (Table 2) could be:

```
Name      Operation  Operand
QUEUE     DCB        DDNAME=DDFILE,
                     DSORG=CQ,
                     MACRF=(G,P)
```

The DCB macro-instruction that defines the parameters of a data control block associated with a message logging device (Table 3) could be:

```
Name      Operation  Operand
MSGLOG    DCB        DDNAME=DDLOG,
                     DSORG=PS,
                     MACRF=(PM),RECFM=V,
                     BLKSIZE=100,
                     BFTEK=SMP,
                     BUFNO=3,BUFL=100,
                     DEVD=TA,DEN=1
```

The DCB macro-instruction that defines the parameters of a data control block associated with an input message processing queue (Table 4) could be:

```
Name      Operation  Operand
PPMQIN    DCB        DDNAME=PRCIN,
                     DSORG=MQ,
                     MACRF=G,
                     BUFRQ=2,SOWA=300,
                     RECFM=G,
                     TRMAD=SOURCE,
                     SYNAD=ERROR
```

The DCB macro-instruction that defines the parameters of a data control block associated with an output message processing queue (Table 5) could be:

```
Name      Operation  Operand
PPMQOUT   DCB        DDNAME=PRCOUT,
                     DSORG=MQ,
                     MACRF=P,SOWA=300,
                     RECFM=G,
                     TRMAD=DESTN
```

OPEN AND CLOSE MACRO-INSTRUCTIONS

To initialize a data set (make it accessible by a problem program), an OPEN macro-instruction that specifies the data control block associated with the data set must be issued. In certain cases, the OPEN macro-instruction must also specify whether the data set is being opened for input or output. When the OPEN macro-instruction is issued, the following general functions will be performed:

- Completion of the initialization of the data control block.
- Acquisition of the access method subroutines.
- Acquisition of main storage for internal control blocks required by the access method.

To terminate use of a data set, the CLOSE macro-instruction is issued. The following general functions will be performed:

- Releasing main storage acquired at open time.
- Releasing subroutines acquired at open time.
- Clearing the data control block fields which were initialized at open time.

OPEN Macro-Instruction

The OPEN macro-instruction makes the following ready for use: queues on direct-access storage devices, data sets on logging devices, processing queues, and communications line groups. Each of these data sets must be opened before any reference can be made to it. This macro-instruction, when used, has the following effects:

- If a communications line group data control block is specified, the OPEN macro-instruction causes all lines in the group to be prepared for operation. The actual activation of the lines for message transmission occurs automatically unless IDLE is specified as the second entry in the sublist in the OPEN macro-instruction for the communications line group DCB. If the line group is specified as an input, or input and output line group, and the IDLE operand is not included, the OPEN macro-instruction initiates polling on those lines in the line group that have an active polling list with terminal entries. (A polling list is specified by the POLL macro-instruction; terminal entries may or may not be included. An active polling list is one in which the second byte of the list is a non-zero character.) If IDLE is specified, all of the lines or particular lines in the line group can be subsequently activated by one or more STRTLN macro-instructions; however, STRTLN must not be included in the Line Procedure Specification section of the message control task macro-instructions. The user can also inhibit polling on a line by changing the second byte of the polling list for that line to zero before issuing the OPEN (see "Change Polling List (CHNGPL) Macro-Instruction"). Lines used for output only (see below) are never polled.
- If a direct-access queue data control block is specified, and it is specified that there are messages from a previous period in the queues, it will update queue status tables to enable these messages to be transmitted.
- If a process data control block is specified, it causes QTAM to be prepared to deliver messages to or receive messages from the message processing task, depending upon whether input or output is specified.
- If a logging data control block is specified, it causes the QSAM routines to be entered into the system and be prepared for placing messages on a logging device. Output must be specified.

Name	Operation	Operand
[symbol]	OPEN	(dcbname$_1$, [({INPUT / OUTPUT / INOUT}) [,IDLE])] ,...,dcbname$_n$, [({INPUT / OUTPUT / INOUT}) [,IDLE])] [,{MF=L / MF=(E,listname)}]

symbol
: Specifies the name for the OPEN macro-instruction.

dcbname$_1$...dcbname$_n$
: Indicates the address of the data control block associated with the communications line group, direct-access queues, processing queue, or message log to be opened. The group of operands that can be repeated begins with dcbname and ends with IDLE.

INPUT
: Indicates an input data set. If neither INPUT, OUTPUT, nor INOUT is specified, the operating system assumes that INPUT is specified. If: 1) a communications line group is opened; 2) the INPUT operand is specified or no operand is included in its place; and 3) the IDLE operand is omitted, polling begins on all lines with an active polling list and terminal entries on that list.

OUTPUT
: Indicates an output data set. If a communications line group is opened, and this operand is specified, the CPOLL keyword operand sublist of the DCB macro-instruction for the line group must point to a polling list with no terminal entries.

INOUT
: Indicates a data set that can be used for input and output. If a communications line group is opened, and this operand is specified, some of the lines can be used for input and others for output. If an entry in the sublist of the CPOLL keyword parameter in the DCB macro-instruction for the line group points to a polling list with terminal entries, the line is a polled input line; polling begins if the polling list is active and the IDLE

operand is not specified in the OPEN macro-instruction for the line group. If an entry in the sublist of the CPOLL keyword parameter of the line group DCB points to a polling list with no terminal entries, the line is an output line.

Note: If neither INPUT, OUTPUT, nor INOUT is indicated for a data set, and another data control block address is specified, two commas must appear between the two data control block addresses.

IDLE

Pertains only to communications line groups. If a communications line group is opened, and the IDLE operand is included, the data set for the line group is initialized but the lines remain inactive until activated by a STRTLN macro-instruction. If the IDLE operand is omitted, all lines in the group are automatically activated as part of the OPEN instruction's functions. If the IDLE operand is included, and neither INPUT, OUTPUT, nor INOUT is specified, a comma must be placed before IDLE to represent the missing preceding operand in the sublist.

Note: If neither the input/output designation nor the IDLE operand is included for a data set, and another data control block address is specified, two commas must appear between the two specified data control block addresses.

MF=L

Causes the creation of a parameter list based on the entries specified in the operand section of the OPEN macro-instruction. No executable code is generated. The functions specified by the preceding parameters are not performed until the problem program encounters an OPEN macro-instruction with a keyword parameter referencing the parameter list (see below). The name assigned to the parameter list is specified in the name field of the OPEN macro-instruction containing the MF=L operand.

MF=(E,listname)

Causes the execution of those functions indicated by the parameter list named "listname." This list was previously created by an OPEN macro-instruction with the MF=L keyword parameter. Parameters included in the OPEN instruction with the MF=(E,listname) entry override corresponding entries in the parameter list.

Note: The absence of the MF keyword parameter causes the OPEN macro-instruction to be executed with all the specified parameters. The advantage of the use of MF=L and MF=(E,listname) operands is that one parameter list (created through MF=L) can serve several OPEN and CLOSE macro-instructions with the MF=(E,listname) operand. For example, an OPEN macro-instruction with the MF=L operand creates a parameter list; several CLOSE macro-instructions, each with the MF=(E,listname) operand and each to be executed depending on a given situation, can subsequently reference the list. If these keyword operands are not used, one parameter list is generated in-line each time an OPEN or CLOSE macro-instruction for the line group is specified.

Note: Data sets transmitted over communications lines and data sets on direct-access storage devices for queueing and on logging devices must be opened in the message control task. Data sets on message processing task queues must be opened in the user's message processing task. Data sets on direct-access devices must be opened before any other data sets used in QTAM.

Examples: An OPEN macro-instruction that could be used in the message control task is:

```
┌────┬─────────┬──────────────────────────┐
│Name│Operation│Operand                   │
├────┼─────────┼──────────────────────────┤
│    │OPEN     │(QUEUE,,GROUPONE,(,IDLE), │
│    │         │MSGLOG,(OUTPUT))          │
└────┴─────────┴──────────────────────────┘
```

An OPEN macro-instruction that could be used in the message processing task is:

```
┌────┬─────────┬──────────────────────────┐
│Name│Operation│Operand                   │
├────┼─────────┼──────────────────────────┤
│    │OPEN     │(PPMQIN,(INPUT),PPMQOUT,  │
│    │         │(OUTPUT)),MF=(E,OPENNAME) │
└────┴─────────┴──────────────────────────┘
```

Note: Refer to examples of the DCB macro-instruction.

CLOSE Macro-Instruction

The CLOSE macro-instruction removes direct-access storage queues, processing queues, and communications line groups from active use; i.e., closes them. When not required for further operations, these queues and line groups should be closed in order to release core storage, clear output areas, and relinquish volumes (where applicable). This macro-instruction, when used to close a communications line group, has the following effects:

Operations on all lines in the group are stopped. If a line was conditioned by a CHNGPL macro-instruction to stop receiving, all messages queued for transmission on the line will be transmitted before control is returned to the program issuing the CLOSE macro-instruction. If the line was not conditioned to stop receiving, control is returned to the program issuing the CLOSE macro-instruction immediately after the completion of any message being transmitted. In the latter case, any messages remaining in the queue for a line will be transmitted when the data control block for the communications line group is reopened.

To close one line within a communications line group, the STOPLN macro-instruction should be used (see "Stop Line (STOPLN) Macro-Instruction" under "Message Processing Task").

Name	Operation	Operand
[symbol]	CLOSE	(dcbname$_1$,,,... dcbname$_n$) [,{MF=L | MF=(E,listname)}]

symbol
Specifies the name for the CLOSE macro-instruction.

dcbname$_1$,,,...dcbname$_n$
Indicates the address of the data control block associated with the direct-access queue, the processing queue, or the communications line group to be closed.

Note: The two commas immediately following dcbname should be omitted if the operand is the last (or only) positional operand.

MF=L
Causes the creation of a parameter list based on the entries specified in the operand section of the CLOSE macro-instruction. No executable code is generated. The functions indicated by the preceding parameters are not performed until the problem program encounters a CLOSE macro-instruction with a keyword parameter referencing the parameter list (see below). The name assigned to the parameter list is specified in the name field of the CLOSE macro-instruction containing the MF=L entry.

MF=(E,listname)
Causes the execution of those functions indicated by the parameter list named "listname". This list was previously created by a CLOSE macro-instruction with the MF=L keyword parameter. Parameters included in the CLOSE instruction with the MF=(E,listname) entry override corresponding entries in the parameter list. (Information contained in the note under "OPEN Macro-Instruction" also pertains to CLOSE.)

RESTRICTION: A direct-access queue data control block must not be closed while any line group or process data control block is open.

Note: Data sets associated with communications lines and data sets on direct-access storage devices for queueing and on logging devices must be closed in the message control task. Data sets on processing program message queues must be closed in the user's message processing task.

Two additional forms of the CLOSE macro-instruction are available: the E form and the L form. (For a description of the use of these forms, refer to the publication IBM System/360 Operating System: Control Program Services.)

Examples: The CLOSE macro-instruction that is used in the message control task could be:

Operation	Operand
CLOSE	(GROUPONE,,QUEUE,,MSGLOG)

The CLOSE macro-instruction that is used in the message processing task could be:

Operation	Operand
CLOSE	(PPMQIN,,PPMQOUT)

Note: Refer to examples of the DCB macro-instruction.

MESSAGE CONTROL TASK

The message control task includes both device handling and message handling routines of QTAM. Messages destined for other terminals (as in a message switching application) can be completely handled by the message control task. Messages that require processing by a user's program can be directed to or taken from the message processing task queues that are referred to by the user's message processing tasks.

Messages arriving from terminals are coded according to the transmission code of the particular terminal. The message control task includes facilities that convert these transmission codes to the extended binary-coded decimal interchange code (EBCDIC) which results in simplification of message analysis. In addition, the message control task can perform the following functions:

- Terminal polling.
- Time and date stamping of incoming messages.
- Sequence checking of incoming messages
- Logging of message traffic on a secondary storage device (disk, tape, or drum).
- Checking validity of source terminal code.
- Routing of incoming messages to their destination (processing queue or terminal).

Messages received from terminals can be routed to one or more destinations. The routines check the validity of the destination codes and place messages in queues according to their destinations. A priority scheme may be included so that processing of certain messages can be expedited.

Messages being sent to a terminal must be converted to the appropriate transmission line code. The following functions are also provided for processing outgoing messages:

- Terminal addressing.
- Time and date stamping.
- Sequence numbering.
- Logging on secondary storage devices such as disks, tapes, or drums.

To define the message control task, a user must specify appropriate system-supplied macro-instructions. The message control task must operate as a high-priority task. A message control task for Tele-processing operations need remain in main storage only as long as message processing is going on.

There are two major categories of macro-instructions that may be selected for inclusion in the message control task: control information and line procedure specifications. Control information macro-instructions create tables, lists, and buffer areas needed in the system; line procedure specification macro-instructions provide for such functions as message header analysis, routing, and error handling.

CONTROL INFORMATION

To use the optional functions of QTAM, a user must provide certain control information. This data includes the following:

- A terminal table that contains all of the valid terminal codes as well as complete information about the terminals connected to the system.
- A polling list that provides the sequence in which terminals on a line are to be polled.
- Buffer specifications that provide the definition of message segment sizes, the type of buffering, and the maximum number of segments needed by QTAM.

Terminal Table Specifications

The terminal table consists of the following parts:

- A control field that defines the length of the terminal table.
- Optional fields that contain information defined by the programmer.
- Terminal descriptions that contain the terminal, the message processing queue, and the distribution list codes, as well as other pertinent control information. (All terminal table entry names must have the same number of characters.)

The macro-instructions presented below may be used to create the terminal table.

Terminal Table (TERMTBL) Macro-Instruction

The TERMTBL macro-instruction causes a control field to be created for the terminal table. It provides a definition of the length of the table. Only one such instruction is necessary for the table.

Name	Operation	Operand
	TERMTBL	entry

entry
Specifies the terminal name of the last entry in the terminal table.

RESTRICTION: The TERMTBL macro-instruction must be the first of those specified to create the terminal table.

Note: TERMTBL is the symbolic address of the terminal table.

Terminal Table Optional Field (OPTION) Macro-Instruction

The OPTION macro-instruction is used to allocate storage for a specified subfield to be included in the user area of each terminal table entry created by the TERM macro-instruction. One OPTION macro-instruction is necessary for each required subfield. The order of these instructions defines the order of the subfields within the user area. A standard definition of this area is provided to permit symbolic references to each subfield.

Name	Operation	Operand
symbolic name of subfield	OPTION	subfield type and length

symbolic name of subfield
: Refers to a specific subfield. It can have a maximum of eight characters.

subfield type and length
: Specifies the standard assembler language format; e.g., XL3 or CL8, etc.

RESTRICTION: The required OPTION macro-instruction must immediately follow the TERMTBL macro-instruction and be in the same order as the optdata field of the TERM macro-instruction. For example:

Name	Operation	Operand
	TERMTBL	KCHI
TRAFFIC	OPTION	XL4
STAT1	OPTION	XL1
ALTNTERM	OPTION	AL3
POLLIMIT	OPTION	XL1

Note: The actual data values to be inserted into each user optional field are specified in each TERM macro-instruction.

Terminal Table Entry (TERM) Macro-Instruction

The TERM macro-instruction causes a terminal name and its parameters to be included as an entry of the terminal table. The terminal name represents a single terminal or a group of terminals. One TERM macro-instruction is required for each terminal that can transmit a message and each terminal or group of terminals that can receive a message.

Name	Operation	Operand
entry	TERM	$\left\{\begin{matrix}T\\L\end{matrix}\right\}$,dcbname,rln, addressing,[(optdata$_1$, ...,optdata$_n$)]

entry
: Represents the symbolic terminal name. It can have from one to eight nonblank characters, however, all terminal names in the system must have the same length.

T
: Indicates that outgoing messages are to be queued by terminals. Highest priority terminal messages are sent first.

L
: Indicates that outgoing messages are to be queued by communications lines. Messages for all terminals on the line are sent on a first-in-first-out basis. When this option is used, all terminals associated with a line must use the same option.

dcbname
: Indicates the symbolic name of the data control block for the communications line group to which the terminal is attached.

rln
: Specifies the relative line number within the line group to which the given terminal is attached. For a terminal on a dialing line, this number should always be 1.

addressing
: Represents the addressing and polling characters for the terminal (see Table 6). The characters must be specified in the hexadecimal equivalent of the particular device code.

optdata$_1$,...,optdata$_n$
: Represents the actual data to be inserted into the user area subfields of this entry. There must be an exact one-to-one correspondence between the data specified here and the specified type and length allocation made in the OPTION macro-instruction.

Note: A comma is used to separate data belonging to each subfield. Character data

must be delimited by single quotation marks.

Terminal Table List (LIST) Macro-Instruction

The LIST macro-instruction causes a name and a list of terminals that it represents to be included as an entry in the terminal table. One LIST macro-instruction is needed for each distribution list to be supplied.

Name	Operation	Operand
entry	LIST	($entry_1$,$entry_2$,...,$entry_n$)

entry
Represents the symbolic name of the distribution list. It can have from one to eight characters, however, all terminal names in the system must be the same length.

$entry_1$,$entry_2$,...,$entry_n$
Indicates the symbolic terminal name to be included in the distribution list of this entry.

RESTRICTION: Terminal names representing another distribution list may not be included as a list entry. All terminal names in the list must be defined by a TERM or PROCESS macro-instruction.

Terminal Table Process (PROCESS) Macro-Instruction

The PROCESS macro-instruction causes a name that represents a processing program message queue to be included as an entry in the terminal table. One PROCESS macro-instruction is needed for each process queue defined.

Name	Operation	Operand
entry	PROCESS	[EXPEDITE]

entry
Represents the symbolic terminal name. It can have from one to eight characters, however, all terminal names of the system must be the same length.

Table 6. Terminal Polling and Addressing Codes Format

Terminal Type	Format
IBM 1050 Data Communications System IBM 1060 Data Communications System AT & T 83B2 Selective Calling Stations Western Union Plan 115A Outstations	2 Addressing Characters | 2 Polling Characters
IBM 1050 Data Communications System (dial service)	Number of Dial Digits | Dial Digits | 2 Addressing Characters
Common Carrier TWX Stations	Number of Dial Digits | Dial Digits | Number of Identification Digits | Identification
IBM 1030 Data Collection System	2 Addressing Characters

EXPEDITE

When provided, indicates that message segments will be routed directly to the processing program. The EXPEDITE option should not be used when multi-segment messages are expected. If queueing on a direct-access device is specified, the messages are not written on the device and the RETRIEVE macro-instruction may not be used. The data control block used to get messages from this queue must specify RECFM=S. (Refer to the "Data Control Block (DCB) Macro-Instruction.")

Example: Table 7 is an example of the coding sequence that might be used to create the terminal table for use with a system. Instruction number 1 of Table 7 is the TERMTBL macro-instruction, which identifies the last entry in the terminal table.

Instruction number 2 defines a 1-byte field named LIMIT for each terminal identified in the terminal table. This is the field which may be referred to in the POLLIMIT macro-instruction to limit the number of times a terminal is consecutively polled during each polling pass.

Instructions number 3 through 7 cause five terminals and their respective parameters to be recorded in the terminal table. In this example, outgoing messages are to be queued by communications lines. The first operand of each TERM macro-instruction, therefore, is an L. If queueing were by terminal, this operand would be a T.

The second operand of each TERM macro-instruction indicates the symbolic name of the data control block for the line group to which the terminal is attached. In this case, there is one communications line group (hence one data control block).

The third operand is the relative line number. The terminals identified by the codes CHI and BOS are attached to one line, which has a relative line number of 1. The other three terminals (PHI, NYC, and WAS) are attached to another line, which has a relative line number of 2. The fourth operand of each TERM macro-instruction contains the addressing and polling codes (in hexadecimal) for the terminal.

The fifth operand of each TERM macro-instruction contains the actual polling limit to be inserted in the field defined by the OPTION macro-instruction. NYC, CHI and PHI can each be consecutively polled twice during each polling pass. The BOS and WAS entries, however, can only be polled once during each polling pass.

Instruction number 8, the LIST macro-instruction, is used to create a distribution list within the terminal table. This macro-instruction causes a distribution list called PBW to be recorded in the terminal table. The terminals on the list are BOS and WAS. When PBW is listed in a message header as a destination code, messages will be routed to Boston and Washington.

Instruction number 9 of Table 7 is a PROCESS macro-instruction. This instruction defines a process queue called CPU. This queue will contain messages for a message processing program.

Polling List Definition

The order in which terminals are polled on a line is determined by the polling list. Terminal names may be entered in the list as often as desired and in any order. Since a polling list must not refer to terminals on more than one line a list must be defined for each line.

The POLL macro-instruction can be used to create a polling list.

Table 7. Terminal Table Creation Format

No.	Name	Operation	Operand	Comments
1		TERMTBL	CPU	Identifies last entry in the terminal table.
2	LIMIT	OPTION	FL1	Specifies name and size of LIMIT field to be used with the POLLIMIT macro-instruction.
3	CHI	TERM	L,DCBGROUP,1,E407E40D,(2)	Identifies terminals in the system as to type of queueing, name of the data control block for the communications line group, relative line number, address and polling characters (in hexadecimal), and maximum number of consecutive polls.
4	NYC	TERM	L,DCBGROUP,2,E207E20D,(2)	
5	PHI	TERM	L,DCBGROUP,2,E407E40D,(2)	
6	BOS	TERM	L,DCBGROUP,1,E207E20D,(1)	
7	WAS	TERM	L,DCBGROUP,2,E707E70D,(1)	
8	PBW	LIST	(BOS,WAS)	Defines a distribution list called PBW for terminals Boston and Washington.
9	CPU	PROCESS		Defines process queue called CPU to receive messages routed to it.

Polling Table Definition (POLL) Macro-Instruction

The POLL macro-instruction defines the order in which the terminals on a line are to be polled. A given terminal can be specified any number of times in the list. One POLL macro-instruction must be included for each line in the system. If the line is used for output only, the operand field of this macro-instruction must be blank.

Name	Operation	Operand
pollname	POLL	($entry_1$,...,$entry_n$)

pollname
: Represents the name of a line's polling list. The name is also a parameter of the DCB macro-instruction for the communications line group.

$entry_1$,...,$entry_n$
: Specifies the terminal table entry names in the order in which the terminals are to be polled for a given line. Each entry must be the name of a TERM macro-instruction that defines a single terminal. If the line is used for output only, the entry operands must be omitted.

Example: The POLL macro-instruction might be used in the following manner to define an order for polling terminals on each line of the sample system:

Name	Operation	Operand
POLLINE1	POLL	(CHI,BOS)
POLLINE2	POLL	(NYC,PHI,NYC,WAS)

Buffer Assignment

The user must specify the amount of main storage to be used for input and output buffering. This buffer area will be a pool of buffers that are used on a dynamic basis for all communications lines, direct-access queueing devices, and processing queues.

The following macro-instruction is provided for specifying the buffer areas.

BUFFER Macro-Instruction

The BUFFER macro-instruction provides the main storage buffer areas for QTAM and specifies direct-access or main storage queueing.

Name	Operation	Operand
	BUFFER	[dcbname],n,length

dcbname
: Indicates the name of the data control block for the direct-access queues used for queueing of QTAM messages. If main storage queueing is used, this field is blank.

n
: Represents the number of buffers to be reserved.

length
: Represents the length, in number of bytes, of each buffer. All buffers in the buffer pool will be of the same length and should be specified as a minimum size equal to or greater than a message header plus a message header prefix. Buffer length determines the message segment size used in the system. The maximum buffer size is 281 bytes.

Note: The entire header must be contained in the first segment of the message. Where applicable, the entire header must appear before the occurrence of the end-of-block (EOB) character.

Example: The coding in the example below defines 60 buffers, each 125 characters long. Since the header prefix is 37 characters long, the header segment cannot exceed 88 characters (125-37 = 88).

Operation	Operand
BUFFER	DCBNAME,60,125

The actual contents of a buffer used by QTAM is dependent on which type of segments occupy the buffer. These are the header segments and the text segments. Figure 4 illustrates the buffer formats for the header prefix and text prefix; the LPS places the header prefix at the beginning of the header segment of a message, and the text prefix at the beginning of each text segment.

Figure 5 illustrates a simplified representation of header and text segments of three messages. The relationship of messages in a queue, if line queueing is specified, is shown. If terminal queueing is specified, the key identifying message destination will always indicate the same terminal.

DATA SET INITIALIZATION MACRO-INSTRUCTIONS

The data set initialization section of coding for the message control task begins with an OPEN macro-instruction and ends with an ENDREADY macro-instruction. If the user is opening a communications line group and indicates, through the IDLE operand, that the entire line group will not be activated at this point, STRTLN macro-instructions for each of the particular lines to be activated can be included in this section. (The OPEN macro-instruction is described under "OPEN and CLOSE Macro-Instructions"; the STRTLN macro-instruction is discussed under "Message Processing Tasks.")

A single OPEN, group of OPENs, or group of OPEN and STRTLN macro-instructions must be followed by an ENDREADY macro-instruction. ENDREADY is essentially a wait-type instruction; the event is the procurement of the first message. Only one ENDREADY macro-instruction can be included, and it must be the last in the group of initialization instructions.

Operation	Operand
ENDREADY	

LINE PROCEDURE SPECIFICATIONS

QTAM provides macro-instructions that may be selected to define the message-handling procedure for a given line group. A group of macro-instructions selected to accomplish this is called a line procedure specification (LPS). An LPS, which is composed of delimiter and functional type macro-instructions, will be required for each communications line group. When the messages and the terminals of several line groups have similar characteristics, however, more than one can use the same LPS.

The user can specify the macro-instructions within a given LPS in any order that will fit the message header format. This enables the user to design his message format in any way that is convenient for his own application. A restriction, however, is that the entire header portion of the message must be in the first segment. (Refer to "BUFFER Macro-Instruction.") In the case of an IBM 1050 Data Communications System, the entire header must appear before the occurrence of the first end-of-block character.

Header Prefix (in Bytes)

0 1 2 3	4 5 6 7	8 9	10 11	12	13 14 15
Queueing Information	Message Priority and Linkage	Segment Size	Source Key	MSTATUS*	Message Address on Direct-Access Device

16 17 18	19 20 21	22 23 24	25	26 27	28 29	30 31
Next Segment Link	Previous Header Link	Next Message Link	Stored Scan Pointer	Destination Key	Message Sequence Number (In)	Message Sequence Number (Out)

Text Prefix (in Bytes)

0 1 2 3	4 5 6 7	8 9	10 11	12	13 14 15
Queueing Information	Message Priority and Linkage	Segment Size	Source Key	MSTATUS*	Message Address on Direct-Access Device

16 17 18	19 20 21
Next Segment Link	Message Header Link

*Significance of the bits within the MSTATUS byte is as follows:

Bit 0

CANCEL
- 0=chain message
- 1=do not chain

Bit 1

REROUTE
- 0=original copy of message header
- 1=duplicate copy of message header

Bit 2 (unused)

Bit 3

SRVCD
- 0=message not previously serviced (for output message)
- 1=message previously serviced (for output message)

Bit 4

TRUNC
- 0=message complete
- 1=message incomplete

Bit 5

PRIORITY
- 0=message with no priority
- 1=message is sent with priority

Bits 6-7

SEGTYP
- 00=header
- 01=text
- 10=header (EOM)
- 11=text (EOM)

Figure 4. Message Buffer Formats

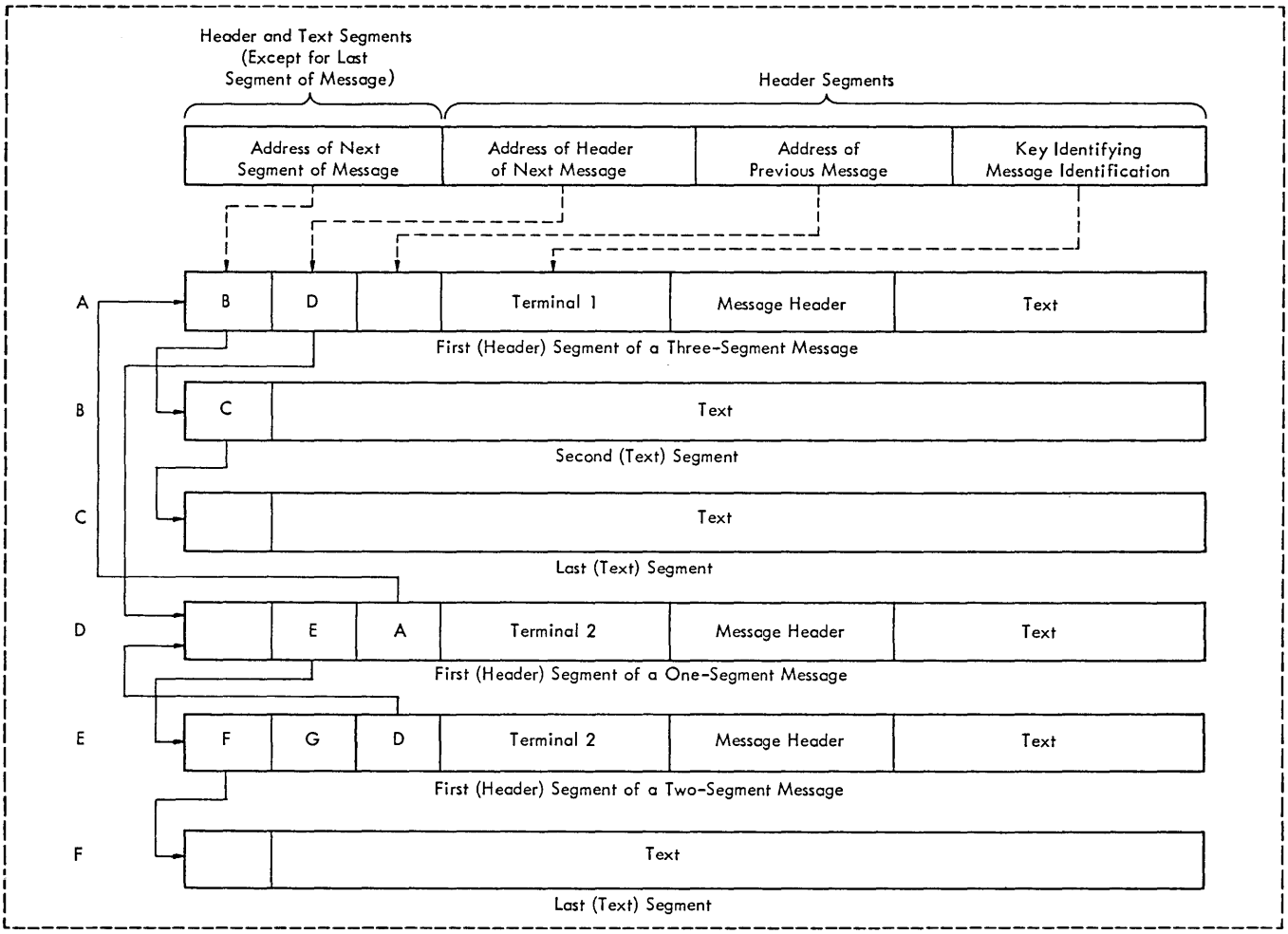


Figure 5. Header and Text Segment Relationships in a Queue

Delimiter macro-instructions are used to logically group the functional macro-instructions. They supply transfer instructions that allow the flow of operation to bypass those functional macro-instructions which do not apply to the given message segment type (header or text) or message direction (receiving or sending).

Functional macro-instructions are those instructions that are selected to satisfy the specific handling requirements of messages directed to the LPS. If a function is needed and is not supplied, the user may supply his own routines.

DELIMITER MACRO-INSTRUCTIONS

Delimiter macro-instructions can be used to identify the type of message segments upon which a group of functional macro-instructions operates. For example, if a group of instructions is intended to operate only on header segments, that group of instructions must be preceded by a delimiter macro-instruction which permits operation only on the header segment.

Table 8 presents an example of the typical order of delimiter macro-instructions within an LPS. The discussion which follows Table 8 presents the delimiter macro-instructions in the order presented in the table. Where functional macro-instructions may not be required, delimiter macro-instructions are indicated in the discussion as optional. Mandatory delimiters indicate the minimum functions required in the LPS.

Table 8. Line Procedure Specification Delimiter Macro-Instructions Order Format

Delimiter Macro-Instructions	Comments
LPSTART	Starting point of the LPS.
RCVSEG . . .	Functional macro-instructions that operate on all segments of an input message. For example, the TRANS macro-instruction could be used here.
RCVHDR . . .	Functional macro-instructions that operate only on the header segment of an input message. For example, the macro-instructions that check on source and destination are used here.
ENDRCVE . . .	Functional macro-instructions that do message error handling.
POSTRCVE	End of input section.
SENDHDR . . .	Functional macro-instructions that operate on the header segment of an output message. For example, the SEQOUT macro-instruction could be used here.
SENDSEG . . .	Functional macro-instructions that operate on all segments of an output message. For example, logging and the translating of the message to the terminal code could be done here.
ENDSEND . . .	Functional macro-instructions that do message error handling.
POSTSEND	End of output section.

Line Procedure Specification Start (LPSTART) Macro-Instruction

The LPSTART macro-instruction is a delimiter that provides an initialization procedure for the LPS. It must be included as the first instruction of the LPS.

Name	Operation	Operand
label	LPSTART	[n]

label
Specifies the symbolic name to be assigned to the LPS. This operand name is the parameter addr specified in the CLPS=addr of the DCB macro-instruction for the line group.

n
Indicates the number of blank characters required for use by the TIMESTMP, DATESTMP, and SEQOUT macro-instructions. If this operand is blank, no space is reserved.

Receive Segment (RCVSEG) Macro-Instruction

The RCVSEG macro-instruction is a delimiter that identifies the beginning of a program sequence that processes both header and text segments of incoming messages.

Operation	Operand
RCVSEG	

Receive Header (RCVHDR) Macro-Instruction

The RCVHDR macro-instruction is a delimiter that identifies the beginning of a program sequence to process the header segment of incoming messages.

Operation	Operand
RCVHDR	

End Receive (ENDRCVE) Macro-Instruction

The ENDRCVE macro-instruction is a delimiter that identifies the beginning of a program sequence to process end conditions for incoming messages.

Operation	Operand
ENDRCVE	

The functional macro-instructions between the ENDRCVE and POSTRCVE macro-instructions are user-selected to test for errors and to control polling if necessary. When such instructions are not selected, this delimiter is not required.

With an IBM 1050 Data Communications System, the user can choose to continue receiving a message after an end-of-block character causes a transmission interruption. When required, the user can also provide line correction procedures. (Refer to "End-of-Block and Line Correction (EOBLC) Macro-Instruction.")

Post Receive (POSTRCVE) Macro-Instruction

The POSTRCVE macro-instruction is a delimiter that identifies the end of a program sequence to process incoming messages. This delimiter is always required in an LPS.

Operation	Operand
POSTRCVE	

Send Header (SENDHDR) Macro-Instruction

The SENDHDR macro-instruction is a delimiter that identifies the beginning of a program sequence to process the header segment of an outgoing message. When this service is not required, this delimiter is not required.

Operation	Operand
SENDHDR	

Send Segment (SENDSEG) Macro-Instruction

The SENDSEG macro-instruction is a delimiter that identifies the beginning of a program sequence to process both header and text segments of outgoing messages.

Operation	Operand
SENDSEG	

End Send (ENDSEND) Macro-Instruction

The ENDSEND macro-instruction is a delimiter that identifies the beginning of a program sequence to process end conditions for outgoing messages.

Operation	Operand
ENDSEND	

The functional macro-instructions between the ENDSEND and POSTSEND macro-instructions are selected by the user for error testing and (if needed) error procedures. When such instructions are not selected, this delimiter is not required.

With an IBM 1050 Data Communications System, the user can choose to continue sending a message after an end-of-block character causes a transmission interruption. (Refer to "End-of-Block (EOB) Macro-Instruction.")

Post Send (POSTSEND) Macro-Instruction

The POSTSEND macro-instruction is a delimiter that identifies the end of a program sequence to process outgoing messages. This delimiter is always required in an LPS.

Operation	Operand
POSTSEND	

FUNCTIONAL MACRO-INSTRUCTIONS

Functional macro-instructions can be used to process message header information. They use a QTAM subroutine to scan the message header and move the characters of header fields into a workarea. Since header scanning is always left to right, the functional macro-instructions must appear in the LPS in an order corresponding to the order of the header fields they service.

A scan pointer used to locate each successive header character is maintained in a general register and is incremented as the header fields are scanned. When the LPS is entered to process a message being received or sent, the scan pointer is automatically initialized.

If the user desires to eliminate the need for header field delimiters within the message header, the user must specify the length of each header field serviced by a functional macro-instruction. When lengths of header fields are specified, the scanner ignores blank characters erroneously embedded within a valid field.

If it is desired to include variable length fields in the message header (for example, when omission of leading zeros is desired in fields that contain numeric data), the end of the field must be defined with a field delimiter. If the length of a field is not specified, it is handled as a variable length field. The scanner treats a blank character erroneously embedded within a variable length field as a field delimiter.

Functional macro-instructions are divided into the following four groups to aid the user in placing them under the correct delimiters:

- Receive, which indicates those macro-instructions that can be placed only under the receive delimiters.
- Receive or send, which indicates those macro-instructions that can be placed under either the receive or the send delimiters.
- Send, which indicates those macro-instructions that can be placed only under the send delimiters.
- Error handling, which indicates those macro-instructions that can be used to handle given error conditions.

Note: For additional information on where each macro-instruction can be used and restrictions that might affect their use, refer to "Summary Charts."

RECEIVE FUNCTIONAL MACRO-INSTRUCTIONS

Sequence In (SEQIN) Macro-Instruction

The SEQIN macro-instruction checks the sequence number of each message from the same terminal as the message arrives in the system. If the sequence number is incorrect, an error flag is set in the error half-word. If the terminals are non-polled devices (e.g., dial devices), the SEQIN macro-instruction, if used, must be preceded by a SOURCE macro-instruction.

Operation	Operand
SEQIN	[n]

n

Specifies the number of characters in the header sequence number field. The maximum value of n is four. If this parameter is not specified a variable length field is assumed. Variable length fields must be followed by a blank character used as a field delimiter.

Note: The sequence number of the first message from a terminal must be 1. The maximum number is 9999.

SOURCE Macro-Instruction

The SOURCE macro-instruction checks the validity of the contents of the source terminal code field received in the message header. If an invalid source terminal code is detected, an error is indicated in the error half-word for the line.

Operation	Operand
SOURCE	n

n

Indicates the (decimal) number of characters in the source terminal code field of the message header. The maximum value of n is eight.

Routing (ROUTE) Macro-Instruction

The ROUTE macro-instruction causes a validity check of the destination code field in the message header. If the destination code is valid, the ROUTE macro-instruction causes the message to be queued for the specified destination. If an invalid destination code is detected in the header field, an error is indicated in the error half-word. This macro-instruction, used in conjunction with the EOA macro-instruction, permits multiple routing.

Operation	Operand
ROUTE	n

n

Specifies the number of characters in the destination code field. The maximum value of n is eight.

The terminal table that includes all valid destination terminal codes must be provided. (Refer to "Control Information.")

A message can be directed to multiple destinations by including more than one terminal code in the destination code field. It is not necessary to state the number of destination codes in the message. An end-of-address character, as described under "End-of-Address (EOA) Macro-Instruction," is used to define the end of the destination field.

Multiple destinations can also be specified by using a single terminal code to identify a distribution list in the terminal table. Each terminal in the distribution list receives the message.

Where special hardware features are available, "group code" transmission is another multiple addressing method that can be used. By using unique address characters, a single message can be sent simultaneously to a prespecified group of terminals on the same line.

Any combination of terminal codes may be used in a header destination field to route a message.

RESTRICTION: Only one ROUTE macro-instruction can be specified for a message type. If the DIRECT macro-instruction is used for the message, the ROUTE macro-instruction cannot be used.

End-of-Address (EOA) Macro-Instruction

The EOA macro-instruction must immediately follow the ROUTE macro-instruction when multiple routing can be expected. This macro-instruction determines if any additional terminals are to receive the message.

Operation	Operand
EOA	C'char'

char

Indicates the EOA character that must appear in the message header after the last destination code.

DIRECT Macro-Instruction

The DIRECT macro-instruction causes a message to be queued for the destination specified in the operand. This macro-instruction may be used in place of the ROUTE macro-instruction when there is no destination data in the message header.

Operation	Operand
DIRECT	{=C'dest'} {field}

dest

Represents the destination code, which may be any terminal name in a terminal table entry.

field

Indicates the symbolic name of an optional field in a terminal table entry; this field, defined by the OPTION macro-instruction, contains the name of the terminal to which the message should be sent. If the source terminal is a non-polled device (e.g., a dial device), the SOURCE macro-instruction must be specified in order for the "field" entry to be used.

Note: Only one DIRECT macro-instruction may be specified for a message type. The DIRECT macro-instruction cannot be used if the ROUTE macro-instruction is used for the same message.

Polling Limit (POLLIMIT) Macro-Instruction

The POLLIMIT macro-instruction is used to determine whether a terminal has sent the maximum number of messages allowed on a single polling pass. When this maximum limit is reached, the next terminal is polled.

Operation	Operand
POLLIMIT	{FL1'n' / field}

n
Specifies a limit for all of the terminals using the LPS. This option can only be used when the number of consecutive polls is the same for all terminals.

field
Specifies the symbolic name of an optional field in the terminal table that contains the limit of consecutive polls for each terminal. This method of specifying the polling limit allows a different polling limit to be set for each terminal.

Note: If no polling limit is set (POLLIMIT macro-instruction is not used), each terminal is polled until the terminal has no more messages for the polling pass.

RESTRICTION: The POLLIMIT macro-instruction has no effect when used with a dialing line.

Halt Receive (BREAKOFF) Macro-Instruction

The BREAKOFF macro-instruction determines if the length of the received message exceeds the specified maximum number of characters (n). It also checks to see if the input buffer is filled with identical characters. If either of these conditions is found, an error flag is set in the error half-word and reception of the message is terminated. (Refer to "Error Handling Functional Macro-Instructions.")

Operation	Operand
BREAKOFF	n

n
Specifies the maximum message length allowed, expressed in characters (n≤32,767).

RECEIVE OR SEND FUNCTIONAL MACRO-INSTRUCTIONS

Date Stamp (DATESTMP) Macro-Instruction

The DATESTMP macro-instruction inserts the date in the header field of incoming or outgoing messages in the form byy.ddd, where b=blank, yy=year, and ddd=day of year (for example, b64.325).

Operation	Operand
DATESTMP	

The user must reserve seven spaces for the date stamp in the buffer. (Refer to "Line Procedure Specification Start (LPSTART) Macro-Instruction.")

Time Stamp (TIMESTMP) Macro-Instruction

The TIMESTMP macro-instruction inserts the time of day in the header field of incoming or outgoing messages in the form bHH.MM.SS.th, where b=blank, HH=hours, MM=minutes, SS=seconds, t=tenths of seconds, and h=hundredths of seconds. When less than twelve spaces are reserved the time will be truncated from the right.

Operation	Operand
TIMESTMP	n

n
Indicates the number of characters to be inserted in the message header stamp. The maximum value of n is twelve. The first character of the time stamp is always a blank. The user must reserve the necessary space for the time stamp in the message buffer. (Refer to "Line Procedure Specification Start (LPSTART) Macro-Instruction.")

RESTRICTION: The TIMESTMP macro-instruction may be used only when the system includes interval timer capability.

SKIP Macro-Instruction

The SKIP macro-instruction causes skipping of either a designated number (n) of nonblank characters, or all characters up to and including a designated character configuration. The SKIP macro-instruction is provided to allow fields in the message header to be skipped during processing.

Operation	Operand
SKIP	[n],[C'chars']

n
: Specifies the number of nonblank characters to be skipped. The maximum value of n is the number of characters remaining in the header.

chars
: Indicates the nonblank character configuration designated to terminate the skip operation. This character configuration must not exceed eight characters.

Example: A SKIP macro-instruction to skip five characters would be:

Operation	Operand
SKIP	5

A SKIP macro-instruction to skip characters up to and including , would be:

Operation	Operand
SKIP	,C','

Message Mode (MODE) Macro-Instruction

The MODE macro-instruction causes execution of a designated function if the next nonblank header character found is the character specified in the char operand, or if the char operand is omitted. Otherwise, control returns to the next macro-instruction in the LPS. The MODE macro-instruction can be used more than once in the same LPS.

Operation	Operand
MODE	{ PRIORITY / CONVERSE / INITIATE / others } ,[C'char']

PRIORITY
: Causes scanning of the header to locate the next nonblank character. This character is the priority value assigned to the message by the subroutine. The sequence of increasing priority is A,...,Z,1,...,9.

CONVERSE
: Causes the line to be placed in the conversational mode. The line is held open until an entire message is received and a reply has been sent. Messages that are already queued for transmission to the line or terminal are not sent during the conversation period.

INITIATE
: Allows segments of an input message to be routed to an output queue before the entire message is received. For multiple-addressed messages, this routing option applies only to the first destination.

others
: Specifies the name of a user-written subroutine.

char
: Represents a specified character. This parameter may be omitted if an unconditional transfer to the designated subroutine is desired.

End-of-Block (EOB) Macro-Instruction

The EOB macro-instruction allows a terminal to continue sending a message after an end-of-block has occurred. This macro-instruction can be used only with an IBM 1050 Data Communications System and must immediately follow the ENDRCVE and ENDSEND delimiter macro-instructions. If the EOB macro-instruction is not supplied, the occurrence of an end-of-block is interpreted as an end-of-transmission.

Operation	Operand
EOB	

End-of-Block and Line Correction (EOBLC) Macro-Instruction

The EOBLC macro-instruction has the same function as the EOB macro-instruction except that it also provides a procedure for line correction if a transmission error is indicated. This procedure provides for a retransmission of the message block that followed the previous end-of-block. The maximum number of retries is three. If the error is not corrected, an error indication is made in the error half-word.

Operation	Operand
EOBLC	

Logging (LOGSEG) Macro-Instruction

The LOGSEG macro-instruction places message segments on a user-specified input/output device. This macro-instruction may be used in the receiving portion of the LPS to log incoming messages, or in the sending portion of the LPS to log outgoing messages.

Operation	Operand
LOGSEG	dcbname

dcbname
Represents the symbolic name of the DCB macro-instruction that the user must supply to define the parameters of the data set necessary for logging segments.

Note: If the LOGSEG macro-instruction is used, the logged message segments are interleaved (i.e., the message segments will be recorded in the log in the order in which they arrive). No attempt is made to group segments of individual messages. This logging is in addition to the queueing procedure of QTAM.

Message Type (MSGTYPE) Macro-Instruction

The MSGTYPE macro-instruction is used when some of the messages on a line require a different LPS sequence from the other messages on the line. It examines the next nonblank character in the message header to determine if the message type character is the same as the character specified in the first operand.

If the characters are the same, the macro-instructions immediately following this macro-instruction are used to process the message. This process is terminated by the next delimiter.

If the character is not the same, a transfer to the next MSGTYPE macro-instruction occurs and the same message type character is compared with the first operand of the new MSGTYPE macro-instruction.

Operation	Operand
MSGTYPE	{=C'type code' {blank

type code
Specifies the character that identifies the message type code.

blank
Indicates that the group of macro-instructions, which immediately follow the MSGTYPE macro-instruction, handle messages that were not identified by the previous MSGTYPE macro-instructions.

Example: Part of a routine using MSGTYPE macro-instructions is shown as:

```
LPS1      LPSTART  4
             .
             .
          RCVHDR
          MSGTYPE  P
message   SOURCE   4     LPS instructions
not type     .            for handling
"P"                       type "P" message
          EOA ------ C'*'
          MSGTYPE
          SEQIN    4     LPS instructions
             .            for handling the
             .            remaining mes-
                          sage types
          ENDRCVE
             .
```

RESTRICTION: The MSGTYPE macro-instruction can only be included after the RCVHDR or SENDHDR macro-instructions.

Translate (TRANS) Macro-Instruction

The TRANS macro-instruction converts the characters of a message segment being received or sent from one code to another. Messages received are translated into an extended binary coded decimal interchange

code (EBCDIC). Those that are sent are translated from this code to that of the destination terminal. Translating is done on a one-to-one character basis.

Operation	Operand
TRANS	table

table
Specifies that the programmer must provide the name of the code conversion table. He may select tables provided by QTAM or define his own. The tables provided by QTAM are:

Tables	Conversion
SEND1050	EBCDIC to 1050
SEND1030	EBCDIC to 1030
SENDT1	EBCDIC to 5-level Baudot
SENDT2	EBCDIC to 8-level TWX
RCVE1050	1050 to EBCDIC
RCVE1030	1030 to EBCDIC
RCVET1	5-level Baudot to EBCDIC
RCVET2	8-level TWX to EBCDIC
RCVF1050	1050 to Monocase EBCDIC

Note: Received messages must be translated into EBCDIC immediately after an RCVSEG macro-instruction, and messages to be transmitted must be translated into the appropriate line code as the last function before an ENDSEND macro-instruction.

The RCVF1050 table permits an IBM 1050 Data Communications System to be effectively used with a 48- or 60-character System/360 installation. The operator keys all input messages in lower case so that no shifts are required between letters and numerals. The RCVF1050 (Receive Folded 1050) table converts upper and lower case letters into upper case EBCDIC.

Numerals and special characters are not modified. If the output must include letter-numeral shifts, the SEND1050 table is used. This mode of operation offers the following three advantages:

- Operator procedure is simplified; case shifts are not required.
- Line time is saved since shift characters are not sent from the terminal.
- Listing at the terminal will show all letters in messages from the terminal in lower case; all letters in messages to the terminal in upper case.

SEND FUNCTIONAL MACRO-INSTRUCTIONS

PAUSE Macro-Instruction

The PAUSE macro-instruction can be used to insert idle characters in an outgoing message upon recognition of a designated character, such as a carriage return. This macro-instruction, if used, must appear after the TRANS macro-instruction.

Operation	Operand
PAUSE	X'char',nX'id'

char
Is the hexadecimal representation of the special character that signals the insertion of the idle character(s). The special character is in the code for the particular terminal device used.

n
Specifies the number of idle characters to be inserted.

id
Indicates the actual transmission code bit configuration of the idle character to be inserted. This is also specified in hexadecimal notation.

Sequence Out (SEQOUT) Macro-Instruction

The SEQOUT macro-instruction is used to number sequentially all outgoing messages for each terminal (or for a group code where applicable).

Operation	Operand
SEQOUT	n

n
Specifies the number of characters to be inserted in the header for the output sequence number. The first character in the output sequence number field is always a blank. The maximum field size that can be specified is five; the range of numbers in the maximum-size field is from 0001 to 9999.

ERROR HANDLING FUNCTIONAL
MACRO-INSTRUCTIONS

After reception or transmission of the entire message is completed, an error half-word for the applicable line indicates whether certain errors have been detected (Figure 6). It is the user's responsibility to check the code (in the error half-word) by using one of the error handling macro-instructions. If errors are detected, the designated action will be taken. An error half-word that contains all zeros indicates a normal completion.

The macro-instructions, for interrogating the error half-word to determine what action should be taken for a given error, are listed below in alphabetical order.

Cancel Message (CANCELM) Macro-Instruction

The CANCELM macro-instruction causes a message to be canceled (the message will not be transmitted) if any of the errors specified by the mask have been detected.

Operation	Operand
CANCELM	X'mask'

mask
: Represents the bit configuration (in hexadecimal) used to test the error half-word.

RESTRICTION: This macro-instruction, if used, must follow the ENDRCVE or ENDSEND macro-instruction delimiters.

Error Message (ERRMSG) Macro-Instruction

The ERRMSG macro-instruction causes an error message to be sent to a designated terminal when any of the errors specified by the mask have been detected.

Operation	Operand
ERRMSG	X'mask',{=C'dest' / field / SOURCE},{=C'errtext' / addr}

mask
: Represents the bit configuration (in hexadecimal) used to test the error half-word.

dest
: Indicates the name of an entry in the terminal table to be used as the destination for the error message.

field
: Indicates the symbolic name of an optional field in a terminal table entry; this field, defined by the OPTION macro-instruction, contains the name of the terminal to which the error message should be sent. If the source terminal is a non-polled device (e.g., a dial device), the SOURCE macro-instruction must be specified in order for the "field" entry to be used.

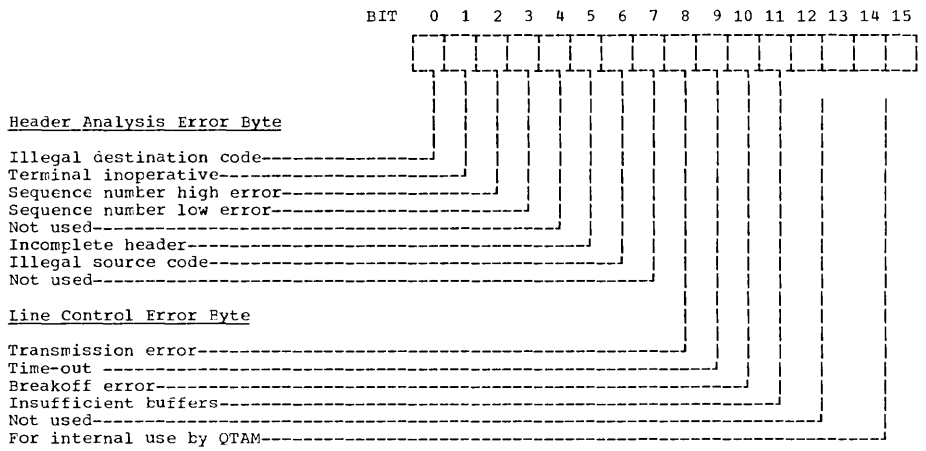
SOURCE
: Indicates that the error message is to be returned to the source terminal. If the source terminal is a non-polled device, 1) the illegal source code bit (bit 6) should not be represented by a one in the bit configuration of the mask; and 2) the SOURCE macro-instruction must be specified.

errtext
: Indicates the actual text of the desired error message. The error message must not be longer than one message segment. If the first character of the message text is a period, QTAM precedes the message with the header of the message in error.

addr
: Indicates the address of the first character of the error message. If the first character of the message text is a period, QTAM precedes the message with the header of the message in error.

Note: If the header of the message in error is to precede the error message, the combined length of the header and error message must not exceed the length of one message segment.

RESTRICTION: This macro-instruction, if used, must follow ENDRCVE or ENDSEND macro-instruction delimiters, or both.



Legend

Illegal destination code (bit 0)
Indicates that the ROUTE macro-instruction found a destination code that was not an entry name in the terminal table. This type of error causes an automatic cancellation of the message for that given destination. If a CANCELM macro-instruction is given for this error condition, any destination codes that follow the invalid one, in the case of multiple-address messages, are also canceled.

Terminal inoperative (bit 1)
Indicates that the message was not sent to its destination because the send bit in the terminal-table entry for the destination terminal was turned off (that is, set to zero).

Sequence number high error (bit 2)
Indicates that the SEQIN macro-instruction found a message sequence number higher than the expected sequence number in the particular terminal entry of the terminal table. When this error is detected, the expected sequence number is unchanged.

Sequence number low error (bit 3)
Indicates that the SEQIN macro-instruction found the message sequence number lower than the expected sequence number in the particular terminal entry of the terminal table. If this message is not canceled, there is more than one message from the given terminal with the same sequence number. When this error is detected, the expected sequence number is unchanged.

Not used (bit 4)

Incomplete header (bit 5)
Indicates the message header did not terminate within the first message segment or prior to the first end-of-block.

Illegal source code (bit 6)
Indicates that the SOURCE macro-instruction found that the message source field was in error. This error occurs when the message source field is not the same as that of the terminal that sent the message or, in the case of a dial system (no polling), the message source field is not an entry name in the terminal table.

Not used (bit 7)

Transmission error (bit 8)
Indicates that an error occurred during transmission (sending or receiving). Such errors as longitudinal and vertical redundancy check cause this bit to be turned on.

Time-out (bit 9)
Indicates that a time-out occurred while sending or receiving. This error should be tested to determine if message rerouting or message intercepting is needed.

Breakoff error (bit 10)
Indicates that the BREAKOFF macro-instruction found a receiving message that either exceeded the maximum allowable message length or in which all of the characters were the same in a buffer.

Insufficient buffers (bit 11)
Indicates that QTAM buffer assignment was not able to provide buffers for an incoming message. This condition, when it occurs infrequently, may be corrected by asking for retransmission. Frequent occurrences of this condition, however, require that QTAM be redefined with more buffers.

Not used (bits 12-13)

For internal use by QTAM (bits 14-15)

Figure 6. Communications Line Error Half-Word As Interrogated by Error Mask

Intercept (INTERCPT) Macro-Instruction

The INTERCPT macro-instruction causes the suppression of all message transmission to a terminal when any of the errors specified by the mask have been detected. The sequence number of the first untransmitted message is stored in the INTERCPT optional field of the terminal table entry, and the intercept bit is set to 1. No messages are transmitted to the terminal until the intercept bit is reset. This may be done by a problem program using the CHNGT macro-instruction, or by the RELEASEM macro-instruction.

Operation	Operand
INTERCPT	X'mask'

mask
: Represents the bit configuration (in hexadecimal) used to test the error half-word.

RESTRICTION: This macro-instruction, if used, must follow the ENDSEND macro-instruction delimiter.

Note: A 2-byte field named INTERCPT must be provided in the optional area of the terminal table. (Refer to the "OPTION Macro-Instruction.") The intercept bit may be reset by a message processing task using the CHNGT macro-instruction or the RELEASEM macro-instruction. If the RELEASEM macro-instruction is used, all intercepted messages are transmitted. If the CHNGT macro-instruction is used, only those messages that have not been bypassed in the queue are transmitted.

REROUTE Macro-Instruction

The REROUTE macro-instruction causes a message to be queued for an alternate destination when any of the errors specified in the mask have been detected.

Operation	Operand
REROUTE	X'mask',{=C'dest' / field}

mask
: Represents the bit configuration (in hexadecimal) used to test the error half-word.

dest
: Indicates the name of an entry in the terminal table to be used as an alternate destination code.

field
: Represents the name of an optional field in a terminal table entry that contains the name of the alternate terminal. If the terminals are non-polled devices, the SOURCE macro-instruction must be specified in order for the "field" entry to be used.

MESSAGE PROCESSING TASKS

It is the user's responsibility to write the message processing tasks for his data communication system. QTAM provides functions to aid in two areas of the message processing task's operation.

- Queue access - macro-instructions provide the ability to obtain messages from and place messages into a message queue.
- Telecommunications system status - macro-instructions provide the ability to examine or change such parts of the communications system as polling lists, terminal table entries, and message queues.

The user must provide all other functions required in his message processing task. All services of System/360 Operating System may be used to aid in the provision of the needed functions. (Refer to IBM System/360 Operating System: Control Program Services.)

QUEUE ACCESS

The main connection between a user's message processing task and the message control task is the message queue. Input (process) queues store incoming messages; output (destination) queues store outgoing messages. Both message control and message processing tasks have access to these queues.

A QTAM user must specify the unit of data with which his processing routines work. The work unit is specified in the data control block associated with the queue. The three options available are: record, message segment, or complete message. A complete message is defined by an

end-of-transmission character. A message segment is defined by a buffer length. A record is defined by either carriage return, line feed, combined carriage return-line feed, or an end-of-block. Record size and segment size have no necessary relationship.

A user may also specify a priority queueing scheme. Instead of queueing messages on a first-in-first-out basis, they may be queued on a first-in-first-out basis within each priority section of the queue (Figure 7).

The priority of each message must be specified in the message header. The MODE macro-instruction is then used in the message control task to check message priority.

Individual macro-instructions provided for access to message queues are as follows.

GET Macro-Instruction

The GET macro-instruction obtains the next sequential segment, record, or message from the input queue associated with the data control block referred to. Reference to an empty queue results in a wait status unless a user's EODAD exit is provided in the associated data control block.

The terminal table entry name of the message source, in the case of a polled terminal, is placed by the GET macro-instruction into the area whose address is contained in the TRMAD field of the data control block. If the source terminal was not a polled type, the terminal table entry name of the message source is placed into the address specified by TRMAD only if the SOURCE macro-instruction was given within the line procedure specification.

Name	Operation	Operand
	GET	dcbname,workarea

dcbname
Specifies the address of the input data control block that contains the parameters necessary to gain access to the desired queue.

workarea
Specifies the address of the area into which the desired segment, record, or message is placed.

Note: The first four bytes in the workarea are either the record, segment, or message prefix. The first two of these bytes specify the number of characters in this record, segment, or message. The third byte specifies the message type (Table 9). The last byte of the prefix is a zero.

When a GET macro-instruction is issued and record work units have been specified in the data control block, the data transferred to the workarea includes all characters through the first carriage return, line feed, combined carriage return-line feed, or end-of-block character.

PUT Macro-Instruction

The PUT macro-instruction places the desired message segment or record into an output queue. The terminal table entry name of the message destination must be in the location specified by the TRMAD parameter.

Name	Operation	Operand
	PUT	dcbname,workarea

dcbname
Specifies the address of the output DCB macro-instruction that contains the parameters necessary to refer to the desired queue.

Sequence into Q	A	B	C	D	E	F	G
Priority	2	3	1	2	3	3	2
Sequence Out of Queue	B	E	F	A	D	G	C

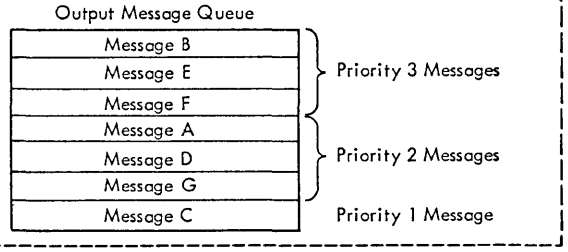


Figure 7. Message Priorities Within a Queue

Table 9. Segment-Type Byte Definition Chart

Segment-Type Byte Contents	Message	Segment	Record
00000000		Header segment of a multiple-segment message	Header record of a multiple-record message
00000001		Intermediate text segment	Intermediate text record
00000010	Complete message	Single-segment complete message	Single-record complete message
00000011		Last text segment of a multiple-segment message	Last text record of a multiple-record message

workarea
Specifies the symbolic name of the area into which the desired segment, record, or message is placed.

Note: The first four bytes in the workarea are either the record, segment, or message prefix. The first two of these bytes specify the number of characters in this record, segment, or message. The third byte specifies the message type (Table 9). The last byte of the prefix is a zero.

RETRIEVE Macro-Instruction

The RETRIEVE macro-instruction causes a message segment to be retrieved from the specified queue and placed into a user-provided workarea. The queue must be located on a direct-access storage device.

Name	Operation	Operand
	RETRIEVE	{source terminal / destination terminal / direct-access device address}, workarea, {IN / OUT / blank}, {number / blank}

source terminal
Specifies the symbolic name assigned to a message source point, whether it be a single terminal device or a process program. Either this parameter or the "destination terminal" parameter must be provided if retrieval is to be made by input or output sequence number.

destination terminal
Specifies the address of an area that contains the name of the message destination point, whether it be a single terminal device, a group code, or a process program. (It may not be the name of a distribution list.) Either this parameter or the "source terminal" must be provided if retrieval is to be made by input or output sequence number. Only the first segment of the message is retrieved. This parameter may be expressed as a symbolic name or as a register designation, where the register contains the address of the name.

direct-access device address
Indicates the relative record address of the segment on the direct-access storage device. This parameter must be provided if retrieval is to be made by direct-access device address. This parameter must be expressed as a register designation, where the register contains the 3-byte record address.

workarea
Indicates the address of the user-provided workarea into which the segment is placed. This parameter may be expressed as an explicit effective address, a symbolic relocatable address, or a designation of a register containing the address.

IN
Indicates retrieval is to be by input sequence number.

OUT
Indicates retrieval is to be by output sequence number.

blank
Represents the parameter left blank if retrieval is to be by direct-access device address.

number
Represents actual input or output sequence number if retrieval is to be by sequence number.

blank
Represents the parameter left blank if retrieval is to be by direct-access device address.

Note: Two methods are used to identify the message segment to be retrieved:

- The terminal name and the input or output sequence number may be specified, in which case the first segment (header segment) of the message is retrieved. The header prefix contains the direct-access device address (relative record address) of the next segment if it is a multisegment message.
- The relative record address of the segment may be specified. Using the direct-access device address provided in the header segment by the first method, the second segment may be retrieved. Each additional segment may then be retrieved by specifying the direct-access device address provided in the previous segment.

Release Intercept (RELEASEM) Macro-Instruction

The RELEASEM macro-instruction causes transmission to a terminal of all messages that have been intercepted for that terminal. (Refer to "INTERCPT Macro-Instruction.") The intercept bit in the terminal table entry is set to zero.

Name	Operation	Operand
	RELEASEM	termname

termname
Represents the terminal name that identifies the terminal table entry.

TELECOMMUNICATIONS SYSTEM STATUS

The following macro-instructions enable the user to: 1) examine polling lists, terminal tables, and message queues; 2) change polling lists and terminal tables; and 3) activate or deactivate a particular line in a communications line group.

Copy Polling List (CPYPL) Macro-Instruction

The CPYPL macro-instruction copies the polling list of a specified line into a specified workarea.

Name	Operation	Operand
	CPYPL	dcbname,rln,workarea

dcbname
Specifies the name of the data control block for the communications line group containing the particular line.

rln
Specifies a register that contains the relative line number of the communications line.

workarea
Specifies the address of the workarea in which the polling list is to be placed. The size of the workarea that must be provided can be determined from the polling list format shown in Figure 8.

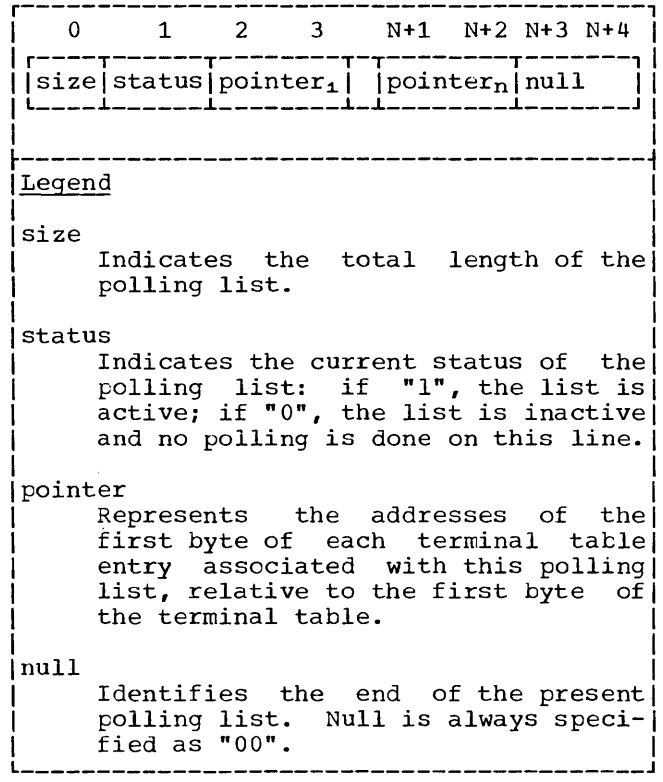


Figure 8. Polling List Formats

Change Polling List (CHNGPL) Macro-Instruction

The CHNGPL macro-instruction places a new polling list into the polling list area specified for the line or changes the line status with regard to polling.

Name	Operation	Operand
	CHNGPL	dcbname,rln,{workarea / =C'0' / =C'1'}

dcbname
: Specifies the name of the data control block that provides control information that pertains to a line polling list.

rln
: Specifies a register that contains the binary relative line number within the communications line group for which the polling list is defined.

workarea
: Specifies the address of the area that contains the new polling list. The new polling list must not contain more entries than the original polling list. It must be of the format shown in Figure 8. If the new polling list is larger, it is not used. Bit 1 of the first byte of the workarea, however, is used as a return code to indicate the error.

0
: Conditions the line not to receive incoming messages.

1
: Conditions the line to start polling with the previously specified polling list.

Note: For common carrier TWX networks, only the condition change can apply (i.e., 0 or 1 can be specified).

Start Line (STRTLN) Macro-Instruction

The STRTLN macro-instruction can be used to: 1) allow message transmission to commence or resume on a particular line in a communications line group; and 2) allow message transmission to occur on all lines in a communications line group. The user must previously have issued an OPEN macro-instruction for the line group.

If the IDLE entry is specified in the OPEN macro-instruction for the line group, the STRTLN macro-instruction must be issued before initial message transmission can take place. If the IDLE entry is not included in the OPEN for the line group, STRTLN is not required for initial message transmission. If a line is deactivated by a STOPLN macro-instruction, STRTLN must be issued before message transmission on that particular line can resume.

In all the above cases, if polling is used, the presence of an active polling list is a prerequisite for message transmission. (An active polling list is one in which the second byte of the list is a non-zero character -- this character is initialized as a 1 and can be changed by the CHNGPL macro-instruction.) If STRTLN is used, polling begins after the execution of that macro-instruction. If the IDLE entry is not specified in the OPEN for the line group, polling begins following the execution of the OPEN macro-instruction.

Name	Operation	Operand
[symbol]	STRTLN	dcbname,{rln / ALL}

symbol
: Specifies the name for the STRTLN macro-instruction.

dcbname
: Specifies the name of the data control block for the communications line group containing the line to be activated.

rln
: Specifies, in decimal, the relative line number of the line to be activated. If this number is enclosed in parentheses, it is interpreted as specifying the number of a register containing, in binary form, the relative line number of the line to be activated.

ALL
: Specifies that all the lines in the communications line group should be activated.

Stop Line (STOPLN) Macro-Instruction

The STOPLN macro-instruction removes a communications line from active use. This macro-instruction has the following effects:

Operations on the line designated are stopped. If the line was conditioned by a CHNGPL macro-instruction to stop receiving, all messages queued for transmission on the line are transmitted before control is returned to the program issuing the STOPLN macro-instruction. If the line was not conditioned to stop receiving, control is returned to the program issuing the STOPLN macro-instruction immediately after the completion of any message being transmitted. In the latter case, transmission of any messages remaining in the queue for the line resumes when a STRTLN macro-instruction reactivates the line.

Name	Operation	Operand
[symbol]	STOPLN	dcbname,rln

symbol
: Specifies the name for the STOPLN macro-instruction.

dcbname
: Specifies the name of the data control block for the communications line group containing the line to be deactivated.

rln
: Specifies, in decimal, the relative line number of the line to be deactivated. If this number is enclosed in parentheses, it is interpreted as specifying the number of a register containing, in binary form, the relative line number of the line to be deactivated.

Copy Terminal Table (COPYT) Macro-Instruction

The COPYT macro-instruction causes the information contained in a specified entry of the terminal table to be moved into a designated area.

Name	Operation	Operand
	COPYT	termname,area

termname
: Specifies the name of the terminal with which the entry is associated. An invalid name will cause the macro-instruction to be terminated. Bit 0 of the first bytes in the workarea will be made a return code to indicate the error.

area
: specifies the address of the area into which the information is to be placed. The first byte of the area is reserved for a return code. The table data always starts in the second byte.

The format of the terminal table entry copied by a COPYT macro-instruction depends on whether it is a single entry terminal, a group code entry, a distribution list entry, or a process program entry.

Since the maximum size of a terminal table entry is 255 bytes, the maximum size of the workarea that must be provided for a type of entry is 256 bytes (255 bytes plus return code byte). The user may, however, conserve storage space by determining the actual size of each entry type from the terminal table entry formats illustrated in Figure 9 and provide workareas for those sizes only.

Change Terminal Table (CHNGT) Macro-Instruction

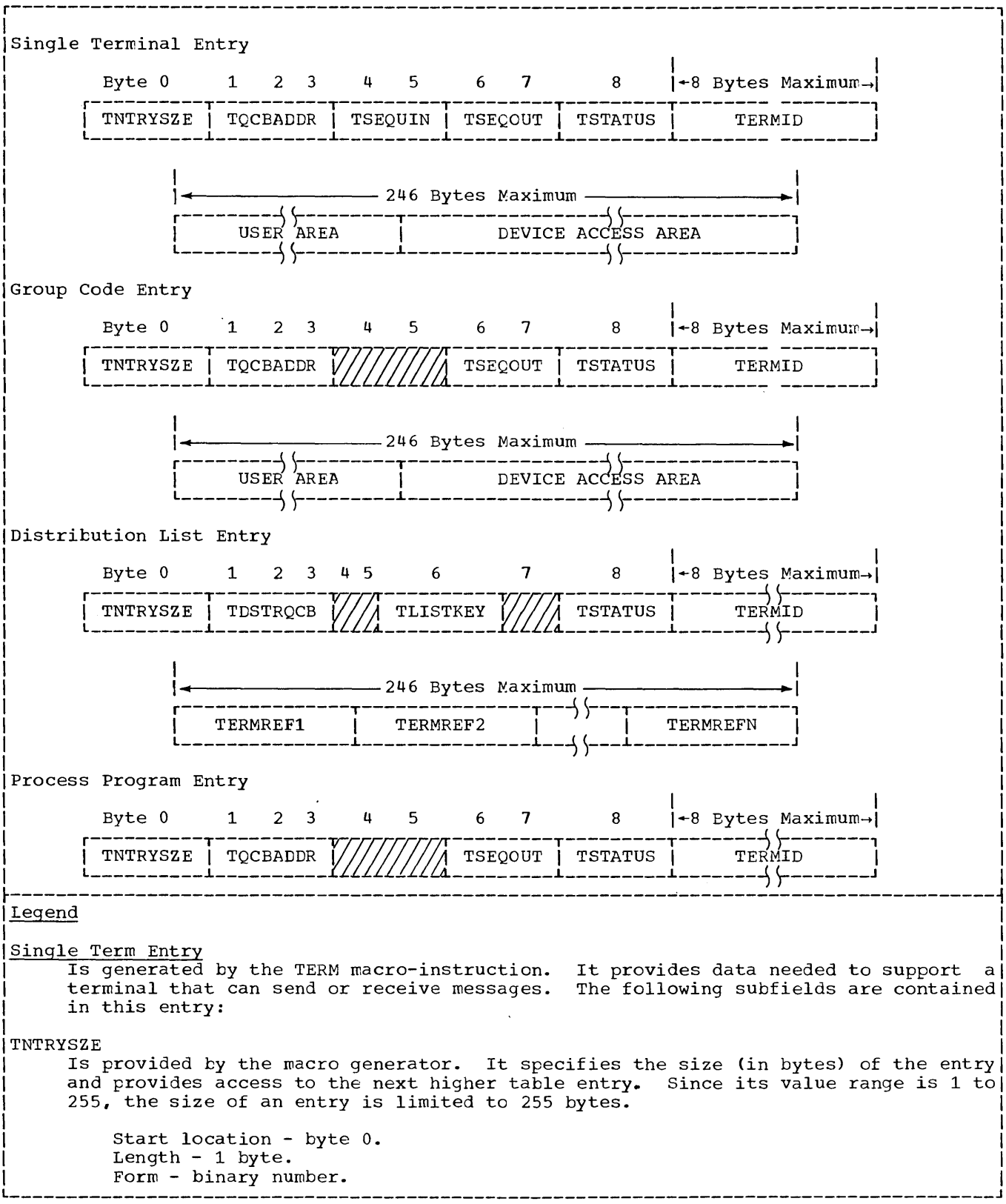
The CHNGT macro-instruction causes a complete terminal table entry to be moved from a designated area into a specified entry of the terminal table.

Name	Operation	Operand
	CHNGT	termname,area

termname
: Specifies the terminal name with which the entry is associated. An invalid terminal name causes the macro-instruction request to be terminated. Bit 0 of the first byte in the workarea is used as a return code to indicate the error.

area
: Specifies the address of the area from which the information is to be moved. The first byte of the area is reserved for a return code. The new table data must always start in the second byte.

Note: The CHNGT macro-instruction causes the entire contents of a current entry to be overlaid. It is normally preceded by a COPYT macro-instruction and some modification to the copied entry. If the new entry is larger than the current one, it is not used and bit 1 of the first byte of the workarea is used as a return code to indicate the error.



Legend

Single Term Entry
Is generated by the TERM macro-instruction. It provides data needed to support a terminal that can send or receive messages. The following subfields are contained in this entry:

TNTRYSZE
Is provided by the macro generator. It specifies the size (in bytes) of the entry and provides access to the next higher table entry. Since its value range is 1 to 255, the size of an entry is limited to 255 bytes.

Start location - byte 0.
Length - 1 byte.
Form - binary number.

Figure 9. Terminal Table Entry Formats (Sheet 1 of 4)

TQCBADDR
Is provided by QTAM. It contains the address of the control block that controls the queue in which outgoing messages to this terminal are stored. Using this address identification, the queueing routine places each message into its appropriate outgoing queue.

Start location - byte 1.
Length - 3 bytes.
Form - binary address.

TSEQUIN
Is provided by QTAM to store and maintain a master sequence number for incoming messages from this terminal. It is initially set to a value of 1.

Start location - byte 4.
Length - 2 bytes.
Form - binary count.

TSEQOUT
Is supplied by QTAM to provide a master sequence number for messages being sent to this terminal. It is initially set to the value of 1.

Start location - byte 6.
Length - 2 bytes.
Form - binary count.

TSTATUS
Is used to indicate various communication conditions associated with the entry.

Start location - byte 8.
Length - 1 byte.
Form - binary status bits.

Bits 0 through 4 (not used).
Bit 5 The intercept bit is initially set to a value of 0. The bit is set to 1 when the user specifies the INTERCPT macro-instruction. This indicates that a message in the queue has been skipped as a result of a transmission failure. It is turned off by the user (CHNGT macro-instruction) when transmission can be resumed, or by the RELEASEM macro-instruction.
Bit 6 The send bit is initially set to a value of 1. This setting indicates that a message may be sent to the terminal. It is set to 0 when the user specifies the INTERCPT macro-instruction. The bit is set to 1 again when the user specifies the CHNGT macro-instruction, or by the RELEASEM macro-instruction.
Bit 7 The receive bit is initially set to a value of 1. This indication means that the terminal may be polled for a message. The user can prevent the polling of a terminal by setting the bit to 0 by specifying the CHNGT macro-instruction. Polling is resumed when the bit is set to 1 again.

TERMID
Indicates the terminal name of the entry that must be specified by the user as a macro-instruction operand. The name of the entry is also the terminal code that may appear in the source or destination field of the message header.

Start location - byte 9.
Length - 1 to 8 bytes.
Form - BCD characters.

Figure 9. Terminal Table Entry Formats (Sheet 2 of 4)

USER AREA

The storage allocation for each optional subfield that comprises the user area is created by repeated contiguous specification of the OPTION macro-instruction. There must be one OPTION macro-instruction specified for each required subfield. They must immediately follow the TERMTBL macro-instruction. The subfield storage allocation requested is repeated in each terminal table entry created by a TERM macro-instruction. A user area for the other entry types cannot be specified.

Data to be inserted into each subfield of the user area must be specified by the user in the relevant operand of each TERM macro-instruction. The data specified must have a one-to-one correspondence with each requested subfield. The user area could be used for such data as alternate terminal code, polling limit parameter, diagnostic information, traffic data, or status bits.

Start location - dependent upon the specified length of the TERMID subfield. Symbolic references may be made to the optional subfields named by the user for this area.
Length - variable as required to store the optional subfields specified.
Form - as specified for each subfield.

DEVICE ACCESS AREA

Contains the address characters needed to poll and address the terminal device. They are specified by the user in the addressing operand of the TERM macro-instruction.

Start location - address of the first byte that follows the last byte of the user area.
Length - variable as required to store the user-defined address data.
Form - transmission code.

Group Entry Code

Is generated by the TERM macro-instruction. The entry created is identical to that of a single terminal with the following considerations:

The terminal name specified by the user represents a prespecified group of terminals on a line that is equipped with special equipment. This feature permits a message to be sent simultaneously to a group of terminals by specifying a single set of unique address characters. Several combinations of prespecified terminals can be grouped for this purpose. Each group is given a group terminal name and a corresponding group code terminal table entry.

Note: This type of line can also be sent messages for a single terminal when ordinary address characters are used.

The master sequence number for outgoing messages (subfield TSEQOUT) is incremented by one when the group is simultaneously sent a message. If any terminal in the group is also associated with a single terminal entry, the message number of that entry is not changed.

The master sequence number for incoming messages (subfield TSEQUIN, bytes 4 and 5, in the single terminal entry) is not applicable to the group code entry because the terminal group cannot collectively send a message to the system. For the same reason, there are no polling characters in the device access area of the entry, and the receive bit of the TSTATUS subfield is never interrogated.

Figure 9. Terminal Table Entry Formats (Sheet 3 of 4)

Distribution List Entry

Is generated by the LIST macro-instruction. The terminal name of the entry represents a list of terminals specified within the entry. When the entry terminal name is used as a destination code, the associated message is sent to all terminals of the list via separate transmissions. Each terminal of the list, therefore, must have a corresponding single terminal entry. The following subfields are valid for this entry:

TNTRYSZE (same as single terminal entry).

TDSTRQCB

Is the same as TQCBADDR in the single terminal entry, except that this address identifies a control block that controls a queue used for all distribution list messages.

TLISTKEY

Provides the access key to start the terminal list (subfield TERMREF). It is a relative address value, relative to byte 0 of the entry.

Start location - byte 6.
Length - 1 byte.
Form - binary number.

TSTATUS

Is the same as the single terminal entry, except that the receive bit (bit 7) is never referred to because the terminals of the list cannot collectively send a message to the system.

TERMID

Is the same as the single terminal entry, except that the terminal name represents the list of terminals in the entry.

TERMREF

Specifies that each terminal is listed as a relative address that locates the entry of that terminal in the terminal table. The address is relative to the base address of the table. The high order bit of the last TERMREF entry will be a 1 indicating the end of the TERMREF list.

Start location - follows TERMID. (See description of TLISTKEY above.)
Length - 2 bytes for each terminal of list.
Form - binary relative address.

Process Program Entry

Is generated by the PROCESS macro-instruction. The terminal name of this entry represents a message queue associated with a message processing program. The following subfields are valid for this entry:

TNTRYSZE (same as single terminal entry).

TQCBADDR (same as single terminal entry).

TSEQOUT (same as single terminal entry).

TSTATUS

Is the same as the single terminal entry, except that the receive bit (bit 7) is not applicable.

TERMID

Is the same as the single terminal entry, except that the terminal name specified identifies a message queue for a message processing program. This name is used as a header destination code to address a message to the queue.

Figure 9. Terminal Table Entry Formats (Sheet 4 of 4)

Copy Queue Status (CPYQ) Macro-Instruction

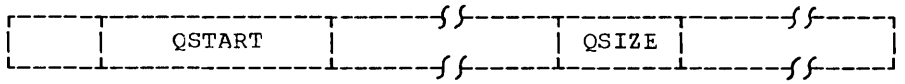
The CPYQ macro-instruction places information that concerns the status of a specified message queue into a workarea.

Name	Operation	Operand
	CPYQ	name,workarea

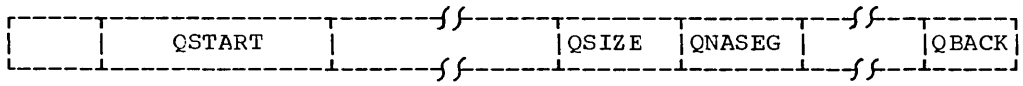
name
Specifies the address of an entry in the terminal table. Only terminal table entry names of TERM or PROCESS macro-instructions may be specified in the CPYQ macro-instruction. An invalid name causes the macro-instruction request to be terminated. Bit 0 of the first byte in the workarea is used as a return code to indicate the error.

workarea
Specifies the address of the area designated to receive the queue status information. The first byte of the area is reserved for a return code. The queue status information always starts in the second byte. The workarea provided must be 7 bytes or 13 bytes long, depending on the type queue to be copied (main storage queue or direct-access queue). The format of the queue status information is illustrated in Figure 10.

Main Storage Queue



Direct-Access Storage Device



Legend

QSTART
Indicates the starting address in main or direct-access storage of the first message in the queue; i.e., the next message to be processed or transmitted. For direct-access storage devices, the address represents the relative record number within the queueing area.

QSIZE
Indicates the number of messages in the queue.

QNASEG
Indicates the starting address, on a direct-access storage device, where the next message for this queue will be placed.

QBACK
Indicates the starting address, on a direct-access storage device, of the last message processed or transmitted from this queue.

Figure 10. Queue Status Information Formats

Note: If the line queueing option is specified in the terminal table entry, the queue status information copied reflects the status of the queue for the entire line.

MESSAGE PROCESSING TASK APPLICATIONS

In addition to writing message processing tasks, a user must perform the following:

- Define all queues used by his program. This is done at assembly time with the DCB macro-instruction.
- Open each queue used by his program with an OPEN macro-instruction, before any reference to the queue is made.
- Close each queue when no further reference to the queue is necessary.

Figure 11 illustrates a possible structure for a user program. It is assumed that no EODAD parameter was specified in the DCB macro-instruction for the data control block referred to in the GET macro-instruction. Thus, if no messages are in the queue, the program is placed in a wait status and is reentered only when additional messages arrive.

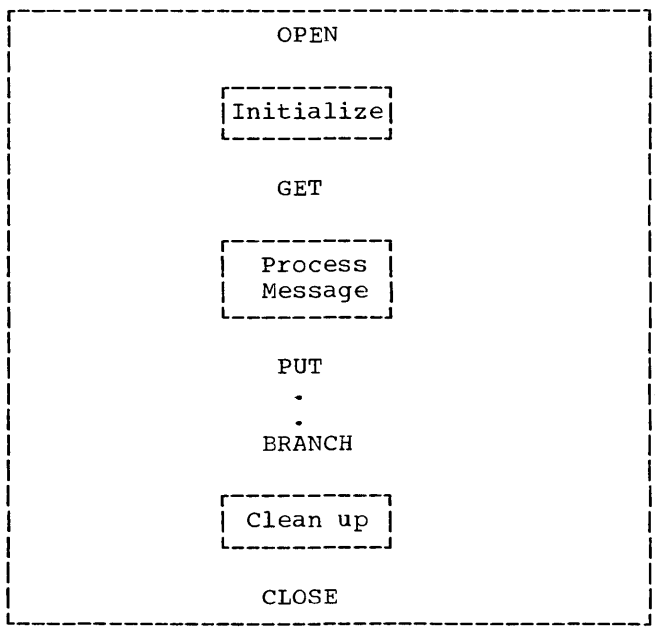


Figure 11. Possible Structure for a User Program

A user program of the structure illustrated in Figure 11 is a task. The "process message" section of the structure may contain any operations available in System/360 Operating System. For additional information regarding task scheduling within the operating system, refer to the publication IBM System/360 Operating System: Concepts and Facilities.

Message processing tasks are considered to be in the following basic areas:

- Data collection application.
- Message switching application.
- Inquiry application.
- Operator control.
- Remote stacked job application.

DATA COLLECTION APPLICATION

In a data collection application, messages are sent from remote terminals to the central system (Figure 12). There they are accumulated to be batch-processed.

Since the input queues specified in the "Message Control Task" section of QTAM may be located on a direct-access storage device, they may also be retained and processed later, thus fulfilling the data collection application. Another method of accumulating this data within QTAM is by use of the LOGSEG macro-instruction to accumulate data on a separate input/output device.

The program used to process the data collected is scheduled as a separate task through the task scheduler.

MESSAGE SWITCHING APPLICATION

In a message switching application, messages are sent from a remote terminal that has as its destination other terminals or groups of terminals. These messages require no intermediate processing (Figure 13).

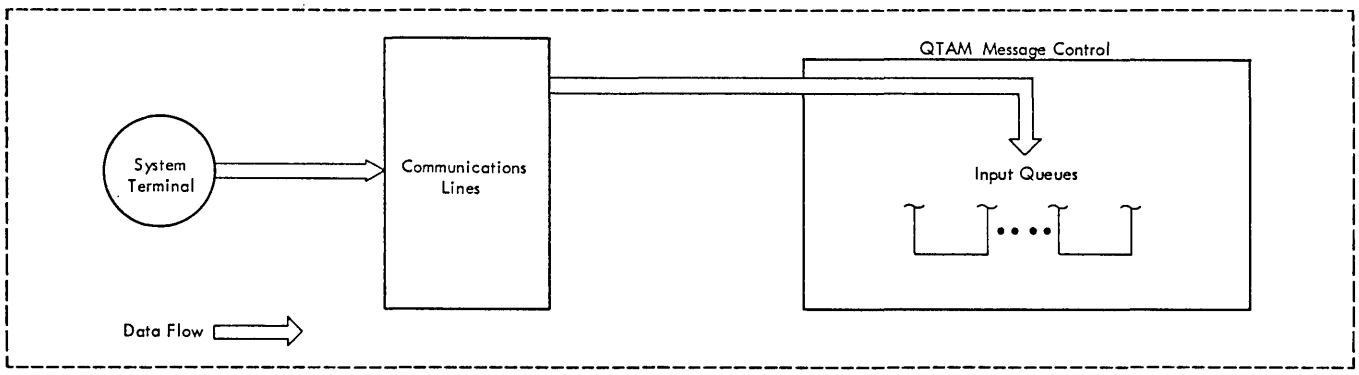


Figure 12. Data Collection

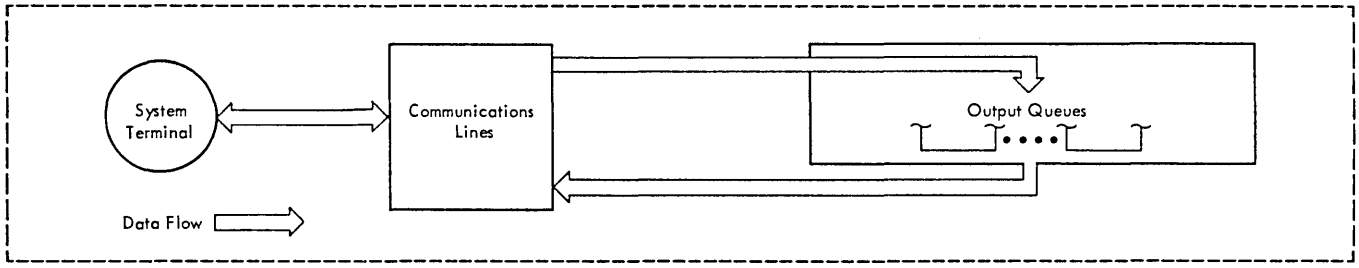


Figure 13. Message Switching

INQUIRY APPLICATION

In an inquiry application, messages are sent from remote terminals. The messages contain data to be processed, and require a reply to be sent back to the terminal. The programs used to process the messages may or may not be resident in main storage.

If desired, an inquiry application can be operated in a conversational mode by inclusion of the CONVERSE parameter in the MODE macro-instruction in the LPS. In this mode, a terminal transmitting a message into the system is held on-line until the response is generated and sent from the central system.

Figure 14 illustrates two basic methods of incorporating user-provided processing programs. With either method, the structure of the user program is as shown. The only difference is the way in which the message is processed after it is obtained from a queue.

The left-hand side of Figure 14 represents a case in which the processing program issues the GET and PUT macro-instructions. This would most likely be a short processing program that is resident in main storage.

A more involved situation is shown on the right-hand side of Figure 14. In this case, the program issuing the GET and PUT macro-instructions links (via the LINK macro-instruction) to another program. As shown in the diagram, the program may use data from input/output devices for processing the message, provided the required data management access method is included in the operating system. (Refer to the publication IBM System/360 Operating System: Data Management.)

The reply message in either case is routed to a terminal (usually the terminal sending the input message) by use of the PUT macro-instruction which places the message in the appropriate output queue. QTAM provides for the actual sending of the message.

If both unsolicited messages and inquiry responses are directed to the same terminal, the order in which they are transmitted is unpredictable.

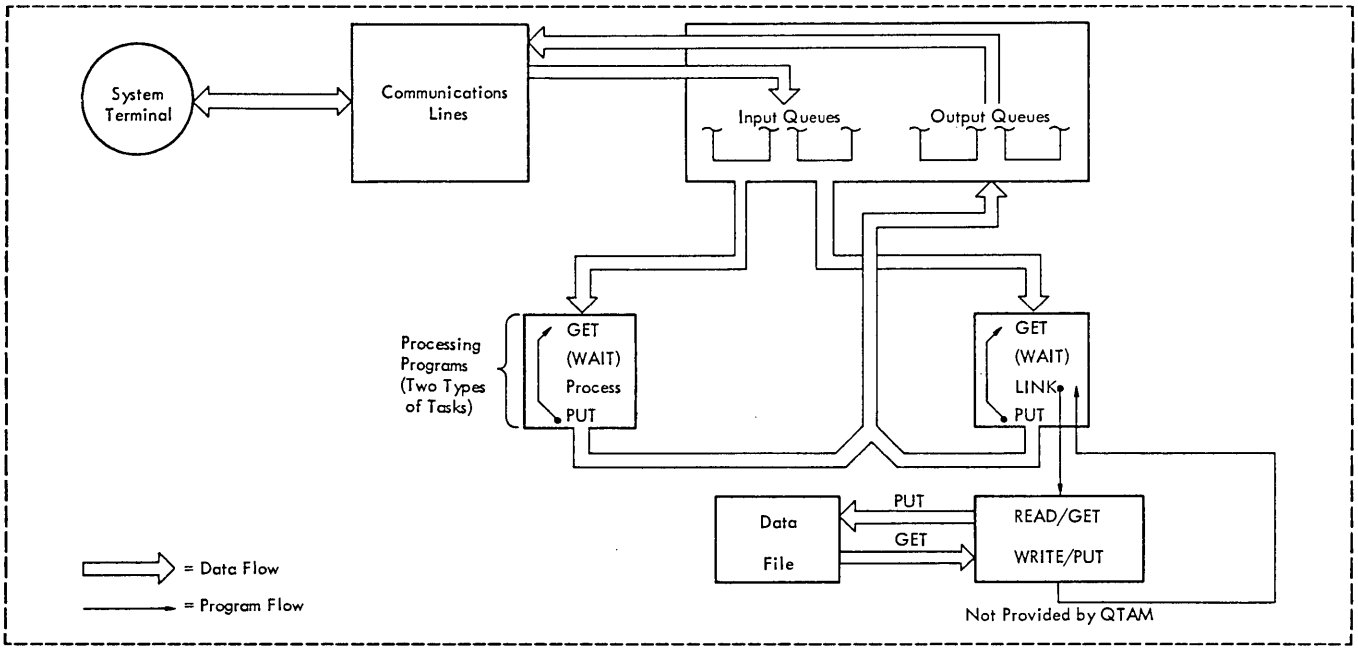


Figure 14. Inquiry Application

OPERATOR CONTROL

Control messages may be entered into the system through a local terminal to obtain and modify the status of the telecommunications system.

The user may provide a task to analyze operator requests that may examine or dynamically modify the status of the telecommunications system (message control task). Upon input of such requests, telecommunications system status macroinstructions can be used within user-provided routines to obtain the system status and generate a report to the operator, or effect a change in the system as specified by the operator.

As shown in Figure 15, the operator request can be entered into the system by a local terminal. The operator request is then handled in the same manner as in the inquiry application.

REMOTE STACKED JOB APPLICATION

A remote stacked job application involves messages that contain programs, or programs and data. These programs may require:

- Execution.
- Compilation or assembly before execution.
- Compilation or assembly only.

An example of a remote stacked job application is the remote entering of a FORTRAN source deck followed by a data deck. This application is identical to a normal stacked job, except the reception and transmission of the job is via communications lines.

System/360 Operating System provides the user with the ability to bring remote stacked jobs into the system, to execute them, and to transmit job output to remote terminals. Job output is independent of the source of input; i.e., any terminal which has been defined in QTAM can receive output from any job which specifies that terminal. Routing of job output to a terminal or local device is controlled by the disposition field of a DD card.

A remote stacked job application is handled in the following manner:

1. The QTAM message control task is assembled by the user with one or more LPS's capable of routing stacked job input to the processing queue which the user has defined. No special processing is required other than translating and routing. The processing queue has no special attributes

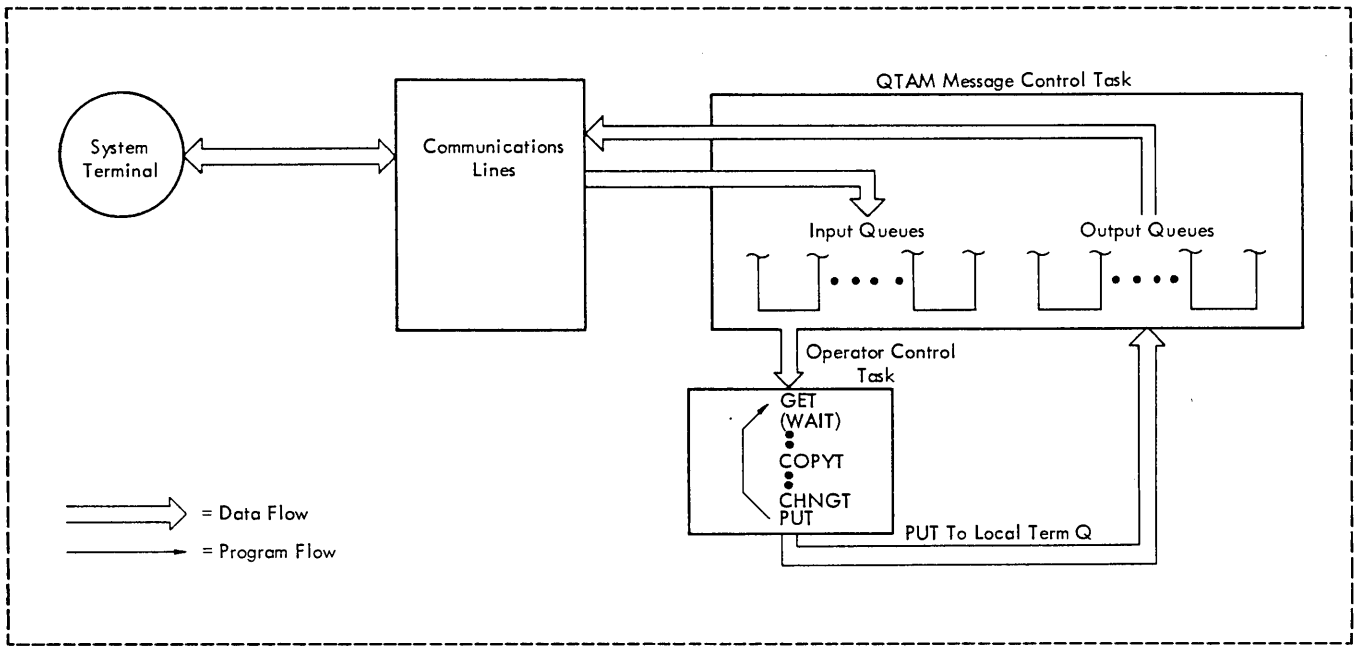


Figure 15. Operator Control

other than its name. No special processing of job output is required other than translating; however, the user may supply any functions that he desires within the SEND LPS structure.

2. The QTAM message control task is introduced into the system through the normal input stream.
3. A Tele-processing reader/interpreter, which uses input from the process queue, is attached in the normal manner for a reader. (Refer to the publication IBM System/360 Operating System: Operating Considerations.)
4. Remote stacked jobs are entered through any of the terminals attached to the system. Normal job input format must be used, except that non-significant columns at the end of cards may be omitted; however, the records must be formed into 80-character card images if this is done. The end of a remote stacked job input is indicated by an end-of-transmission.
5. The job is scheduled for execution by the job scheduler.
6. A Tele-processing output writer, which dispatches output to a terminal through QTAM, is attached in the normal manner for a writer. (Refer to the publication IBM System/360 Operating System: Operating Considerations.)

The following considerations should be noted:

1. QTAM must be in the system whenever a Tele-processing reader/interpreter or a Tele-processing output writer is attached, or when a job specifying a terminal is executed.
2. QTAM can stack remote jobs indefinitely; i.e., a Tele-processing reader interpreter need be attached only when execution is desired.
3. The Tele-processing reader/interpreter and the Tele-processing output writer can be detached in any of the following ways:
   - By a command from a remote terminal, which is routed to the process queue for the reader/interpreter.
   - By a command from the system console.
   - By a command to another reader/interpreter.

SUMMARY CHARTS

Summary charts list all macro-instructions that can be used with QTAM. They show where each macro-instruction can be used and note any restrictions that might affect their use. The macro-instructions are listed in alphabetical order within each chart so that they may be used for reference. The summary charts are divided into the following categorized macro-instruction charts:

- Telecommunications system specification macro-instructions (Table 10), which include those macro-instructions that must be specified for both the message control and message processing tasks.
- Message control macro-instructions for specifying line procedure specifications (Table 11). This table lists all the delimiter and functional macro-instructions. The functional macro-instructions are identified with the appropriate delimiter macro-instructions that they can follow. Macro-instructions, which are delimiters, are indicated in the last column of the table.
- Message control macro-instructions for specifying QTAM control information (Table 12), which includes those macro-instructions needed to provide detailed information in the form of tables and lists and to provide buffers for message handling.
- Message processing macro-instructions (Table 13), which includes those macro-instructions that may be specified within the user's message processing programs to communicate with the message control task. These macro-instructions are separated into queue access and telecommunications system status categories.

Table 10. Telecommunications System Specification Macro-Instructions

Name	Operation	Operand	Message Control Task	Message Processing Task
[symbol]	CLOSE	(dcbname$_1$,,...dcbname$_n$) [,{MF L MF (E,listname)}]	x	x
diskqueue dcbname	DCB	keyword parameters	x	
linegroup dcbname	DCB	keyword parameters	x	
log dcbname	DCB	keyword parameters	x	
process dcbname	DCB	keyword parameters		x
[symbol]	OPEN	(dcbname$_1$,[({INPUT OUTPUT INCUT}[,IDLE])],..., dcbname$_n$,[({INPUT OUTPUT INCUT}[,IDLE])]) [,{MF=L MF=(E,listname)}]	x	x

Table 11. Message Control Macro-Instructions for Specifying Line Procedure Specifications

Name	Operation	Operand	Functional Macro-Instructions and the Delimiter Macro-Instructions Each May Follow: RCVSEG	RCVHDR	ENDRCVE	SENDSEG	SENDHDR	ENDSEND	DELIMITERS
	BREAKOFF	n	x						
	CANCELM	X'mask'			x				
	DATESTMP			x			x		
	DIRECT	dest		1					
	ENDRCVE								x
	ENDSEND								x
	EOA	C'char'		2					
	EOB				3			3	
	EOBLC				3			3	
	ERRMSG	X'mask',{=C'dest' / field / SOURCE},{=C'errtext' / addr}			x			x	
	INTERCPT	X'mask'						x	
	LOGSEG	dcbname	x	x		x	x		
label	LPSTART	[n]							4
	MODE	{PRIORITY / CONVERSE / INITIATE},[C'char']							
		others		x			x		
	MSGTYPE	{=C'type code' / blank}		x			x		
	PAUSE	X'char',nX'id'				x			
	POLLIMIT	{FL1'n' / field}			x				
	POSTRCVE								5
	POSTSEND								6
	RCVHDR								x
	RCVSEG								x
	REROUTE	X'mask',{=C'dest' / field}			x			x	
	ROUTE	n		x					
	SENDHDR								x
	SENDSEG								x
	SEQIN	[n]		7					
	SEQOUT	n					x		
	SKIP	[n],[C'chars']		x			x		
	SOURCE	n		x					
	TIMESTMP	n		x			x		
	TRANS	table	x			x			

Legend

1. Used in place of ROUTE macro-instruction.
2. First macro-instruction after ROUTE macro-instruction when multiple addressing is possible.
3. Specified only when 1050 Data Communications Systems are used and must immediately follow ENDRCVE and ENDSEND macro-instruction delimiters.
4. First macro-instruction of an LPS.
5. Last macro-instruction for LPS receive.
6. Last macro-instruction for LPS send.
7. In a dialing system, SEQIN must follow the SOURCE macro-instruction.

Table 12. Message Control Macro-Instructions for Specifying QTAM Control Information

Name	Operation	Operand	Function of Given Message Control Macro-Instructions		
			Terminal Table Specifications	Polling List Definition	Buffer Assignment
	BUFFER	[dcbname],n,length			x
entry	LIST	($entry_1$,...,$entry_n$)	x		
symbolic name of subfield	OPTION	subfield type and length	1		
pollname	POLL	($entry_1$,...,$entry_n$)		x	
entry	PROCESS	[EXPEDITE]	x		
entry	TERM	{T / L},dcbname,rln,addressing, [($optdata_1$,...,$optdata_n$)]	x		
	TERMTBL	entry	2		

Legend

1. Must directly follow TERMTBL macro-instruction.
2. Must be the first macro-instruction specified for terminal table.

Table 13. Message Processing Macro-Instructions

Name	Operation	Operand	Function of Given Message Processing Macro-Instructions		
			Queue Access	Telecommunications System Status	Line Activation and Deactivation
	CHNGPL	dcbname,rln,{workarea / =C'0' / =C'1'}		x	
	CHNGT	termname,area		x	
	CPYPL	dcbname,rln,workarea		x	
	CPYQ	name,workarea		x	
	COPYT	termname,area		x	
	GET	dcbname,workarea	x		
	PUT	dcbname,workarea	x		
	RELEASEM	termname	x		
	RETRIEVE	{source terminal / destination terminal / direct-access device address}, workarea,{IN / OUT / blank},{number / blank}	x		
symbol	STRTLN	dcbname,{rln / ALL}			x
symbol	STOPLN	dcbname,rln			x

Routines for handling line, terminal, control unit, and device oriented functions are included in the basic telecommunications access method (BTAM). This method also provides macro-instructions that enable a user to read a message segment into the computer and write a message segment onto a communications device. The user, who does not require the queueing and analysis functions of QTAM, may use BTAM.

Facilities provided by BTAM include:

- Terminal polling.
- Terminal addressing.
- Answering.
- Message receiving.
- Dynamic buffering.
- Message transmitting.
- Dialing.
- Communications system status changing.

The BTAM user is responsible for:

- Providing communications system specifications.
- Analyzing messages to determine the processing programs required.
- Writing message processing programs for specific applications.
- Writing message processing programs for changing communications system status.
- Routing messages to the appropriate processing program.
- Providing error checking and error corrections.
- Providing other message handling functions such as code translating, time and date stamping, header analysis, and error checking.

TELECOMMUNICATIONS SYSTEM SPECIFICATIONS

A user must provide certain system specifications through the use of macro-instructions which cause the following:

- Creation of system control information.
- Preparation of control units and communications lines for message transmission.
- Deactivation of communications lines.

One data control block must be created for each communications line group. A line group is a group of communications lines that meet the following operational requirements:

- All lines and their associated terminal devices have the same terminal control characteristics.
- All lines share the same buffer pool if one is used.
- All lines use the same buffering technique. (Refer to the publication IBM system/360 Operating System: Data Management.)

The user must supply a DD statement that indicates which communications lines constitute a communications line group and the order in which they are to be associated with the data control block for the line group.

To create a data control block for a communications line group, the user must issue a DCB macro-instruction. Before any messages can be sent or received over a communications line of the group, however, the line group must be activated by execution of an OPEN macro-instruction. When all operations on the lines have been completed, the lines are deactivated by issuing a CLOSE macro-instruction.

The DCB, OPEN, and CLOSE macro-instructions provide the user with a convenient method of specifying, initiating, and terminating his communications control system.

Data Control Block (DCB) Macro-Instruction

The DCB macro-instruction causes the creation of a data control block. A data control block is required for each communications line group. This macro-instruction reserves space for the data control block and assigns values to those fields for which the user provides parameters.

Name	Operation	Operand
dcbname	DCB	keyword parameters

dcbname
Represents the symbolic name of the data control block being created.

keyword parameters
Indicate the parameters listed in Table 14. When an alternate source code is shown with a parameter, it indicates the other sources of information that concern the parameter: C indicates that the parameter may be specified in a DD statement; E indicates the parameter whose value may be supplied at any time up to and including the DCB exit provided at open time. (Refer to the publication IBM System/360 Operating S Control Program Services.)

Table 14. DCB Macro-Instruction Keyword Parameters

Keyword Parameter	Alternate Source Code	Description
DDNAME=name		name Specifies the name that appears in the DD statement associated with the data control block. (Refer to the publication IBM System/360 Operating System: Job Control Language.)
DSORG=CX		CX identifies the data set organization to be that of a communications line group.
[{MACRF=(R) / MACRF=(W) / MACRF=(R,W)}]		(R), (W), or (R,W) Specifies that the access to this communications line group is to be gained through the READ or WRITE macro-instructions.
[BUFNO=n]	C,E	n Specifies the number of buffers to be provided for the buffer pool. The maximum value of n is 255.
[BUFL=l]	C,E	l Specifies the length (in bytes) of each buffer provided for the buffer pool or a standard length for user-provided buffers. The maximum value of l is 32,767 bytes.
[BUFCB=addr]	E	addr Specifies the name of a buffer pool control block to be provided at assembly time by the user. If this parameter is not provided, and if both BUFNO and BUFL are provided, an OPEN macro-instruction provides a buffer pool.
[BFTEK=D]	C,E	D Specifies that dynamic buffer allocation is to be provided. If this parameter is not provided, it is assumed that the user will handle buffering.
[EXLST=addr]		addr Specifies the symbolic address of the exit list. (Refer to the publication IBM System/360 Operating System: Control Program Services.)

OPEN Macro-Instruction

The OPEN macro-instruction prepares communications line groups for use. Each line group must be opened before message transmission can begin. The OPEN macro-instruction, together with the CLOSE macro-instruction, can be used to control the operative or nonoperative status of communications lines in the system.

Name	Operation	Operand
[name]	OPEN	(dcbname$_1$,,...dcbname$_n$) [,{MF=L / MF=(E,listname)}]

dcbname$_1$,,...dcbname$_n$
: Specifies the address of the data control block associated with a communications line group to be initiated. This address may be specified as an absolute address, explicit effective address, or a parenthesized symbolic name or integer designating a register that contains the address.

MF=L
: Specifies that this macro-instruction results in the creation of a parameter list as specified in the operand. This form of OPEN macro-instruction does not result in the execution of the open function. The name assigned to the parameter list is the name specified in the name field of the OPEN macro-instruction.

MF=(E,listname)
: Specifies that this macro-instruction results in the execution of the function. The parameter list constructed prior to the issuance of this macro-instruction is obtained and the function is executed for each parameter in the list (listname is the name assigned to the parameter list).

Note: The absence of the MF keyword parameter causes the macro-instruction to be executed with all the parameters specified in this issuance of the instruction. (For additional information, refer to the publication IBM System/360 Operating System: Control Program Services.)

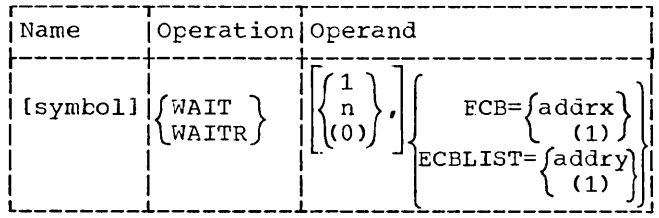
WAIT and WAITR Macro-Instructions

Two macro-instructions are available for signaling that the BTAM problem program can relinquish control of the Central Processing Unit; control can be relinquished because the program must wait for the occurrence of an event (e.g., completion of an input/output operation). The two macro-instructions are WAIT and WAITR.

WAIT results in the suspension of processing in the task from which the instruction is issued; processing resumes after the indicated event(s) occurs. The WAITR macro-instruction, in addition to performing the same functions as WAIT, provides two control features.

The type of control feature implemented by WAITR depends on the type of system used. In a system with the shared dynamic storage option (two job steps alternately occupy a single area of core storage), issuance of the WAITR macro-instruction in the BTAM problem program results in the writing (or "rolling out") of a portion of this task on an auxiliary storage device. A non-BTAM task (background task) is read into core storage (that is, "rolled in"), and processing of this task begins. If the event for which the BTAM problem program is waiting occurs, and the background task has no I/O in progress, the background task is written on the auxiliary storage device and the BTAM task is read back into core storage. If the event for which the BTAM problem program is waiting occurs, and the background task has I/O in progress, the BTAM problem program is not read back into core storage until the background-task event occurs. The BTAM problem program should not issue a WAITR macro-instruction unless all events for which it has issued WAITs have been completed.

In a system in which the available core storage is divided into distinct sections (partitions) to concurrently process job steps, the first issuance of the WAITR macro-instruction from the BTAM problem program has the following effects: 1) the partition in which the BTAM problem program resides becomes permanently assigned to BTAM; and 2) a processing program for a job step in another job is read into the other (or next) partition and processing of that job step commences. To use the partition technique, core storage size must be greater than 32,768 bytes. Under this partitioned dynamic storage option, no portion of the BTAM task is written out to provide more space. As in the case of the shared dynamic storage option, however, occurrence of an event for which the BTAM problem program is waiting signals that control is to be returned to the BTAM program. Subsequent issuance of WAITR by the BTAM problem program is treated as a WAIT.



symbol
Specifies the name for the WAIT or WAITR macro-instruction.

n
specifies the number of events that must occur before the problem program issuing the WAIT or WAITR macro-instruction can regain control of the system. If (0) is specified, the number of events must be loaded into register 0 prior to execution of this macro-instruction. If no entry is made for the number of events, 1 is assumed.

addrx
Specifies the address of an event control block (ECB) representing the only event that must occur before processing can continue. If ECB=addrx is specified, n cannot be greater than 1. If ECB=(1) is specified, the address of the ECB must be loaded into register 1 prior to execution of this macro-instruction.

addry
Specifies the address of a variable-length list that can contain the addresses of up to 255 event control blocks; each event control block represents an event to be waited for. The WAIT or WAITR macro-instruction is satisfied when the number of events posted to ECB's specified in the list equals the number indicated by the n entry in the macro. If ECBLIST=(1) is specified, the list address must be loaded into register 1 prior to the execution of this macro-instruction. (For additional information, see the publication Control Program Services.)

CLOSE Macro-Instruction

The CLOSE macro-instruction removes communications line groups from use. The CLOSE macro-instruction can be used with the OPEN macro-instruction to control the operative status of communications lines in the system.

dcbname$_1$,,...dcbname$_n$
Specifies the address of the data control block associated with a communications line group to be closed. This address may be specified as an absolute address, explicit effective address, or a parenthesized symbolic name or integer designating a register that contains the address.

MF=L
Specifies that this macro-instruction results in the creation of a parameter list as specified in the operand. This form of CLOSE macro-instruction does not result in the execution of the close function. The name assigned to the parameter list is the name specified in the name field of the CLOSE macro-instruction.

MF=(E,listname)
Specifies that this macro-instruction results in the execution of the close function. The parameter list constructed prior to the issuance of this macro-instruction is obtained and the function is executed for each parameter in the list (listname is the name assigned to the parameter list).

Note: The absence of the MF keyword parameter causes the macro-instruction to be executed with all the parameters specified in this issuance of the instruction. (For additional information, refer to the publication IBM Operating System/360 Operating System: Control Program Services.)

MESSAGE CONTROL

The message control section of BTAM polls terminals, answers calling devices, addresses terminals, transmits and receives messages, and dials devices. To use the facilities provided by BTAM, the user must provide terminal lists. (Refer to "Terminal List Structures.")

Programming for any additional message handling, such as code translating, message editing, format checking, and routing, must also be provided by the user. The number of data event control blocks (DECBs) that must be defined is equal to the maximum number of reads and/or writes that can be occurring simultaneously.

The format of the data event control block is shown in Figure 16.

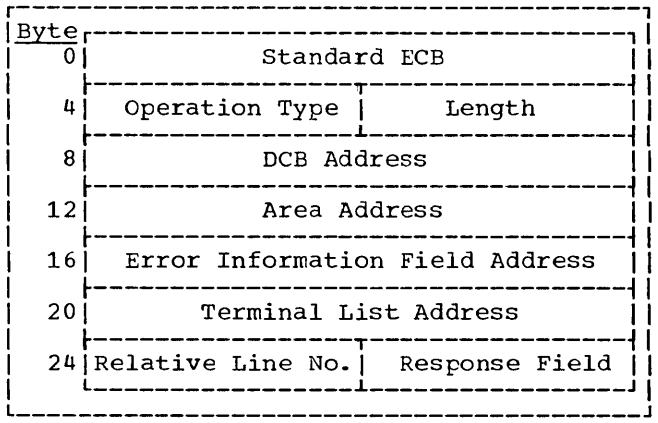


Figure 16. Data Event Control Block Format

POLLING

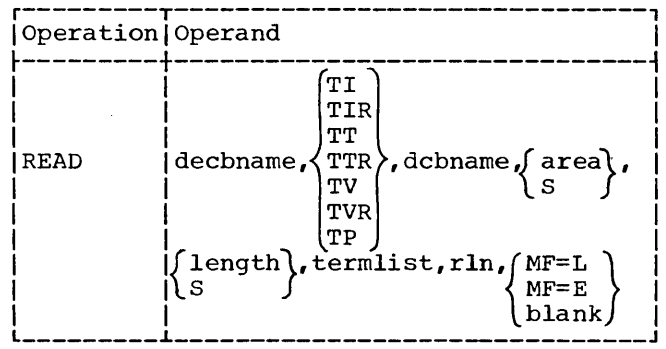
Macro-instructions for input operations are provided by BTAM. The user must provide polling lists that determine the order in which terminals on each line are interrogated for messages. (Refer to "Direct Polling and Addressing Lists.") The list is processed sequentially by BTAM, starting at the point specified by the user in his problem program.

Each terminal on the list is polled by BTAM. If a terminal has a message to send, contact is maintained until either an end-of-transmission (EOT) or an end-of-block (EOB) character is received. When one of these characters is received, completion is indicated by BTAM in the line DECB.

If a terminal has no message to send, the next terminal in the polling list is polled. Polling continues until a message is received, the polling list is exhausted, or an error condition is detected. To initiate polling, a user must issue a READ macro-instruction.

READ Macro-Instruction

The READ macro-instruction causes contact to be established with a terminal. If the terminal has a message to send, contact is maintained until an EOT or EOB character is received. The READ macro-instruction is then considered satisfied and completion is posted in the appropriate line DECB.



decbname
Specifies the address of the DECB in which completion is to be posted.

TI
Specifies initial READ. This option indicates that initial contacts must be made with the terminal. If required, contact is made by dialing or answering. Otherwise, initial contact is made by polling.

TIR
Specifies initial READ with RESET at completion. (Refer to the "RESET Macro-Instruction.")

TT
Specifies continued READ. This option indicates that a read operation is continued. When an EOB is transmitted as a message delimiter, the central processing unit must respond to the longitudinal redundancy check generated by the control unit before transmission can be continued. This option allows BTAM to provide the response and to maintain contact with the sending device, thus eliminating the need for repolling.

TTR
Specifies continued READ with RESET at completion.

TV
Specifies conversational READ. This option applies to dialing or common carrier TWX. Initial contact is made by dialing or answering. This option makes it possible to hold the line for subsequent communication without re-establishing contact.

TVR
Specifies conversational READ with RESET at completion.

TP
Specifies repeat READ. This option can be used for terminals that can recognize negative responses to be

transmitted to the terminal. It provides the problem program with the means of requesting retransmission of data received in error. To accept the erroneous data, a continued READ macro-instruction can be used.

dcbname
Specifies the address of the data control block for the line.

area
Represents the address of the first byte of the input area used to receive the message.

S
Specifies that BTAM will provide the buffer.

length
Indicates the number of bytes in the input area used to receive the message.

S
Indicates that BTAM will provide the buffer length specified in the data control block.

termlist
Indicates the address of an entry in the terminal list.

rln
Specifies the relative line number (within a line group for the data control block referred to) to which this macro-instruction applies.

MF=L
Specifies that this macro-instruction results in the creation of a parameter list as specified in the operand. This form of macro-instruction does not result in the execution of the read function.

MF=E
Specifies that this macro-instruction results in the execution of the function without the creation of a parameter list. The parameter list created prior to the issuance of this macro-instruction will be updated by inserting any parameters preceding the MF parameter.

blank
Specifies that this parameter may be left blank. If this parameter is omitted, the macro-instruction results in the creation of a parameter list and the execution of the function.

Note: For additional information, refer to the description of the "L" and "E" forms of macro-instructions found in the introduction to IBM System/360 Operating System: Control Program Services. Table 15 shows the operands permitted for each of these additional forms and their restrictions.

ADDRESSING

A convenient method is provided by BTAM for directing output messages to a terminal

Table 15. READ Macro-Instruction Forms

	Macro-Instruction Form		
Parameter	L	E	Neither L nor E
decb	Must be stated, will be used as the name of the decb, or must not be in register notation.	Must be stated, will be used as a pointer to the decb.	Required.
type	Required.	Acceptable, will override.	Required.
dcbname	Must not be in register notation.	Acceptable, will override.	Required.
area	Must not be in register notation.	Acceptable, will override.	Required.
length	Must not be in register notation.	Acceptable, will override.	Required.
termlist	Must not be in register notation.	Acceptable, will override.	Required.
rln	Must not be in register notation.	Acceptable, will override.	Acceptable.
MF=	MF=L.	MF=E.	Not acceptable.
Note: Type represents one of the following options: TI, TIR, TT, TTR, TV, TVR, TP.			

or group of terminals on a line. The terminals to be addressed must be identified in an addressing list provided by the user. This list may contain the addresses of one or more of the terminals on a line.

All terminals on the specified addressing list are addressed by BTAM before a message is sent. To initiate addressing, a user must issue a WRITE macro-instruction. When a message is destined for more than one line, it must be transmitted by separate WRITE macro-instructions.

WRITE Macro-Instruction

The WRITE macro-instruction causes transmission of a message segment to the terminals specified in an addressing list. The list may refer to one or more terminals on the line.

```
┌─────────┬──────────────────────────────────────────────────────────┐
│Operation│Operand                                                   │
├─────────┼──────────────────────────────────────────────────────────┤
│         │          ┌ TI  ┐                                        │
│         │          │ TIR │                                        │
│         │          │ TT  │                                        │
│WRITE    │decbname,{│ TTR │},dcbname,{area},                       │
│         │          │ TV  │          {  S }                        │
│         │          │ TVR │                                        │
│         │          └ TB  ┘                                        │
│         │{length},termlist,rln,{MF=L }                            │
│         │{  S   }              {MF=E }                            │
│         │                      {blank}                            │
└─────────┴──────────────────────────────────────────────────────────┘
```

decbname
: Indicates the name of the line DECB in which completion is to be posted.

TI
: Specifies initial WRITE. This option indicates that initial contact must be made with the terminal. If required, contact is made by dialing or answering. Otherwise, initial contact is made by addressing.

TIR
: Specifies initial WRITE with RESET at completion. (Refer to "Reset (RESETPL) Macro-Instruction.")

TT
: Specifies continued WRITE. When an EOB is transmitted, the control unit provides a longitudinal redundancy check. The remote terminal must respond to the longitudinal redundancy check before sending can continue. This option provides for monitoring the response and maintaining contact with the terminal.

TTR
: Specifies continued WRITE with RESET at completion.

TV
: Specifies conversational WRITE. This option applies to dialing or common carrier TWX. Initial contact is made by dialing or answering. When contact with the line has been established, this option provides the ability to carry on multimessage transmission with one or more components without redialing.

TVR
: Specifies conversational WRITE with RESET at completion.

TB
: Specifies SPACE signal write. (Refer to the "Commands" section in the publications: IBM 2701 Data Adapter Unit, Principles of Operation, or IBM 2702 Transmission Control.)

dcbname
: Indicates the address of the data control block for the line.

area
: Represents the address of the first byte of the output area.

S
: Specifies that BTAM will write a chain of buffers.

length
: Specifies the number of bytes in the output area.

S
: Specifies that BTAM will provide the length parameter based upon BUFL.

termlist
: Indicates the address of the terminal list.

rln
: Specifies the relative line number (within a line group for the data control block referred to) to which this macro-instruction is to apply.

MF=L
: Specifies that this macro-instruction results in the creation of a parameter list as specified in the operand. This form of WRITE macro-instruction does not result in the execution of the write function.

MF=E

Specifies that this macro-instruction results in the execution of the function without the creation of a parameter list. The parameter list created prior to the issuance of this macro-instruction will be updated by inserting any parameters preceding the MF parameter.

blank

Specifies that this parameter may be left blank. If this parameter is omitted, the macro-instruction results in the creation of a parameter list and the execution of the function.

ANSWERING

In order for a terminal to dial and establish contact with the computer, the user must enable the line to transmit messages to the system. This may be accomplished by issuing a READ or WRITE macro-instruction with the proper terminal list specification. Contact with the terminal is established by BTAM, and messages are received. For subsequent communication with the terminal conversational-type READ or WRITE macro-instructions car be used.

The line, which has been enabled, may be disabled by issuing a RESETPL macro-instruction or a CNTRL (disable) macro-instruction. A line may also be disabled by issuing a READ or WRITE macro-instruction that specifies a RESET at completion.

Reset (RESETPL) Macro-Instruction

The RESETPL macro-instruction provides the ability to free a dial-type line or to interrupt a polling sequence. If a message is being transmitted when a RESETPL macro-instruction is issued, no action is taken. Otherwise, the line is freed or polling is interrupted and the preceding READ or WRITE macro-instruction is posted as complete. Control is then returned to the problem program.

If TIR, TTR, or TVR options of the READ or WRITE macro-instructions are specified, the reset function will be performed at the end of message transmission. If, however, the RESETPL macro-instruction is issued for a dial-type line which is being polled, the polling will be terminated and the line will be freed.

Operation	Operand
RESETPL	dcbname,rln

dcbname

Indicates the address of the data control block for the communications line group.

rln

Specifies the relative line number (within the line group for the data control block referred to) to whicn this macro-instruction applies.

TERMINAL LIST STRUCTURES

The data transmission macro-instructions (READ or WRITE) provided by BTAM refer to the user-provided terminal list. The structure of the list is determined by both the terminal equipment involved and the type of operation to be performed. In all macro-instructions used to generate terminal lists, component addressing, polling, and identification characters must be specified as the hexadecimal equivalents of the particular terminal device code for that character. Tables of the various code structures are contained in the publication IBM 2701 Data Adapter Unit, Principles of Operation, Form A22-6864.

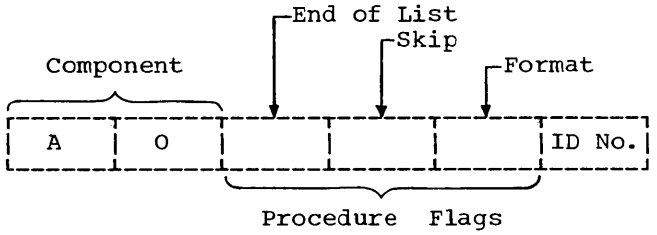
Direct Polling and Addressing Lists

Direct polling and addressing apply to data transmission operations where the line connection is permanently established (non-dial). Read operations involve the polling of terminals. When a READ macro-instruction is specified, it must refer to a polling list. A polling list determines the order in which remote terminals are interrogated to send data.

Write operations require the addressing of terminals to prepare them for the receipt of a message. When a WRITE macro-instruction is specified, it must refer to an addressing list. An addressing list consists of addresses of one or more terminals on a line, which are the destinations of a message. All terminals in the list are contacted before the message is transmitted.

A polling list may be either an open or a wrap-around list. An addressing list is identical in structure to the open list.

OPEN LIST: In an open list, each entry is assigned a fixed field size. An index, consisting of three procedure flags plus an identification number that indicates the entry's relative position in the list, is entered for each entry.



The last entry in the list has a flag in the index. The open list is processed sequentially and completion is posted by BTAM either when the end of a list has been reached, a message has been received, or an error condition has been detected.

WRAP-AROUND LIST: A wrap-around list is identical to the open list, except that the last entry contains a pointer to the beginning of the list. Completion of the wrap-around list is posted by BTAM when a message has been received or when an error condition has been detected. When the end of a wrap-around list has been reached, polling automatically restarts with the first entry on the list. To free a line during a wrap-around polling sequence, a RESET macro-instruction must be issued.

Figure 17 is an example of a wrap-around polling list. The entry for each terminal contains two polling characters, and a control byte. The polling characters for entries 1 through 5 are, in IBM 1050 code, A0, B0, C0, D0, and E0, respectively; these are represented in Figure 17 in their hexadecimal equivalents. The end-of-list bit in the control byte is 0 in all entries. When the skip bit is 1 as in entries 2 and 3, it indicates that the entry is to be deleted from the polling sequence. The format bit is 1 in entry 5, indicating that the contents of the next two bytes are used to locate the start of the list. Thus, the polling sequence can continue without problem program intervention when no terminal responds with a message.

Note: If in entry 5 the format bit was 0 and the end of list bit was 1, the list would be an open list and polling would cease after entry 5. An addressing list would contain addressing characters rather than polling characters.

Dialing and Answering Lists

Dialing and answering apply where the line connection must be established (by dialing) before data transmission can take place.

Read operations involve:

- Dialing from computer to terminal (and polling) to receive a message. (In a TWX application, terminal identification verification must also be performed.)
- Answering a call from a terminal that has dialed the computer and polling the terminal to receive a message. (In a TWX application, terminal identification verification must also be performed.)

When this type of READ macro-instruction is specified, it must refer to an appropriate terminal list.

Figure 18 is an example of a terminal list for dialing from computer to terminal

	Polling Characters		Control Byte: End of List	Skip	Format	Relative Location	
Entry 1	E2	15	0	0	0	00000	
Entry 2	E4	15	0	1	0	00001	
Entry 3	E7	15	0	1	0	00010	
Entry 4	E8	15	0	0	0	00011	
Entry 5	EB	15	0	0	1	00100	Start of List

Figure 17. Wrap-Around Polling List Example

(and addressing) to transmit a message. The dial digits may be followed by one or more open list entries. The component address is shown as the hexadecimal equivalent of IBM 1050 code characters A9.

Figure 19 is an example of a terminal list for answering a call from a terminal that has dialed the computer, and polling the terminal to receive a message. The answer digit (zero) may be followed by one or more open list entries. The component address is shown as the hexadecimal equivalent of IBM 1050 code characters A0.

Figure 20 is an example of a terminal list for dialing in a TWX application. The terminal identification is shown as the hexadecimal equivalent of the Eight-Bit Data Interchange Code Characters CHI.

Figure 21 is an example of a terminal list for answering in a TWX application, except the answer digit (zero) is added, and the number of dial digits and the dial digits are not included. The terminal identification is shown as the hexadecimal equivalent of the Eight-Bit Data Interchange Code Characters CHI.

Write operations involve:

- Dialing from computer to terminal (and addressing) to transmit a message. (In a TWX application, terminal identification verification must also be performed.)
- Answering a call from a terminal that has dialed the computer and asking the terminal to transmit a message. (In a TWX application, terminal identification verification must also be performed.)

Note: The list structure for the write operation is identical to that for the read operation except that in the examples shown, the field called component address contains addressing characters instead of polling characters.

		Open List Entry		
No. of Dial Digits	Dial Digits	Component Address	Procedure Flags	Entry ID
7	4635514	E213		

Figure 18. Dial DIALST Example

	Open List Entry		
Answer	Component Address	Procedure Flags	Entry ID
0	E215		

Figure 19. Answer DIALST Example

No. of Dial Digits	Dial Digits	No. of ID Characters	ID From Terminal	ID of Terminal	Procedure Flags	Entry ID
7	4634414	3		C31393		

Figure 20. Dial IDLST Example

Answer	No. of ID Characters	ID From Terminal	ID of Terminal	Procedure Flags	Entry ID
0	3		C31393		

Figure 21. Answer IDLST Example

Define Terminal List (DFTRMLST) Macro-Instruction

The DFTRMLST macro-instruction provides the capability for defining a polling or addressing list, and entering the physical addresses and device specifications into the list.

Name	Operation	Operand
entry	DFTRMLST	{OPENLST / WRAPLST / DIALST / IDLST}, arg_1,...,arg_n

entry
Represents the symbolic name assigned to the beginning of the list.

OPENLST
Indicates that the list is to be an open list structure.

WRAPLST
Indicates that the list is to be a wrap-around list structure.

DIALST
Indicates that the list is to be a dial or answer list in which verification of the identification of the terminal is not required.

IDLST
Indicates that the list is to be a dial or answer list of TWX terminals; therefore, verification of the identification of the terminal is required.

arg_1,...,arg_n
Specify the parameters needed to construct the terminal lists. The parameters that must be included for each type of list are as follows:

Open list (OPENLST entry): arg_1 through arg_n indicate the addressing or polling characters. The actual characters are in the code for the particular type of terminal, and must be specified in hexadecimal. Example: arg_1 through arg_4 are E202,E402,E702,E802 (for A1, B1, C1 and D1, respectively, in IBM 1050 code).

Wrap-around list (WRAPLST entry): arg_1 through arg_n indicate the addressing or polling characters. As in the case of the open list arguments, the characters must be specified in hexadecimal. Example: arg_1 through arg_4 are E20B,E40B,E70B,E80B (for A5, B5, C5, and D5, respectively, in IBM 1050 code).

Dial or answer list (DIALST entry): for a dial list, arg_1 specifies the number of dial digits, arg_2 the dial digits, and arg_3 through arg_n the addressing or polling characters. As in the open list arguments, the addressing or polling characters must be specified in hexadecimal. Example: arg_1 through arg_4 are 7,4635514,E207,E202 (arg_3 and arg are the hexadecimal representation of A3 and A1 in IBM 1050 code). For an answer list, arg_1 is a zero, and arg_2 through arg_n (in hexadecimal) are the addressing or polling characters. Example: arg_1 through arg_3 are 0,E20D,E20B (arg_2 and arg_3 are the hexadecimal representation of A6 and A5 in IBM 1050 code).

ID dial or answer list (IDLST entry): for a dial list, arg_1 specifies the number of dial digits, arg_2 the dial digits, arg_3 the number of identification digits, and arg_4 the identification of the terminal. The identification of the terminal must be the hexadecimal equivalent of the particular terminal device code for the characters used. Example: arg_1 through arg_4 are 7,4635514,2,43CD (arg_4 is the hexadecimal representation of B3 in Eight-Bit Data Interchange Code). For an answer list, arg_1 is zero, arg_2 the number of identification digits, and arg_3 the identification of the terminal (in hexadecimal). Example: arg_1 through arg_3 are 0,2,43CD (arg_3 is the hexadecimal representation of B3 in Eight-Bit Interchange Code).

Change Terminal Entry (CHGNTRY) Macro-Instruction

The CHGNTRY macro-instruction provides the means for deleting or reactivating a terminal entry in a polling or addressing list without redefining the existing list. The list referred to in list ID will be located and the entry specified in arg_1 will be altered to the conditions specified in arg_2.

Name	Operation	Operand
	CHGNTRY	entry,type,arg_1, {SKIP / ACTIVATE}

entry
Represents the symbolic name for the beginning of the list. It is defined using the DFTRMLST macro-instruction.

type
Signifies the type of list specified in the DFTRMLST macro-instruction used to create list.

arg_1
Indicates the relative position in the list of the entry to be changed.

SKIP
Indicates that this entry is to be skipped when polling or addressing.

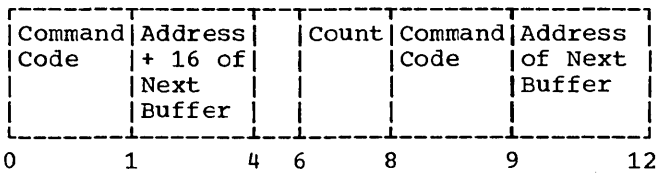
ACTIVATE
Indicates that this entry is to be reactivated.

BUFFERING

When the data control block parameters BUFNO and BUFL or BUFCB are provided, a buffer pool is shared by all lines in the communications line group. The user may manipulate the pool using the GETBUF and FREEBUF macro-instructions.

If the data control block parameter BFTEK=DYN is also specified, BTAM dynamically allocates buffers to read operations and dynamically writes a chain of buffers. The first eight bytes of each buffer contain the read or write channel command word, and the next four bytes contain a transfer in channel command. Thus, the user's definition of buffers length (BUFL=l) must be twelve bytes larger than that required for data.

For read operations with dynamic buffering, the user may specify an initial input area in the READ macro-instruction. Alternatively, BTAM provides the initial area from the buffer pool if an area parameter of S is specified. Any additional buffers needed will be supplied by BTAM from the buffer pool.



At the end of the read operation, the first twelve bytes of each buffer will be in the format shown above. For the last buffer, the addresses will both be null. The address of the first byte of the first buffer will be contained in the data event control block.

For write operations with dynamic buffering, the user must provide the address of the first byte of the first buffer in the line DECB and specify the macro-instruction with an area parameter of S. The first twelve bytes of each buffer must also be in the format shown above. BTAM will provide the command codes and terminate the operation when all buffers have been transmitted. The address fields in the last buffer must be null. If the address fields of any buffer are non-zero, they will be assumed to contain valid buffer addresses.

ERROR HANDLING

Error conditions are posted in the line error information field by BTAM routines. These conditions must be recognized by the user to ensure proper operation (Figure 22).

An all-zero-sense byte indicates that the operation was completed successfully and no error conditions occurred. In a non-zero-sense byte, the bit positions indicate command reject (bit 0), intervention required (bit 1), parity error (bit 2), equipment check (bit 3), data check (bit 4), overrun (bit 5), receiving (bit 6), and time-out (bit 7). (Refer to the publications IBM 2701 Data Adapter Unit, Principles of Operation and IBM 2702 Transmission Control.)

Completion is posted whether termination was caused by a completed transmission or by an error condition. The terminal response character is placed in the response field byte of the data event control block and it is the responsibility of the user to test these conditions and take proper action.

If successive macro-instructions are issued for a line before the preceding macro-instruction has been posted as complete, they are rejected.

When using dynamic buffer allocation, it is the problem programmer's responsibility to provide sufficient buffers to accommodate maximum traffic for the system. If buffers are requested and none are available, no buffer can be assigned. If no buffers are available, a null link is inserted in the last message buffer. This results in an incomplete message transmission and the error condition is posted in the sense byte of the error information field (Figure 22).

The FREEBUF macro-instruction must be used to free buffers and return them to the buffer pool. Concerning buffer pool usage, the first eight bytes of the buffer contain the channel command word for this transmission, and the next four bytes of the buffer contain the linkage to the next buffer for this transmission.

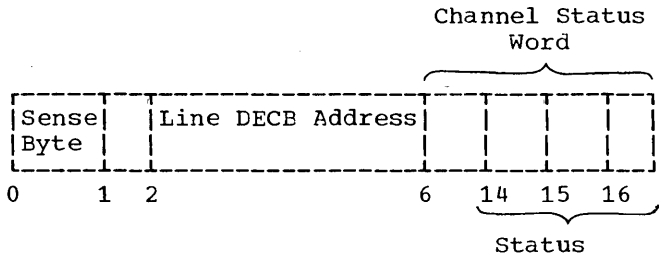


Figure 22. Line Error Information Field Format

Free Buffer (FREEBUF) Macro-Instruction

The FREEBUF macro-instruction returns the buffer whose address is contained in the specified register to the buffer pool associated with the specified data control block. Buffers assigned using dynamic buffering must be returned to the pool by the problem program.

Operation	Operand
FREEBUF	dcbname,T

dcbname
: Specifies the symbolic name of the data control block through which the buffer was assigned.

T
: Specifies the register that contains the address of the buffer to be freed.

SYSTEM CONFIGURATION

The system configuration for the QTAM sample problem (Figure 23) is as follows:

Communications

1050 Data Communications System (five)
Half-Duplex Communications Line (two)
2701 Data Adapter Unit (one)
Multiplexor Channel (one)
2311 Disk Storage Drive (one)

Operating System

Computing System/360 Model F30(64k) (one)
Selector Channel (one)
1052 Console (one)
1402 Card Read-Punch (one)
1403 Printer (one)
2311 Disk Storage Drive (one)

The sample program is capable of handling message switching and inquiry message type applications. Message switching refers to messages that do not require processing of message text but are to be routed directly to their destinations. Destinations which are specified in the input header may be any of the following:

- Single destination specified in the destination field of the header. For example; NYC.
- Multiple destinations specified in sequence in the header. For example, NYC PHI ... .
- Distribution list specified in the header. For example, PBW would specify destinations contained in the terminal table list for PBW; i.e., Boston and Washington.

Inquiry message refers to process messages that require processing of the message by problem programs resident in the central processing system. Reply messages must be generated for transmission back to the sending terminal. (Refer to Figures 24 and 25 and to Tables 16 and 17.)

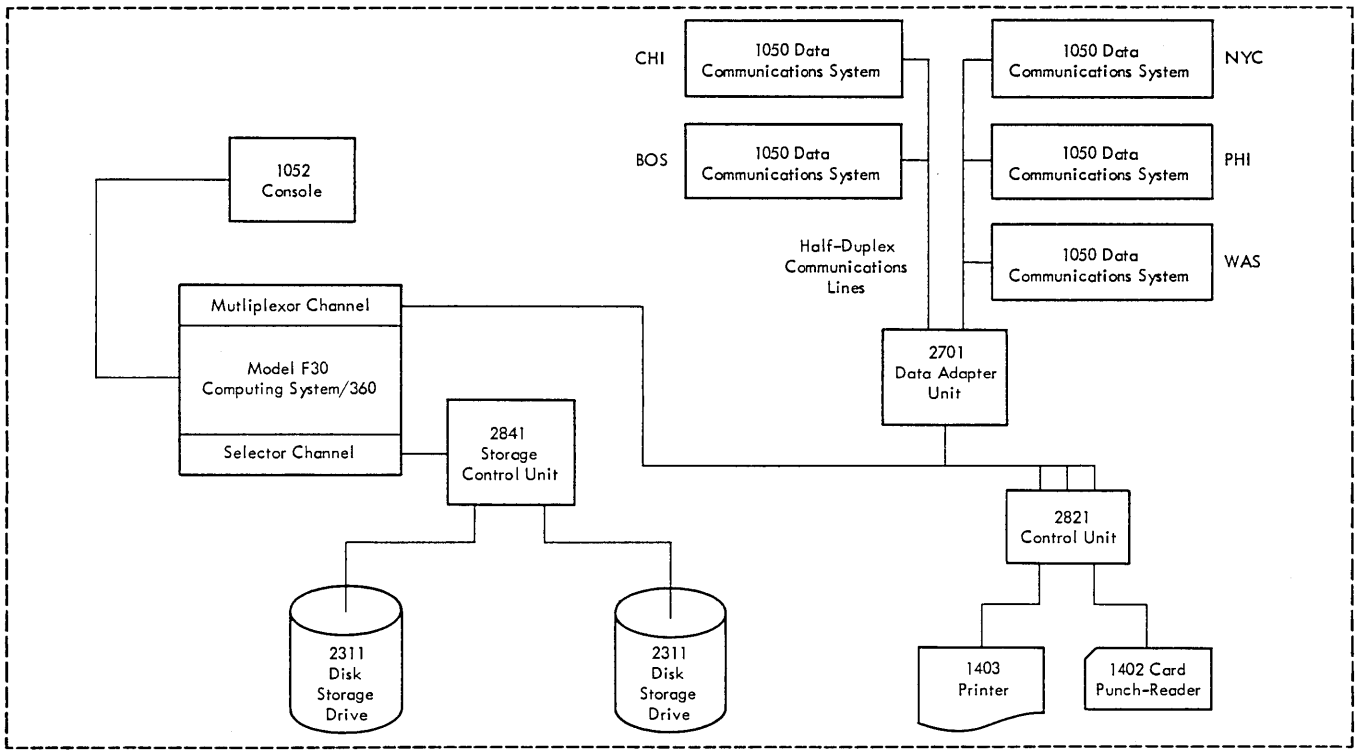


Figure 23. Queued Telecommunications Access Method Sample Problem - System Configuration

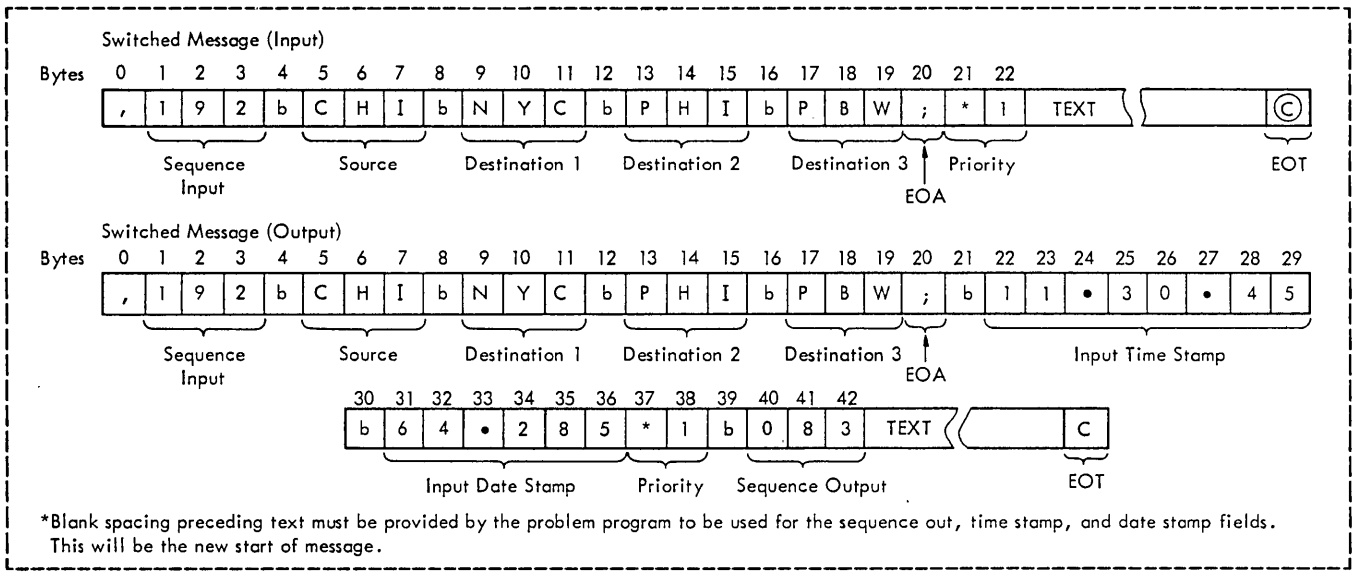


Figure 24. Switched Message Formats

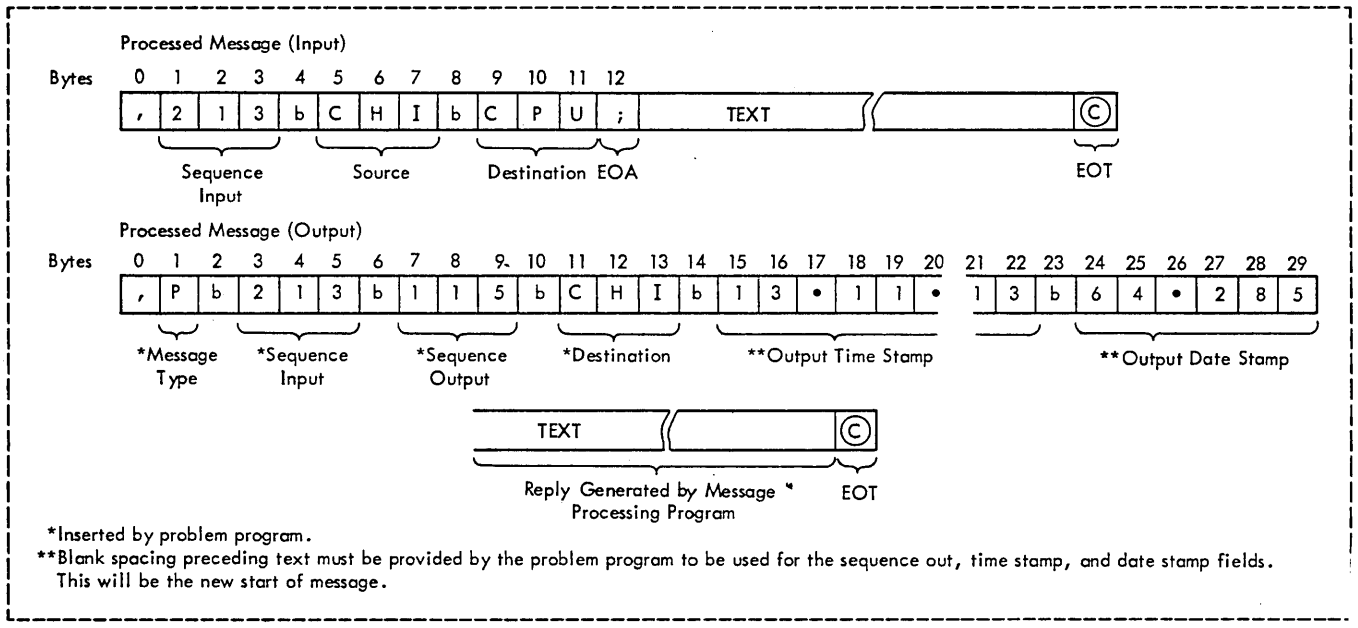


Figure 25. Processed Message Formats

Table 16. Message Control Sample Problem

Name	Operation	Operand	Comments
Define DCB for Communications Line Group			
DCBGROUP	DCB	DDNAME=DDGROUP1	Gives the DDNAME associated with the data control block.
		DSORG=CX,	Defines the data control block as a communications line group type.
		MACRF=(G,P),	Indicates that GET and PUT macro-instructions will be used to refer to this data set.
		BUFRQ=3,	Specifies the number of buffers to be requested in advance for each line.
		CPRI=E,	Gives send and receive equal priority.
		CPOLL=(POLLINE1, POLLINE2)	Represents the symbolic names assigned to polling lists for each line.
		CLPS=LPS1	Identifies LPS1 as the name of this line group LPS.
		ACLOC=13	Indicates the relative address of the device in the terminal entries.
Define DCB Direct-Access Queues			
QUEUE	DCB	DDNAME=DDFILE	Gives the DDNAME associated with the data control block.
		DSORG=CQ	Defines data control block as a direct-access device type.
		MACRF=(G,P)	Indicates that GET and PUT macro-instructions will be used to refer to this data set.
Define Terminal Table			
	TERMTBL	CPU	Specifies the extent of the terminal table; CPU is defined as the last entry in the terminal table.
LIMIT	OPTION	FL1	LIMIT is the symbolic name of the field in this terminal table that contains the limit of consecutive polls for each terminal. Field length equals one (FL1).
CHI NYC PHI BOS WAS	TERM TERM TERM TERM TERM	L,DCBGROUP,1,E407E40D,(2) L,DCBGROUP,2,E207E20D,(2) L,DCBGROUP,2,E407E40D,(2) L,DCBGROUP,1,E207E20D,(1) L,DCBGROUP,2,E707E70D,(1)	The five terminals and their parameters are entered in the terminal table. The explanation of the first line is as follows: L indicates that the outgoing messages are to be queued by line. DCBGROUP specifies the name of the data control block. 1 is the relative line number within the line group to which the terminal is attached. E40DE407 is the hexadecimal representation of IBM 1050 code-characters B6 and B3; B6 is the polling code and B3 the addressing code. 2 is the maximum number of continuous polls to be inserted in the field LIMIT defined above.

(continued)

Table 16. Message Control Sample Problem (continued)

Name	Operation	Operand	Comments
PBW	LIST	(BOS,WAS)	Defines a distribution list of message destinations as Boston and Washington.
CPU	PROCESS		Defines symbolic name of a process data control block in the terminal table. Allows CPU as valid destination.
Define Polling Order			
$POLLINE_1$	POLL	(CHI,BOS)	Defines order of polling of terminals attached to line 1. Identifies terminals.
$POLLINE_2$	POLL	(NYC,PHI,NYC,WAS)	Defines order of polling of line 2.
Define Buffering			
	BUFFER	QUEUE,60,125	Provides main storage buffer areas for the data set (QUEUE) used for message queueing. Sixty buffers are specified, 125 bytes per buffer.
Define Beginning of Message Control Task			
STQTAM	OPEN	(QUEUE,,DCBGROUP)	Initializes the data control blocks for the data set to be used for message queueing and the communications line group. It causes polling to be initiated on the lines, and updating of queue status tables.
	ENDREADY		Specifies the end of the initialization instructions.
LPS1	LPSTART	20	Identifies start of message segment; 20 spaces for the time stamp, date stamp, and sequence out number.
	RCVSEG		Instructions following will service header and text segments of the input message.
	TRANS	RCVE1050	Converts 1050 message characters to EBCDIC (for all segments, header and text).
	RCVHDR		Instructions following will service the header segment of the input message.
	SKIP	,C','	Causes all characters up to and including the , to be skipped.
	SEQIN	3	Checks sequence of numbered messages for each terminal as they arrive. Operand is the number of characters in header sequence number field.
	SOURCE	3	Checks the validity of the source terminal code received in the message header against the terminal table provided by the user. The operand specifies the number of characters in the source field of the header. If invalid an error is indicated in the error half-word.

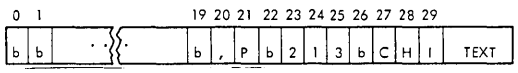
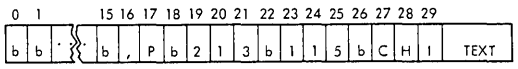
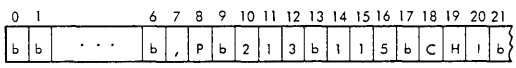
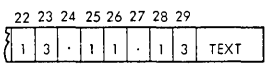
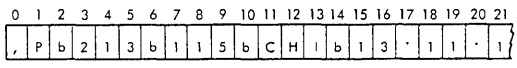
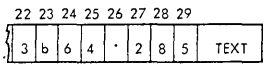
(continued)

Table 16. Message Control Sample Problem (continued)

Name	Operation	Operand	Comments
	ROUTE	3	Checks the validity of the destination code in the message header. If the destination code is valid the message is subsequently queued for the specified destination. Operand specifies the number of characters in the destination code.
	EOA	C';'	Causes messages to be routed to any additional destination. Operand identifies; as the end of address character. This character must appear in the message header after the last destination code.
	TIMESTMP	9	Inserts the time-of-day stamp in the header field. First character is blank. Operand indicates number of characters to be inserted.
	DATESTMP		Inserts date. (See TIMESTMP.)
	MODE	PRIORITY,C'*'	If next character is an * then the character following will be the priority.
	ENDRCVE		Specifies that following macro-instructions will service the message after the end of message is received.
	EOBLC		Allows the 1050 Data Communications System to continue receiving after an end-of-block. It also provides a procedure for line correction if a transmission error is detected. If the error is not corrected, an error is indicated in the error half-word for this line.
	ERRMSG	=X'3000',SOURCE,=C'& MESSAGE NUMBER NOT IN SEQUENCE'	Sends the error text to the specified terminal when error type specified by the mask is detected. The message header will replace the &. The mask is the bit configuration (in hexadecimal) used to test the half-word error indicator.
	ERRMSG	=X'8600',SOURCE,=C'& MESSAGE NUMBER IN ERROR CORRECT AND RESEND'	Sends message on detection of error indicated by error mask X'8600'.
	CANCELM	X'8600'	Specifies that any message mask X'8600 is canceled.
	POLLIMIT	LIMIT	Determines whether the terminal has sent the maximum number of messages allowed on a single polling pass. Operand is the symbolic name of a field in the terminal table which contains the limit of consecutive polls for each terminal.

(continued)

Table 16. Message Control Sample Problem (continued)

Name	Operation	Operand	Comments
	POSTRCVE		Indicates the end of the input section of the LPS.
	SENDHDR		Specifies that following macro-instructions will service the header segment of the output message.
	MSGTYPE	=C'P'	Determines if the message is a type P message. If the next nonblank character is a P, the following LPS group will handle the message. The problem program must leave 20 spaces at the beginning of the message for the out time stamp, date stamp, and sequence number.
	SKIP	3	Skips sequence input number.
	SEQOUT	4	Sequentially numbers outgoing message destination address. A 3-character sequence number plus a leading blank is inserted. Space must be reserved at the beginning by the problem program. Before:  After: 
	SKIP	3	Skips destination field.
	TIMESTMP	9	Inserts a 6-character time-of-day stamp in the outgoing header field plus a leading blank. (See SEQOUT operand.) Before:  After: 
	DATESTMP		Inserts a 5-character date stamp in the message. Before:  After: 

(continued)

Table 16. Message Control Sample Problem (continued)

Name	Operation	Operand	Comments
	MSGTYPE		All messages except type P messages will be handled by the following LPS group.
	SEQOUT	4	Inserts a 3-character sequence number of message plus a leading blank.
	SENDSEG		Specifies that following macro-instructions will service header and text segments of the output message.
	TRANS	SEND1050	Translates output message using code conversion table named SEND1050.
	PAUSE	X'15',20X'17'	Upon recognition of carriage return character, this routine inserts 20 idle characters to provide time for carriage return. 15=carriage return in hexadecimal; 17=IDLE in hexadecimal.
	ENDSEND		Identifies the end of instructions used to service output messages.
	EOBLC		Allows the 1050 Data Communications System to continue sending after receipt of an end-of-block. It also provides a procedure for line correction if a transmission error is detected. If the error is not corrected an error is indicated in the error half-word for this line.
	REROUTE	=X'0040',=C'BOS'	Causes a message to be queued for the terminal specified when the error type specified in the mask X'0040' is detected.
	POSTSEND		Identifies the end of the sending portion of the LPS. It also indicates the last instruction of this LPS.

Table 17. QTAM Sample Problem Programs

Name	Operation	Operand	Comments
STPROCES	OPEN . . .	(PROCESSQ)	Opens data set that contains the queue of messages to be processed.
LOOP	GET . . .	PROCESSQ,WORKA1	Gets the next sequential segment from the queue referred to in the data control block and places it in workarea WORKA1. Message is now available for processing by the problem program. The message is obtained from the queue specified by the parameter ddname (CPU). The program would be executed under the normal protection features offered by the control program. Data sets to be used in preparing the reply would be referred to under normal procedures of System/360 Operating System.
	PUT	REPLYQ,WORKA2	Places the processed message on the appropriate destination queue specified by the DCB parameter TRMAD. The terminal table entry name in location specified by TRMAD will be the destination.
	B	LOOP	Gets next message for processing.
Constant Definition			
WORKA1	DS	CL300	Provides storage for message processing.
WORKA2	DS	CL300	Provides storage for message processing.
SOURCE	DS	CL3	Provides area that will contain the destination terminal table entry name.
DCB Definition PROCESSQ			
PROCESSQ	DCB	DDNAME=CPU	Identifies the name of the process queue terminal table entry and the DDNAME associated with this process data control block.
		DSORG=MQ	Defines data control block as process type.
		MACRF=G	Indicates that the program uses the GET macro-instruction.
		BUFRQ=2	Indicates two buffers to be queued in core.
		RECFM=G	Specifies working unit as a message.
		SYNAD=ERROR	Identifies the name of the routine that will handle overflow messages.
		TRMAD=SOURCE	Specifies the name of the location that will contain origin of the message.
		SOWA=300	Specifies the workarea size (in bytes).
Reply			
REPLYQ	DCB	RPYOUT,DSORG=MQ, MACRF=P,RECFM=G, TRMAD=SOURCE,SOWA=300	Defines parameters for data control block associated with an output message processing queue.

SYSTEM CONFIGURATION

The system configuration for the BTAM sample problem (Figure 26) is as follows:

Communications

1050 Data Communications System (three)
Half-Duplex Communications Line (one)
2701 Data Adapter Unit (one)
Multiplexor Channel (one)
2311 Disk Storage Drive (one)

Operating System

Computing System/360 Model E30 (32K) (one)
Selector Channel (one)
1052 Console (one)
1402 Card Read-Punch (one)
1403 Printer (one)
2311 Disk Storage Drive (one)

BTAM provides macro-instructions to read a message into the computer and to write the message out to a communications device.

The program presented in Table 18 provides an example of the use of BTAM macro-instructions to read messages from remote terminals as depicted in Figure 26. The problem program processes the message and sends a reply to the remote terminal on the line.

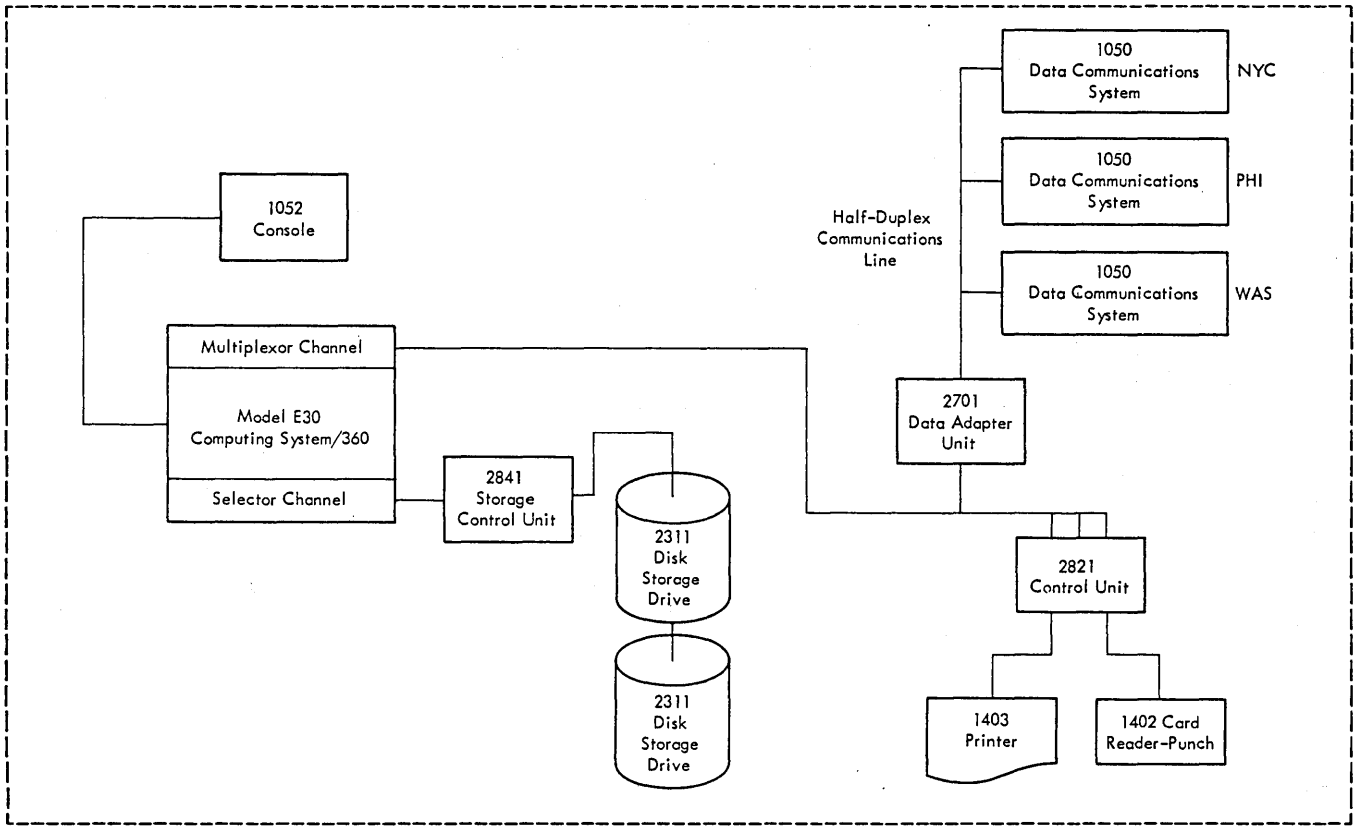


Figure 26. Basic Telecommunications Access Method Sample Problem - System Configuration

Table 18. BTAM Sample Problem Programs

Name	Operation	Operand	Comments
Define Polling List and Data Control Block			
LINEA	DFTRMLST	OPENLST,	LINEA specifies the symbolic name of start of polling list. OPENLST indicates that the polling list is to be an open list structure.
		E20B,E40B,E70B	The first two characters of each operand (E2, E4, E7) are the hexadecimal representation of the IBM 1050 code characters (A, B, and C, respectively), identifying the station on the line. The last two characters of each operand (OB) are the hexadecimal representation of the IBM 1050 code character (5) identifying the component to be polled. In this case, the component at each station is the 1050 keyboard.
LINE1	DCB	DDNAME=JOBA,	Gives the DDNAME to be associated with this data control block.
		DSORG=CX,	Defines the data organization as that of a communications line group.
		MACRF=(R,W),	Specifies user-supplied READ and WRITE macro-instructions.
		BUFNO=10,	Specifies the number of buffers to be assigned to the line.
		BUFL=92,	Specifies the length of each buffer to be provided for the buffer pool.
		BFTEK=D	Indicates dynamic buffer allocation.
INITIATE	OPEN	(LINE1)	Opens the communications line.
Problem Program to Read From Line			
ICOP	READ	INPA,	Indicates the name of the data event control block.
		TI,	Specifies that initial contact with the terminal is to be established by the READ macro-instruction.
		LINE1,	Specifies the name of the data control block associated with the communications line.
		S,	Represents the input area to be provided from the message pool by BTAM.
		S,	Indicates the size of the input area to be provided by BTAM.
		LINEA,	Names the polling list.
		1	Specifies the relative line number.
	WAIT	ECB=INPA	Specifies that the user must wait for completion to be posted in the block named INPA.

(continued)

Table 18. BTAM Sample Problem Programs (continued)

Name	Operation	Operand	Comments
	B	PROC	
Problem Program to Write to Line			
WRA	WRITE	OUTA,	Specifies the name of the data event control block.
		TI,	Indicates initial contact with the terminal is to be established for the WRITE macro-instruction.
		LINE1,	Specifies the address of the data control block for the line.
		(OUTAREA),	Represents the name of the register that contains the address of the output area.
		(OUTSIZE),	Represents the name of the register that contains the number of bytes in the OUTAREA.
		(TERMADDR)	Specifies the name of the register that contains the address of the addressing list.
		1	Specifies the relative line number.
	WAIT	ECB=OUTA	Specifies that the user waits for completion to be posted in the block named OUTA.
	FREEBUF	LINE1,OUTAREA	Frees the buffer used for the output message.
		. . .	Specifies additional processing.
User-Written Routines to Process Message			
PROC			Indicates that the processing of the input message is to be performed. The buffer for the output message must be obtained by the user, but it may be done using the GETBUF macro-instruction.
	GETBUF	LINE1,OUTAREA	Specifies the address of the data control block for the line. OUTAREA indicates the name of the register that will contain the address of the output buffer.
		. . .	Specifies additional processing.
End-of-Job Shutdown of System			
END	CLOSE	(LINE1)	Specifies that when the problem program has completed all message processing, it must close the communications line.

(continued)

Table 18. BTAM Sample Problem Programs (continued)

Name	Operation	Operand	Comments
Storage and Symbol Definitions			
OUTAREA OUTSIZE TERMADDR	EQU EQU EQU	4 6 7	Indicates that the problem program must define symbolic register names.
Define Addressing Lists			
TERMA TERMB TERMC	DFTRMLST DFTRMLST DFTRMLST	OPENLST,A1 OPENLST,B1 OPENLST,C1	Specifies that the appropriate addressing list address must be provided in the register named TERMADDR when the WRITE macro-instruction is executed.

Where more than one page reference is listed, the major reference is given first.

READER'S COMMENTS

IBM Operating System/360, Telecommunications: Preliminary Specifications

C28-6553-2

Your comments will help us to produce better publications for your use. Please check or fill in the items below and add explanations and other comments in the space provided.

Name: _____

Address: _____

Which of the following terms best describes your job?

¤ Programmer	¤ Systems Analyst	¤ Customer Engineer
¤ Manager	¤ Engineer	¤ Systems Engineer
¤ Operator	¤ Mathematician	¤ Sales Representative
¤ Instructor	¤ Student/Trainee	¤ Other (explain) _____

Does your installation subscribe to the SRL Revision Service? ¤ Yes ¤ No

How did you use this publication?

¤ As an introduction
¤ As a reference manual
¤ As a text (student)
¤ As a text (instructor)
¤ For another purpose (explain) _____

Did you find the material easy to read and understand? ¤ Yes ¤ No (explain below)

Did you find the material organized for convenient use? ¤ Yes ¤ No (explain below)

Specific Criticisms (explain below)

Clarifications on pages
Additions on pages
Deletions on pages
Errors on pages

Explanations and Other Comments

No postage necessary if mailed in U.S.A.

FOLD FOLD

FIRST CLASS
PERMIT NO. 155
RALEIGH, N. C.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
P.O. BOX 9361
RALEIGH, NORTH CAROLINA 27603

ATTN: PROGRAMMING DOCUMENTATION
DEPARTMENT 840

OLD FOLD

Printed in U.S.A. C28-6553-2

IBM®
International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601

Printed in U.S.A. C28-6553-2

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