File No. S360-30
Order No. GC30-2003-4

OS

IBM

Systems Reference Library

IBM System/360
Operating System
Queued Telecommunications Access Method
Message Processing Program Services

Program Number 360S-CQ-519

PREFACE

This publication is intended for the problem programmer assigned to write a message processing program to support a QTAM-controlled telecommunications system operating under the IBM System/360 Operating System. Included is a general discussion of message processing programs, followed by a detailed description of the services QTAM provides in support of a message processing program. The QTAM services are provided through standard macro-language statements such as GET, PUT, OPEN, and CLOSE.

The first four sections of a companion publication, IBM System/360 Operating System: QTAM Message Control Program, GC30-2005, contain general information of interest to the programmer writing a message processing program, i.e., telecommunications applications, concepts and terminology, and message formats.

The prerequisite for a thorough understanding of this publication is a basic knowledge of System/360 machine concepts and of the System/360 Operating System.

Fifth Edition (June 1971)

This is a major revision of, and obsoletes, C30-2003-3 and Technical Newsletter GN30-2532. Besides general maintenance changes, an appendix has been added that contains QTAM error messages and abend codes.

Significant new material has been added throughout, and existing material has been changed extensively; therefore, no vertical lines or bullets appear in the margins, and the manual should be reread in its entirety.

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems or equipment, refer to the latest SRL Newsletter for the editions that are current and applicable.

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

This manual has been prepared by the IBM Systems Development Division, Publications Center, Department E01, P.O. Box 12275, Research Triangle Park, North Carolina 27709. A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be sent to the above address. Comments become the property of IBM.

CONTENTS

FIGURES

In the IBM System/360 Operating System, an access method is a procedure for transferring data between main storage and an input/output device. A variety of access methods is available to the user of the operating system. One of these, the Queued Telecommunications Access Method (QTAM), controls data transfer between main storage and remote terminals.

QTAM is a generalized input/output control system that extends the techniques of data management to the telecommunications environment. Data sets used by the problem programmer are queues of messages incoming from, or outgoing to, remote terminals via communication lines. Even though the time and order of the arrival and departure of messages to and from the central processing unit (CPU) are unpredictable, the programmer handles them as if they were organized sequentially.

Unlike other commonly used access methods, QTAM furnishes more than just the mechanics for input/output operations. In addition to the standard GET/PUT macro instruction support for message processing programs, QTAM provides a high-level and flexible message control language. QTAM-supplied macro instructions can be used to construct a complete message control program that controls the flow of message traffic from one remote terminal to another (message switching application), and between remote terminals and any message processing programs (message processing applications).

A QTAM message control program is completely device-dependent, with all communication lines and terminals identified to the system. Through data set definition and control information macro instructions, the user specifies his equipment configuration and the main storage areas (buffers) required for his applications. These macros generate the tables and lists of control information that define the environment of the system for the QTAM logic. The number and size of the buffers required are specified by the user, and are one of the primary resources in the telecommunications system. The buffers are allocated to a common buffer pool from which QTAM automatically and dynamically uses them in accordance with immediate requirements.

The message processing program services of QTAM enable a programmer to process messages from a telecommunications network with the same easy-to-use macro instructions that he uses for his local input/output devices. Because a QTAM message control program performs the input/output operations, a completely device-independent message processing program can be written. The programmer is shielded from the time- and device-dependent aspects of the telecommunications environment.

For a QTAM message control program to handle the flow of message data between a message processing program and the remote terminals in a system, there must be an interface between the message control program and the message processing program. QTAM (in the form of macro instructions) provides facilities that enable the programmer to establish this interface from the message processing program.

This publication describes in detail the services QTAM provides in support of a message processing program. The message control program is discussed in general terms when necessary to give a complete picture of how a message processing program fits into a QTAM-controlled telecommunications system. For detailed information on the message control program, refer to the publication IBM System/360 Operating System: QTAM Message Control Program, GC30-2005.

In telecommunications terminology, a message is the unit of work with which the programmer is concerned when he writes a processing program. A message is composed of two parts: the message header and the message text. The header portion contains control information about a particular message used by the message control program in performing its functions. This information can include a destination code (for example, a message processing program), the code name of the originating terminal (source code), a message-type indicator, and other fields containing control-type data. The text portion of a message consists of the information of concern to the party receiving the message. This party can be a message processing program.

A message processing program normally consists of an analysis routine or processing routines (or both) that take action on the text portion of a message. A response message may or may not be generated.

An analysis routine is user-written coding that examines the content of a message to determine which course of action is to be taken. With this decision, the analysis routine establishes linkage to the processing routine required to perform the necessary action on the message. The complexity of the analysis routine varies directly with the total number of courses of action that can be required by the incoming messages. The same method should be used for detecting all message types. For example, a message-type character can always appear in a prespecified position in the message header or in the first position of the text.

The processing required may be standard for all messages routed to a message processing program. In this case, an analysis routine is not required.

All processing routines are also user-provided. There must be one processing routine for each specific course of action required by a message. A message processing routine is required when the user wishes to cause a shutdown of the QTAM message control program. At execution time, a processing routine resides either in main storage, as an integral part of the message processing program, or on a DASD library. If the latter method is selected, each processing routine is assembled or compiled independently of the rest of the message processing program. It is link-edited onto a DASD library and brought into main storage dynamically as needed via a LINK macro instruction. Unlike a message control program, the message processing program may contain control program services (CPS) macro instructions as well as QTAM macro instructions.

Note: The user must ensure that the operating system subroutine error trace scheme can function. This may be done by making a SAVE macro the first instruction in each message processing program. For detailed information, see IBM System/360 Operating System: Supervisor and Data Management Services, GC28-6646.

QTAM provides certain functions in support of a message processing program and the telecommunications system. These functions include:

1. Obtaining a message for processing,
2. Placing a response message, if any, on a destination queue or another process queue.

These two functions (unlike the functions performed by the processing routine and the analysis routine) are peculiar to QTAM, which provides macro instructions to aid in performing them.

The GET macro instruction obtains a message from the main storage process queue and places it in a user-specified work area, where it is then analyzed and processed. The PUT macro instruction causes a response message to be placed on a destination queue. These macro instructions are described in detail under the Obtaining Messages and Placing Response Messages section.

A QTAM message control program performs the actual input/output operations required by a message processing program. The message control program must be executed in the highest priority partition or region. As many message processing programs as there are partitions or regions remaining can operate concurrently with the message control program. In MFT, each message processing program must operate in a partition separate from other message processing programs and from the message control program in order for QTAM to provide asynchronous operation for all programming components of the system. In MVT, each processing pro-

gram may operate in a separate region or may be attached by the message control program. If the processing programs are attached, the message control program must be the calling program and must have the highest priority.

After being assembled, link-edited, and placed on a library, a message processing program can be executed by placing the appropriate job control cards in the input job stream, following the job control cards for the message control program, or for a different message processing program.

Figure 1 shows a sample structure for a message processing program. The DCB macro instructions define the data control block for the main storage process and destination queues. The user must define all queues referred to by his program. The OPEN macro instruction prepares these data control blocks for use in processing. The statement "other initialization instructions" represents any instructions the user wishes to include to clear storage areas, etc. GET obtains a message. The "analysis" instructions determine which processing routine is needed to process the message. The appropriate LINK macro instruction brings the specified processing routine into main storage and executes the routine. Response messages generated by the processing routine are placed on the appropriate destination queue by the PUT macro instruction.

The user then performs a test to determine whether processing should terminate. If processing should continue, the program branches to GET to obtain the next message. If processing should terminate, the program performs any necessary termination functions.

The program structure shown in Figure 1 assumes that no EODAD keyword operand was specified in the DCB macro instruction for the main storage process queue. If no message has been placed in the main storage process queue by the message control program, the message processing program enters a wait state, and is reentered only when a message arrives for this main storage process queue.

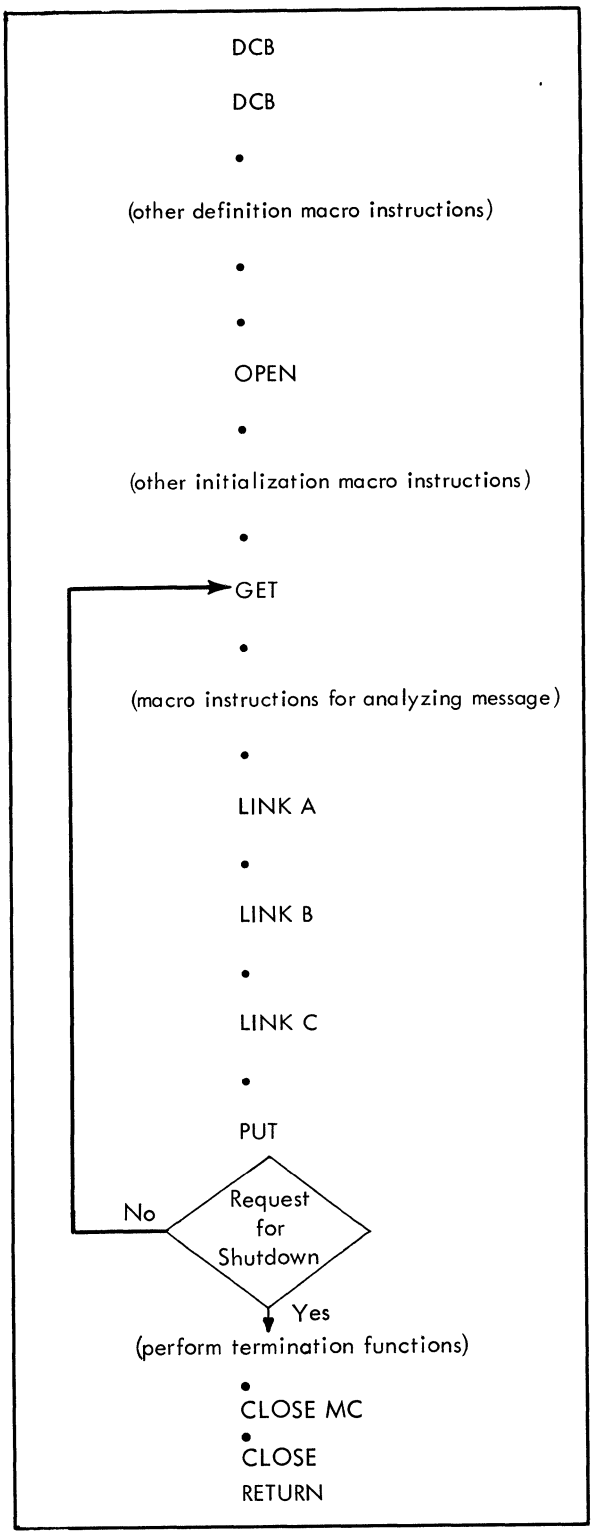


Figure 1. Sample Structure for a Message Processing Program

MESSAGE FLOW WITHIN THE SYSTEM

This section describes the flow of a message between a remote terminal and a message processing program operating under QTAM. The manner in which a QTAM message control program acts as an intermediary between the terminal and the message processing program is discussed in some detail. Figure 2 illustrates this flow.

The input message is prepared at the remote terminal location. Messages are of variable length and consist of two parts: header and text. The terminal sends the message to the computer via a communication line. Step 1 of Figure 2 shows the message passing through an IBM 2701, 2702, or 2703 control unit and the multiplexer channel, and filling available buffers from the QTAM buffer pool defined in the message control program.

The user defines the size of his buffers in the message control program. QTAM inserts control information (known as a prefix) in the first portion of each buffer. The first 32 bytes of a buffer, used to contain a message header, are set aside for a header prefix generated by QTAM. This buffer may contain text data in addition to the header. However, the entire header must be contained in this buffer. The characters transmitted by the remote terminal begin to fill the buffer in the thirty-third byte. The first 22 bytes of a buffer used to contain text data only, are set aside for a text prefix generated by QTAM. Message data begins to fill the buffer in the twenty-third byte.

The user transmits single-segment or multisegment messages. A message segment is message data that occupies one buffer. In single-segment messages, the entire message is contained within one buffer. In multisegment messages, more than one buffer is needed for a message.

In all but the last buffer for a multisegment message, the segment containing a header is shorter than a segment containing text only, because the header prefix generated by QTAM is ten bytes longer than the text prefix. In each buffer used to contain intermediate text, the segments are the same size. In the last buffer for a multisegment message, the message text portion of the segment can be any length equal to, or less than, the buffer length minus 22.

The buffers shown in Figure 2 are each 80 bytes. The first input buffer thus accommodates a message segment of 48 characters (26 constitute the header portion of the message and 22 constitute the text portion). In the second input buffer, the message segment is 58 characters, all of which are text data. The third and last input buffer contains the remaining characters in the message. Because the input message is 150 characters, the message segment size for this buffer is 44.

As soon as a buffer is filled with the first segment of a message, the receive group portion of the line procedure specification (LPS) section of the message control program performs user-selected functions such as code conversion, logging, updating of message counts, incorporation of time-received and date-received information, and input-sequence-number checking. The first three functions can also be performed for text segments. In the example shown in Figure 2, the user has specified that six characters of time-received information be incorporated into the message header (see Step 2).

After performing these functions, the receive group of the LPS routes the prefix (minus the first eight bytes[1]) and message segment to a DASD process queue on the DASD message queues data set. Each DASD process queue is associated with a message processing program. Messages requiring text processing should be routed to the DASD process queue associated with the message processing program that processes that type of message. The user controls this routing either via the message header (the destination code is the name of the DASD process queue) or via LPS macro instructions (which direct messages of a particular type to a particular queue). Step 2 shows the LPS routing a message to a DASD process queue.

For each DASD process queue maintained, QTAM maintains a corresponding queue in main storage. Each main storage (MS) process queue is maintained in buffers from the QTAM buffer pool in the message control program. The number of buffers allocated to a MS process queue is specified in a data control block defined in the message processing program that gets messages from that queue. After the data control block for the MS process queue has been opened by the message processing program, a QTAM routine in the message control program automatically passes the message segment from the DASD process queue to a buffer in the MS process queue (see Step 3). In moving the prefix and segment to the buffer, the eight bytes that were deleted when the prefix and segment were placed on the DASD

---

[1]The first eight bytes of a header or text prefix contain control information used only in main storage buffer handling. Therefore, these bytes are not placed on the direct access device.

process queue are restored, so that the prefix length is once again 32 (header prefix) or 22 (text prefix).

Each time the message processing program gains control and issues a GET (Step 4), QTAM passes message data from the MS process queue to a user-specified work area in the message processing program. Message data is provided in the work unit specified by the user in the data control block. The work unit may be a complete message, a message segment, or a record. Before moving the message data to the work area, QTAM strips the header and text prefixes from the message segments. QTAM places a four-byte prefix in the first four bytes of this work area. This prefix indicates the size and type of the work unit on which the processing program is to operate. After receiving the message data, the message processing program processes it as required by the application.

A message processing program that generates a response message must define and open a data control block governing message transfer before it attempts to place the message on a DASD destination queue. This data control block contains information needed by QTAM to establish an MS destination queue. When a PUT macro instruction is issued by a message processing program (Step 5), QTAM moves the message data from the user-specified work area into the MS destination queue. The header and text prefixes are attached to the message segments in the buffer areas that make up the MS destination queue.

As the message data fills the buffers, QTAM inserts chaining addresses and other necessary control information into the prefix fields. The response message generated by a message processing program can be any size. (The one used in Figure 2 is 120 characters.)

After the header or text prefixes have been added in the MS destination queue, QTAM places the segment into the appropriate DASD destination queue on the DASD message queues data set (Step 6).

QTAM retrieves message segments from the DASD destination queues on a first-in first-out basis within priority groups. The message segments are brought in from the direct access device and placed in available buffers (Step 7). The "send group" of the LPS section in the message control program then performs user-selected functions such as converting the code of the message to the transmission code of the terminal, incorporating time-sent and date-sent information in the header, message logging, and updating of message counts. These operations are performed in the buffers that receive the message segments from the direct access device. QTAM then strips the header and text prefixes from the message segments and transmits the message to the appropriate terminal (Step 8).

The header and text prefixes described in this section are generated automatically and are used by QTAM routines. No programming considerations are required by the user for the manipulation of the buffers and their prefixes. They are described here to give a complete view of the message flow through the system.

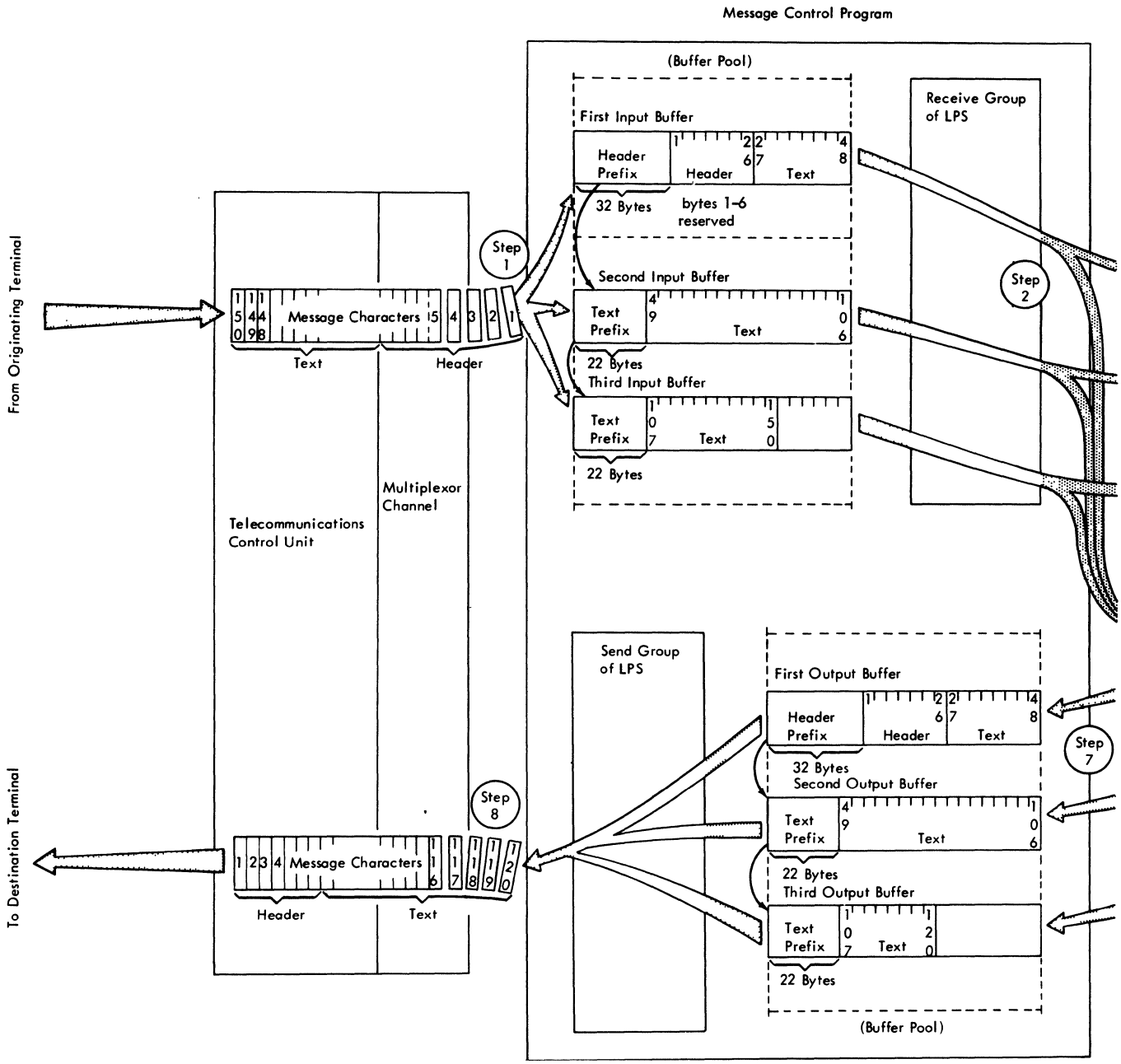


Figure 2. QTAM Message Flow (Part 1 of 2)

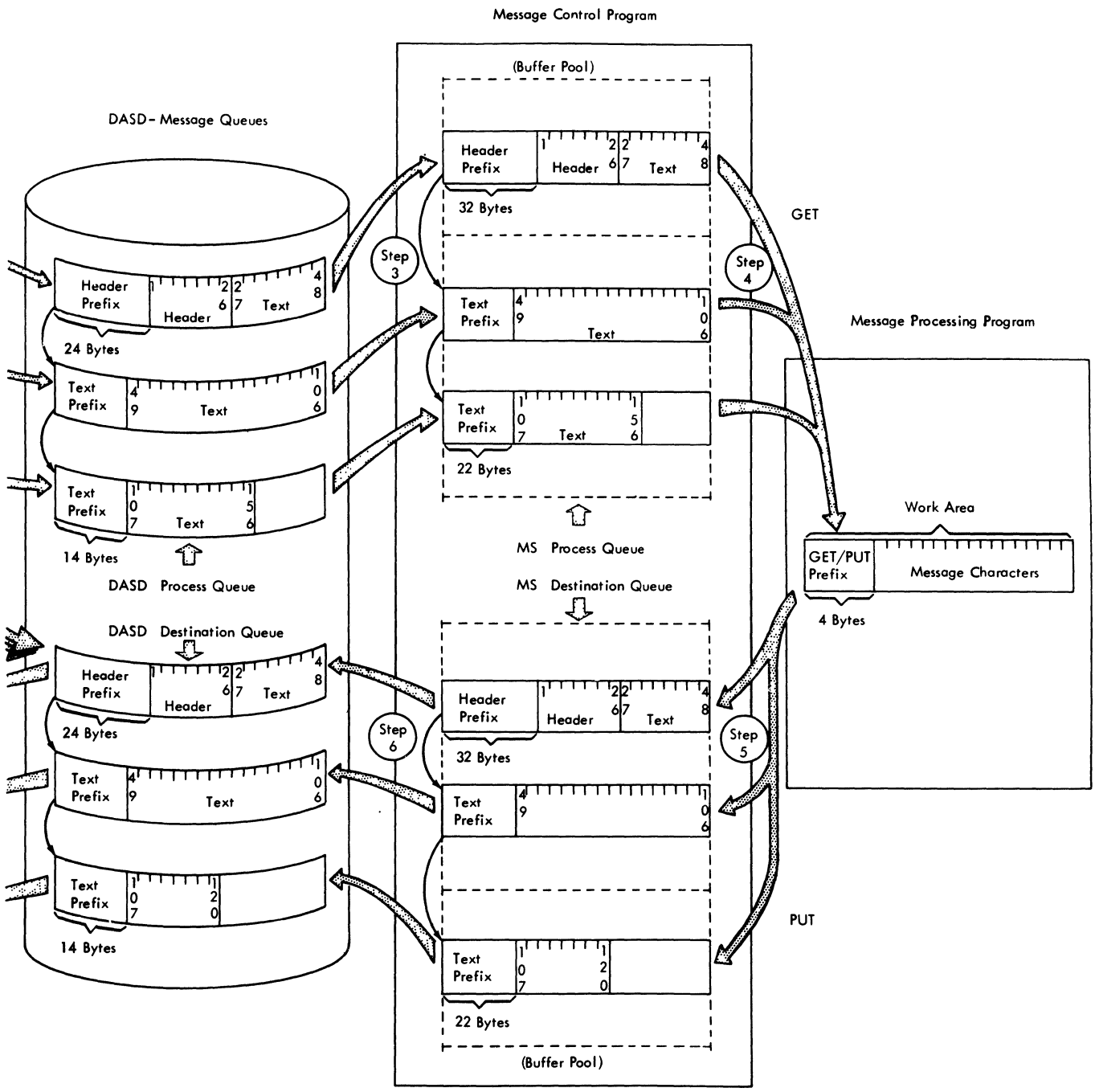


Figure 2. QTAM Message Flow (Part 2 of 2)

The routing of messages between a message processing program and remote terminals is handled by the QTAM message control program. Because a message processing program depends on the message control program to perform its input/output operations, an interface must be established between a message processing program and the message control program. QTAM provides the following services that allow this interface to be established from a message processing program:

- Defines the message control program interface (DCB macro instruction)
- Initializes and activates the message control program interface (OPEN macro instruction)
- Obtains messages and transfers response messages (GET and PUT macro instructions)
- Deactivates the message control program interface (CLOSEMC macro instruction)

Unlike the functions performed by the analysis and processing routines of a message processing program, these functions are partially or wholly peculiar to QTAM and the telecommunications environment. Therefore, QTAM provides routines to accomplish these functions. Linkage to these routines is established by QTAM macro instructions in a message processing program. The remainder of this section discusses these macro instructions in detail.

DEFINING THE MESSAGE CONTROL PROGRAM INTERFACE

Message transfer from a main storage (MS) process queue to a message processing program is controlled by a data control block. If the message processing program generates a response message, message transfer from the message processing program to a main storage (MS) destination queue is governed by another data control block. The user must define, open, and close these data control blocks in the message processing program.

A DCB macro instruction must be specified for each MS process queue (queue providing input to the message processing program) and for each MS destination queue (output queue) referred to by a message processing program. A DD statement must be provided for each DCB. The DD statement must contain only DUMMY in the operand field. This indicates that no I/O device is being assigned. The name of the DD statement must be identical with the name specified in the DDNAME keyword operand of its associated DCB macro instruction.

Examples of DD statements for a message processing program are:

Name	Operation	Operand
MAINPQ	DD	DUMMY
RESPMT	DD	DUMMY

The data control blocks generated by the expansion of these macro instructions are not associated with data sets themselves. Instead, they contain the necessary control information to establish the interface to the QTAM message control program, which uses this control information in transferring data to and from the message processing program.

The message control program performs the actual input/output operations needed to receive and send messages over communication lines. The incoming messages that must be routed to a message processing program are first placed in the DASD process queue associated with the message processing program. After the data control block for the MS process queue has been opened, and the first GET macro instruction is issued in the message processing program, the message control program begins transferring messages from the DASD process queue to the MS process queue. While the MS process queue remains open, the message control program automatically replenishes it with messages from the DASD process queue in anticipation of the next GET.

Similarly, the MS destination queue must be defined (by a DCB macro instruction) and opened in the message processing program if a response message is to be sent. When a PUT macro instruction is issued, the PUT routine transfers the message to the MS destination queue and signals the message control program that a message is ready to be placed on a DASD destination queue. The message control program places the message on the appropriate DASD destination queue. Finally, the message control program

retrieves the message from the DASD destination queue and transmits the message to the appropriate terminal.

Messages generated by the message processing program can also be sent to another message processing program. This is accomplished via a PUT macro instruction. The data flow is identical with that described previously for sending a message to a terminal, except that message transfer from the MS destination queue is to a DASD process queue.

Thus, the MS process and MS destination queues defined in the message processing program serve as the connectors between the message control program and the message processing program. When the QTAM message control program is used as an intermediary between the message processing program and the remote terminals, the message processing program is completely device-independent.

Data Control Block (DCB) Macro Instruction

In a message processing program, the DCB macro instruction defines two types of data control blocks.

One data control block contains the information needed to create the MS process queue from which messages can be obtained for processing. The other data control block contains the information needed to create the MS destination queue, and is required only if response messages are to be generated.

Normally, only one MS process queue is defined in a message processing program. All message types to be processed by a particular message processing program are obtained from the same MS process queue. An analysis routine determines the type of each message and establishes linkage to the appropriate processing routine. However, it is possible to have multiple MS process queues in the same message processing program; that is, one MS process queue for each type of message to be processed. In this case, the ECDAD keyword operand should be used to regain control if no message appears in this MS process queue. Execution of another GET from a different MS process queue can be effected, and so forth.

Only one MS destination queue is required (and only if a response message is to be generated), regardless of the number of MS process queues.

The DCB macro instruction causes the allocation of main storage space for a data control block at assembly time. Parameters based on the operands in the macro instruction are included in the data control block. No executable code is generated through this macro.

Figures 3 and 4 show the operands for DCB macro instructions for two data control blocks: the MS process queue DCB, and the MS destination queue DCB, respectively.

Name	Operation	Operand
dcb	DCB	keyword operands

dcb
: The address of the DCB macro instruction. The name must be specified.

keyword operands
: The operands that can be included to facilitate the control of message transfer between the message processing program and a DASD process or DASD destination queue. The operands for the two types of DCB macro instructions in the message processing program are described in Figures 3 and 4.

When a parameter is provided by an alternate source, one or more symbols appear in the table. When there is no alternate source, no symbols are shown. The symbols have the following meanings:

Symbol	Meaning
PP	The value of the operand can be provided by the user's problem program any time before the data control block exit at open time.
OE	The value can be provided by the user's problem program any time up to and including the data control block exit provided at open time.

Keyword Operands	Alternate Source	Value Description
DSORG=MQ		MQ Identifies the data control block as one governing message transfer to or from a telecommunications message processing queue. If this operand is omitted, the telecommunications job, when executed, is terminated.
MACRF=G		G Specifies that access to the MS process queue is to be gained with the GET macro instruction. If this operand is omitted, the telecommunications job, when executed, is terminated.
DDNAME=ddname	PP	ddname Is the name that appears in the DD statement associated with the data control block. This name is also the name used in the PROCESS macro instruction to identify the MS process queue. If this operand is omitted and the value is not provided through an alternate source, the telecommunications job, when executed, is terminated.
[BUFRQ=absexp BUFRQ=1]	OE	absexp Is the number of buffers to be requested in advance for the GET macro instruction. "absexp" must be less than 256. When BUFRQ=0 or if this operand is omitted and the value is not supplied by an alternate source, BUFRQ=1 is assumed.
SOWA=absexp	OE	absexp Is the size in bytes of the user-provided input work area; "absexp" must be less than 32,768. This value must include the four-byte user prefix. If this operand is omitted and the value is not provided by an alternate source, the telecommunications job, when executed, is terminated.

Figure 3. Keyword Operands for the Main Storage Process Queue DCB Macro Instruction (Part 1 of 2)

Keyword Operands	Alternate Source	Value Description
[RECFM=G RECFM=S RECFM=R]	OE	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 (CR), line feed (LF), new line (NL), or end-of-block (EOB) character]. If this operand is omitted and the value is not supplied by an alternate source, RECFM=S is assumed.
[EODAD=relexp]	OE	relexp Is the symbolic address of a user-provided routine to be entered if no messages are available when a GET macro instruction is issued. If this operand is omitted and the value is not supplied through an alternate source, a WAIT macro instruction is implied.
[TRMAD=relexp]	OE	relexp Is the symbolic address of a user-provided area to contain the terminal name. When a GET macro instruction is issued, QTAM places the source terminal name at the specified address. The length of the area must be equal to or larger than the maximum size terminal name or process queue name used. If the Auto Poll facility or switched line groups are being used, the SOURCE macro instruction is required in the LPS for these line groups. If this operand is omitted and the value is not provided through the alternate source, the telecommunications job, when executed, is terminated.
[SYNAD=relexp]	OE	relexp Is the symbolic address of a user-provided routine to be entered if a work unit is longer than the work area provided for input. If this operand is omitted and the value is not provided through the alternate source, the remainder of the work unit is supplied when the next GET macro instruction is issued. Data remaining in the buffer, however, will be lost for a GET segment.

Figure 3. Keyword Operands for the Main Storage Process Queue DCB Macro Instruction (Part 2 of 2)

Keyword Operands	Alternate Source	Value Description
DSORG=MQ		MQ Identifies the data control block as one governing message transfer to or from a telecommunications message processing queue. If this operand is omitted, the telecommunications job, when executed, is terminated.
MACRF=P		P Specifies that messages are to be transferred to an MS destination queue by the PUT macro instruction. If this operand is omitted, the telecommunications job, when executed, is terminated.
[DDNAME=ddname]	PP	ddname Is the name that appears in the DD statement associated with the data control block. If this operand is omitted and no value is provided through an alternate source, the telecommunications job, when executed, is terminated.
[RECFM=G RECFM=S RECFM=R]	OE	G, S, or R Specifies the work unit as follows: G -- message (the contents of the work area is considered to be a full message). S -- segment (the contents of the work area is considered to be a segment and it should fit into the buffer an even number of times). R -- record (the contents of the work area is considered to be a complete record). If this operand is omitted and the value is not provided through an alternate source, RECFM=S is assumed.
[TRMAD=relexp]	PP	relexp Is the symbolic address of a user-provided area to contain the terminal table entry name. The length of the area must be equal to, or larger than, the maximum size terminal name or process queue name used. If this operand is omitted, the telecommunications job, when executed, is terminated. When a PUT macro instruction is issued, the destination terminal name must be provided at the specified address. The name must be defined in a terminal table entry within the message control program.

Figure 4. Keyword Operands for the Main Storage Destination Queue DCB Macro Instruction

Examples:

1. A DCB macro instruction that defines the parameters of a data control block associated with a main storage process queue (see Figure 3 above).

   Note: For optimum performance, the message control program should always attempt to read ahead to keep one full work unit ready to be transferred in response to a GET. To achieve this, the BUFRQ operand should specify twice the number of buffers required to hold a single work unit. For example, a message of 200 data characters would require three 100-character buffers (200 data characters plus 76 characters of QTAM control information). The BUFRQ operand, in this case, should specify six buffers.

```
┌────────┬─────────┬──────────────────────┐
|Name    |Operation|Operand               |
├────────┼─────────┼──────────────────────┤
|PPMQIN  |DCB      |DDNAME=PRCIN,         |
|        |         |DSORG=MQ,             |
|        |         |MACRF=G,              |
|        |         |BUFRQ=6,SOWA=300,     |
|        |         |RECFM=G,              |
|        |         |TRMAD=SOURCE,         |
|        |         |SYNAD=ERROR           |
└────────┴─────────┴──────────────────────┘
```

2. A DCB macro instruction that defines the parameters of a data control block to govern response message transfer (see Figure 4 above).

```
┌────────┬─────────┬──────────────────────┐
|Name    |Operation|Operand               |
├────────┼─────────┼──────────────────────┤
|PPMQOUT |DCB      |DDNAME=PRCOUT,        |
|        |         |DSORG=MQ,MACRF=P,     |
|        |         |RECFM=G,              |
|        |         |TRMAD=DESTN           |
└────────┴─────────┴──────────────────────┘
```

EODAD Routine

The EODAD operand of the DCB for the main storage process queue allows the user to specify a routine that will receive control when a GET is issued for that queue and there is no message there. This routine may do some processing and then attempt to get messages from some other queue.

For example:

```
PRCPROG   .
          .
          .
          GET   QUEUE1     EODAD=SUBRTN1
          .
SUBRTN1   .
          .
          GET   QUEUE2     EODAD=SUBRTN2
          .
SUBRTN2   .
          .
          GET   QUEUE3     EODAD=PRCPROG
          .
```

In this situation, if all the queues are empty, this process program loops continually, relinquishing control only for I/O interrupts.

It is suggested that the user issue some sort of wait before completing the loop. This might be accomplished by issuing an SVC STIMER for 3 milliseconds or so to allow the higher priority partition to get control. This will allow the message control program to process at least one message before giving control back to the processing program.

If this GET-EODAD loop is absolutely necessary, then the job in which it occurs must be in the lowest priority job in the system.

HANDLING THE MESSAGE CONTROL PROGRAM INTERFACE

QTAM provides macro instructions to handle the interface between the message processing program and the message control program. These macro instructions have the following functions:

- Initializing and activating (OPEN macro instruction)
- Deactivating (CLOSE and CLOSEMC macro instructions)

Initialization and activation of the interface to the message control program is accomplished by issuing an OPEN macro instruction for the MS process queue(s), and if response messages are to be generated, for the MS destination queue. The OPEN routine performs the initialization functions that are necessary to activate the interface. No QTAM macros (including OPEN, CLOSE, GET, and PUT) may be processed before the DCB for the direct access device message queue has been opened or after it has been closed. After the MS process and MS destination queues have been opened, the transfer of data to and from the message processing program can commence.

When not required for further operations, the MS process and MS destination queues should be deactivated. The CLOSE macro instruction accomplishes this by removing the queues from active use. The interface to the message control program is effectively destroyed, and no further messages may be obtained from the MS process queue or placed on the MS destination queue.

When a processing program abends or is canceled and adequate main storage is not available, the abend may take a part of the processing program storage. If the area taken contains the processing program's DCB, the DCB will not be closed, causing either a program check when the next checkpoint is taken or an abend with a system completion code of 0A5 when the processing program is reloaded. To avoid such a situation, more main storage must be allocated or the DCBs placed at the end of the processing program.

The deactivation of the MS process and MS destination queues can be performed in a special deactivation routine within the message processing program. This special routine can be entered as the result of a message received from a terminal operator directing that termination functions be performed. The analysis routine is coded to recognize this message (by a unique character indicating the message-type) as one requiring the execution of the deactivation routine. Linkage is established to the deactivation routine, which includes the necessary CLOSE macro instructions and instructions to perform other required termination functions.

OPEN Macro Instruction

The data control blocks for MS process queues must be opened before the message processing program can issue GET instructions to receive messages from the queue. The MS destination queue must be opened before the message processing program can PUT response messages to any destination. The OPEN macro instruction causes the initialization of the message control program interface to be completed.

All data control blocks in the message processing program can be opened collectively with one OPEN or opened separately by individual OPEN macro instructions.

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

symbol
: Either the name of the first instruction generated by OPEN or the name of a parameter list created by OPEN. If the MF=L operand is specified, "symbol" must be included. It becomes the name of the parameter list. If no MF operand is specified, or the MF=(E, listname) operand is specified, "symbol" is optional. If included, it becomes the name of the first instruction generated by OPEN.

dcb$_1$
: The address of the data control block to be opened. If register notation is used, the register must contain the address.

INPUT
: Specifies that this data set can be used for input. INPUT should be specified for MS process queue data sets.

OUTPUT
: Specifies that this data set can be used for output. OUTPUT should be specified for MS destination data sets.

MF=L
: Causes creation of a parameter list based on the OPEN operands. No executable code is generated. The user must specify this form of the OPEN with his program constants. The parameters in the list are not used until the problem program issues an OPEN (or CLOSE) macro with an MF=(E, listname) operand referring to the list (see following text). The name specified in the name field of the OPEN macro becomes the name assigned to the parameter list.

MF=(E,listname)
: Causes execution of the OPEN routine, using the parameter list referred to by "listname." This list was created by a macro having the MF=L operand specified, as described previously. Parameters specified through a macro having an MF=(E,listname) operand override corresponding parameters in the list. An OPEN macro with the MF=(E,listname) operand can also refer to a parameter list created by a CLOSE macro with an MF=L operand.

CLOSE Macro Instruction

The CLOSE macro instruction deactivates the interface between the message processing program and the message control program. After the MS process queue has been closed, no further messages can be obtained from it for processing. Similarly, after the MS destination queue has been closed, no further response messages can be placed on it. When a CLOSE is issued, main storage and subroutines acquired at open time are released; fields in the data control block that were initialized at open time are cleared.

Name	Operation	Operand
[symbol]	CLOSE	([dcb_1,,]...) [,MF=L] [,MF=(E,listname)]

symbol
: Either the name of the first instruction generated by CLOSE or the name of a parameter list created by CLOSE. If the MF=L operand is specified, "symbol" must be included. It becomes the name of the parameter list. If no MF operand is specified, or the MF=(E,listname) operand is specified, "symbol" is optional. If included, it becomes the name of the first instruction generated by CLOSE.

dcb_1
: The address of the data control block to be closed. If register notation is used, the register must contain the address.

MF=L
: Causes creation of a parameter list based on the CLOSE operands. No executable code is generated. The user must specify this form of the CLOSE with his program constants. The parameters in the list are not used until the problem program issues a CLOSE (or OPEN) macro instruction with an MF=(E,listname) operand referring to the list (see following text). The name specified in the name field of the CLOSE macro becomes the name assigned to this parameter list.

MF=(E,listname)
: Causes execution of the CLOSE routine by using the parameter list referred to by "listname." This list was created by a macro having the MF=L operand specified, as previously described. Parameters specified through a macro having an MF=(E, listname) operand override corresponding parameters in the list. A CLOSE macro with the MF=(E, listname) operand can also refer to a parameter list created by an OPEN macro with an MF=L operand.

OBTAINING MESSAGES AND PLACING RESPONSE MESSAGES

QTAM provides the message processing program user with facilities for obtaining messages for processing and placing response messages on a DASD destination queue. Even though the messages are received from (and sent to), remote terminals via communication lines, the programmer uses GET/PUT macro instructions for obtaining and sending messages. A QTAM message control program performs the device-dependent input/output operations for the message processing program.

The main connectors between a message control program and a message processing program are the MS process and MS destination queues. After the MS process queue has been defined and opened, a message is obtained from this queue by issuing a GET macro instruction. Once obtained, the message is analyzed and processed as required by the application. If a response message is generated, it is placed on the MS destination queue (after it has been defined and opened) by a PUT macro instruction. The message control program then transfers the message to the appropriate DASD destination queue, and finally sends the message to the remote terminal.

GET Macro Instruction

GET obtains the next sequential work unit from the MS process queue indicated by the first operand. This operand is the name of the data control block associated with the MS process queue. The user specifies in the DCB macro instruction for the MS process queue, the work unit with which he is operating (message segment, record, or complete message).

If the user specifies "segment" in the DCB, the work area must be large enough to accommodate an entire segment. Thus, for a buffer of 100 bytes, the work area must be at least 82 bytes.

Maximum text segment = buffer size - size of text prefix = 78

Work area size = maximum text segment + work area prefix = 82

If all the data in a segment cannot be accommodated in a work area, the remaining data is lost. There is no advantage in defining a work area of greater size, since no more than one text segment is transferred into the work area for each GET.

If the EOT character happens to be the last character in the message, an additional buffer with no data in it will be carried through the LPS with the other buffers for that message. If the user has specified segment in the DCB and issues a GET for that data set, this empty buffer will be placed in the work area. This condition can be detected by checking the first two bytes of the GET/PUT prefix of that segment for a count of four.

If the user specifies "message" in the DCB, the data is transferred until the work area is full or the entire message is moved, whichever occurs first. For a given application, if the size of a message is known, the work area size should be set equal to this value. If the work area can not accommodate the full message, the remaining data is moved into the work area when the next GET is issued. Since a message is the logical unit being considered, the question of whether the work area should be smaller, equal to, or greater than a segment (buffer) size is not relevant.

If the user specifies "record" in the DCB, the message data is moved into the work area until one of the following occurs:

1. EOB, ETX (X'03'), new line (NL), carriage return (CR), or line feed (LF) is encountered. (Data is transferred up to and including the first of these characters.)
2. An entire message has been transferred.
3. The work area has been filled.

Since the logical unit considered is a record, the size of the work area should be defined to accommodate a complete record.

When "record" has been specified in the DCB, the record must have been translated to EBCDIC in the Receive group of the LPS.

The synchronous error exit (SYNAD) will be taken, if specified, whenever any of the three forms of the GET macro is completed with data remaining in the buffers.

The second operand is the address of the work area into which the work unit is to be placed. The user must define this work area in his problem program, and the size of the work area is specified in the SOWA keyword operand of the DCB macro instruction for the MS process queue.

If there is no work unit on the MS process queue, and no user-written routine is provided for this situation, the message processing program in which the GET appears enters a wait state. A user-written routine can be specified via the EODAD keyword operand in the DCB macro instruction for the MS process queue.

If the work unit has been received via a nonswitched line, GET causes the name of the source terminal to be placed in the area specified by the TRMAD keyword operand of the DCB macro instruction for the MS process queue. If a processing program puts the work unit in a process queue, zeros are placed in the user-specified area. If the work unit has been received over a switched line or a nonswitched line with the Auto Poll feature, GET causes the name of the source terminal to be placed in the user-specified area only if the SOURCE macro instruction is included in the message control program LPS section that governs that terminal. If the SOURCE macro instruction has not been included, GET causes zeros to be placed in the user-specified area. (See the IBM System/360 Operating System: QTAM Message Control Program publication.)

GET uses the first four bytes in the user-specified work area to record information about the work unit. The first two bytes of the four-byte prefix (called the GET/PUT prefix), contain the number of characters in the work unit. The third byte identifies the type of segment or record (see Figure 5). As shown in the figure, all completed messages are identified by a one in bit position 6. The last byte of the four-byte prefix is not modified by QTAM. The user's message data starts with byte five of the work area.

If RECFM=R is specified in the MS process queue DCB macro instruction, the GET routine transfers to the work area all characters up to and including the first of the following characters: carriage return, line feed, new line, or end-of-block.

Name	Operation	Operand
[symbol]	GET	dcb,workarea

symbol
The name of the macro instruction.

dcb

The symbolic address of the data control block associated with the MS process queue from which the processing program is to obtain a work unit. If register notation is used, the register must contain the address of the DCB.

workarea

The address of the user-defined area in which the work unit is to be placed. If register notation is used, the address of the work area must have been previously loaded intc the register specified.

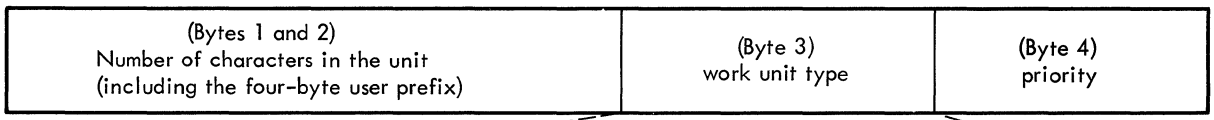
PUT Macro Instruction

The PUT macro instruction causes the processed message, message segment, or record to be transferred from the work area specified to an MS destination queue. The message control program then transfers the work unit to the appropriate DASD destination queue. Before issuing a PUT, the user must ensure that the name of the terminal (to which the work unit is being sent) is in the location specified by the TRMAD keyword operand in the DCB macro instruction for the MS destination queue. Data transfer starts with the fifth byte of the work area.

Prior to issuing a PUT, the user must also place the necessary information in the four-byte GET/PUT prefix. The length of the work unit to be transferred (including the four bytes in the prefix) must be specified in bytes one and two, and the type of work unit must be specified in byte three (see Figure 5). The fourth byte may contain a priority to be associated with the message. The value for this priority is chosen in the same way as for messages originating at remote terminals.

The PUT macro instruction is concerned with one of three logical units, that is, segment, message, or record.

If the user specifies "segment" in the DCB, the data will be transferred from a work area to a buffer. If the work area is larger than the buffer, an error code of X'10' is set in register 15, right-adjusted, and no data is transferred. If the work area is smaller than the buffer, the buffer will not be completely filled and the extra space wasted. It is recommended that the same relationship exist between work area size and buffer size as with the GET segment. If the segments are not in correct sequence, that is, a segment



Third Byte Contents	RECFM=G (complete messages)	RECFM=S (message segments)	RECFM=R (message record)
X'00'		Multiple-segment message. This segment contains a header.	Multiple-block message. This block contains a header.
X'01'		Multiple-segment message. This is an intermediate segment containing text only.	Multiple-block message. This is an intermediate block containing text only.
X'02'	complete messages	Single-segment complete message.	Single-block complete message.
X'03'		Multiple-segment message. This is the last segment and contains text only.	Multiple-block message. This is the last block and contains text only.

Figure 5. Meaning of the Bytes in the GET/PUT Prefix

with EOM does not come before a header segment, an error code of X'40' is set in register 15, right-adjusted.

If the user specifies "message" in the DCB, the contents of the work area are considered to be a full message. If the work area is larger than the buffer, the contents of the work area will be transferred to several buffers. If the work area is smaller than a buffer, space is wasted because the partially filled buffer is transmitted. The size of the work area should therefore be defined to accommodate a single message.

If the user specifies "record" in the DCB, the contents of the work area are considered to be a complete record. PUT does not check EOB , line feed (LF), new line (NL), or carriage return (CR) to delimit the record. If the work area is larger than the buffer, the contents of the work area are transferred to several buffers. If the work area is smaller than the buffer, the buffer is filled in with other message blocks by succeeding PUT macros.

Restriction: Avoid the interleaving of message segments or message records of two different messages being put into the same MS destination queue. For example, assume two MS process queues, A and B. Also assume one MS destination queue, C. Message segments obtained from A and B must not be intermixed while being PUT to C. However, the interleaving of complete messages is permissible.

The format of a record being put from a message processing program must be considered in defining the send sections of the LPS in the message control program. For example, if the TIMESTMP, DATESTMP, or SEQOUT macro instructions are used in the SENDHDR section of the LPS, the number of bytes required by these macros must be reserved at the beginning of the record before issuing the PUT. These fields must be filled with idle characters (X'17' in EBCDIC). Any other macro instructions in the SENDHDR section of the LPS will bypass these idle characters in scanning for the beginning of the header field.

Note also that if a message is to be written just as it was received by a GET, the SENDHDR section of the LPS should not include macro instructions that refer to the header portion of the message. Alternatively, if the header must be worked on in the SENDHDR section, those fields that were processed in the RCVHDR section prior to the GET should be overlaid with idle characters before the subsequent PUT.

Another point to be noted is that an EOA (end of address) sequence should not be generated in a header by a message processing program prior to a PUT. If required, the EOA should be placed in the header by the SENDHDR portion of the LPS after execution of any TIMESTMP, DATESTMP, or SEQOUT macro instructions.

A message cannot be routed to multiple destinations or a distribution list entry by a PUT macro instruction.

Name	Operation	Operand
[symbol]	PUT	dcb,workarea

symbol
: The name of the macro instruction.

dcb
: The symbolic address of the data control block associated with the MS destination queue. If register notation is used, the register must contain the address of the DCB.

workarea
: The address of the user-defined area from which the work unit is to be transferred. If register notation is used, the register must contain the address.

The user may find it advisable during processing to examine control information used by QTAM and to make necessary modifications to the system. QTAM provides a set of macro instructions for this purpose. The macro instructions enable the user to dynamically:

- Activate or deactivate a particular line in a communication line group (STARTLN and STOPLN macro instructions).
- Examine and modify terminal-table entries (COPYT, CHNGT, and RELEASEM macro instructions).
- Examine and modify polling lists (COPYP and CHNGP macro instructions).
- Examine queue control blocks for DASD destination and process queues (COPYQ macro instruction).
- Retrieve for retransmission messages previously sent to terminals (RETRIEVE macro instruction).

In order to dynamically examine and modify the status of the system, these macro instructions must be used in a message processing program.

Routines containing the examination and modification macro instructions can be executed at any time during the operation of the system as the result of a message sent to the message processing program by a terminal operator. Such messages are handled in the same manner as any other message that enters the system from a remote terminal. In other words, the message control program performs the necessary control functions and routes the message to the appropriate message processing program. The message processing program is specified as the destination in the message header. At some prespecified position in the message (for example, in the header or in the first position of the text), a message-type character is specified that identifies the message as one requiring the execution of an examination or modification routine. The analysis routine of the message processing program can be written to recognize this message-type character and branch to the desired routine.

The terminal sending such a message is usually located on the same premises as the CPU and is designated as the "control" terminal. The operator of the control terminal and the CPU could be the same individual. This centralizes the overall control of the telecommunications system. However, the user can designate any terminal in the system as the control terminal.

Some of the macro instructions discussed in this section may also be issued in the message control program. Execution of these macro instructions in the message control program must occur between the OPEN and ENDREADY macro instructions.

WARNING: It is possible that the message control program might use the terminal table, polling list, queue control block, or threshold counters at the same time that the processing program is executing a COPY or CHANGE macro instruction. Accordingly, the line should first be stopped by issuing a STOPLN macro instruction. The change or copy macro instruction is then issued and followed by a STARTLN to restart the line.

LINE ACTIVATION AND DEACTIVATION

Normally, the lines in a line group are automatically prepared for message transmission when the line group is opened in the message control program. When issued in a message processing program, the STOPLN and STARTLN macro instructions enable the user to dynamically deactivate and reactivate a specific line (or all the lines) within the line group at any point during the operation of the system.

STOPLN effects a temporary deactivation of a specific line when the line is expected to be reactivated by a subsequent STARTLN macro instruction.

Note: The STARTLN/STOPLN module (IECKLNCH) must be included in the same overlay as the STARTLN or STOPLN macro instructions.

Stop Line (STOPLN) Macro Instruction

STOPLN removes a communication line from active use. This macro instruction causes operations on the designated line to stop immediately after completion of any message currently being received or transmitted. Transmission of any messages remaining in

the queue for the line resumes when a STARTLN macro instruction reactivates the line.

A message processing DCB must be opened before a STOPLN can be issued.

Name	Operation	Operand
[symbol]	STOPLN	termname, {rln / ALL}

symbol
Is the name of the macro instruction.

termname
Is the name of any terminal in the line group, but not necessarily the name of a terminal on the line being stopped. If register notation is used, the register must contain the address of the data control block for the line group. It cannot contain the terminal name. (The COPYQ macro instruction may be used to locate the DCB.)

If an invalid terminal name is specified, an error code of X'20', right-adjusted, is set in register 15. If the DCB for the line group has not been opened, an error code of X'01' is set in register 15. In either case, the STOPLN has no effect.

rln
Is the relative line number (within the line group) of the line to be deactivated. If register notation is used, the general register specified must contain the relative line number in binary form. If an invalid rln is specified, a code of X'08', right-adjusted, is set in register 15.

ALL
Specifies that all lines in the line group are to be deactivated.

If no errors were detected in the STOPLN macro, register 15 contains all zeros.

Start Line (STARTLN) Macro Instruction

STARTLN:

1. Allows message transmission to resume on a particular line in a communication line group.

2. Allows message transmission to resume on all lines in a communication line group. The user must have previously opened the line group in the message control program.

If a line is deactivated by a STOPLN macro instruction, STARTLN must be issued before message transmission on that particular line can resume.

In each of the preceding 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 nonzero character -- this character is initialized to 1 and can be changed by the CHNGP macro instruction.) If STARTLN is used, polling or enabling of input lines begins after the execution of that macro instruction. Initial polling or enabling of input lines in a line group begins when the line group is opened in the message control program. If activation of a line group was deferred by inclusion of the IDLE operand in the OPEN macro for the line group, a STARTLN macro must be issued to activate the lines.

Restriction: When a STARTLN is issued for a stopped line, QTAM loops for a maximum of 30 seconds waiting for the line to indicate 'ending' status. If no interrupt has been received after 30 seconds, the following system error message will be printed.

IEC806I XXX LINE UNAVAILABLE, ENDING STATUS NOT RECEIVED.

An interrupt must be forced on the line before attempting another STARTLN.

The line will be ignored until a STARTLN macro is issued to that line or a STARTLN operator control message for that line is sent.

An attempt to initiate input/output operations on a line on a control unit that is not operational will result in the following system error message:

IEC809I xxx CONTROL UNIT NOT OPERATIONAL

Where xxx is the line address of the attempted I/O operation.

The line will be ignored until a STARTLN macro is issued to that line or a STARTLN operator control message for that line is sent.

Name	Operation	Operand
[symbol]	STARTLN	termname, {rln / ALL}

symbol
Is the name of the macro instruction.

termname
Is the name of any terminal in the line group, but not necessarily the name of a terminal on the line being started. If register notation is used the register must contain the address of the data control block for the line group. (The COPYQ macro instruction may be used to locate the DCB.) It cannot contain the terminal name. If an invalid terminal name is specified, an error code of X'20', right-adjusted, is set in register 15. If the DCB for the line group has not been opened, an error code of X'01' is set in register 15. In either case, the STARTLN has no effect.

rln
Is the relative line number, within the line group, of the line to be reactivated. If register notation is used, the general register specified must contain the relative line number in binary form. If an invalid relative line number is specified, a code of X'08', right-adjusted, is set in register 15.

ALL
Specifies that all lines in the line group are to be activated.

If no errors were detected in the STARTLN macro, register 15 contains all zeros.

EXAMINING AND MODIFYING THE TERMINAL TABLE

QTAM provides macro instructions that enable the user to examine and dynamically change the control information contained in a terminal-table entry.

The COPYT macro instruction causes the contents of a specified terminal-table entry to be copied into a work area. This macro instruction can be used in conjunction with the CHNGT macro instruction, which substitutes a new terminal-table entry for a superseded one. The user issues a COPYT, examines the information, changes it if necessary, and issues a CHNGT.

The user can also change terminal-table information via the RELEASEM macro instruction. RELEASEM causes the intercept bit for a specified terminal-table entry to be set to 0. All messages intercepted for that terminal are then transmitted to the terminal.

Copy Terminal-Table Entry (COPYT) Macro Instruction

COPYT moves the information contained in a specified terminal-table entry into a designated work area. (The information copied into the work area remains in the same form as in the terminal table: binary fields remain binary and EBCDIC fields remain EBCDIC.) The terminal-table entry can be either a single-terminal, group-code, distribution list, or process program entry. Formats for each of these entries are shown in Appendix A of the OS QTAM Message Control Program publication.

Name	Operation	Operand
[symbol]	COPYT	termname,workarea

symbol
Is the name of the macro instruction.

termname
Is the name of the terminal whose terminal-table entry is to be copied. If register notation is used, the general register designated must contain the address of a location containing the name of the terminal. The field containing the name must be "n" bytes, where "n" equals or exceeds the longest name of any terminal-table entry. The name must be left-adjusted and must be padded with blanks to the length of the longest TERM entry in the terminal table. If an invalid terminal name is specified, no data movement takes place; the routine linked by the COPYT macro instruction returns an error code of X'20' right-adjusted, in register 15. If no error is detected, register 15 contains zero.

workarea
Is the address of the area into which the information is placed. The first byte of the work area receives the first byte of data from the terminal-table entry. The maximum size of the work area is 252 bytes (the maximum size of a terminal-table entry). If register notation is used, the general register designated must contain the address of the work area.

Change Terminal-Table Entry (CHNGT) Macro Instruction

CHNGT moves the information for a terminal-table entry from a designated work area to

the terminal-table area allocated for that entry. CHNGT causes the entire contents of the superseded terminal-table entry, except for the TSEQUIN and TSEQOUT fields, to be changed. The TSEQUIN and TSEQOUT fields are not changed because of the possibility that a message may be received between the time the entry is copied and the time it is changed. This would cause a sequence number error to occur. In order to change the entire contents, including TSEQUIN and TSEQOUT, the user must precede the CHNGT macro with a STOPLN macro for the line on which the affected terminal is located.

CHNGT is normally preceded by the COPYT macro instruction and instructions to examine and modify the contents of the copied terminal-table entry. The user must be certain that the new terminal-table entry contains all the information required for proper execution of QTAM. The information copied into the terminal table should be formatted appropriately in binary or EBCDIC form. The format of the terminal-table entries and the information contained in each field are contained in Appendix A of the OS QTAM Message Control Program publication.

Name	Operation	Operand
[symbol]	CHNGT	termname,workarea

symbol
Is the name of the macro instruction.

termname
Is the name of the terminal whose terminal-table entry is to be replaced. It must be the same as a name that appears in the name field of a TERM, PROCESS, or DLIST macro instruction. If register notation is used, the address of a location containing the name must be in the general register designated. The field containing the name must be "n" bytes, where "n" equals or exceeds the longest name of any terminal-table entry. The name must be left-adjusted and must be padded with blanks to the length of the longest TERM entry in the terminal table.

If an invalid name is specified, the routine generated by CHNGT returns an error code of X'20', right-adjusted, in register 15. QTAM subsequently disregards the new terminal-table entry and continues to use the old.

workarea
Is the address of the area from which the information is moved. If register notation is used, the general register specified must contain the address of the work area. If the new entry does not equal the size of the old entry, no data movement takes place. An error code of X'10' is returned in register 15, and QTAM continues to use the old entry.

If no errors were detected in the CHNGT macro, register 15 contains all zeros.

Release Messages (RELEASEM) Macro Instruction

RELEASEM sets the 'release pending' bit in the terminal table entry for the specified terminal to one. This causes all intercepted messages with that terminal as the destination to be sent. All suppressed messages are sent as well as any new messages. The 'release pending' bit and the 'intercept' bit in the terminal table entry are set to zero and the 'send' bit is set to one when the first intercepted message is read from the disk for sending to the terminal.

The intercept bit is turned on (that is, set to one) by the INTERCPT macro instruction in the LPS section of the message control program.

If the terminal is free, the messages on the queue are transmitted by priority. If the terminal is busy, the messages will not be transmitted at that time.

Name	Operation	Operand
[symbol]	RELEASEM	termname

symbol
Is the name of the macro instruction.

termname
Is the name of the terminal that can now receive its intercepted messages. It must be the same as a name that appears in the name field of a TERM macro instruction. If register notation is used, the address of a location containing the name must be in the general register designated. The field that contains the name must be "n" bytes, where "n" equals or exceeds the longest name of any terminal-table entry. The name must be left-adjusted and must be padded with blanks to the length of the longest TERM entry in the terminal table. If termname is invalid, an error code of X'20', right-adjusted, is set in register 15. If the line was not intercepted, an

error code of X'04' will be returned in register 15. If RELEASEM is issued during a restart, an error code of X'02', right-adjusted, is set in register 15. RELEASEM should be issued until a normal return code is received. A normal return code indicates that the RELEASEM macro instruction is being executed.

EXAMINING AND MODIFYING POLLING LISTS

QTAM provides macro instructions that enable the user to examine and modify the contents of the polling list for a line.

The COPYP macro instruction causes the contents of a specified polling list to be copied into a work area. This macro instruction can be used in conjunction with the CHNGP macro instruction, which can substitute a new polling list for a superseded one (the new list must be the same size as the old one). The user issues a COPYP, examines the information, changes it if necessary, and issues a CHNGP. CHNGP can also be used to stop or restart polling of the terminals on a line.

Copy Polling List (COPYP) Macro Instruction

COPYP causes the polling list for a specified line to be copied into a user-designated work area. The format of the polling list is shown in Appendix A of the OS QTAM Message Control Program publication.

Name	Operation	Operand
[symbol]	COPYP	termname,rln,workarea

symbol
Is the name of the macro instruction.

termname
Is the name of any terminal in the line group, but not necessarily the name of a terminal in the polling list being copied. If register notation is used, the general register designated must contain the address of the data control block for the line group. It cannot contain the terminal name.

If an invalid terminal name is specified, an error code of X'20', right-adjusted, is set in register 15. If the DCB for the line group has not been opened, an error code of X'01' is set in register 15. In either case, the COPYP has no effect.

rln
Is the relative line number, within the line group, of the line whose polling list is to be copied. If register notation is used, the user previously must have placed the relative line number (in binary form) in the general register designated. If the rln specified is invalid, a code of X'08', right-adjusted, will be set in register 15.

workarea
Is the address of the work area into which the polling list is to be copied. The first byte of the work area receives the first byte of data in the polling list. The size of the area necessary can be determined from the polling list format shown in Appendix A of the OS QTAM Message Control Program publication. If register notation is used, the general register specified must contain the address of the work area.

If no errors were detected in the COPYP macro, register 15 contains all zeros.

Change Polling List (CHNGP) Macro Instruction

CHNGP can either:

1. Place a new polling list in the polling list area for a specified line.
2. Change the status of a polling list for a specified line.

Name	Operation	Operand
[symbol]	CHNGP	termname, rln, {workarea / =C'0' / =C'1'}

symbol
Is the name of the macro instruction.

termname
Is the name of any terminal in the line group, but not necessarily the name of a terminal in the polling list being changed. If register notation is used, the general register designated must contain the address of the data control block for the line group. It cannot contain the terminal name.

If an invalid terminal name is specified, an error code of X'20' right-adjusted, is set in register 15. If the DCB for the line group has not been opened, an error code of X'01' is set in register 15. In either case, the CHNGP has no effect.

rln
Is the relative line number, within the line group, of the line whose polling list is to be modified. If register notation is used, the user previously must have placed the relative line number (in binary form) in the general register specified. If the relative line number is invalid (the line group has no such line number) an error code of X'08', right-adjusted, is set in register 15.

workarea
Is the address of the area that contains the new polling list. The first byte of the polling list area receives the first byte of data in the work area. If the new polling list does not equal the size of the old, no data movement takes place. An error code of X'10' is set in register 15. QTAM subsequently disregards the new polling list and continues to use the old.

=C'0'
Causes the second byte of the polling list to be changed to a zero. This results in the deactivation of the polling list. No further messages are received until the list is reactivated.

=C'1'
Causes the second byte of the polling list to be changed to a one. This results in the activation of the polling list. QTAM begins polling the terminals on the line and accepting incoming messages.

If no errors were detected in the CHNGP macro, register 15 contains all zeros.

EXAMINING QUEUE CONTROL BLOCKS

Each terminal-table entry defined by a TERM or PROCESS macro instruction contains the address of the queue control block (QCB) for the DASD destination or DASD process queue on which outgoing messages are placed. QTAM uses the QCB for:

1. Placing each message on its appropriate DASD queue.
2. Maintaining information on the status of the queue.

The COPYQ macro instruction enables the user to examine a QCB to ascertain the status of the DASD destination or DASD process queue associated with the QCB.

Figure 6 shows the contents and relative displacement of each field in the QCB that is of interest to the user. After issuing a COPYQ macro instruction to copy the QCB into a user-specified work area, the user can determine the contents of the fields from which he needs information. For example, the user can determine the number of messages in the queue, or can use the address of the queue on the disk to retrieve a message (see the RETRIEVE macro instruction description).

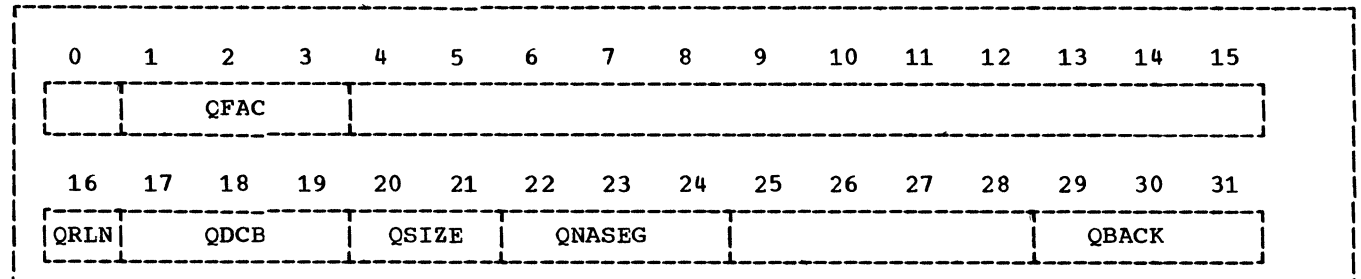
Copy Queue Control Block (COPYQ) Macro Instruction

COPYQ places the contents of a QCB in a specified work area. The user indicates the QCB desired by specifying the name of a terminal or the name of a DASD process queue. If the name of a terminal is specified, COPYQ places the QCB for the DASD destination queue associated with that terminal in the work area. If the name of a DASD process queue is specified, the QCB for the DASD process queue is placed in the work area. In both cases, the entire contents of the 32-byte QCB are provided. However, certain fields are used internally by QTAM routines and are not of concern to the user (see Figure 6).

Name	Operation	Operand
[symbol]	COPYQ	termname,workarea

symbol
Is the name of the macro instruction.

termname
Is the name of the terminal or DASD process queue whose associated QCB is to be copied. Only the name of a single-terminal or process program terminal-table entry can be specified, that is, the name specified in a TERM or PROCESS macro instruction. If specified, no data movement takes place. If register notation is used, the address of a location containing the name must be in the designated general register. The field containing the name must be "n" bytes where "n" equals or exceeds the longest name of any terminal table entry. The name



Legend:

QFAC
Is the relative record address of the next message to be read from this DASD queue; that is, the next message to be processed or transmitted.

QRLN
Is the relative line number of the line associated with this queue. (Field used only for a DASD destination queue.)

QDCB
Is the address of the data control block associated with this QCB. (Field used only for a DASD destination queue.)

QSIZE
Is the number of complete messages on this queue.

QNASEG
Is the relative record address on the DASD message queue data set where the header segment of the next message for this queue will be placed.

QBACK
Is the relative record address of the last message placed on this DASD queue.

Figure 6. Format of Queue Control Block (QCB)

must be left-adjusted and must be padded with blanks to the length of the longest TERM entry in the terminal table. If an invalid termname is specified, a code of X'20', right-adjusted, is set in register 15.

workarea
Is the address of the area in which the contents of the QCB are placed. The area must be 32 bytes long (the size of the QCB). If register notation is used, the general register specified must contain the address of the work area.

If no errors were detected in the COPYQ macro, register 15 contains all zeros.

RETRIEVING MESSAGES

During the operation of a telecommunications system, it may be necessary to retrieve a message that has already been placed on a DASD destination or DASD process queue. The RETRIEVE macro instruction is used to perform this function.

For example, a terminal operator may misplace a message that was previously sent to his terminal. He can send a message (with a message processing program as the destination) requesting that the missing message be sent again. In the request message, he provides the name of his terminal and the sequence number of the message to be retransmitted. The message processing program uses the provided information to RETRIEVE the message from the DASD destination queue that contains messages for that terminal. Subsequently, the message is retransmitted to the terminal via a PUT macro instruction.

Retrieve Message Segment (RETRIEVE) Macro Instruction

RETRIEVE transfers a message segment already placed in a DASD destination queue or DASD process queue to a user-provided work area.

A message segment can be retrieved either by specifying the terminal name to

which the message was sent, or by providing the relative record address of the message segment on the direct-access storage device. In the first method, the input or output sequence number must be specified. The relative record address in the second method is based on zero and reflects the position of the segment in relation to other segments on the storage device. In operation, the second method is faster.

The most common implementation of the RETRIEVE macro instruction uses both of these methods. A terminal name and input or output sequence number are specified for retrieval of the first message segment. This segment contains the header prefix and header portion of the message. RETRIEVE uses the input or output sequence number to find the desired message. When found, the segment containing the header prefix and message header is placed in the work area specified by the user. The header prefix (the first 24 bytes of the segment containing the header[1]) contains, in the MSLINK field, the relative record address of the next segment of the message. The user can load the contents of this field into a register and obtain the next segment by issuing a RETRIEVE macro that specifies the relative record address. Subsequent text segments can be retrieved by means of the relative record address found in the MSLINK field of the text prefix (first 14 bytes of the segment containing the text[1]) in each previous segment. The last segment of a message can be detected by examining the seventh bit (X'02') of the status field (MSTATUS) in the buffer prefix. This bit is set on for the last segment of each message. The formats of the header and text prefixes are shown in Appendix A of the OS QTAM Message Control Program publication. Figure 7 gives an example of the use of RETRIEVE.

Name	Operation	Operand
[symbol]	RETRIEVE	{termname / (dasdaddr)} ,workarea, [{IN / OUT} ,number,] TYPE={S / D}

symbol
Is the name of the macro instruction.

---

[1]When the segment was placed on the direct access queue, the first eight bytes of the 32-byte header prefix (or of the 22-byte text prefix) were deleted.

termname
Is the name of the destination point of the message segment to be retrieved. This is the name specified in the TERM or PROCESS macro instruction and included in the entry in the terminal table. The type of entry referred to can be either a single-terminal, group-code, or process program entry. (The name of a distribution list cannot be specified nor can the name of a process program entry that is defined with the EXPEDITE operand.) If this operand is specified, the first segment of the message is retrieved. If register notation is used, the address of a location containing the name must be in the general register designated. The field that contains the name must be "n" bytes where "n" equals or exceeds the longest name of any terminal table entry. The name must be left-adjusted and must be padded with blanks to the length of the longest TERM entry in the terminal table. If an invalid termname is specified, a return code of X'20' is placed in register 15.

(dasdaddr)
Is the parenthesized name or number of a register containing the relative record address of the message segment to be retrieved. The user must have previously placed the 3-byte record address in this designated register. If an invalid termname is specified, an error code of X'20', right-adjusted, is returned in register 15.

workarea
Is the address of the user-provided work area in which the message segment is to be placed. If register notation is used, the general register specified must contain the address of the work area. The work area must be at least the size of the record defined in the disk initialization process. If an invalid relative record number has been specified, a return code of X'02', right-adjusted, is placed in register 15.

IN
Indicates that the "number" operand that follows specifies the input sequence number of the message to be retrieved. If the "termname" operand is included, either the IN or OUT operand and the "number" operand must be specified. If the "(dasdaddr)" operand is included, IN (as well as OUT and "number") must be omitted.

OUT
Indicates that the "number" operand that follows specifies the output

sequence number of the message to be retrieved. If the "termname" operand is included, either the OUT or IN operand and the "number" operand must be specified. If the "(dasdaddr)" operand is included, OUT (as well as IN and "number") must be omitted. If the sequence number is invalid, an error code of X'40', right-adjusted, is set in register 15.

number
Is the input or output sequence number of the message to be retrieved. If the "termname" operand is included, the "number" operand must be specified. If the "(dasdaddr)" operand is included, the "number" operand must be omitted. If the sequence number is invalid, an error code of X'40', right-adjusted, is set in register 15. If register notation is used, the user must have placed the number (in binary form) in the general register specified. Messages sent from the telecommunications control or alternate terminal may not be retrieved by sequence number.

TYPE=S
Specifies that the message segment is to be retrieved using the terminal name-sequence number method.

TYPE=D
Specifies that the message segment is to be retrieved using the relative record address method.

Name	Operation	Operand	Comments
RETRMSG	RETRIEVE	BOS,CNSLPRT,IN,121,TYPE=S	Retrieves the first segment of a message with input sequence number 121, destined for the Boston terminal, and places the segment in the CNSLPRT area. The segment contains the QTAM shortened header prefix and the message header.
	LA	5,CNSLPRT	Loads the address of the user's work area into general register 5.
	USING	IECKPREF+8,5	Establishes addressability with respect to the DSECT for the QTAM header prefix.
	.		Processing of the header segment according to the current application.
	.		
	.		
TEST	TM	MSTATUS,2	Tests the status field of the buffer prefix to determine if this is the last segment of the message.
	BO	OUT	If yes, branches out of the loop.
	MVC	LINKINFO+1(3),MSLINK	Moves the MSLINK field of the QTAM header prefix (containing the address of the next segment of the message) into an area from which it can be loaded into a general register.
	L	4,LINKINFO	Loads the contents of MSLINK into general register 4.
	RETRIEVE	(4),CNSLPRT,TYPE=D	Retrieves the next segment of the message and places it in the work area.
	.		Processing of the text segment according to the current application.
	.		
	.		
	B	TEST	Loop back to test for last segment.
	.		
OUT	.		
	.		
	.		
This example assumes that each segment is processed after retrieval so that the work area may be reused in retrieving the next segment.			

Figure 7. Example of the Use of the RETRIEVE Macro Instruction

QTAM provides the facility for writing checkpoint records either at specified intervals of time or at certain points in one or more processing programs (see the OS QTAM Message Control Program publication).

The checkpoint records contain the information necessary to record the status of the queues and the telecommunications network. In particular, the checkpoint record includes the polling lists, the terminal table, disk pointers and status information associated with each queue, and disk pointers and status information associated with each line. Note that the data in the buffers is not included in the checkpoint record. Two such checkpoint records are maintained in the checkpoint data set along with a pointer to the most recent record. The format of the checkpoint records is shown in Appendix A.

The user may specify that checkpoint records are to be taken at desired points in the processing programs by:

1. Allocating space on the DASD for the checkpoint data set.
2. Defining the data set.
3. Opening and closing the data set.
4. Using the CKPART operand in the TERMTBL macro instruction in the message control program and issuing a CKREQ macro instruction in each processing program that is to determine when to take the checkpoint records.

The control program is checkpointed when all the required partitions have issued the CKREQ macro instruction. For instance, if CKPART=2 is specified in the TERMTBL macro instruction, and Processing Program 1 issues a CKREQ macro instruction, it will go into a wait state until some other partition also issues a CKREQ macro instruction. At that point the checkpoint will be taken. This puts both of these processing programs in synchronization with the message control program. The processing programs know the exact circumstances at the checkpoint and can take steps to guard against duplicate messages following a restart.

Note that the CKREQ macro instruction cannot be issued in the message control program. The CPINTV and CKPART operands may not both be specified; if both are specified, CPINTV will take precedence and CKPART will be ignored. If the CPINTV keyword is used and the processing program issues a CKREQ macro instruction, the checkpoint records will be taken at the intervals of time specified by the CPINTV operand.

Name	Operation	Operand
(name)	CKREQ	

name
Is the name of the first instruction generated by CKREQ.

When the correct number of CKREQ macro instructions have been issued, checkpoint records are written on a checkpoint data set maintained on a direct access storage device (DASD). A new checkpoint record is written over the oldest existing record and the pointer is updated to reflect the most recent record. Should a system failure occur during checkpoint itself, restart may still be accomplished using the alternate checkpoint record.

Restart allows the user to reestablish the queues and the telecommunications network to its status just prior to the last checkpoint.

For information on allocating space for the checkpoint data set, defining, opening, and closing the checkpoint data set, and restarting the message control program, see the OS QTAM Message Control Program publication, GC30-2005.

Note: If checkpointing is specified at time intervals (CPINTV operand in TERMTBL macro), no additional instructions are necessary in the processing programs; checkpointing is thus made independent of the processing programs.

DEACTIVATING THE TELECOMMUNICATIONS SYSTEM

In order to terminate operation of the telecommunications system, the communication line group, checkpoint, and direct access message queues data sets must be closed. Before they may be closed, all message traffic in the system must cease. To accomplish this, the user issues a CLOSEMC macro instruction in a user-written termination routine. CLOSEMC controls and monitors line activity and checks the status of all data sets opened in the message processing programs. When all data sets opened in the message processing programs are closed, and line activity has ceased, the routine returns control to the user to permit him to close the line group and message queues data sets. (See Appendix D for a sample program.) Deactivation of the system proceeds in the following manner.

When the system is to be deactivated, a CLOSEMC macro instruction must be issued in a program other than the message control program. A recommended procedure is to send a special message to a process queue from which a message processing program, containing a user-written termination routine, may obtain the message.

This termination routine should do the following:

1. Ensure that all other message processing programs, and all their QTAM data sets, are closed.

2. Issue the CLOSEMC macro instruction (only one CLOSEMC is required to deactivate the entire system).

3. Close the MS destination and MS process queues data sets and any other data sets opened in that message processing program. If the processing program does not require a main storage queue data set, a dummy one must be supplied and opened. When this data set is closed, the message processing program requests the message control program to close down.

4. Issue a RETURN macro instruction in order to end the message processing job.

When the QTAM termination routine that is called by the CLOSEMC macro is entered, the following action occurs. Outgoing message traffic continues on any lines that are not currently receiving messages. Meanwhile, incoming message traffic on each line is limited to the message currently being received over that line. When the last block of the current message is received, no more incoming messages are accepted (that is, the line is not repolled or reenabled). As each such line becomes free, any outgoing messages that have been queued for that line are sent. In this manner, incoming message traffic declines to nothing, while outgoing message traffic continues until all messages have been sent.

Note: Since CLOSEMC issues STOPLN and STARTLN macro instructions, the STARTLN/STOPLN module (IECKLNCH) cannot be in a different overlay than the CLOSEMC module (IECKCLOS).

The QTAM termination routine monitors the closing of the QTAM data sets opened in the message processing programs. When it finds that all of these data sets have been closed, and all outgoing message traffic has ended, the routine issues a STOPLN macro instruction for each line in the system. When all lines have been stopped, control returns to the first instruction following the ENDREADY macro instruction in the message control program. This instruction must begin a user-written routine (or branch to a routine) that deactivates the message control program. This deactivation routine must issue CLOSE macro instructions for each of the data sets opened in the message control program (that is, the line group, checkpoint, and direct access message queues data sets).

The last QTAM data set to be closed must be the direct access message queues data set. This is important, because closing this data set constitutes deactivation of the telecommunications system. After the message queues data set has been closed, no further references can be made to queues, control blocks, terminal table, polling lists, etc.

The deactivation routine should end with a RETURN macro instruction to end the message control job. Each of the message processing programs should also end with a RETURN.

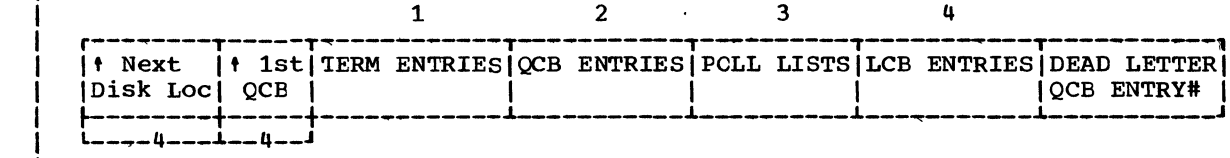
CLOSEMC Macro Instruction

Name	Operation	Operand
[symbol]	CLOSEMC	

symbol
Is the name of the macro instruction.

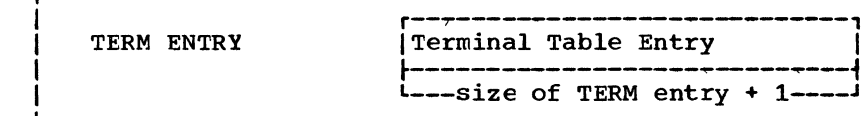
Note: The user is cautioned against using the CLOSEMC macro instruction in the message control program. A communications line on a transmission control unit may not be in CE mode when the CLOSEMC macro instruction is issued. If the TCU is in CE mode, the close procedure will never complete, and the QTAM message control program will remain in a wait state. A line may not be in conversational mode (see the MODE macro instruction in the MCP) when the CLOSEMC macro instruction is issued.

Record Format:

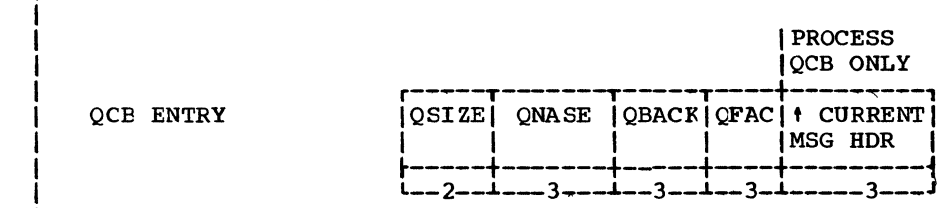


Formats of Fields:

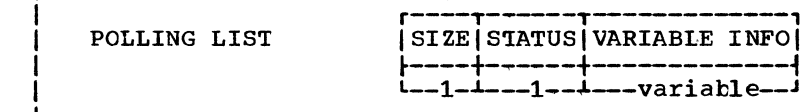
1. Save all terminal entries (except distribution lists).



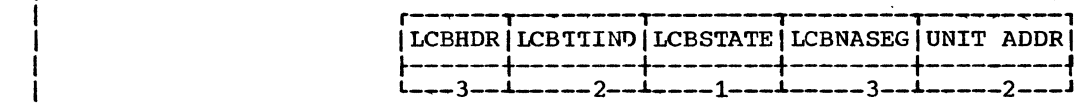
2. Save QCBs only if current QCB is not the same as the last QCB saved.



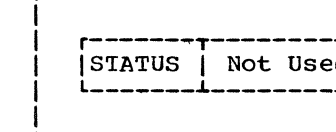
3. Save polling list only if the list is not the same as the last polling list saved.



4. Save LCB information based on QRLN of current QCB being checkpointed.



Control Record:



Status

00 Normal Close
01 Abnormal Termination: (checkpoint record 1 is good)
02 Abnormal Termination: (checkpoint record 2 is good)

APPENDIX B: FORMAT AND SUMMARY OF MACRO INSTRUCTIONS

A format illustration accompanies each macro instruction description in this publication. The illustrations indicate which operands must be coded exactly as shown, which are required, which are variable, etc. The conventions stated to describe the operands are as follows:

1. Keyword operands are described either as the single word that must be coded as shown or by a three-part structure that consists of the keyword operand, followed by an equal sign (both of which must be coded), followed by a value mnemonic or a coded value.

   Examples:

   a. ALL
   b. TYPE=PQ

2. Positional operands are described by a lowercase name that is merely a convenient reference to the operand and is never coded by the programmer. The programmer replaces the positional operand by an allowable expression. Expressions allowed are indicated at the left of the foldout page. The chart shows what expressions are allowable for each operand.

3. Uppercase letters and punctuation marks (except as described in these conventions) represent information that must be coded exactly as shown.

4. Lowercase letters and terms represent information that must be supplied by the programmer. More specifically, n indicates a decimal number, nn a decimal number with at most two digits, nnn with at most three digits, etc.

5. An ellipsis (a comma followed by three periods) indicates that a variable number of items may be included.

6. { } Options contained within braces represent alternatives, one of which must be chosen.

7. [ ] Information contained within brackets represents an option that can be included or omitted, depending on the requirements of the program. If more than one option is contained within brackets, any one or none of the options may be chosen.

8. [A / B / C] Underlined elements represent an assumed value in the event a parameter is omitted.

MACRO INSTRUCTION	OPERANDS	WRITTEN AS									
				Register							
		Sym	Dec Dig	(2-12)	(1)	RX Type	Rel Exp	Abs Exp	Char	Hex Char	W/S
CHNGP	termname			X					X		
	rln		X	X							
	workarea			X		X					
	=C'0'										X
	=C'1'										X
CHNGT	termname			X					X		
	workarea			X		X					
CKREQ											
CLOSE	dcb			X			X				
	MF=										X
CLOSEMC											
COPYP	termname			X					X		
	rln		X	X							
	workarea			X		X					
COPYQ	termname			X					X		
	workarea			X		X					
COPYT	termname			X					X		
	workarea			X		X					
GET	dcb			X			X				
	workarea			X		X					
	(0)										X
OPEN	dcb			X			X				
	INPUT										X
	OUTPUT										X
	MF=										X
PUT	dcb			X			X				
	workarea			X		X					
	(0)										X
RELEASEM	termname			X					X		
RETRIEVE	termname			X					X		
	(dasdaddr)			X			X				
	workarea			X		X					
	IN										X
	OUT										X
	number		X	X							
	type = S										X
	type = D										X
STARTLN	termname			X					X		
	rln		X	X							
	ALL										X
STOPLN	termname			X					X		
	rln		X	X							
	ALL										X

Abbreviations used in Chart

Abbreviations	Meaning
SYM	Any symbol valid in the Assembler Language.
DEC DIG	Any decimal digits, up to the value indicated in the associated macro instruction description.
REGISTER	A general register, always coded within parentheses, as follows:
(2-12)	one of general registers 2 through 12, previously loaded with the right-adjusted value or address indicated in the macro instruction description. The unused high-order bits must be set to zero. The register may be designated symbolically or with an absolute expression.
(1)	general register 1, previously loaded as indicated above. The register can be designated only as (1).
(0)	general register 0, previously loaded as indicated above. The register can be designated only as (0).
RX type	Any address that is valid in the RX form of instruction (for example, LA) may be designated.
REL EXP	A relocatable expression (acceptable as an A-type or V-type address constant by the assembler).
ABS EXP	Any absolute expression as defined by the assembler: self-defining terms (decimal, hexadecimal, binary, character), length attributes, absolute symbols, paired relocatable terms in the same CSECT, and arithmetic combinations of absolute terms.
CHAR	A character string (the framing characters, C' ', are not coded unless specifically indicated in the individual macro instruction description).
HEX CHAR	Concatenated hexadecimal digits (the framing characters, X' ', are not coded unless specifically indicated in the macro instruction description).
W/S	Written as shown.

Upon return to the message processing routine that issued the macro instruction, the following return codes are set in the low-order byte, right-adjusted, in register 15. All numbers in Figure 8 appear in hexadecimal notation.

Macro	Normal Return	Unopened DCB	Invalid Disk Address	Line not Intercepted	Invalid Relative Line Number	Invalid Count	Invalid Terminal Table Entry or DCB Name	Invalid Sequence Number
CHNGP	X'00'	X'01'			X'08'	X'10'	X'20'	
CHNGT	X'00'					X'10'	X'20'	
COPYP	X'00'	X'01'			X'08'		X'20'	
COPYQ	X'00'						X'20'	
COPYT	X'00'						X'20'	
RELEASEM	X'00'			X'04'			X'20'	
RETRIEVE By disk address	X'00'		X'02'					
RETRIEVE By terminal table entry	X'00'						X'20'	X'40'
STARTLN	X'00'	X'01'			X'08'		X'20'	
STOPLN	X'00'	X'01'			X'08'		X'20'	

Figure 8. Return Codes for Macro Instructions Used to Modify and Examine System Status

APPENDIX D: QTAM SAMPLE PROGRAM

```
*************************************************************************
*                                                                       *
* PROGRAM: OS QTAM CLOSE ROUTINE MPP                                    *
*                                                                       *
* OBJECTIVE: SYSTEM CLOSEDOWN                                           *
*                                                                       *
* PROCEDURE: THE GET MACRO INSTRUCTION CAUSES QTAM TO PASS MESSAGE      *
*            DATA (DESTINED FOR THE CLOSE ROUTINE) FROM THE MS PROCESS*
*            QUEUE TO THE USER-SPECIFIED WORK AREA (WORKAREA). IF NO   *
*            MESSAGE DATA IS AVAILABLE FOR THE CLOSE ROUTINE, THE GET  *
*            MACRO INSTRUCTION IS UNSUCCESSFUL AND CONTROL RETURNS TO  *
*            THE SUPERVISOR                                            *
*                                                                      *
*            IF THE GET MACRO INSTRUCTION IS SUCCESSFUL THE CLOSEMC    *
*            MACRO INSTRUCTION IS EXECUTED. CLOSEMC STOPS ALL          *
*            INCOMING MESSAGE TRAFFIC AS SOON AS ALL CURRENTLY         *
*            INCOMING MESSAGES ARE RECEIVED. THEN CLOSEMC CAUSES ALL   *
*            OUTGOING MESSAGES TO BE SENT.                             *
*                                                                      *
*            THE CLOSE MACRO INSTRUCTION CLOSES ALL DATA SETS OPENED   *
*            IN THE CLOSE ROUTINE.                                     *
*                                                                      *
*            THE RETURN MACRO INSTRUCTION RETURNS CONTROL TO THE FIRST *
*            INSTRUCTION AFTER ENDREADY IN THE MCP.                    *
*                                                                      *
*************************************************************************
```

```
CLOSRTN  CSECT
         SAVE  (14,12)
         BALR  12,0
         USING *,12
         ST    13,SAVEAREA+4
         LA    13,SAVEAREA
         B     OPENN
WORKAREA DC    CL20' '
SAVEAREA DC    18F'0'
SOURCE   DC    CL3' '
INDCB    DCB   DSORG=MQ,     *
               MACRF=G,      *
               DDNAME=EOJ,   *
               SOWA=20,      *
               RECFM=R,      *
               TRMAD=SOURCE
OPENN    OPEN  (INDCB,(INPUT))
         GET   INDCB,WORKAREA
FINISH   CLOSEMC
         CLOSE (INDCB)
         RETURN
         END   CLOSRTN
```

Remarks

Location provided for the name of the sending terminal.

Data set organization is a TP message process queue.

A GET is used to gain access to the message process queue.

Name of DD card and process macro instruction in term table.

Size of the user-provided work area.

Name of the work unit is a record.

Location provided for the name of the sending terminal.

Opens the MPP data set.

GET is performed when a message is sent to EOT.

Stops the message traffic in the MCP.

Closes the MPP data set.

Return to the MCP.

ABEND CODES

0A0 Explanation: The error occurred during execution of a QTAM OPEN macro instruction.

The DCBTRMAD field in the data control block (DCB) had not been filled before the system attempted to open the block. (Register 2 contains the address of the data control block in error.)

System Action: The system terminated the task but did not produce a dump.

Programmer Response: Reassemble the program, specifying the TRMAD parameter in the DCB macro instruction or filling the DCBTRMAD field in the data control block before the OPEN macro instruction is executed. Then execute the job step again.

If the problem recurs, do the following before calling IBM for programming support:

- Execute the stand-alone program, IMDSADMP, with TYPE=HI option to produce a storage dump to tape. If a tape is not available, execute the stand-alone program IMDSADMP TYPE=LO option to produce a storage dump to a printer.
- Execute IMDPRDMP with the 'GO' option after you restart the system. The input to IMDPRDMP is the dump tape from IMDSADMP. Save the formatted dump output.

0A1 Explanation: The error occurred during execution of a QTAM OPEN macro instruction that specified the data control block (DCB) for the main-storage process queue.

The DCBSOWA field in the data control block had not been filled before the system attempted to open the block. (Register 2 contains the address of the data control block in error).

System Action: The system terminated the task but did not produce a dump.

Programmer Response: Reassemble the program, specifying the SOWA parameter in the DCB macro instruction or filling the DCBSOWA field in the data control block before the OPEN macro instruction is executed. Then execute the job step again.

If the problem recurs, do the following before calling IBM for programming support.

- Execute stand-alone program, IMDSADMP, with type=HI option to produce a storage dump to tape. If a tape is not available, execute the stand-alone program IMDSADMP TYPE=LO option to produce a storage dump to a printer.
- Execute IMDPRDMP with the 'GO' option after you restart the system. The input to IMDPRDMP is the dump tape from IMDSADMP. Save the formatted dump output.

0A2 Explanation: The error occurred during execution of a QTAM OPEN macro instruction that specified the data control block (DCB) for the main-storage process queue.

The data definition name specified in the name field of the DD statement or as an entry in the terminal table is not the name of the process entry in the terminal table. (Register 2 contains the address of the data control block in error.)

System Action: The system terminated the task, but did not produce a dump.

Programmer Response: Reassemble the program, specifying the correct data definition name in the DD statement or terminal table. Then execute the job step again.

If the problem recurs, do the following before calling IBM for programming support:

- Execute the stand-alone program, IMDSADMP, with type=HI option to produce a storage dump to tape. If a tape is not available, execute the stand-alone program IMDSADMP TYPE=LO option to produce a storage dump to a printer.

- Execute IMDPRDMP with the "GO" option after you restart the system. The input to IMDPRDMP is the dump tape from IMDSADMP. Save the formatted dump output.

0A5 Explanation: The error occurred during execution of a QTAM OPEN macro instruction issued by a message processing program.

The open routine attempted to open a data control block (DCB) for a main-storage process queue or a main-storage destination queue that had previously been opened. (Register 2 contains the address of the data control block in error.)

System Action: The message processing program was terminated with a dump.

Programmer Response: Make sure that a message processing program does not attempt to open a data control block for a main-storage queue that had previously been opened either by itself or another message processing program. Correct the error and execute the job step again.

If the problem recurs, do the following before calling IBM for programming support:

- Have the associated dump and message processing program listings available.

0A6 Explanation: The error occurred during execution of (1) a QTAM OPEN macro instructuon or (2) a QTAM CLOSE macro instruction.

This error condition occurred for either of 2 reasons:

- In case (1), the data control block (DCB) for the direct access message queues data set was opened after a data control block for another QTAM data set was opened.

- In case (2), the data control block for the direct access message queues data set was closed while a data control block for a main-storage process queue or a main-storage destination queue was still open.

System Action: Either the message control program or the message processing program was terminated with a dump.

Programmer Response: The error, in case (1), can be corrected by one of the following:

- Correcting an invalid DD statement for the direct access storage device.

- Changing the message control program to open the data control block for the direct access message queues data set before opening any other data control block in that program.

- Opening the data control block for the direct access message queues data set before any message processing program opens any QTAM data control block.

The execute the job step again.

The error, in case (2), can be corrected by closing all the data control blocks in the message processing programs before closing the data control block for the direct access message queues data set in the message control program. Then execute the job step again.

If the problem in (1) or (2) recurs, do the following before calling IBM for programming support:

- Have the associated dump and program listings available.

ASSEMBLY ERROR MESSAGES

IHB001 XXX OPERAND REQ'D--NOT SPECIFIED

Explanation: A required positional or keyword operand was omitted. The position or name of the operand is xxx.

System Action: The macro instruction was partially expanded; expansion stopped on detection of the error. Severity code=12.

Programmer Response: Probable User Error. Provide the required operand and reassemble. If the problem recurs, do the following before calling IBM for programming support:

- Have the associated program listing available.

IHB002 INVALID XXX OPERAND SPECIFIED - YYY

Explanation: An operand, whose position or name is xxx, was specified as yyy. The specified operand is invalid.

System Action: The macro instruction was partially expanded; expansion stopped on detection of the error. Severity code=12.

Programmer Response: Probable User Error. Correct the invalid operand and reassemble. If the problem recurs, do the following before calling IBM for programming support:

- Have the associated program listing available.

IHB004 REQUIRED OPERAND(s) NOT SPECIFIED

Explanation: One or more required operands were omitted.

System Action: The macro instruction was partially expanded; expansion stopped on detection of the error. Severity code=12.

Programmer Response: Probable User Error. Provide all required operands and reassemble. If the problem recurs, do the following before calling IBM for programming support:

- Have the associated program listing available.

Where more than one page reference appears, the major reference appears first.

IBM S/360 OS QTAM MPPS Printed in U.S.A. GC30-2003-4

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

READER'S COMMENT FORM

IBM System/360 Operating System Order No. GC30-2003-4
QTAM Message Processing Program Services

- How did you use this publication?

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

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

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

- What is your occupation?

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

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

YOUR COMMENTS, PLEASE . . .

Your answers to the questions on the back of this form, together with your comments, help us produce better publications for your use. Each reply is carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

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

Fold Fold

FIRST CLASS
PERMIT NO. 569
RESEARCH TRIANGLE PARK
NORTH CAROLINA

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

POSTAGE WILL BE PAID BY . . .

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

Attention: Publications Center, Dept. E01

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

Cut Along Line

IBM S/360 OS QTAM MPPS Printed in U.S.A. GC30-2003-4