IBM Data Processing Techniques

Operating System/360
QTAM User's Guide -- Message Control Task Specification
Preliminary Edition

The Queued Telecommunications Access Method (QTAM) provides macro instructions for the programming of a communications-based data processing system within the Operating System/360. This book is a compilation of logic flowcharts and explanations designed to instruct the programmer in the coding of a program using QTAM. In a tutorial manner, the reader is led through the necessary decisions and a workbook-like development of the system macro coding on the queued access method level.

Programming

Copies of this and other IBM publications can be obtained through IBM branch offices. Address comments concerning the contents of this publication to IBM, Technical Publications Department, 112 East Post Road, White Plains, N.Y. 10601

CONTENTS

INTRODUCTION i
SECTION A: COMMUNICATION LINE GROUP DCB
DCB. 1
SECTION B: DIRECT ACCESS STORAGE
DEVICE QUEUE DCB 4
DCB. 4
SECTION C: LOGGING DEVICE DCB . . . 6
DCB. 7
SECTION D: TERMINAL TABLE 10
TERMTBL. 10
OPTION 11
TERM. 12
PROCESS 16
LIST 17
TERMTBL. 18
DCB. 18
SECTION E: POLLING LIST 20
POLL 20
SECTION F: BUFFERS. 22
BUFFER. 22
DFNSUBT 24
SECTION G: RECEIVE SEGMENT. 26
LPSTART 27
RCVSEG 27
TRANS 27
LOGSEG. 28
BREAKOFF 29
SECTION H: RECEIVE HEADER 30
RCVHDR. 30
SKIP 31
LOGSEG. 32
MSGTYPE 33
MSGTYPE 34
MODE. 35
DATESTMP 37
SEQIN. 38
ROUTE 39
MODE. 40
MODE. 41
MODE. 42
MODE. 44
TIMESTMP 45
SOURCE. 46
EOA 47
DIRECT 47
LOGSEG. 48
RCVSEG. 49
LOGSEG. 49
SKIP 50
SECTION J: END RECEIVE. 52
ENDRCVE 52
EOBLC 53
EOB 53
ERRMSG. 54
REROUTE 57
REROUTE. 58
CANCELM. 59
POLLIMIT. 60
POSTRCVE 60
SECTION K: SEND HEADER 62
SENDHDR 62
LOGSEG 63
MSGTYPE. 64
MSGTYPE. 65
SKIP 66
SKIP 67
SEQOUT. 67
TIMESTMP 68
DATESTMP 69
MODE 70
LOGSEG. 71
SECTION L: SEND SEGMENT 74
SENDSEG 74
LOGSEG. 74
TRANS 75
PAUSE 75
SECTION M: END SEND 78
ENDSEND 78
EOBLC 79
EOB 79
ERRMSG 80
REROUTE. 83
REROUTE. 84
INTERCPT 85
POSTSEND 86
LPSTART 87
SECTION N: DATA SET INITIALIZATION . 88
OPEN 88
ACTSUBT 92
ENDREADY 93
SECTION P: STRUCTURING 94
APPENDIX A: SAMPLE PROGRAM 96
Application Implementation 96
System Configuration 96
Job Definition. 98
Message Formats. 99
Program Flowchart 100
Message Processing Program 101
Operator Control Message Processing
Program 102
Macro Coding. 103
APPENDIX B: MESSAGE HEADERS
FOR QTAM 111

INTRODUCTION

The Operating System/360 Queued Telecommunications Access Method (QTAM) provides macro instructions for specifying the operation of a communications-based data processing system. This book is a compilation of logic flowcharts and explanations designed to instruct the programmer in the use of these macros. In a tutorial manner, the reader is led through the necessary decisions in a workbook-like development of the system macro coding on the queued access method level.

A sample program using QTAM is included in the Appendix along with some guidelines to designing message formats for efficient use of QTAM.

Upon completing this book, the user will have completely specified and put together all the coding needed to perform the following functions:

Polling and addressing of terminals
Dialing and answering of terminals
Allocation of core storage buffers
Routing of messages
Queuing of messages
Header analysis and synthesis
Message logging
Error checking
Error procedures

Functions not provided for in this book are:

Processing of message contents
Formulating replies to inquiry messages
Operator control of the communications system

This book is largely a presentation of information found in the SRL document IBM Operating System/360: Telecommunications (C28-6553). It is intended that the SRL document be a reference for further detail in specific areas.

Prerequisites for using this book are:

- Knowledge of a system configuration and its application-oriented operating procedures.
- Layouts of the message formats that will be sent and received via communication lines.
- An understanding of the principles of Operating System/360 (IBM Operating System/360, Concepts and Facilities, C28-6535).
- A general, but not extensive, knowledge of the System/360 Assembly Language (IBM Operating System/360; Assembler Language, C28-6514).

The QTAM macro coding produced with this book specifies the operation of a Message Control Task within the framework of Operating System/360. A Message Control Task encompasses all the communications-oriented functions listed but does not include user programs to process the data content of messages received from communications lines. The processing of the data content of the messages is performed by Message Processing Tasks that are user-provided and operated as separate tasks within Operating System/360. These Message Processing Tasks are not obtainable through use of this book.

There is, however, the facility within QTAM to incorporate programs as subtasks of the Message Control Task in order to do a moderate amount of data processing. These subtasks will not then operate as separate tasks of the Operating System, and consequently will make it possible to operate in a task-restricted environment.

Instruction is given at the appropriate points in this book for inclusion of such subtasks. The reader is again referred to C28-6553 for further description of the details and restrictions of such operation.

The Message Control Task to be developed here consists of the following parts:

Data set definition
Control information
Line procedure specification

Data Set Definition is concerned with the writing of DCB (data control block) statements. These statements specify the operation of direct access storage devices, logging devices, and communication lines.

Control Information is necessary for operation of communication lines. It consists of terminal device information, polling list descriptions, and buffer assignments.

Line Procedure Specification (LPS) uses standard delimiter and functional macros to provide the necessary logic flow for header analysis/synthesis and for message handling.

This book breaks the above three main parts into the following sections:

A. Communication Line Group DCB
B. Direct Access Storage Device Queue DCB
C. Logging Device DCB
D. Terminal Table
E. Polling List
F. Buffers
G. Receive Segment
H. Receive Header
J. End Receive
K. Send Header
L. Send Segment
M. End Send
N. Data Set Initialization
P. Structuring

Using this book to develop a Message Control Task, the reader will proceed through each of the above sections as directed. As part of the job of progressing through a section, macro statements for that section are to be filled out. These macro statements will then be collected and ordered to form the three parts: Data Set Definition, Control Information, and LPS(s). These, in turn, are gathered to form the Message Control Task.

The first part to be considered will be Data Set Definition. Here we will be concerned with the writing of DCB macro statements. This is in harmony with the control procedures for Operating System/360.

SECTION A. COMMUNICATION LINE GROUP DCB

BEGIN

Write "Section A. Communication Line Group DCB" in the margin at the top of the first coding sheet. This will identify the macro statements for Section A.

Each communication line group in the system must have a statement defining its characteristics. This is done with a Data Control Block (DCB) statement. (A line group consists of all communication lines in the system that have the same channel programs, the same buffer requirements, the same line procedure specifications, the same send-receive relative priority, the same polling intervals, and the same type terminals.)

Choose symbolic names for each line group. Enter each name in a Name field of macro statements for this section. (Allow about two lines per statement.) Each of these statements will be a DCB statement — one for each line group.

EXAMPLE DCB

Name	Operation	Operand
DCBGRUP1	DCB	DDNAME=DDGROUP,DSORG=CX,
		MACRF=(G,P),CPOLL=(POLLINE1,
		POLLINE2),INTVL=5,BUFRQ=3,
		ACLOC=bb,CLPS=LPS1

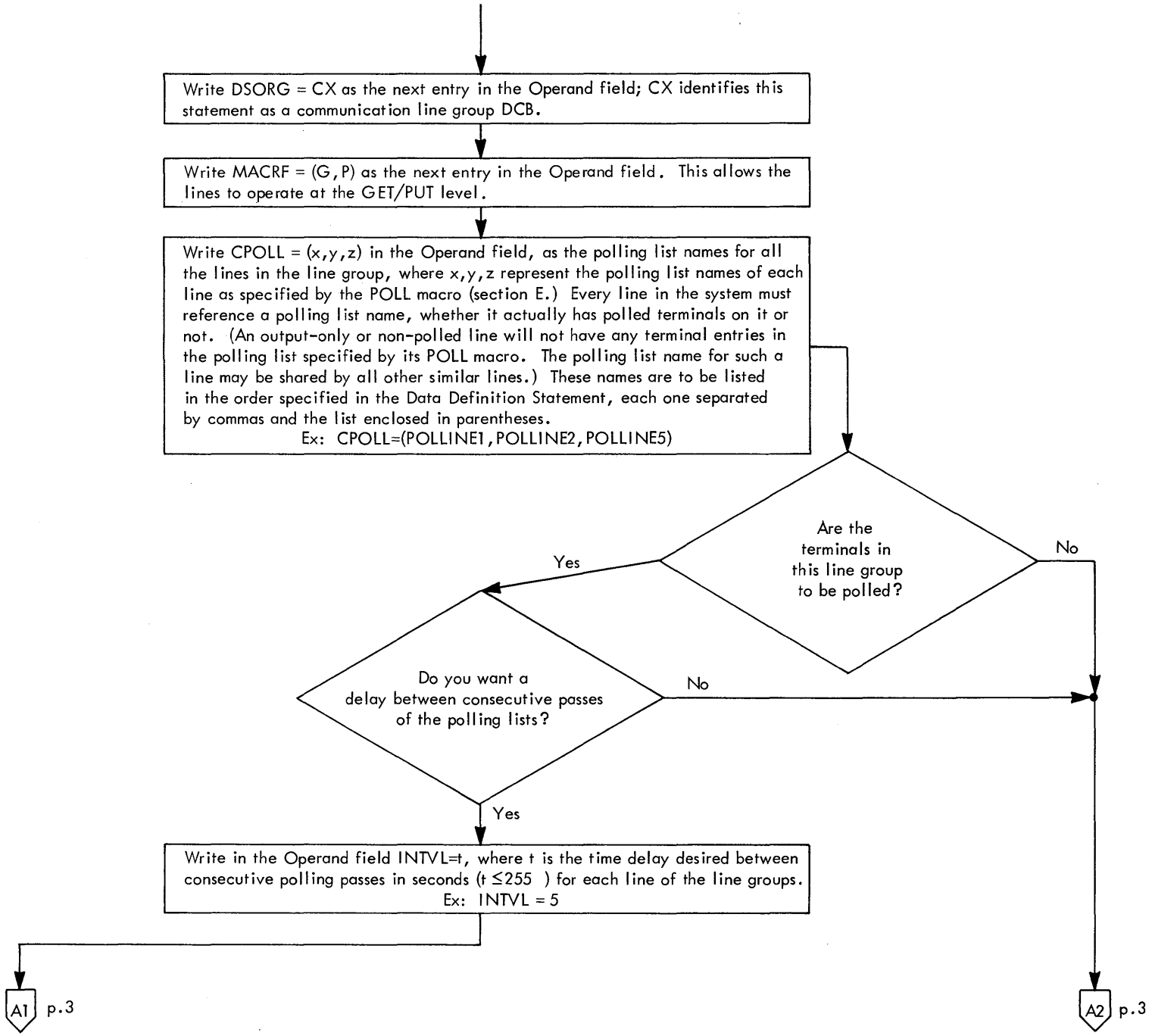
From here on, the discussion will be per DCB. For more than one DCB a similar procedure should be followed for each.

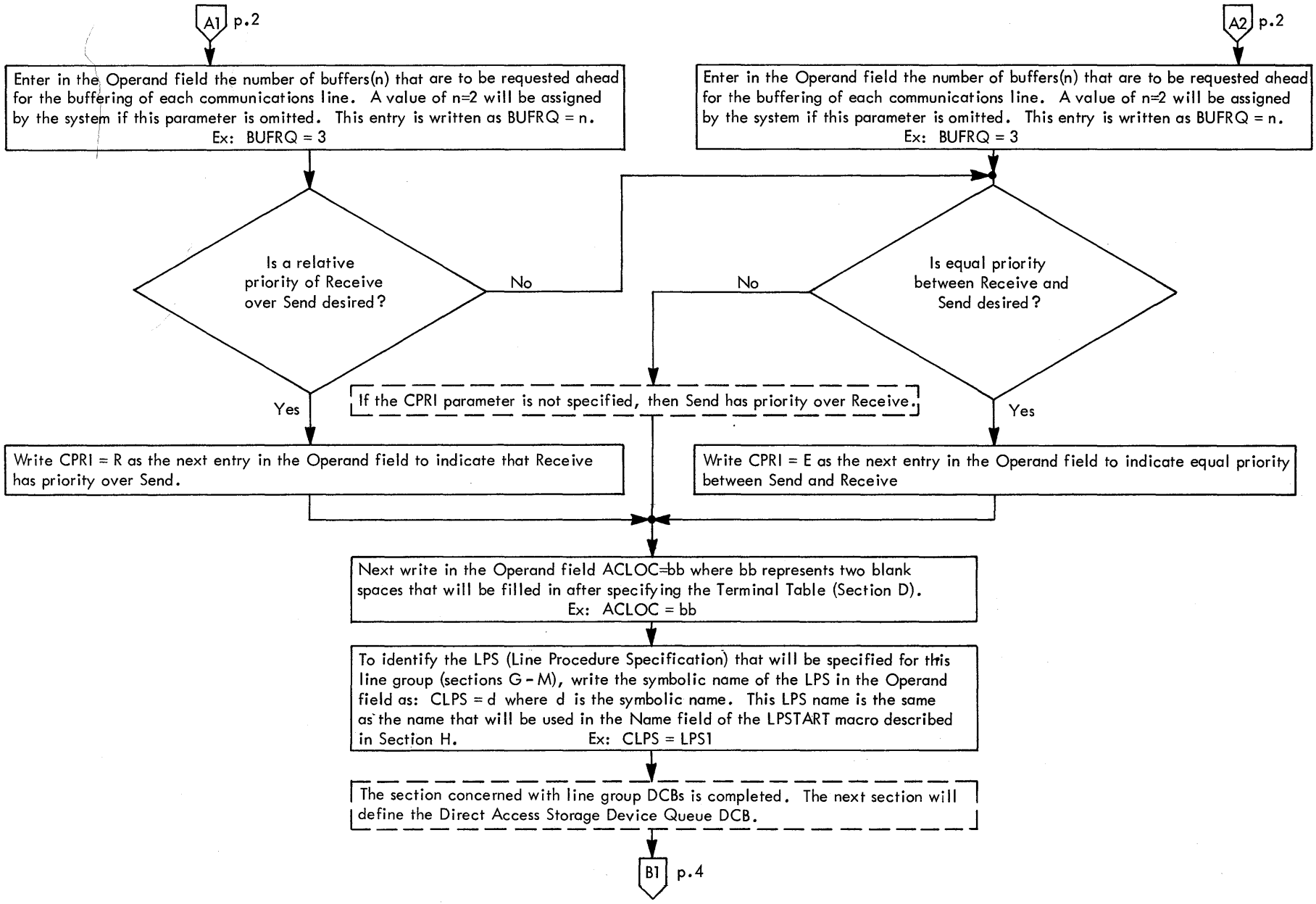
Write DCB in the Operation field of the macro statement.

Write DDNAME = name in the Operand field, where name is the symbolic name for the line group that will be in the Data Definition Statement (DD card) for this line group. (DD cards are job control cards that will be prepared when the Message Control Task being written is entered into the Operating System for execution. There will be a DD card entered for every Data Set defined by a DCB statement in the Message Control Task.)

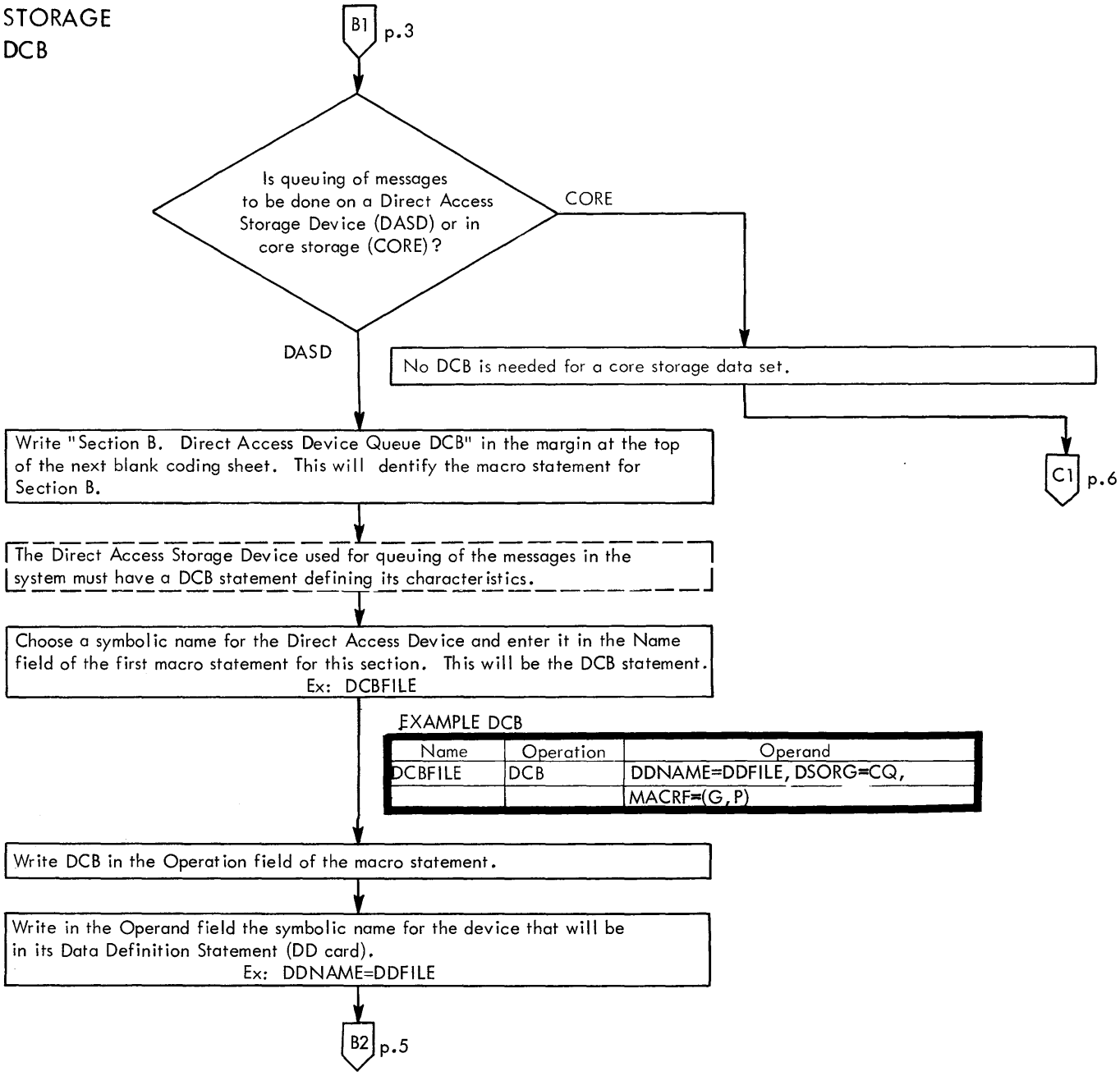
Ex: DDNAME = DD GROUP

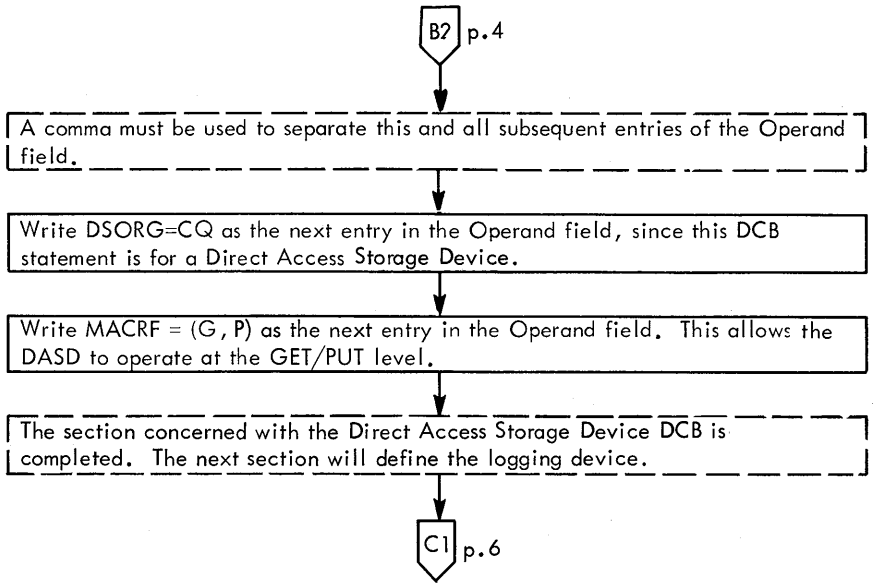
Use a comma to separate this and all future entries of the Operand field.


Write DSORG = CX as the next entry in the Operand field; CX identifies this statement as a communication line group DCB.
Write MACRF = (G,P) as the next entry in the Operand field. This allows the lines to operate at the GET/PUT level.
Write CPOLL = (x,y,z) in the Operand field, as the polling list names for all the lines in the line group, where x,y,z represent the polling list names of each line as specified by the POLL macro (section E.) Every line in the system must reference a polling list name, whether it actually has polled terminals on it or not. (An output-only or non-polled line will not have any terminal entries in the polling list specified by its POLL macro. The polling list name for such a line may be shared by all other similar lines.) These names are to be listed in the order specified in the Data Definition Statement, each one separated by commas and the list enclosed in parentheses.
Ex: CPOLL=(POLLINE1,POLLINE2,POLLINE5)
Are the terminals in this line group to be polled?
Yes
No
Do you want a delay between consecutive passes of the polling lists?
No
Yes
Write in the Operand field INTVL=t, where t is the time delay desired between consecutive polling passes in seconds (t ≤255) for each line of the line groups.
Ex: INTVL = 5
A1 p.3
A2 p.3

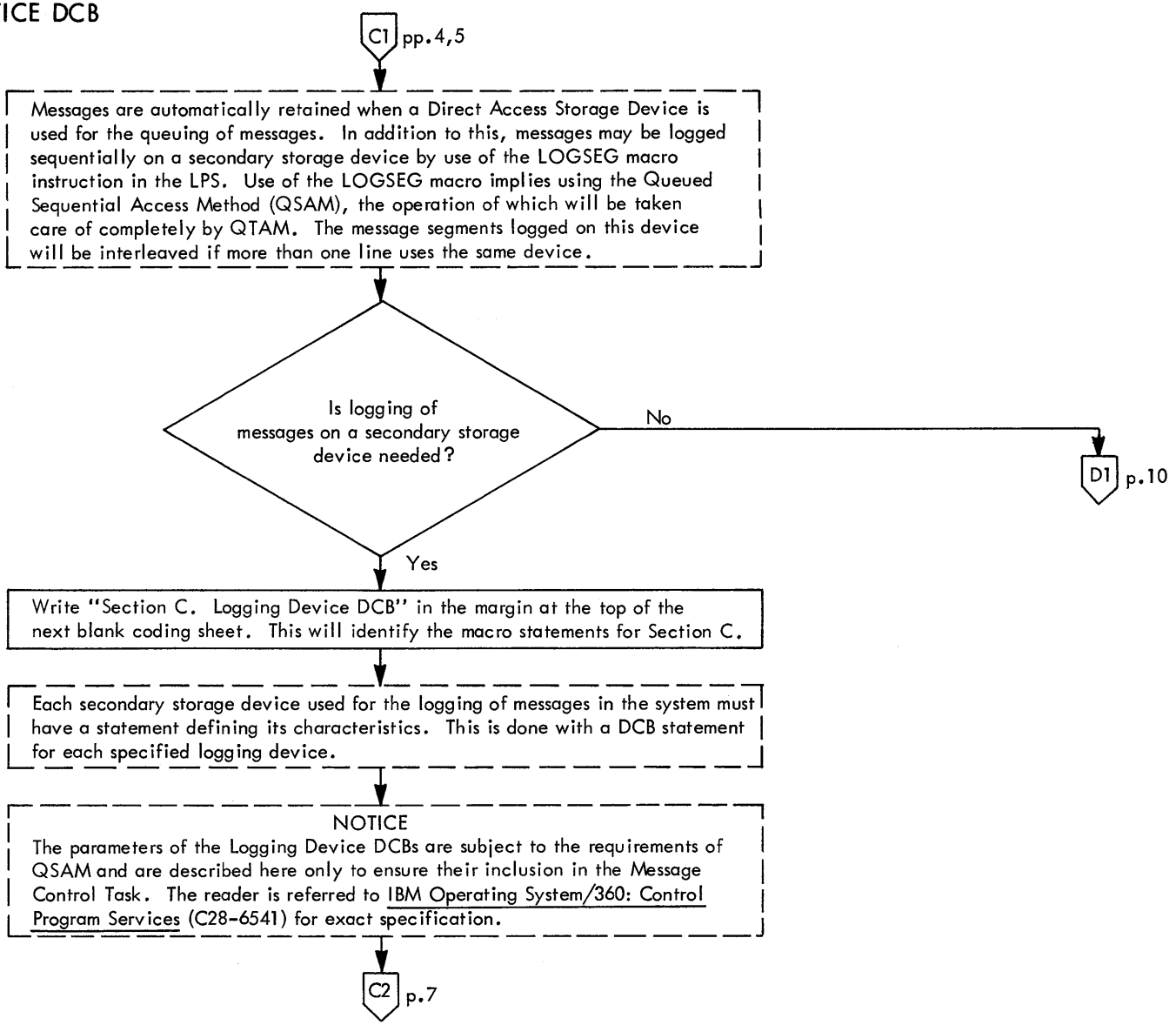

A1 p.2
A2 p.2
Enter in the Operand field the number of buffers(n) that are to be requested ahead for the buffering of each communications line. A value of n=2 will be assigned by the system if this parameter is omitted. This entry is written as BUFRQ = n.
Ex: BUFRQ = 3
Enter in the Operand field the number of buffers(n) that are to be requested ahead for the buffering of each communications line. A value of n=2 will be assigned by the system if this parameter is omitted. This entry is written as BUFRQ = n.
Ex: BUFRQ = 3
Is a relative priority of Receive over Send desired?
No
Yes
Is equal priority between Receive and Send desired?
No
Yes
If the CPRI parameter is not specified, then Send has priority over Receive.
Write CPRI = R as the next entry in the Operand field to indicate that Receive has priority over Send.
Write CPRI = E as the next entry in the Operand field to indicate equal priority between Send and Receive
Next write in the Operand field ACLOC=bb where bb represents two blank spaces that will be filled in after specifying the Terminal Table (Section D).
Ex: ACLOC = bb
To identify the LPS (Line Procedure Specification) that will be specified for this line group (sections G - M), write the symbolic name of the LPS in the Operand field as: CLPS = d where d is the symbolic name. This LPS name is the same as the name that will be used in the Name field of the LPSTART macro described in Section H. Ex: CLPS = LPS1
The section concerned with line group DCBs is completed. The next section will define the Direct Access Storage Device Queue DCB.
B1 p.4

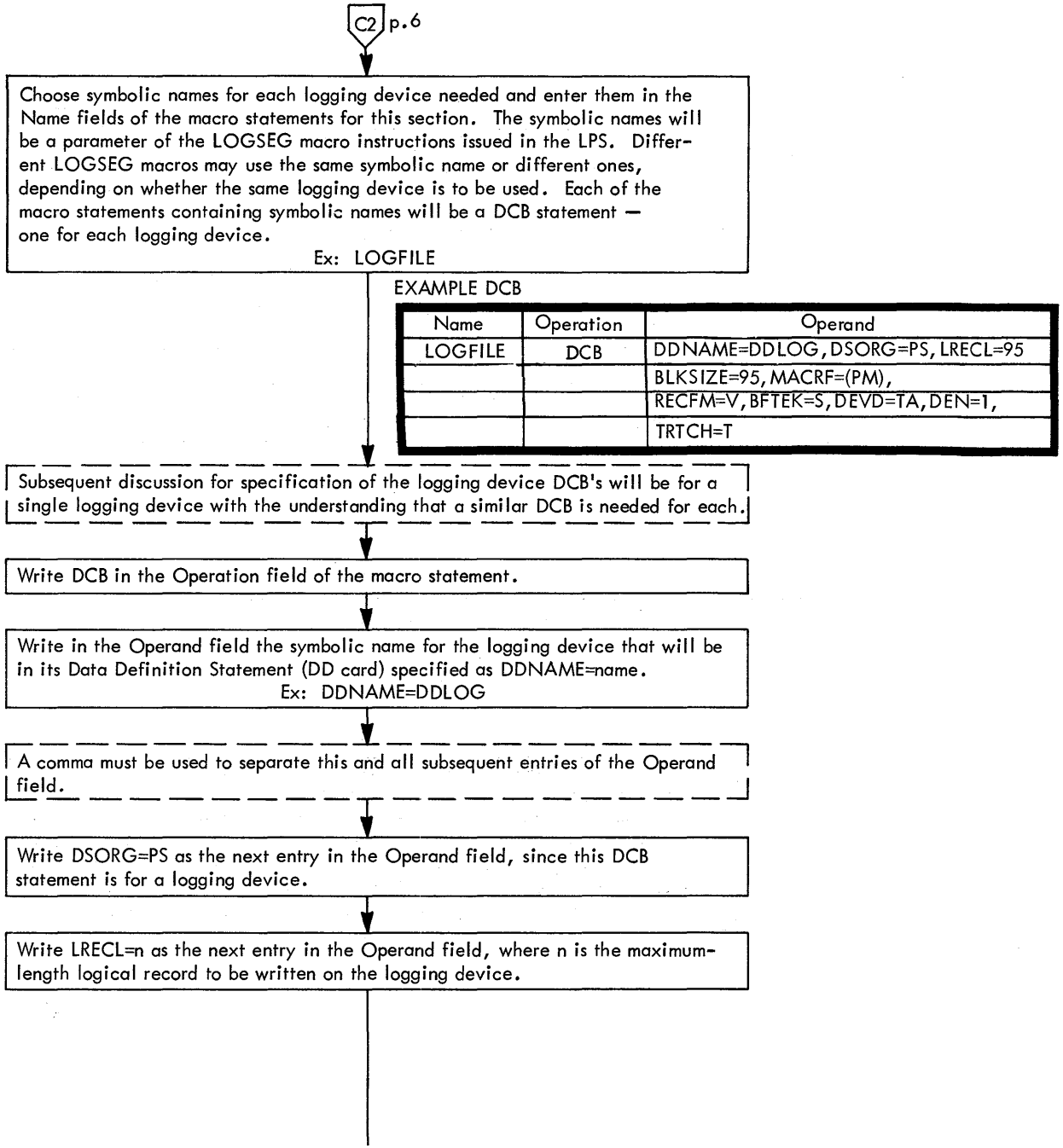
SECTION B. DIRECT ACCESS STORAGE DEVICE QUEUE DCB

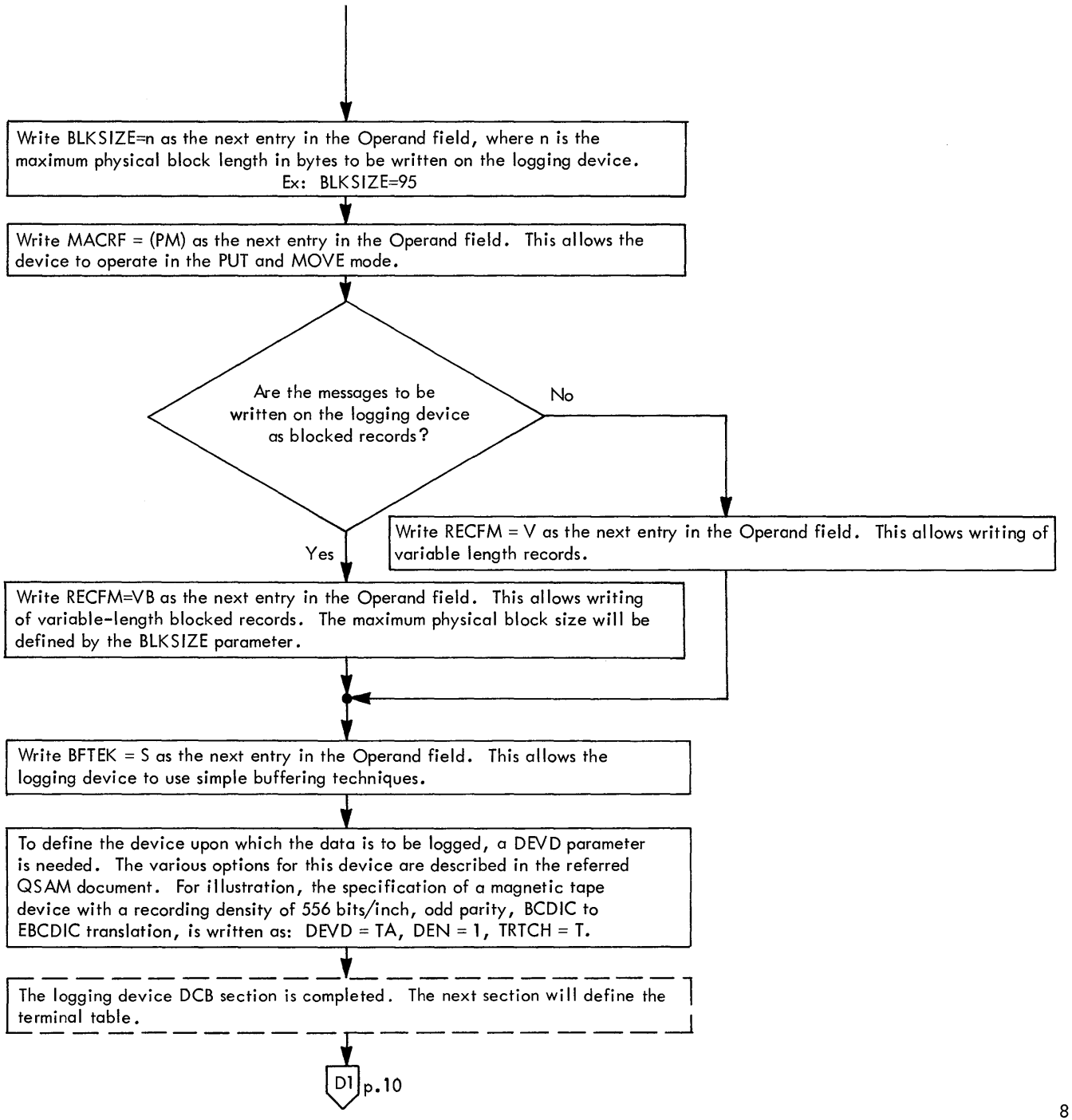



B2 p.4
A comma must be used to separate this and all subsequent entries of the Operand field.
Write DSORG=CQ as the next entry in the Operand field, since this DCB statement is for a Direct Access Storage Device.
Write MACRF = (G, P) as the next entry in the Operand field. This allows the DASD to operate at the GET/PUT level.
The section concerned with the Direct Access Storage Device DCB is completed. The next section will define the logging device.
C1 p.6

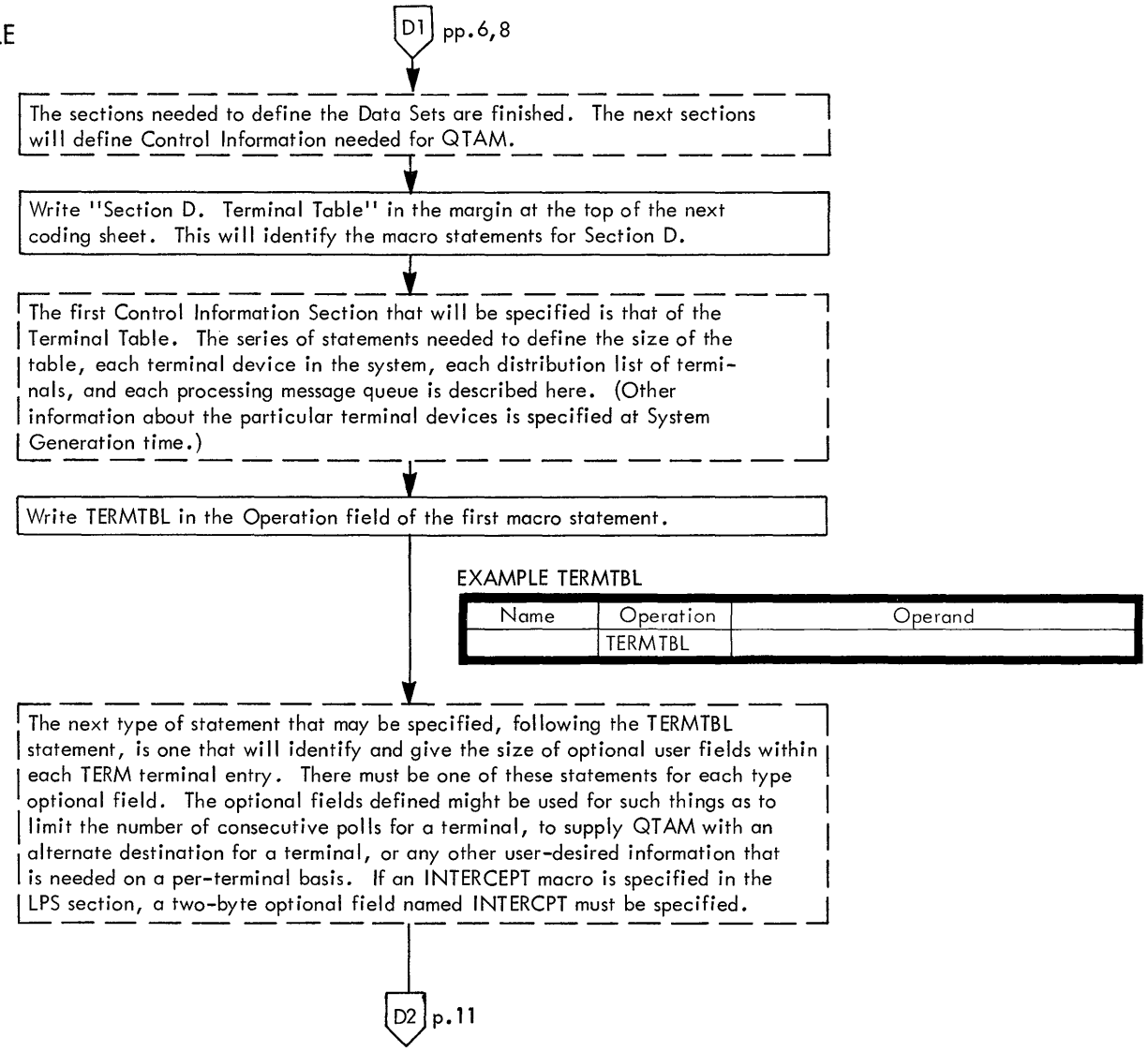
SECTION C. LOGGING DEVICE DCB

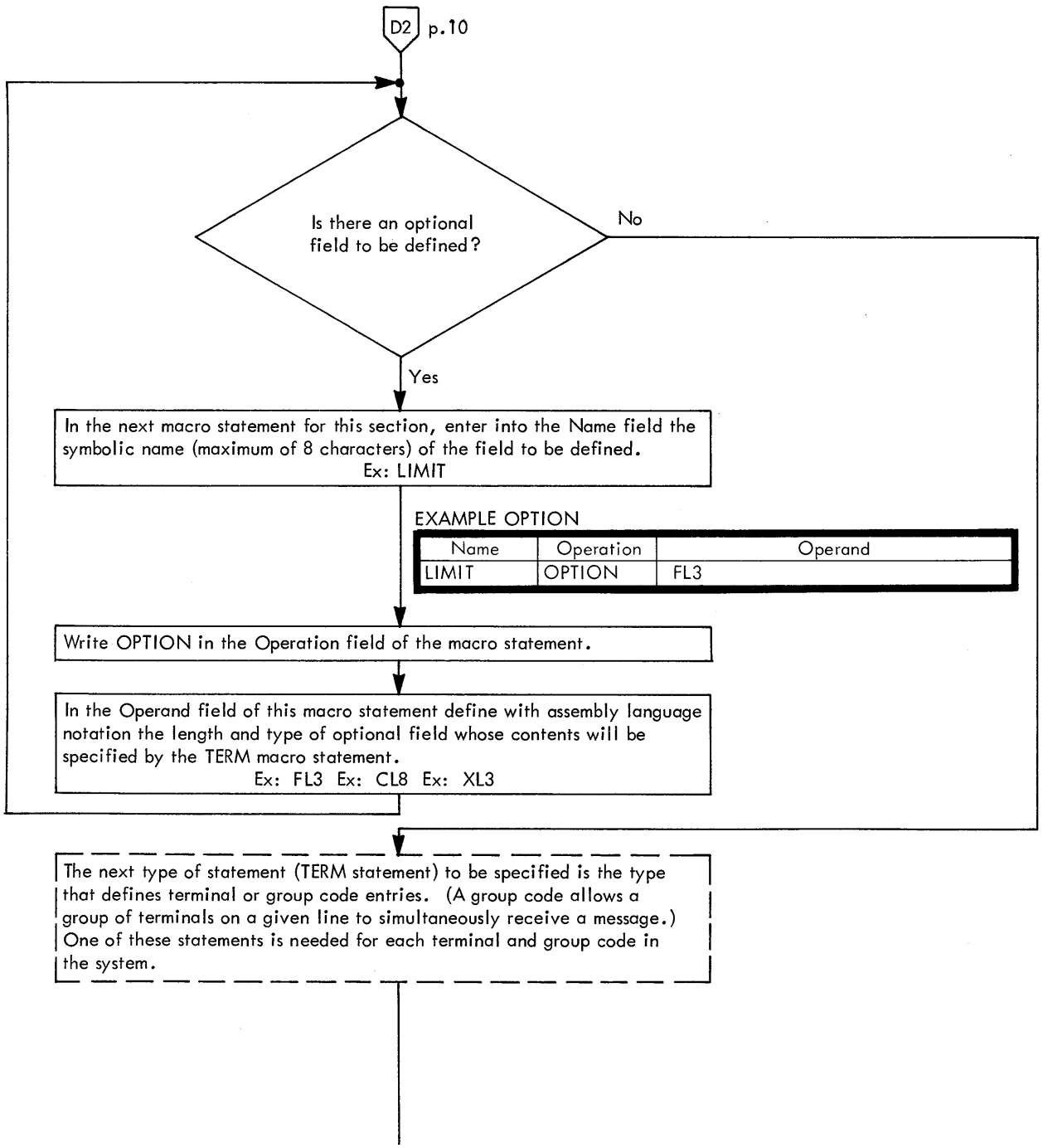



C2 p.6
Choose symbolic names for each logging device needed and enter them in the Name fields of the macro statements for this section. The symbolic names will be a parameter of the LOGSEG macro instructions issued in the LPS. Different LOGSEG macros may use the same symbolic name or different ones, depending on whether the same logging device is to be used. Each of the macro statements containing symbolic names will be a DCB statement — one for each logging device.
Ex: LOGFILE
EXAMPLE DCB
Name | Operation | Operand
LOGFILE | DCB | DDNAME=DDLOG,DSORG=PS,LRECL=95
BLKSIZE=95,MACRF=(PM),
RECFM=V,BFTEK=S,DEVD=TA,DEN=1,
TRTCH=T
Subsequent discussion for specification of the logging device DCB's will be for a single logging device with the understanding that a similar DCB is needed for each.
Write DCB in the Operation field of the macro statement.
Write in the Operand field the symbolic name for the logging device that will be in its Data Definition Statement (DD card) specified as DDNAME=name.
Ex: DDNAME=DDLOG
A comma must be used to separate this and all subsequent entries of the Operand field.
Write DSORG=PS as the next entry in the Operand field, since this DCB statement is for a logging device.
Write LRECL=n as the next entry in the Operand field, where n is the maximum-length logical record to be written on the logging device.


Write BLKSIZE=n as the next entry in the Operand field, where n is the maximum physical block length in bytes to be written on the logging device.
Ex: BLKSIZE=95
Write MACRF = (PM) as the next entry in the Operand field. This allows the device to operate in the PUT and MOVE mode.
Are the messages to be written on the logging device as blocked records?
No
Yes
Write RECFM = V as the next entry in the Operand field. This allows writing of variable length records.
Write RECFM=VB as the next entry in the Operand field. This allows writing of variable-length blocked records. The maximum physical block size will be defined by the BLKSIZE parameter.
Write BFTEK = S as the next entry in the Operand field. This allows the logging device to use simple buffering techniques.
To define the device upon which the data is to be logged, a DEVD parameter is needed. The various options for this device are described in the referred QSAM document. For illustration, the specification of a magnetic tape device with a recording density of 556 bits/inch, odd parity, BCDIC to EBCDIC translation, is written as: DEVD = TA, DEN = 1, TRTCH = T.
The logging device DCB section is completed. The next section will define the terminal table.
D1 p.10

SECTION D. TERMINAL TABLE




D2 p.10
Is there an optional field to be defined?
No
Yes
In the next macro statement for this section, enter into the Name field the symbolic name (maximum of 8 characters) of the field to be defined.
Ex: LIMIT
EXAMPLE OPTION
Name | Operation | Operand
LIMIT | OPTION | FL3
Write OPTION in the Operation field of the macro statement.
In the Operand field of this macro statement define with assembly language notation the length and type of optional field whose contents will be specified by the TERM macro statement.
Ex: FL3 Ex: CL8 Ex: XL3
The next type of statement (TERM statement) to be specified is the type that defines terminal or group code entries. (A group code allows a group of terminals on a given line to simultaneously receive a message.) One of these statements is needed for each terminal and group code in the system.

Although the following discussion is written per statement, a TERM statement should be written concurrently for each terminal (or group code) in the system.

For each TERM statement to be defined, write in the Name field the symbolic name for each terminal or group code to be defined. The symbolic names assigned will be referred to as "Terminal Table entry names" and will be stated in message headers to identify message source and destination(s). Each symbolic name may be of a different length, up to a maximum length of 8 characters, provided that they are delineated by blank character(s) when stated in the message header. (A field equivalent to the maximum name specified will be reserved for each TERM statement entry.) If they are not delineated by blank characters in the message header, each name must be of the same length, and that length must be specified in the functional macro statements that reference them.

Ex: NYC

EXAMPLE TERM

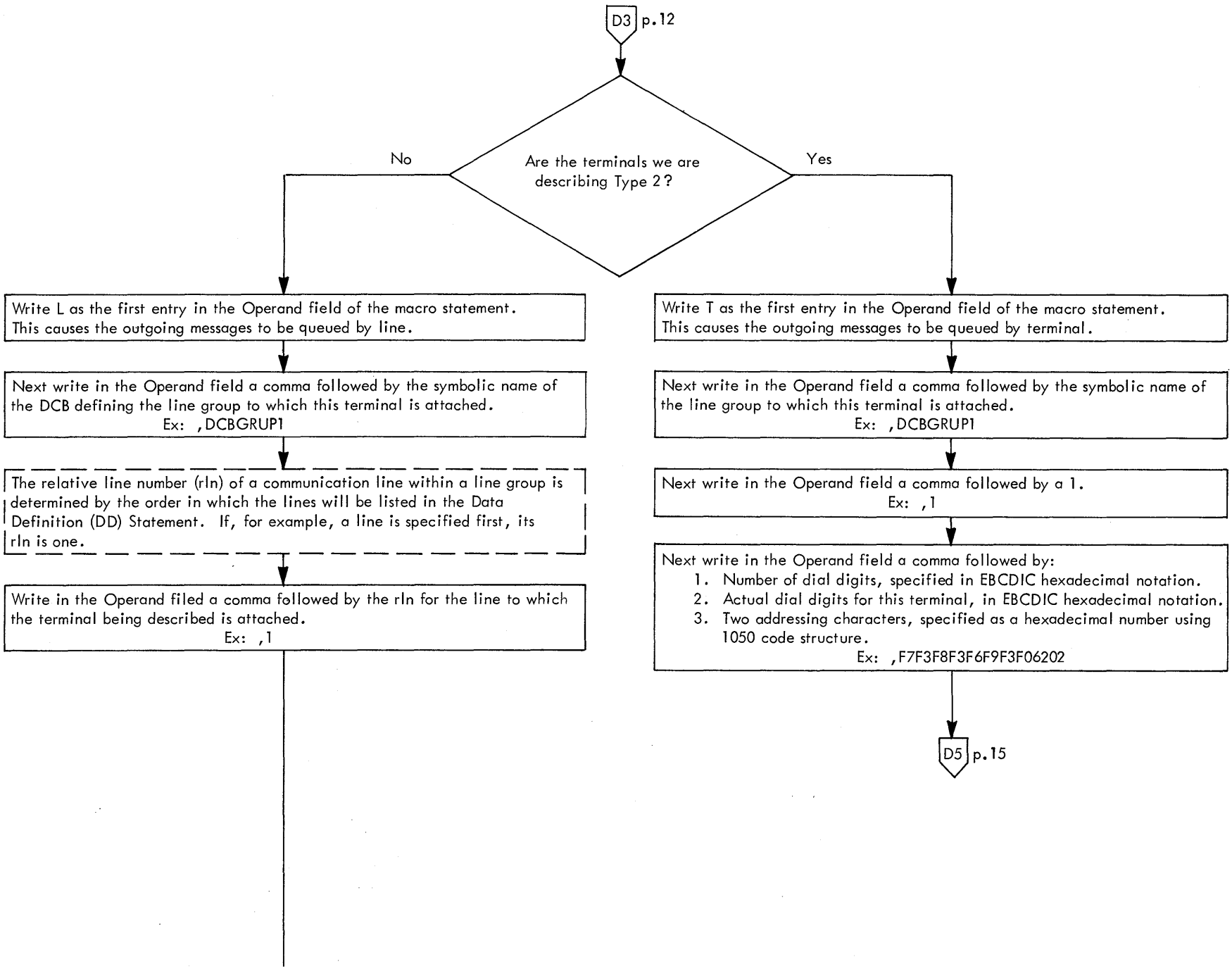
Name	Operation	Operand
NYC	TERM	L, DCBLINE, 1, 6202620E, (1)

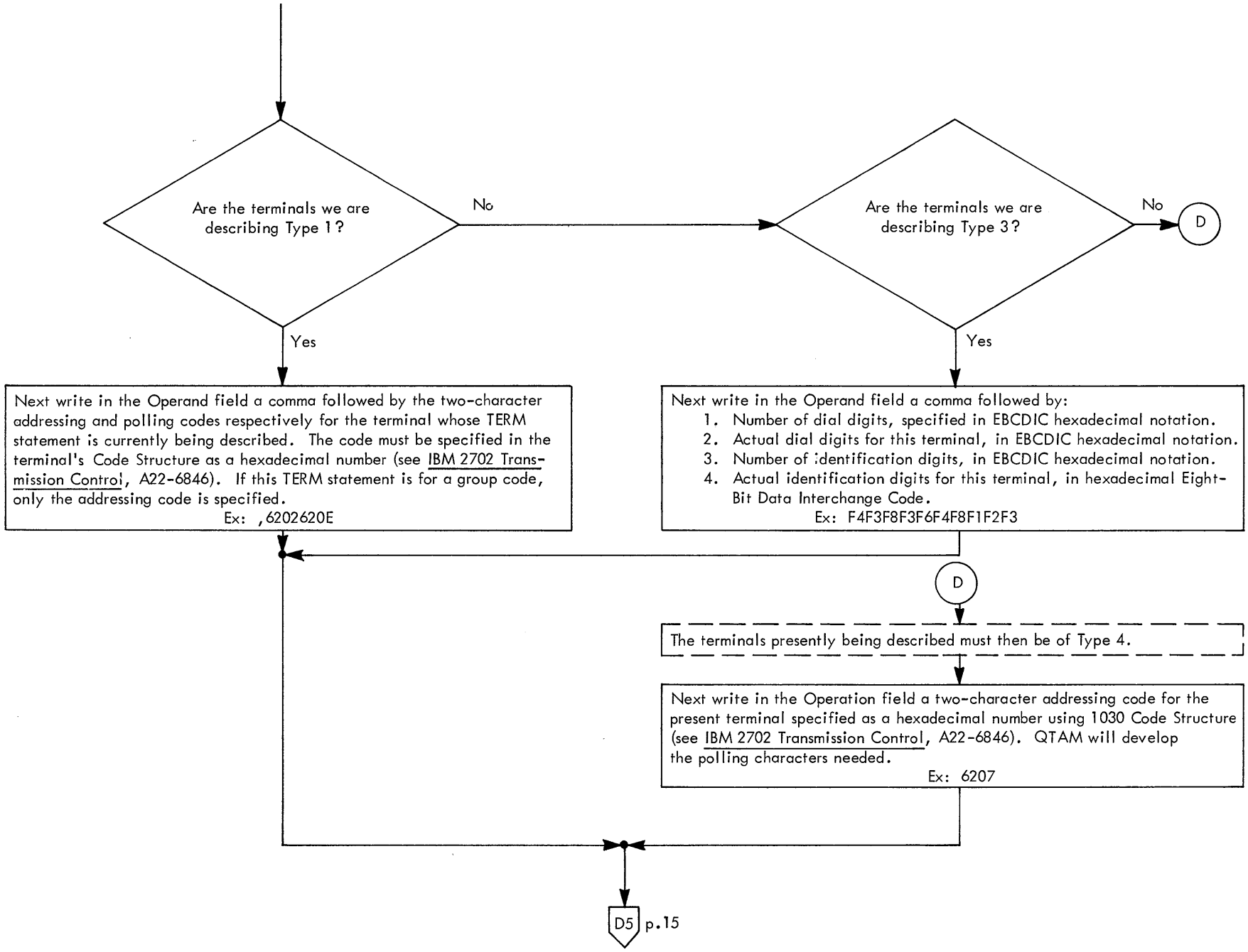
Write TERM in the Operation field of the macro statement.

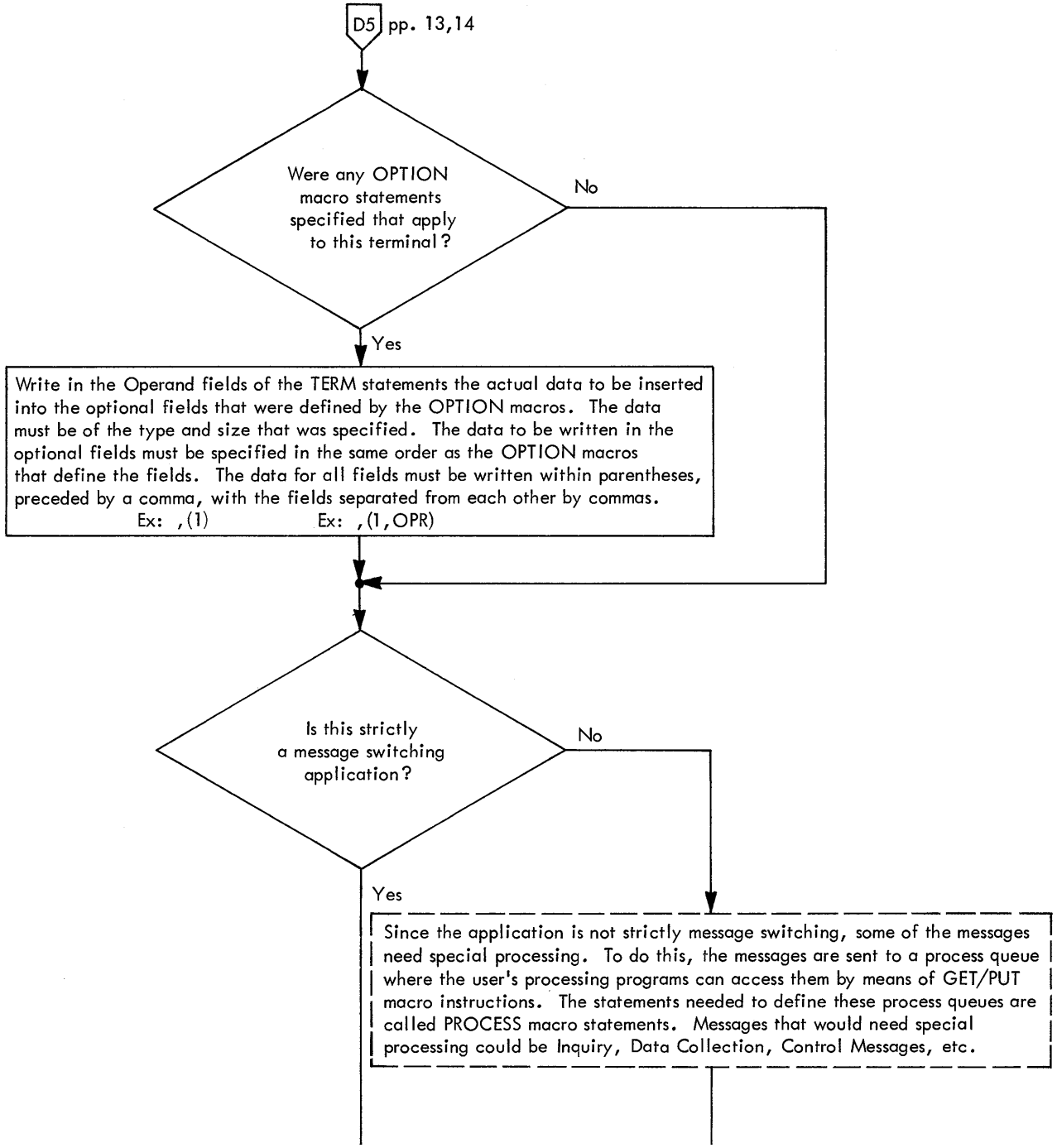
Information to be entered in the Operand field of the Term statement depends on the terminal type and desired method of operation. Those currently supported will be classified here as:

Type 1. IBM 1050 - Polled Terminals
IBM 1060 - Polled Terminals
AT&T 83B3 - Polled Terminals
WU 115A - Polled Terminals
Type 2. IBM 1050 - Dialing
Type 3. Common Carrier TWX
Type 4. IBM 1030

D3 p.13

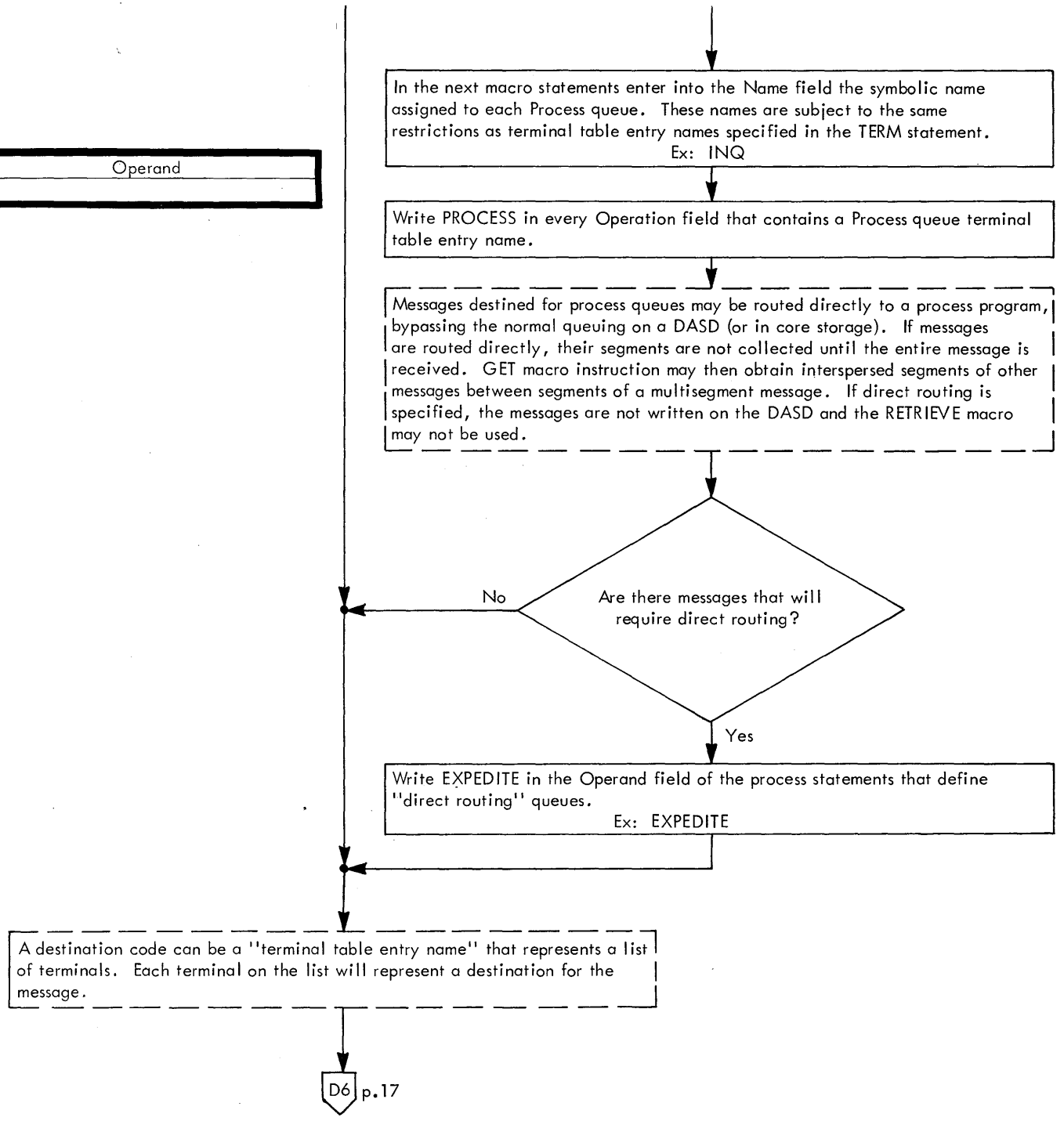

D3 p.12
Are the terminals we are describing Type 2?
No
Yes
Write L as the first entry in the Operand field of the macro statement. This causes the outgoing messages to be queued by line.
Next write in the Operand field a comma followed by the symbolic name of the DCB defining the line group to which this terminal is attached.
Ex: ,DCBGRUP1
The relative line number (rln) of a communication line within a line group is determined by the order in which the lines will be listed in the Data Definition (DD) Statement. If, for example, a line is specified first, its rln is one.
Write in the Operand filed a comma followed by the rln for the line to which the terminal being described is attached.
Ex: ,1
Write T as the first entry in the Operand field of the macro statement. This causes the outgoing messages to be queued by terminal.
Next write in the Operand field a comma followed by the symbolic name of the line group to which this terminal is attached.
Ex: ,DCBGRUP1
Next write in the Operand field a comma followed by a 1.
Ex: ,1
Next write in the Operand field a comma followed by:
1. Number of dial digits, specified in EBCDIC hexadecimal notation.
2. Actual dial digits for this terminal, in EBCDIC hexadecimal notation.
3. Two addressing characters, specified as a hexadecimal number using 1050 code structure.
Ex: ,F7F3F8F3F6F9F3F06202
D5 p.15

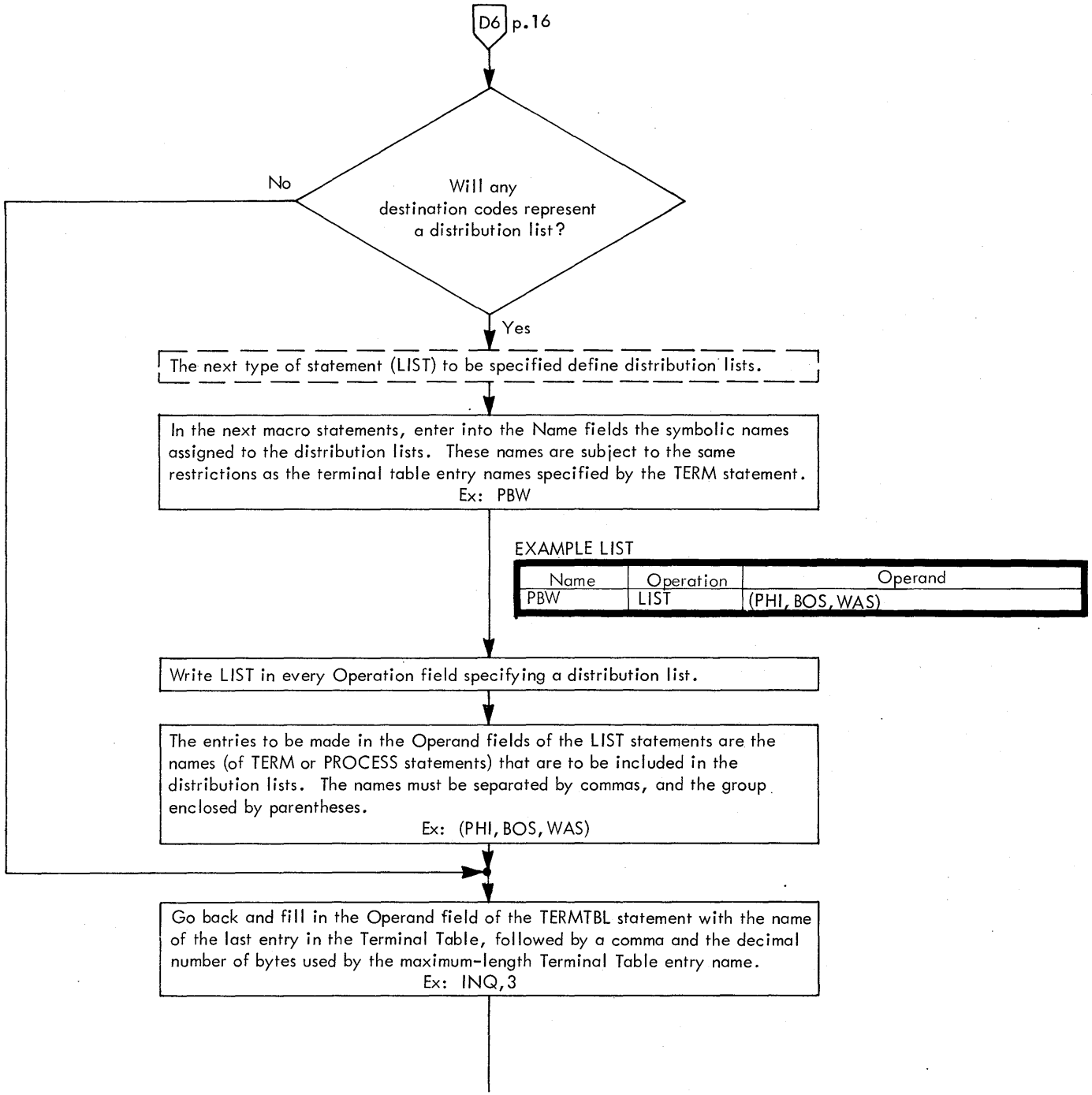

Are the terminals we are describing Type 1?
No
Are the terminals we are describing Type 3?
No
D
Yes
Yes
Next write in the Operand field a comma followed by the two-character addressing and polling codes respectively for the terminal whose TERM statement is currently being described. The code must be specified in the terminal's Code Structure as a hexadecimal number (see IBM 2702 Transmission Control, A22-6846). If this TERM statement is for a group code, only the addressing code is specified.
Ex: ,6202620E
Next write in the Operand field a comma followed by:
1. Number of dial digits, specified in EBCDIC hexadecimal notation.
2. Actual dial digits for this terminal, in EBCDIC hexadecimal notation.
3. Number of identification digits, in EBCDIC hexadecimal notation.
4. Actual identification digits for this terminal, in hexadecimal Eight-Bit Data Interchange Code.
Ex: F4F3F8F3F6F4F8F1F2F3
D
The terminals presently being described must then be of Type 4.
Next write in the Operation field a two-character addressing code for the present terminal specified as a hexadecimal number using 1030 Code Structure (see IBM 2702 Transmission Control, A22-6846). QTAM will develop the polling characters needed.
Ex: 6207
D5 p.15


D5 pp. 13,14
Were any OPTION macro statements specified that apply to this terminal?
No
Yes
Write in the Operand fields of the TERM statements the actual data to be inserted into the optional fields that were defined by the OPTION macros. The data must be of the type and size that was specified. The data to be written in the optional fields must be specified in the same order as the OPTION macros that define the fields. The data for all fields must be written within parentheses, preceded by a comma, with the fields separated from each other by commas.
Ex: ,(1) Ex: ,(1,OPR)
Is this strictly a message switching application?
No
Yes
Since the application is not strictly message switching, some of the messages need special processing. To do this, the messages are sent to a process queue where the user's processing programs can access them by means of GET/PUT macro instructions. The statements needed to define these process queues are called PROCESS macro statements. Messages that would need special processing could be Inquiry, Data Collection, Control Messages, etc.

EXAMPLE PROCESS

Name	Operation	Operand
INQ	PROCESS	




D6 p.16
Will any destination codes represent a distribution list?
No
Yes
The next type of statement (LIST) to be specified define distribution lists.
In the next macro statements, enter into the Name fields the symbolic names assigned to the distribution lists. These names are subject to the same restrictions as the terminal table entry names specified by the TERM statement.
Ex: PBW
EXAMPLE LIST
Name
Operation
Operand
PBW
LIST
(PHI, BOS, WAS)
Write LIST in every Operation field specifying a distribution list.
The entries to be made in the Operand fields of the LIST statements are the names (of TERM or PROCESS statements) that are to be included in the distribution lists. The names must be separated by commas, and the group enclosed by parentheses.
Ex: (PHI, BOS, WAS)
Go back and fill in the Operand field of the TERMTBL statement with the name of the last entry in the Terminal Table, followed by a comma and the decimal number of bytes used by the maximum-length Terminal Table entry name.
Ex: INQ, 3

EXAMPLE TERMTBL

Name	Operation	Operand
	TERMTBL	INQ,3

We are now able to finish specification of the ACLOC parameters that were not written in the Communication Line Group DCB's in Section A. The numbers to be written in the blank spaces reserved will define the "Device Address" field of each terminal relative to the first character of its Terminal Table Entry.

To calculate this decimal number n, use the formula n = 9+x+y, where
x = the maximum number of characters used for a TERM statement Terminal Table entry name and y = the number of characters used for optional fields in each TERM statement.

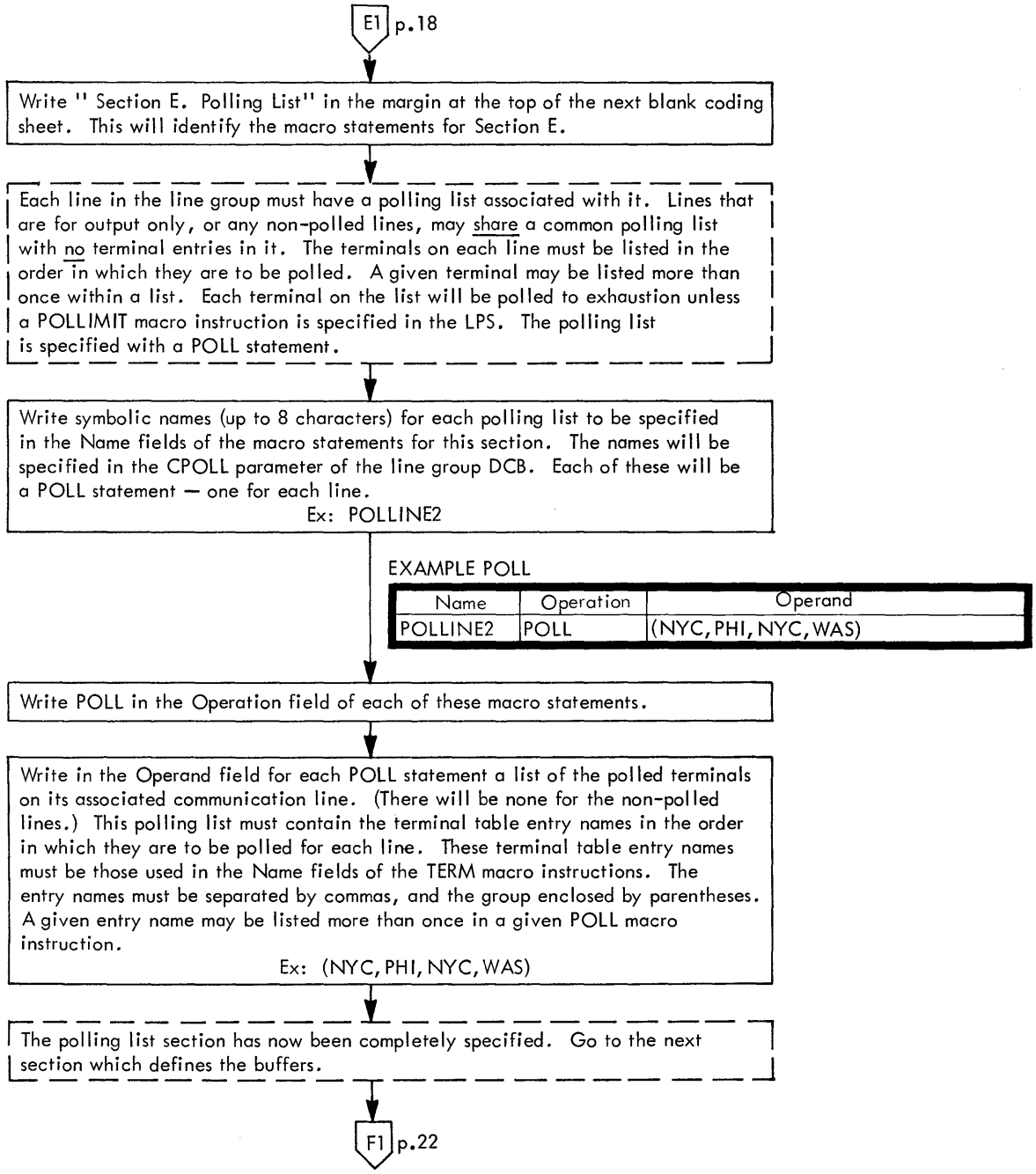
Ex: ACLOC=14

EXAMPLE DCB

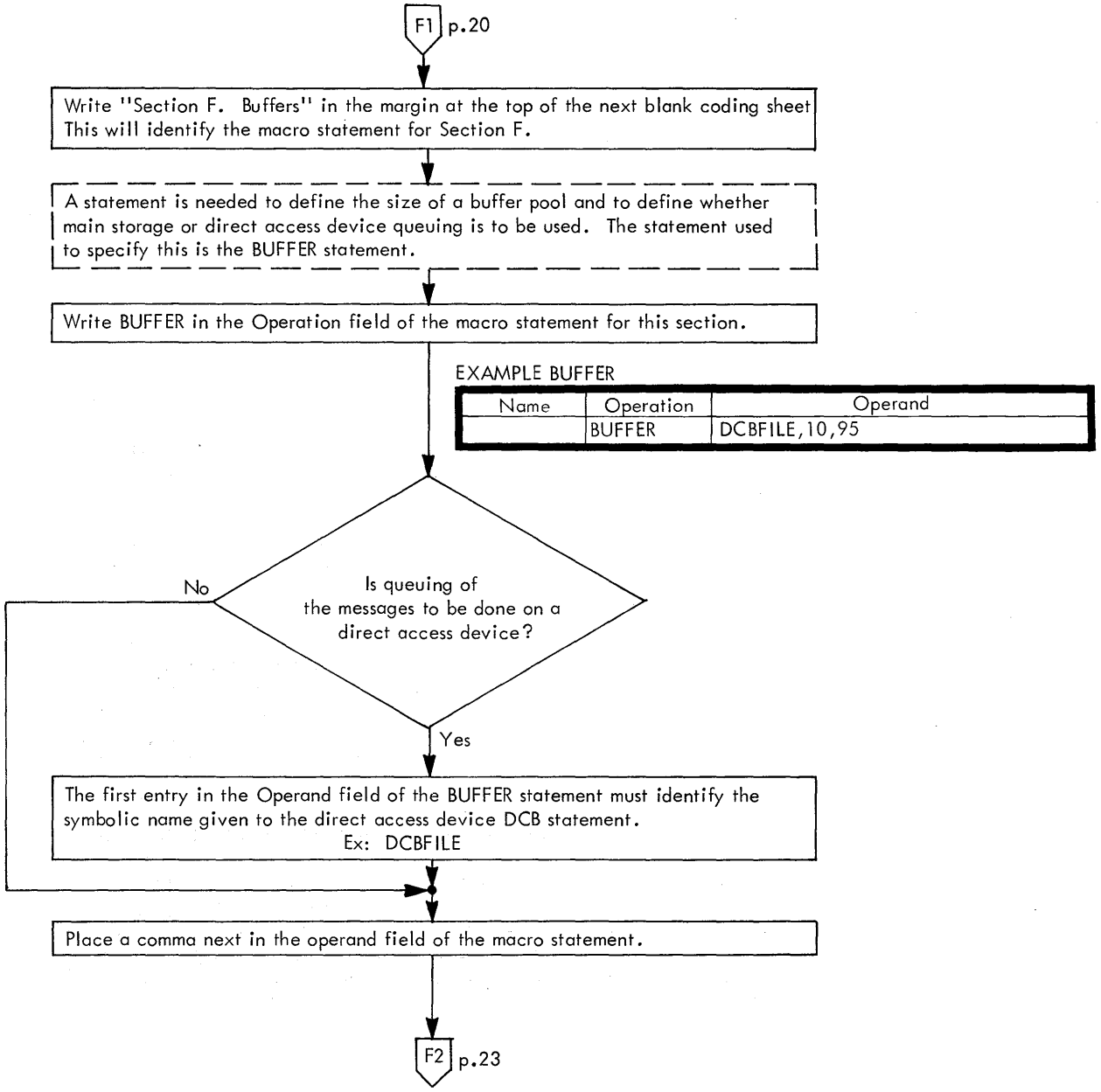
Name	Operation	Operand
DCBGRUP1	DCB	DDNAME=DDGROUP,DSORG=CX,
		MACRF=(G,P),CPOLL=
		(POLLINE1,POLLINE2),INTVL=5,
		BUFRQ=3,ACLOC=14,
		CLPS=LPS1

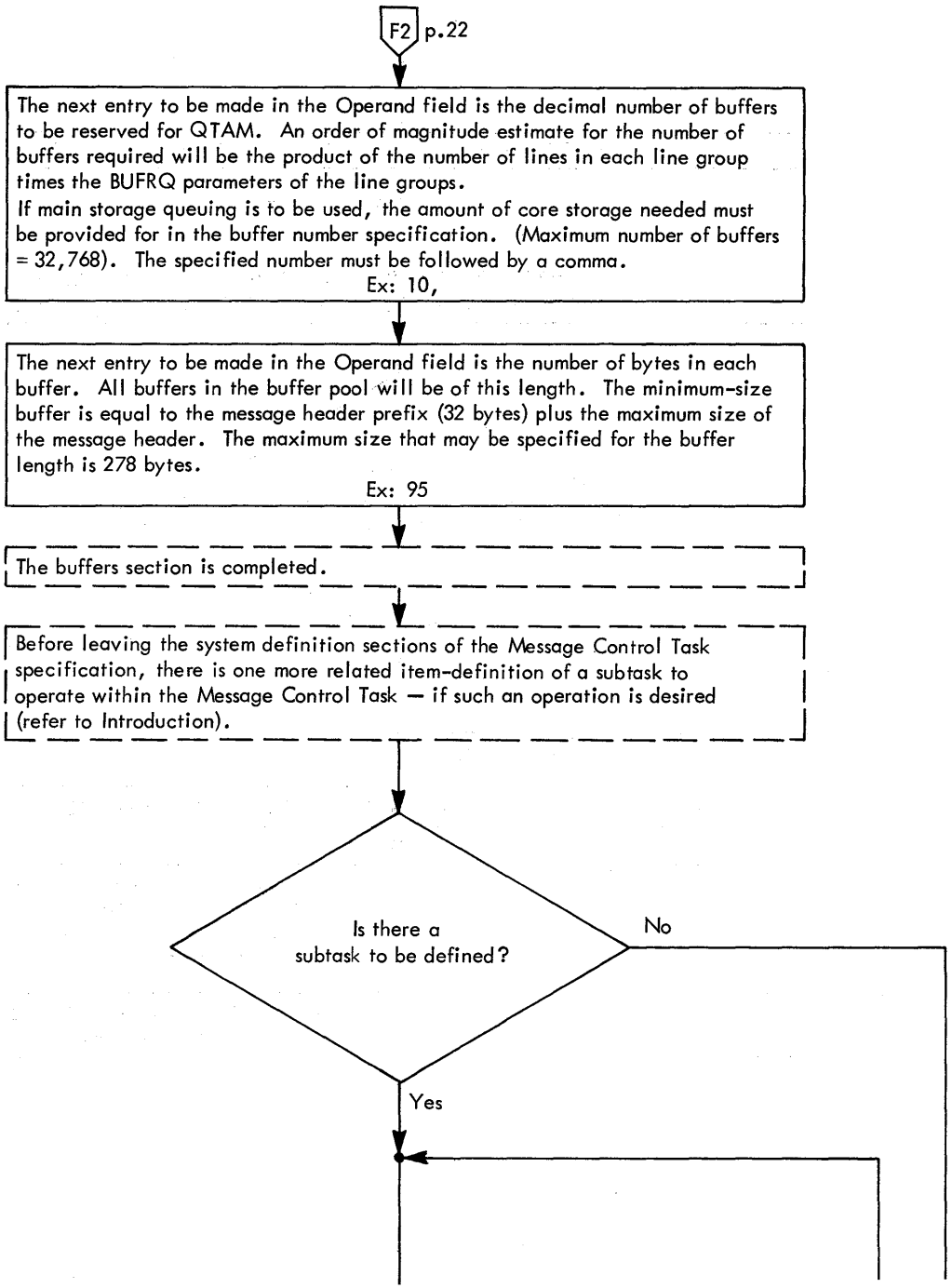
E1 p.20

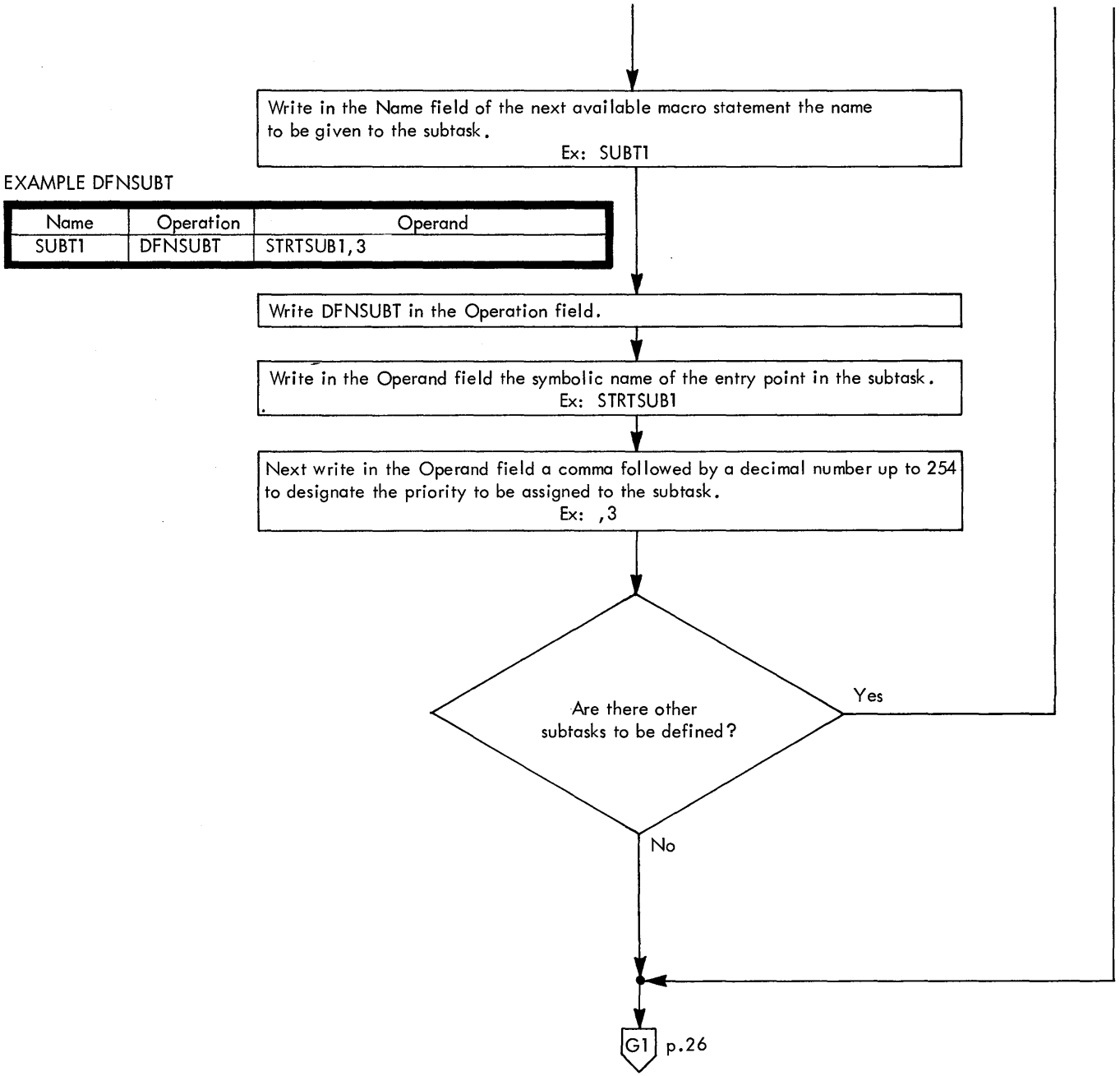
SECTION E. POLLING LIST



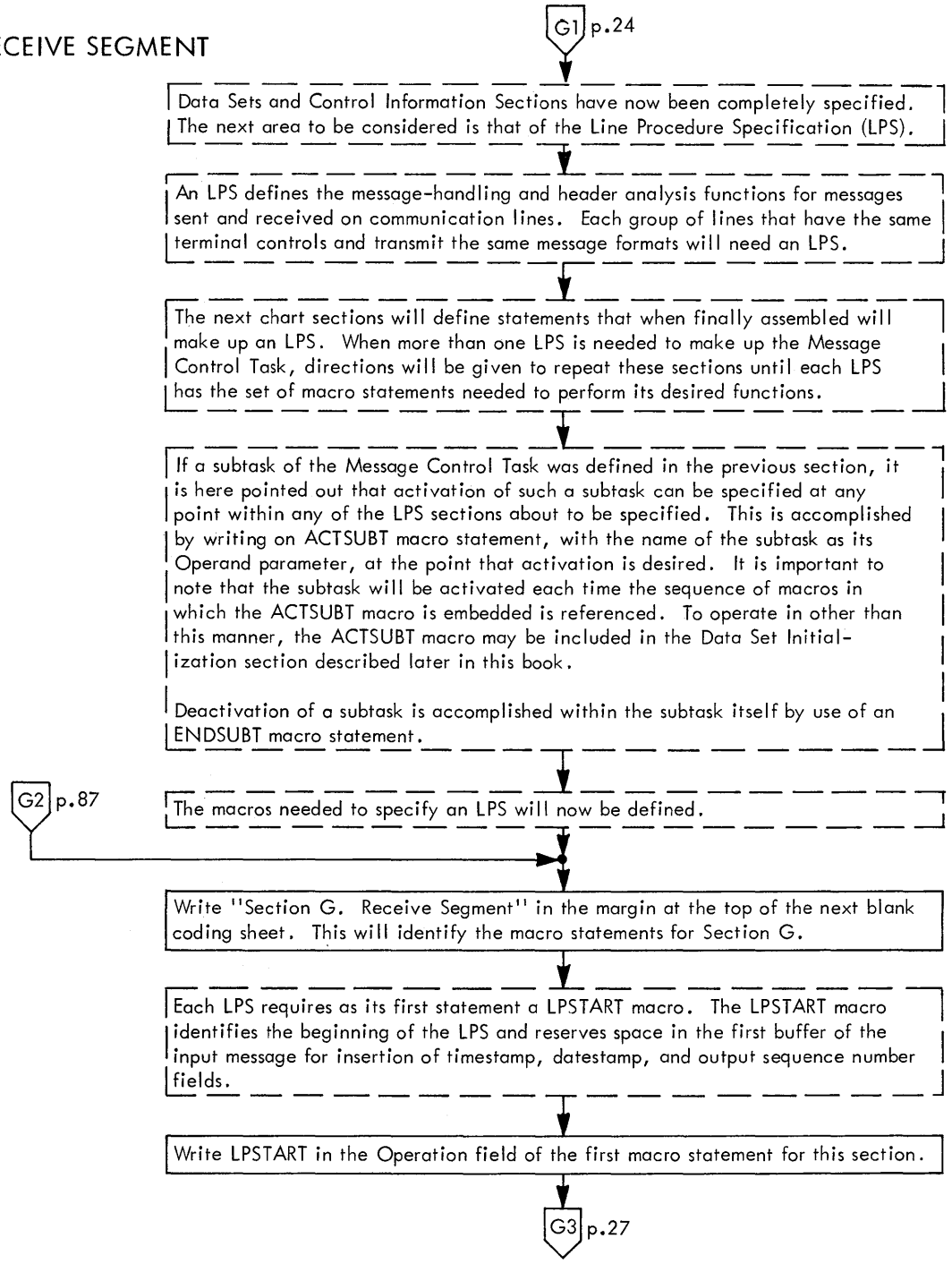
SECTION F. BUFFERS

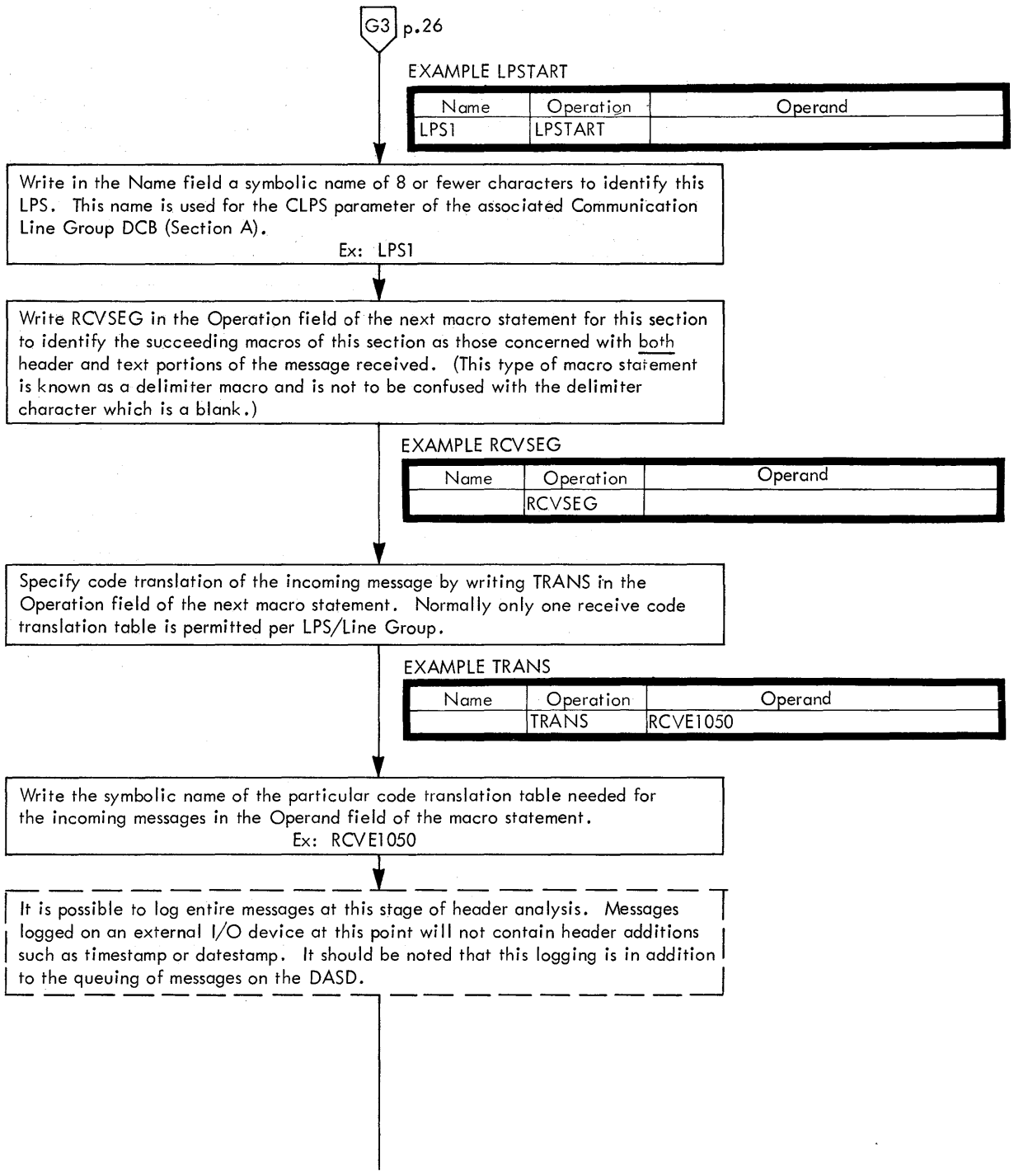



F2 p.22
The next entry to be made in the Operand field is the decimal number of buffers to be reserved for QTAM. An order of magnitude estimate for the number of buffers required will be the product of the number of lines in each line group times the BUFRQ parameters of the line groups.
If main storage queuing is to be used, the amount of core storage needed must be provided for in the buffer number specification. (Maximum number of buffers = 32,768). The specified number must be followed by a comma.
Ex: 10,
The next entry to be made in the Operand field is the number of bytes in each buffer. All buffers in the buffer pool will be of this length. The minimum-size buffer is equal to the message header prefix (32 bytes) plus the maximum size of the message header. The maximum size that may be specified for the buffer length is 278 bytes.
Ex: 95
The buffers section is completed.
Before leaving the system definition sections of the Message Control Task specification, there is one more related item-definition of a subtask to operate within the Message Control Task — if such an operation is desired (refer to Introduction).
Is there a subtask to be defined?
No
Yes

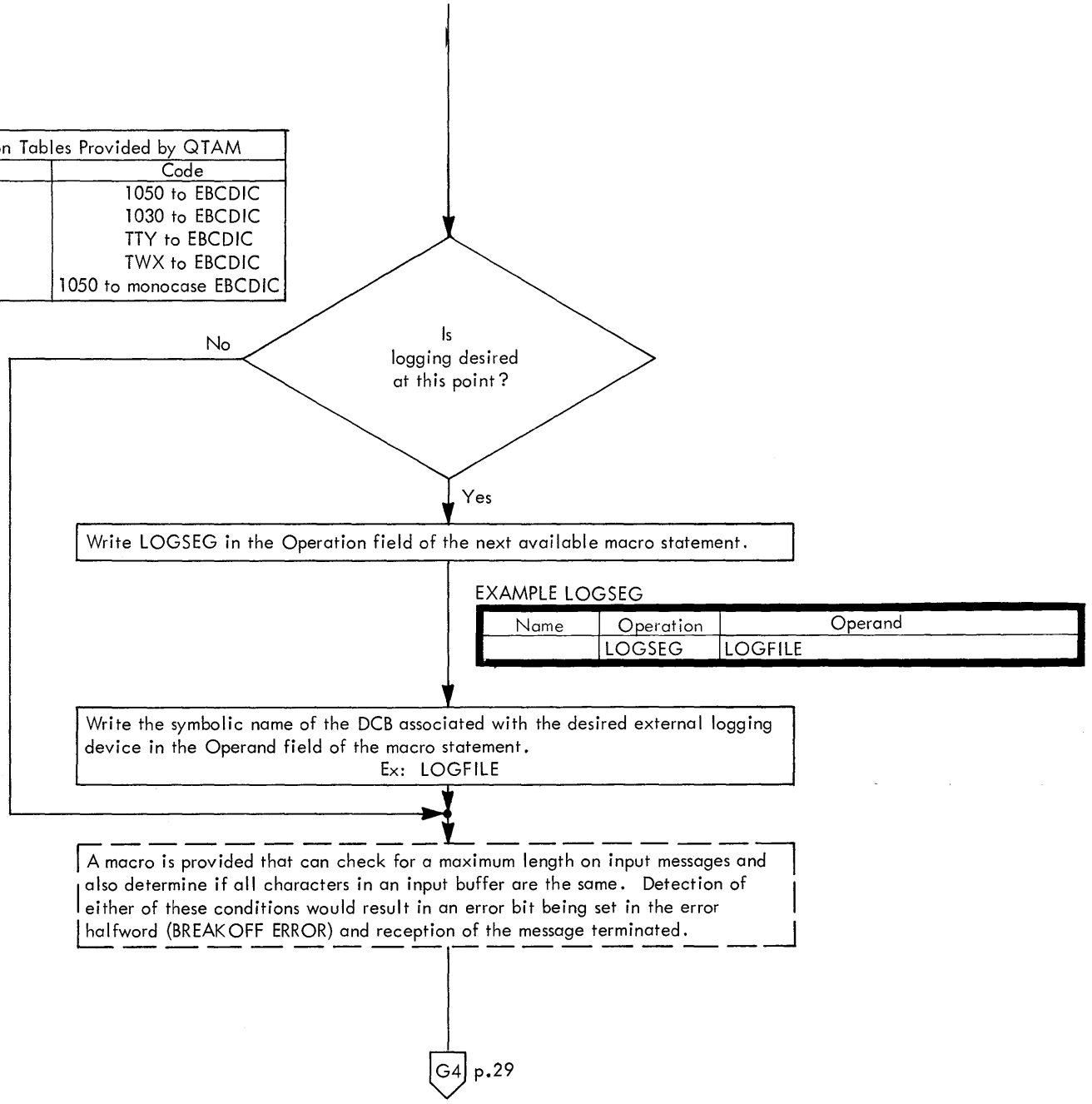

Write in the Name field of the next available macro statement the name to be given to the subtask.
Ex: SUBT1
EXAMPLE DFNSUBT
Name | Operation | Operand
SUBT1 | DFNSUBT | STRTSUB1,3
Write DFNSUBT in the Operation field.
Write in the Operand field the symbolic name of the entry point in the subtask.
Ex: STRTSUB1
Next write in the Operand field a comma followed by a decimal number up to 254 to designate the priority to be assigned to the subtask.
Ex: ,3
Are there other subtasks to be defined?
Yes
No
G1
p.26

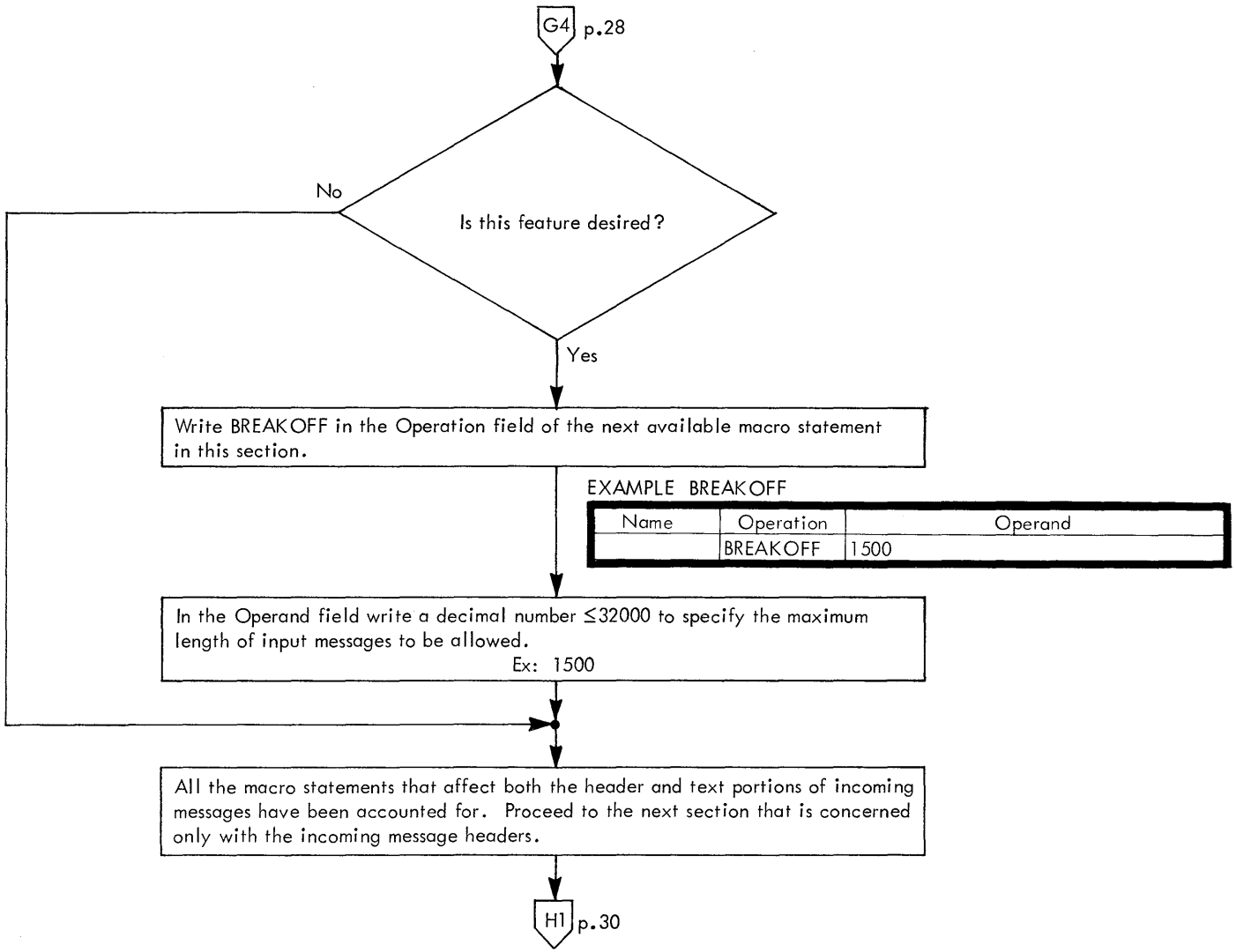
SECTION G. RECEIVE SEGMENT




G3 p.26
EXAMPLE LPSTART
Name | Operation | Operand
LPS1 | LPSTART |
Write in the Name field a symbolic name of 8 or fewer characters to identify this LPS. This name is used for the CLPS parameter of the associated Communication Line Group DCB (Section A).
Ex: LPS1
Write RCVSEG in the Operation field of the next macro statement for this section to identify the succeeding macros of this section as those concerned with both header and text portions of the message received. (This type of macro statement is known as a delimiter macro and is not to be confused with the delimiter character which is a blank.)
EXAMPLE RCVSEG
Name | Operation | Operand
| RCVSEG |
Specify code translation of the incoming message by writing TRANS in the Operation field of the next macro statement. Normally only one receive code translation table is permitted per LPS/Line Group.
EXAMPLE TRANS
Name | Operation | Operand
| TRANS | RCVE1050
Write the symbolic name of the particular code translation table needed for the incoming messages in the Operand field of the macro statement.
Ex: RCVE1050
It is possible to log entire messages at this stage of header analysis. Messages logged on an external I/O device at this point will not contain header additions such as timestamp or datestamp. It should be noted that this logging is in addition to the queuing of messages on the DASD.

Code Translation Tables Provided by QTAM	
Name	Code
RCVE1050	1050 to EBCDIC
RCVE1030	1030 to EBCDIC
RCVET1	TTY to EBCDIC
RCVET2	TWX to EBCDIC
RCVF1050	1050 to monocase EBCDIC




G4 p.28
Is this feature desired?
No
Yes
Write BREAKOFF in the Operation field of the next available macro statement in this section.
EXAMPLE BREAKOFF
Name
Operation
Operand
BREAKOFF
1500
In the Operand field write a decimal number ≤32000 to specify the maximum length of input messages to be allowed.
Ex: 1500
All the macro statements that affect both the header and text portions of incoming messages have been accounted for. Proceed to the next section that is concerned only with the incoming message headers.
H1 p.30

SECTION H. RECEIVE HEADER

H1 p.29

Write "Section H. Receive Header" in the margin at the top of the next blank coding sheet. This will identify the macro statements for Section H.

Write the delimiter macro RCVHDR in the Operation field of the first macro statement for this section to identify the succeeding macros of this section as those concerned only with incoming message headers.

EXAMPLE RCVHDR

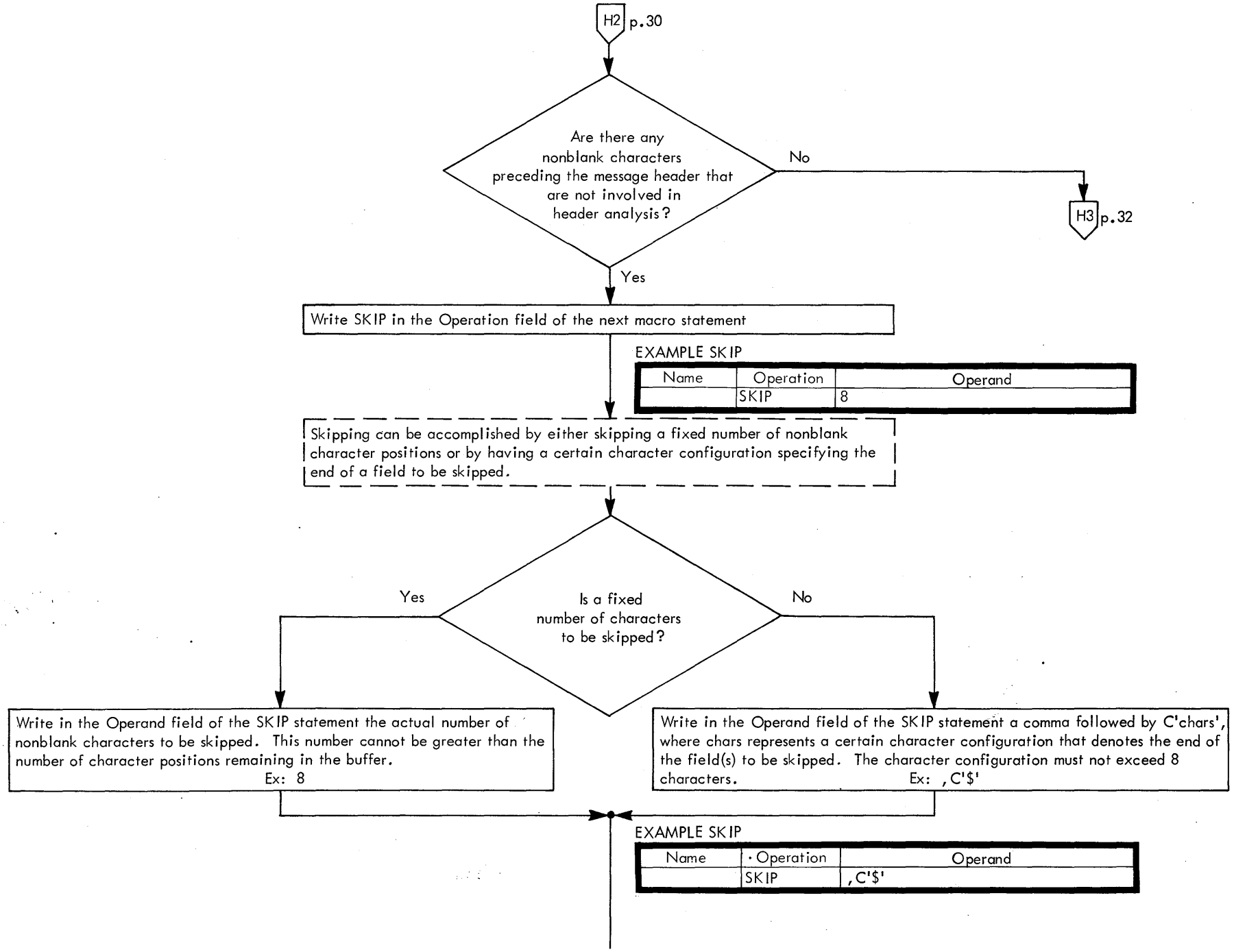
Name	Operation	Operand
	RCVHDR	

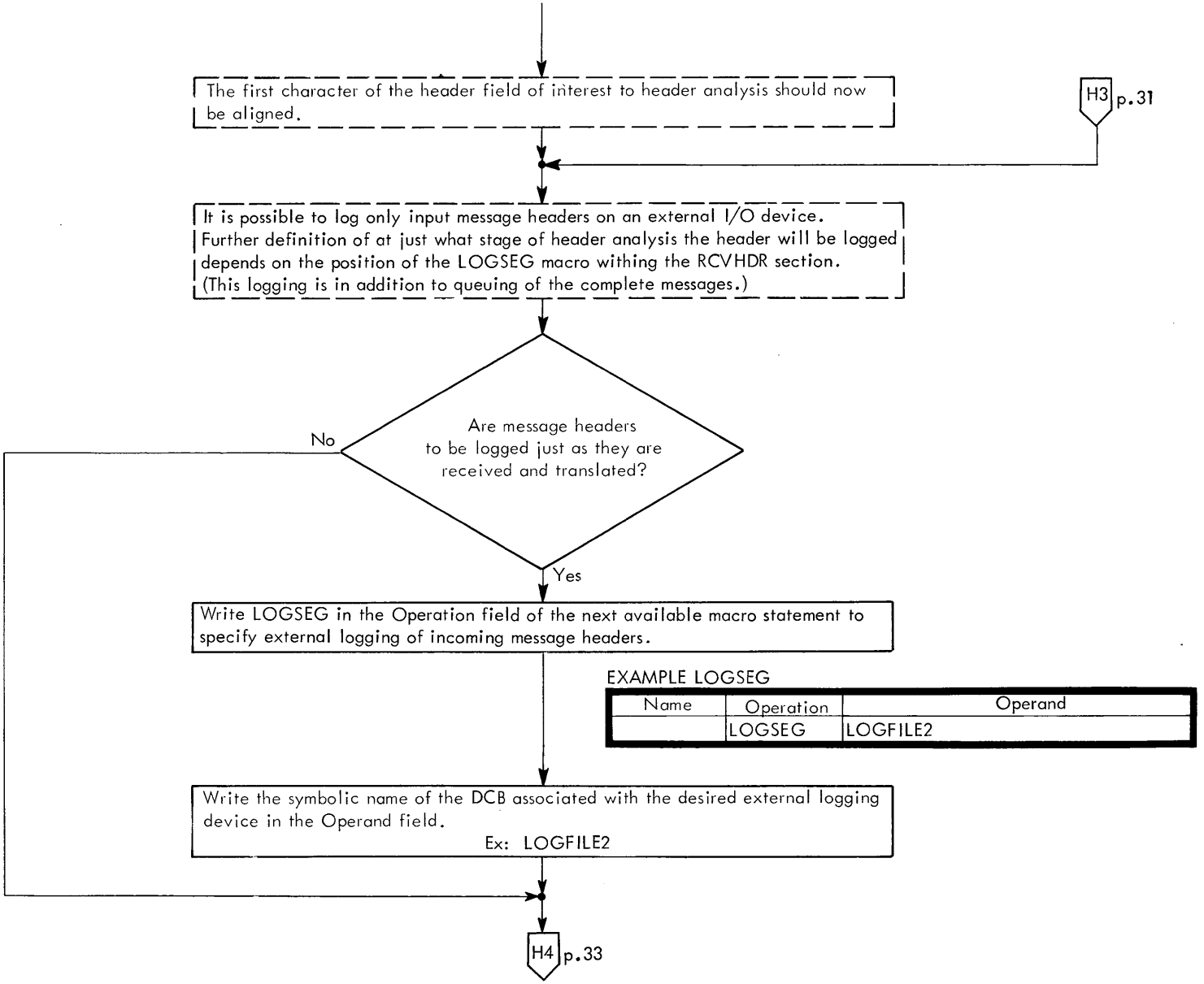
This section of the LPS will perform the desired header analysis/synthesis of incoming message headers. To do this, it is necessary to start at the beginning of the message header and proceed through it (left to right) by specifying the appropriate functional macro instructions in the same order in which they apply to the header. As each macro that operates on a particular field is executed, the LPS will advance to the beginning of the next field of the header before the next macro will be executed. The beginning of the next field will be the first nonblank character after the end of the field being operated on. It should be noted that all blank characters will automatically be skipped between the fields.

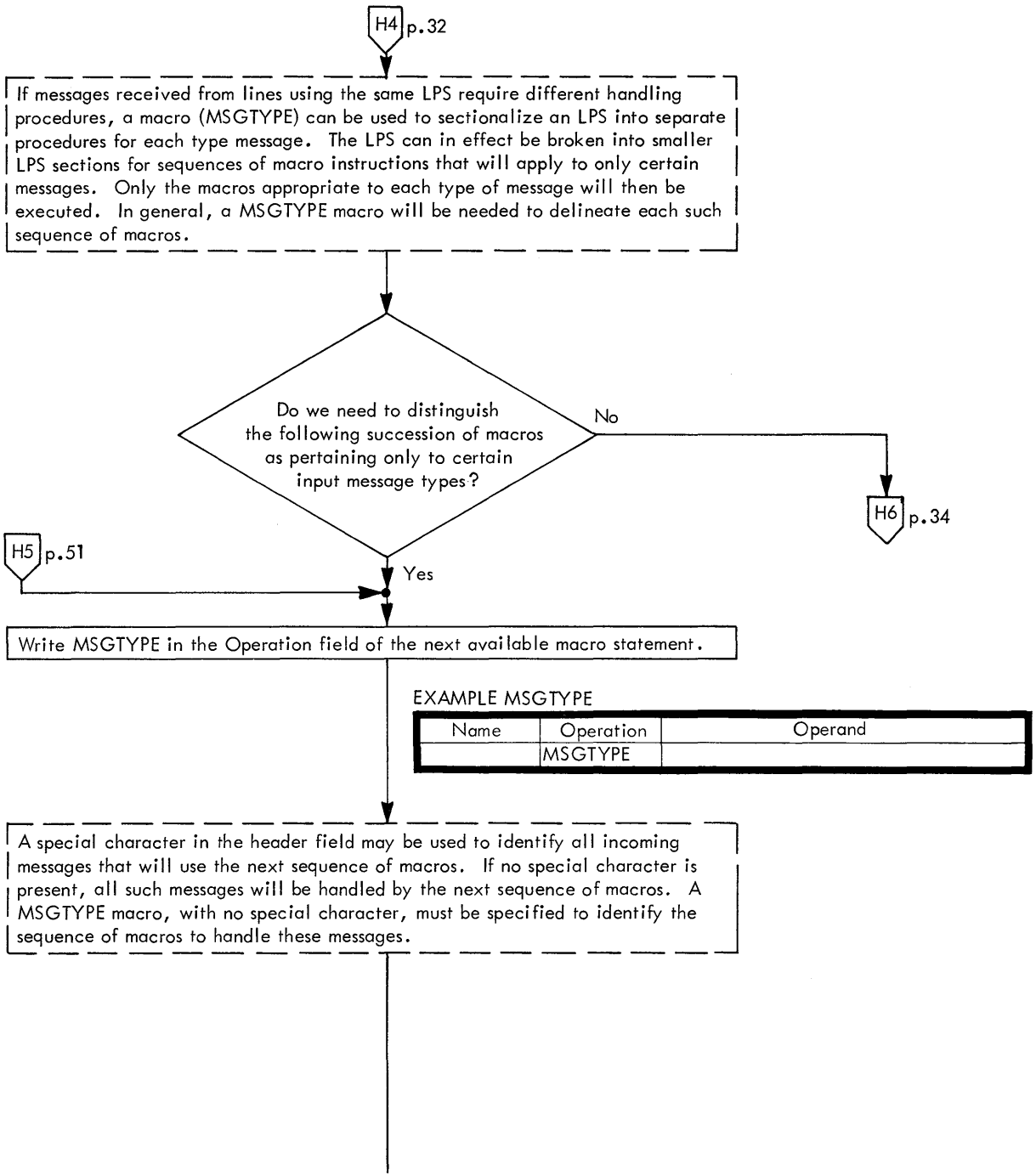
It is important that the LPS be kept aligned with the actual message header by proper delineation of the header fields. This is done by specifying field lengths within the macro statements and by skipping over nonblank fields not involved in header analysis with the SKIP macro.

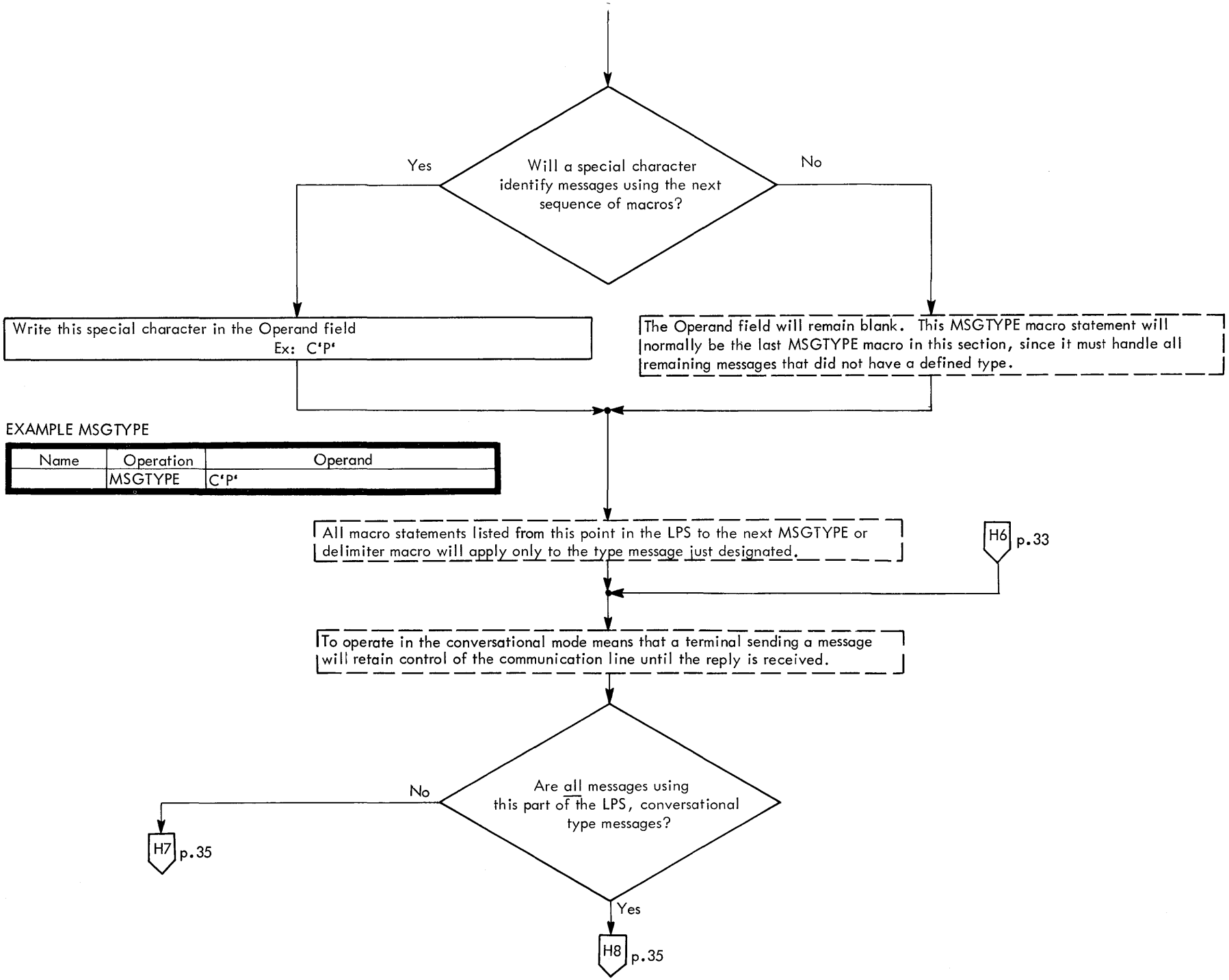
At this point the LPS program is automatically aligned with the first character of the received message (not necessarily the first meaningful character of the header).

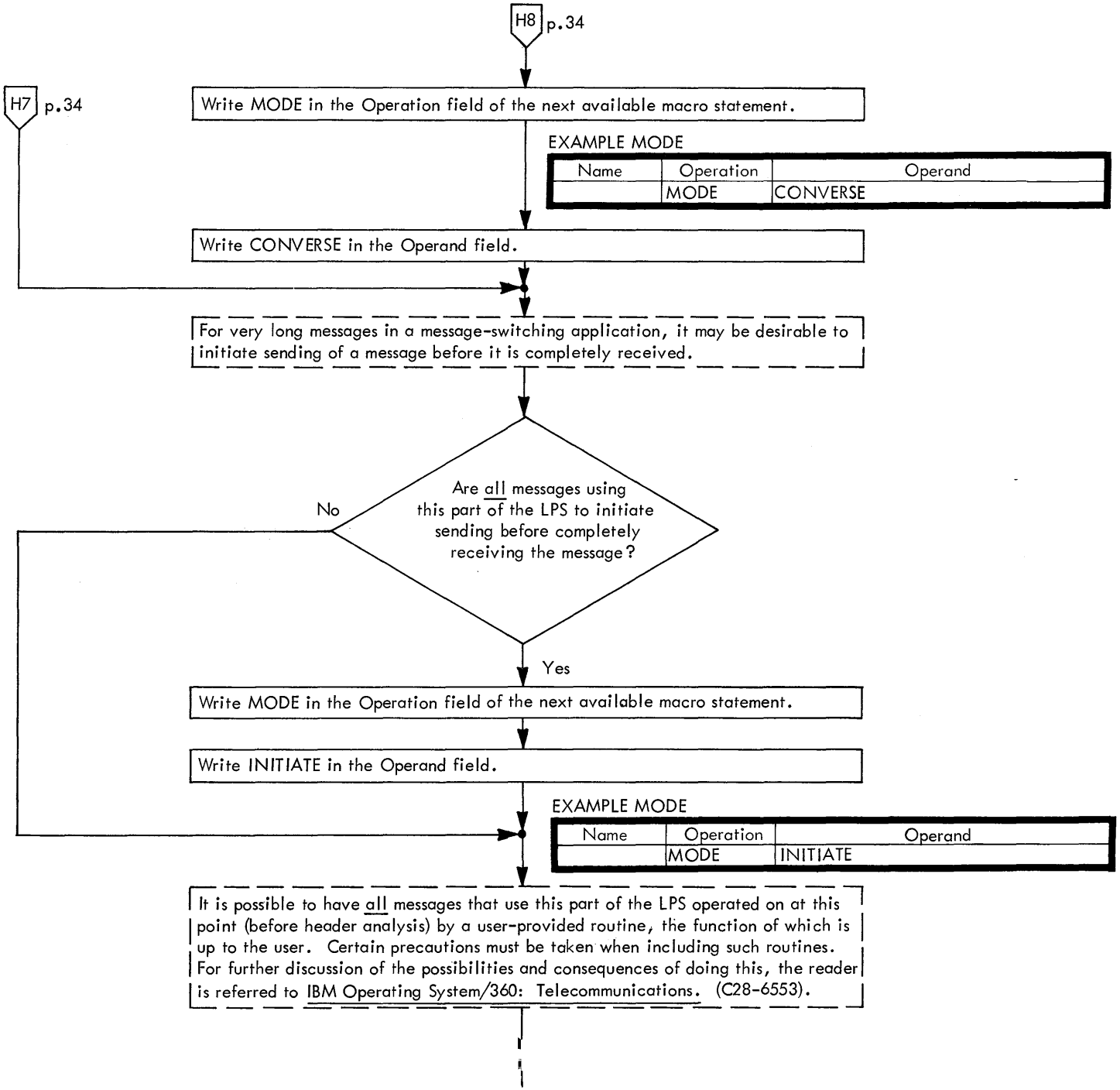
H2 p.31

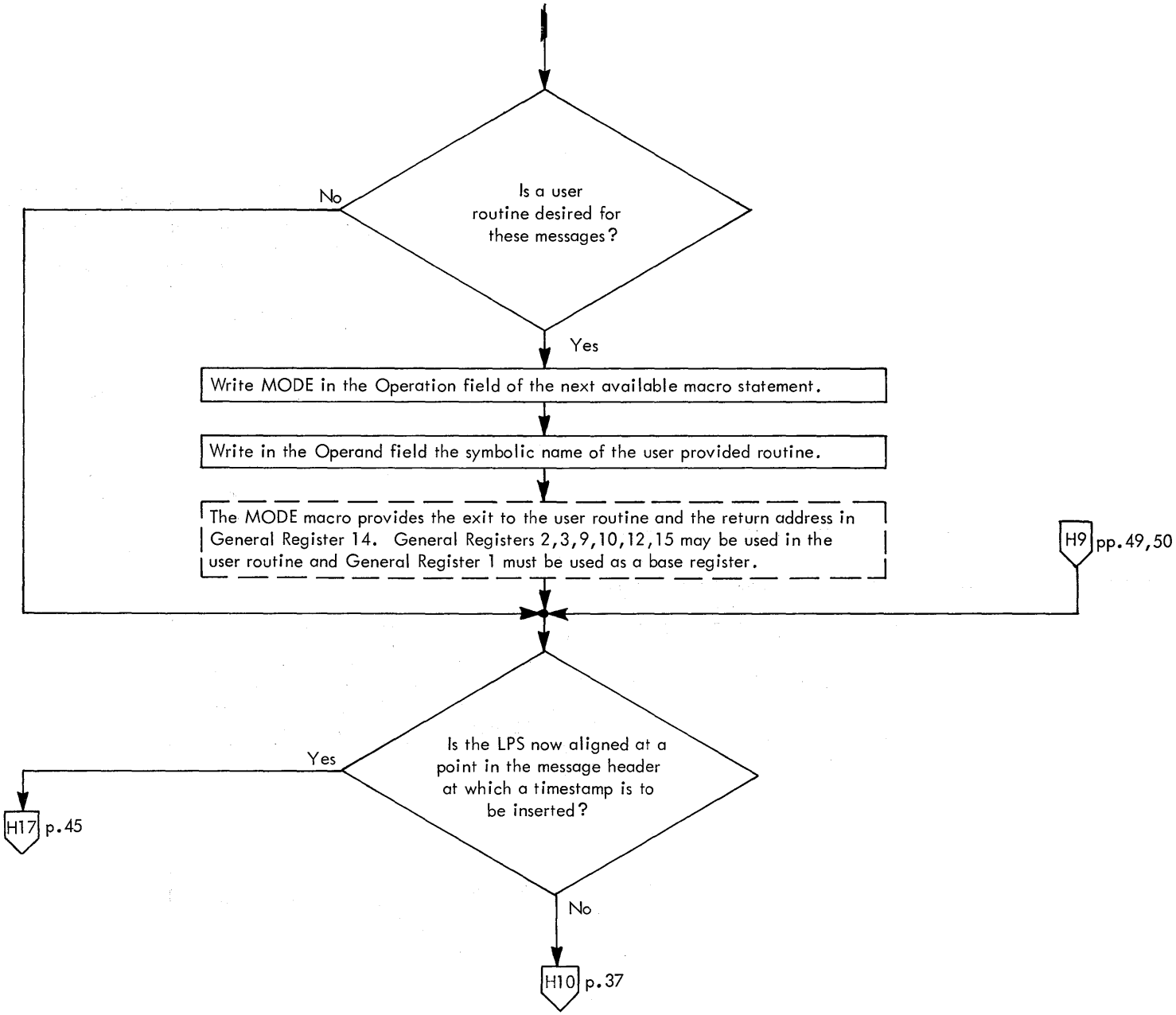

H2 p.30
Are there any nonblank characters preceding the message header that are not involved in header analysis?
No
H3 p.32
Yes
Write SKIP in the Operation field of the next macro statement
EXAMPLE SKIP
Name | Operation | Operand
SKIP | 8
Skipping can be accomplished by either skipping a fixed number of nonblank character positions or by having a certain character configuration specifying the end of a field to be skipped.
Is a fixed number of characters to be skipped?
Yes
No
Write in the Operand field of the SKIP statement the actual number of nonblank characters to be skipped. This number cannot be greater than the number of character positions remaining in the buffer.
Ex: 8
Write in the Operand field of the SKIP statement a comma followed by C'chars', where chars represents a certain character configuration that denotes the end of the field(s) to be skipped. The character configuration must not exceed 8 characters.
Ex: ,C'$'
EXAMPLE SKIP
Name | Operation | Operand
SKIP | ,C'$'

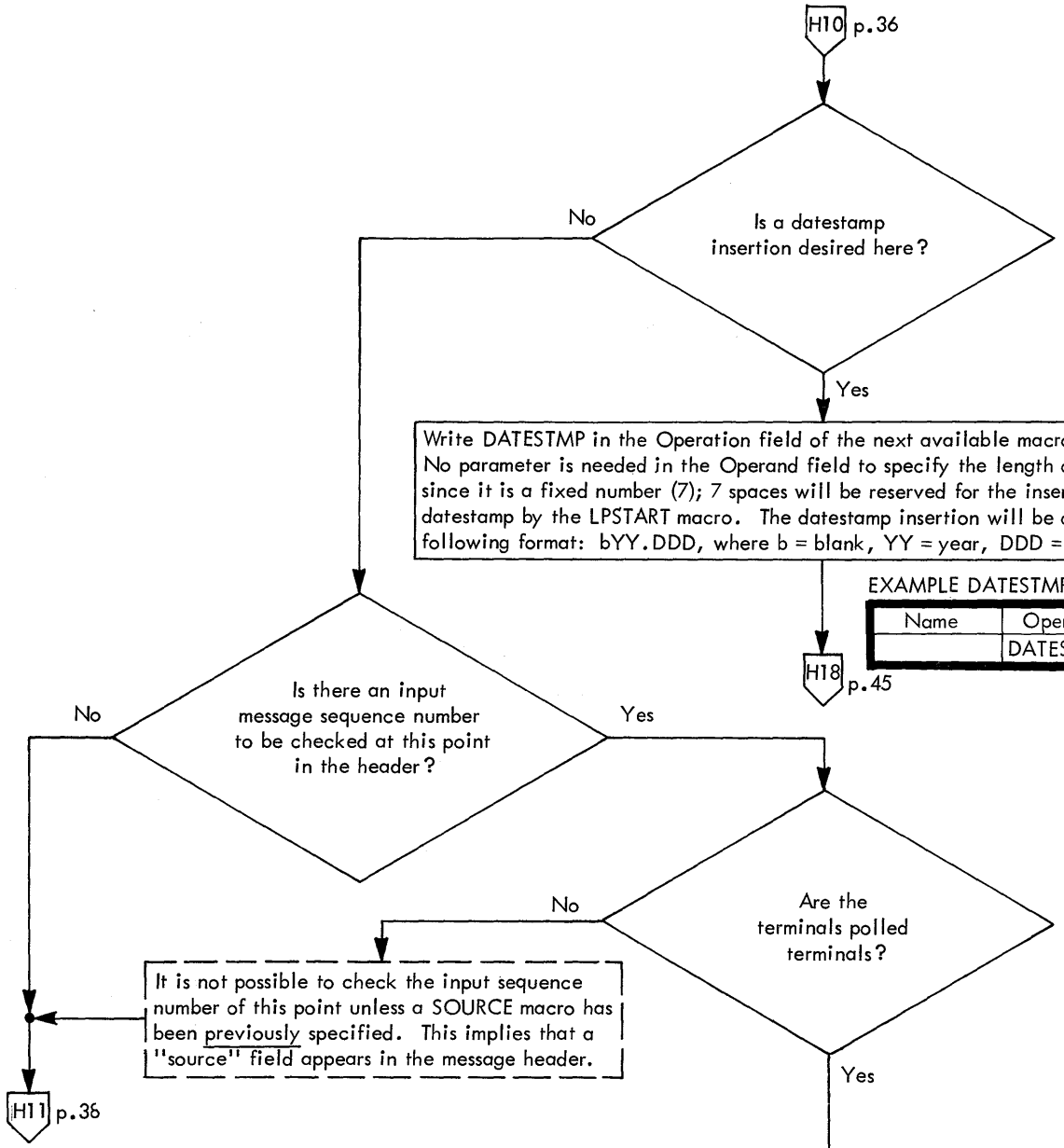

The first character of the header field of interest to header analysis should now be aligned.
H3 p.31
It is possible to log only input message headers on an external I/O device. Further definition of at just what stage of header analysis the header will be logged depends on the position of the LOGSEG macro withing the RCVHDR section. (This logging is in addition to queuing of the complete messages.)
Are message headers to be logged just as they are received and translated?
No
Yes
Write LOGSEG in the Operation field of the next available macro statement to specify external logging of incoming message headers.
EXAMPLE LOGSEG
Name
Operation
Operand
LOGSEG
LOGFILE2
Write the symbolic name of the DCB associated with the desired external logging device in the Operand field.
Ex: LOGFILE2
H4 p.33


H4 p.32
If messages received from lines using the same LPS require different handling procedures, a macro (MSGTYPE) can be used to sectionalize an LPS into separate procedures for each type message. The LPS can in effect be broken into smaller LPS sections for sequences of macro instructions that will apply to only certain messages. Only the macros appropriate to each type of message will then be executed. In general, a MSGTYPE macro will be needed to delineate each such sequence of macros.
Do we need to distinguish the following succession of macros as pertaining only to certain input message types?
No
H6 p.34
H5 p.51
Yes
Write MSGTYPE in the Operation field of the next available macro statement.
EXAMPLE MSGTYPE
Name
Operation
Operand
MSGTYPE
A special character in the header field may be used to identify all incoming messages that will use the next sequence of macros. If no special character is present, all such messages will be handled by the next sequence of macros. A MSGTYPE macro, with no special character, must be specified to identify the sequence of macros to handle these messages.


Will a special character identify messages using the next sequence of macros?
Yes
No
Write this special character in the Operand field
Ex: C'P'
The Operand field will remain blank. This MSGTYPE macro statement will normally be the last MSGTYPE macro in this section, since it must handle all remaining messages that did not have a defined type.
EXAMPLE MSGTYPE
Name
Operation
Operand
MSGTYPE
C'P'
All macro statements listed from this point in the LPS to the next MSGTYPE or delimiter macro will apply only to the type message just designated.
H6 p.33
To operate in the conversational mode means that a terminal sending a message will retain control of the communication line until the reply is received.
Are all messages using this part of the LPS, conversational type messages?
No
H7 p.35
Yes
H8 p.35

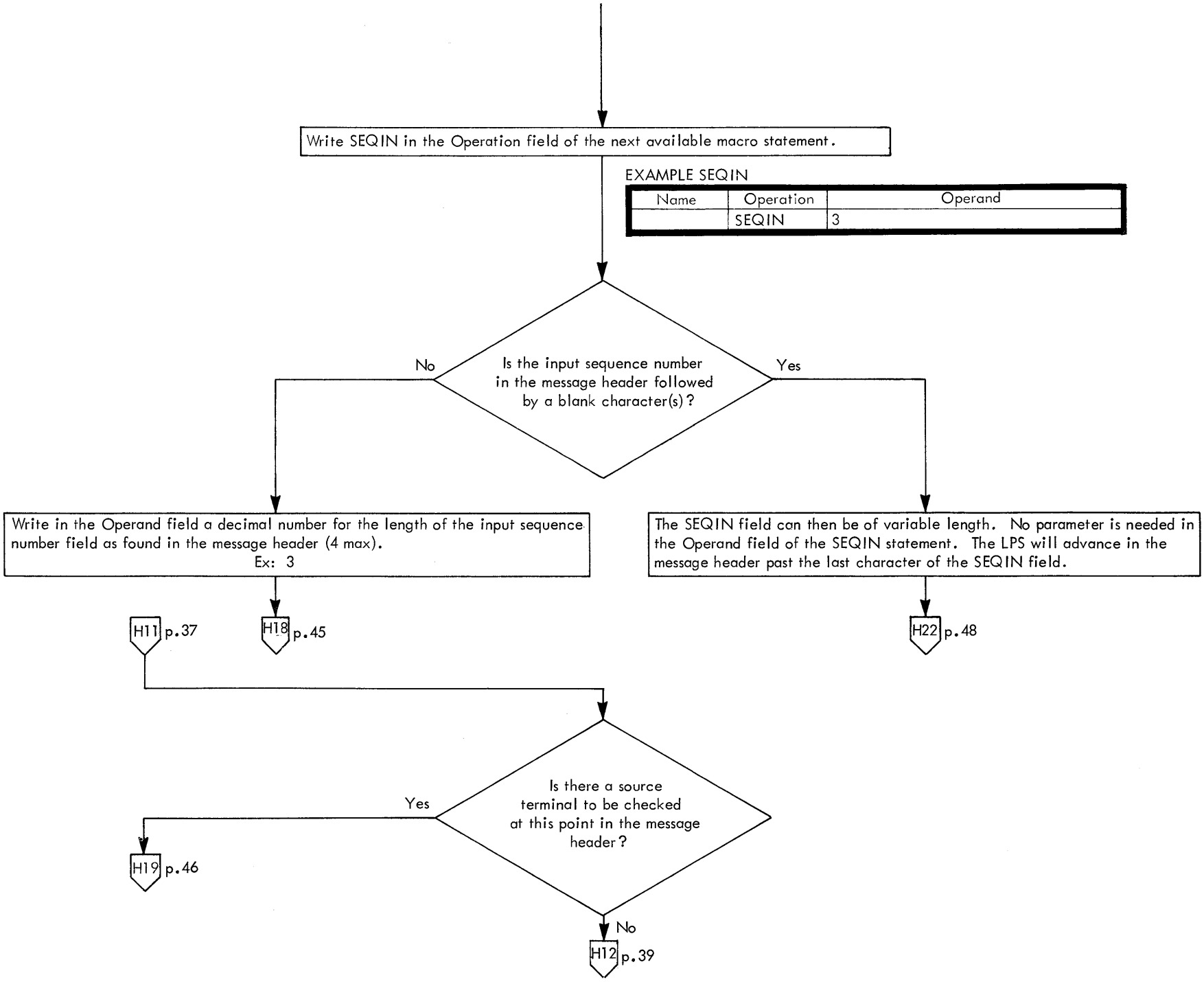

H8 p.34
H7 p.34
Write MODE in the Operation field of the next available macro statement.
EXAMPLE MODE
Name | Operation | Operand
MODE | CONVERSE
Write CONVERSE in the Operand field.
For very long messages in a message-switching application, it may be desirable to initiate sending of a message before it is completely received.
Are all messages using this part of the LPS to initiate sending before completely receiving the message?
No
Yes
Write MODE in the Operation field of the next available macro statement.
Write INITIATE in the Operand field.
EXAMPLE MODE
Name | Operation | Operand
MODE | INITIATE
It is possible to have all messages that use this part of the LPS operated on at this point (before header analysis) by a user-provided routine, the function of which is up to the user. Certain precautions must be taken when including such routines. For further discussion of the possibilities and consequences of doing this, the reader is referred to IBM Operating System/360: Telecommunications. (C28-6553).

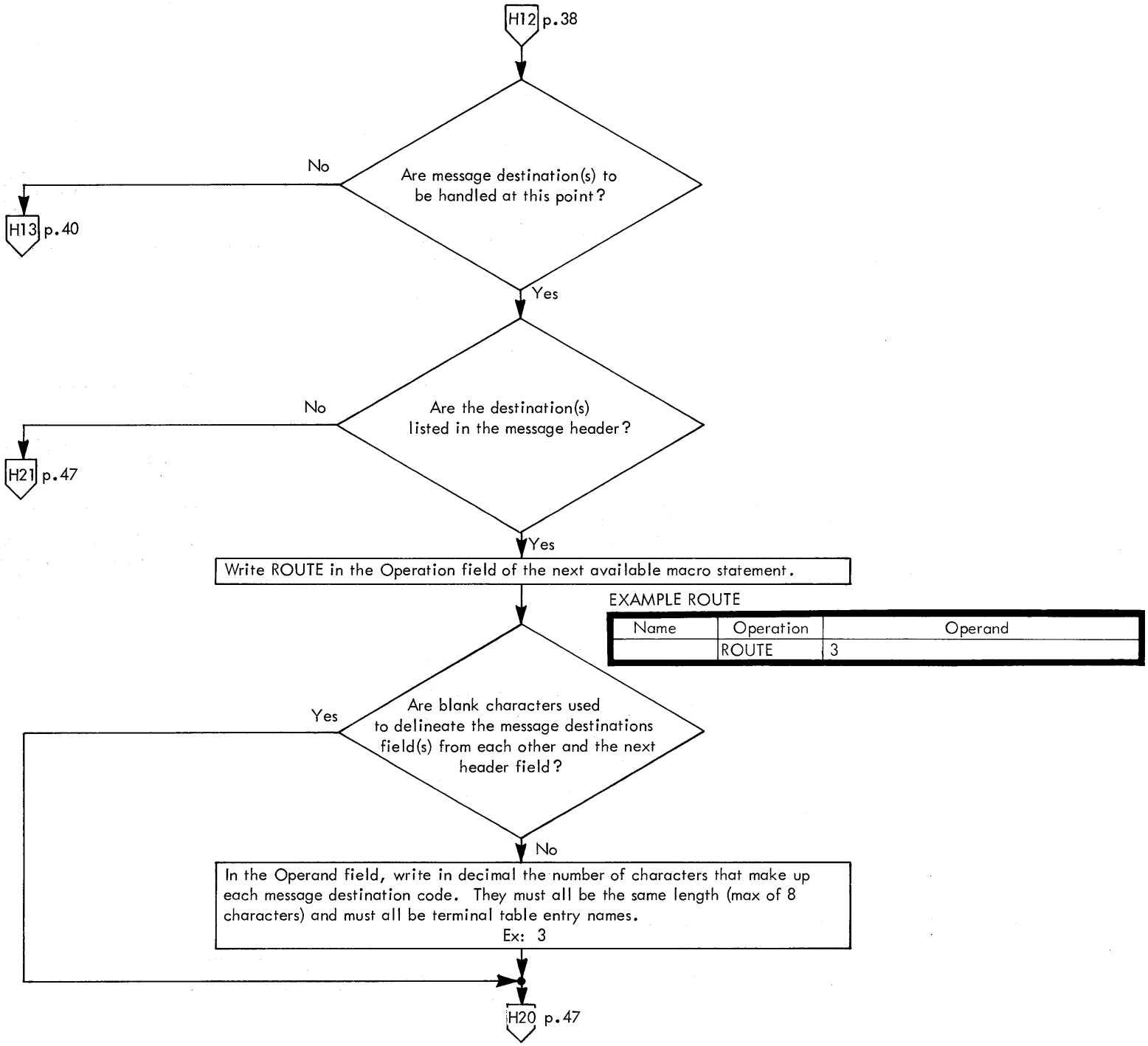

Is a user routine desired for these messages?
No
Yes
Write MODE in the Operation field of the next available macro statement.
Write in the Operand field the symbolic name of the user provided routine.
The MODE macro provides the exit to the user routine and the return address in General Register 14. General Registers 2,3,9,10,12,15 may be used in the user routine and General Register 1 must be used as a base register.
H9 pp.49,50
Is the LPS now aligned at a point in the message header at which a timestamp is to be inserted?
Yes
H17 p.45
No
H10 p.37



EXAMPLE DATESTMP

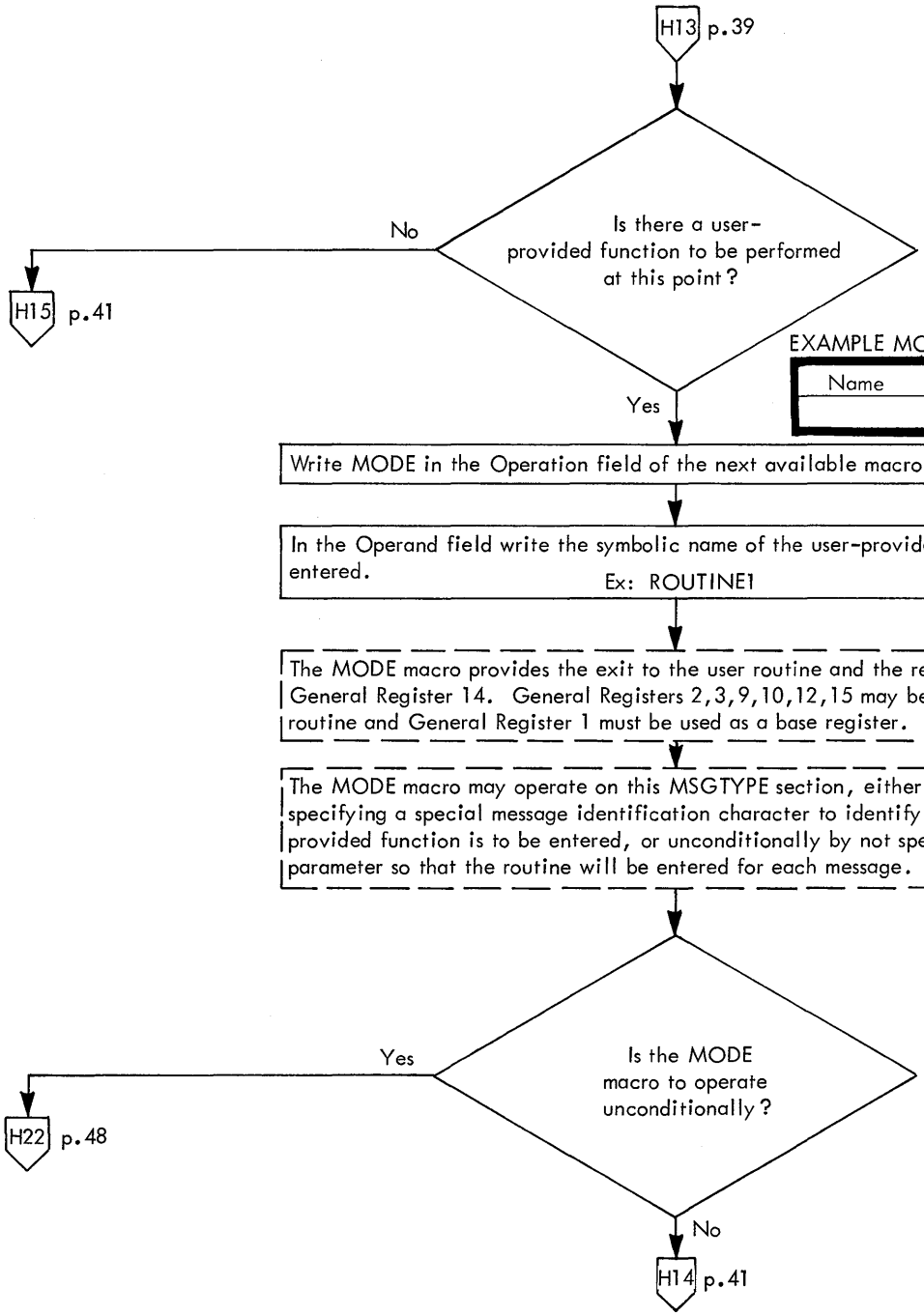
Name	Operation	Operand
	DATESTMP	


Write SEQIN in the Operation field of the next available macro statement.
EXAMPLE SEQIN
Name | Operation | Operand
SEQIN | 3
Is the input sequence number in the message header followed by a blank character(s)?
No
Yes
Write in the Operand field a decimal number for the length of the input sequence number field as found in the message header (4 max).
Ex: 3
The SEQIN field can then be of variable length. No parameter is needed in the Operand field of the SEQIN statement. The LPS will advance in the message header past the last character of the SEQIN field.
H11 p.37
H18 p.45
H22 p.48
Is there a source terminal to be checked at this point in the message header?
Yes
No
H19 p.46
H12 p.39



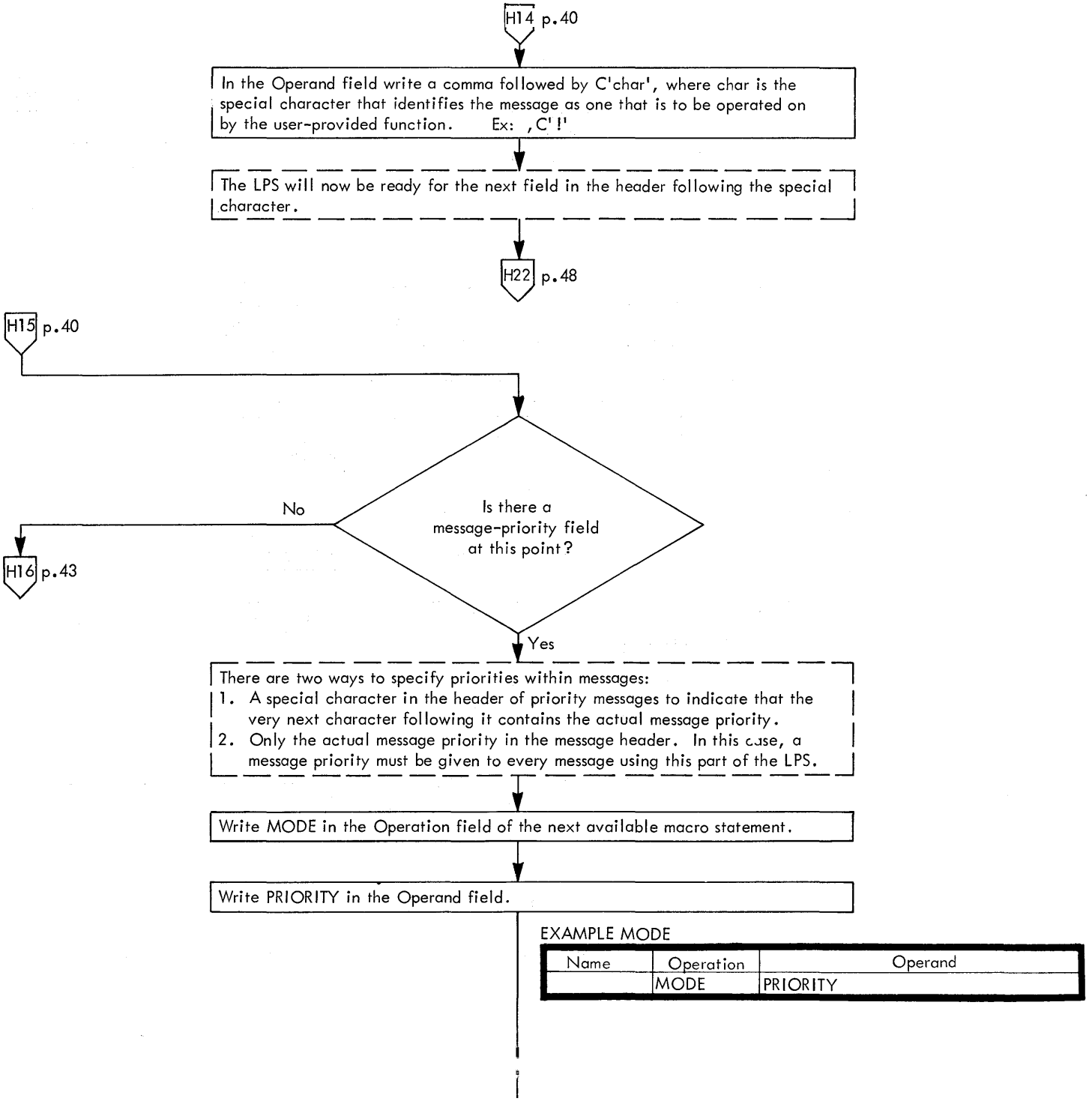
EXAMPLE ROUTE

Name	Operation	Operand
	ROUTE	3



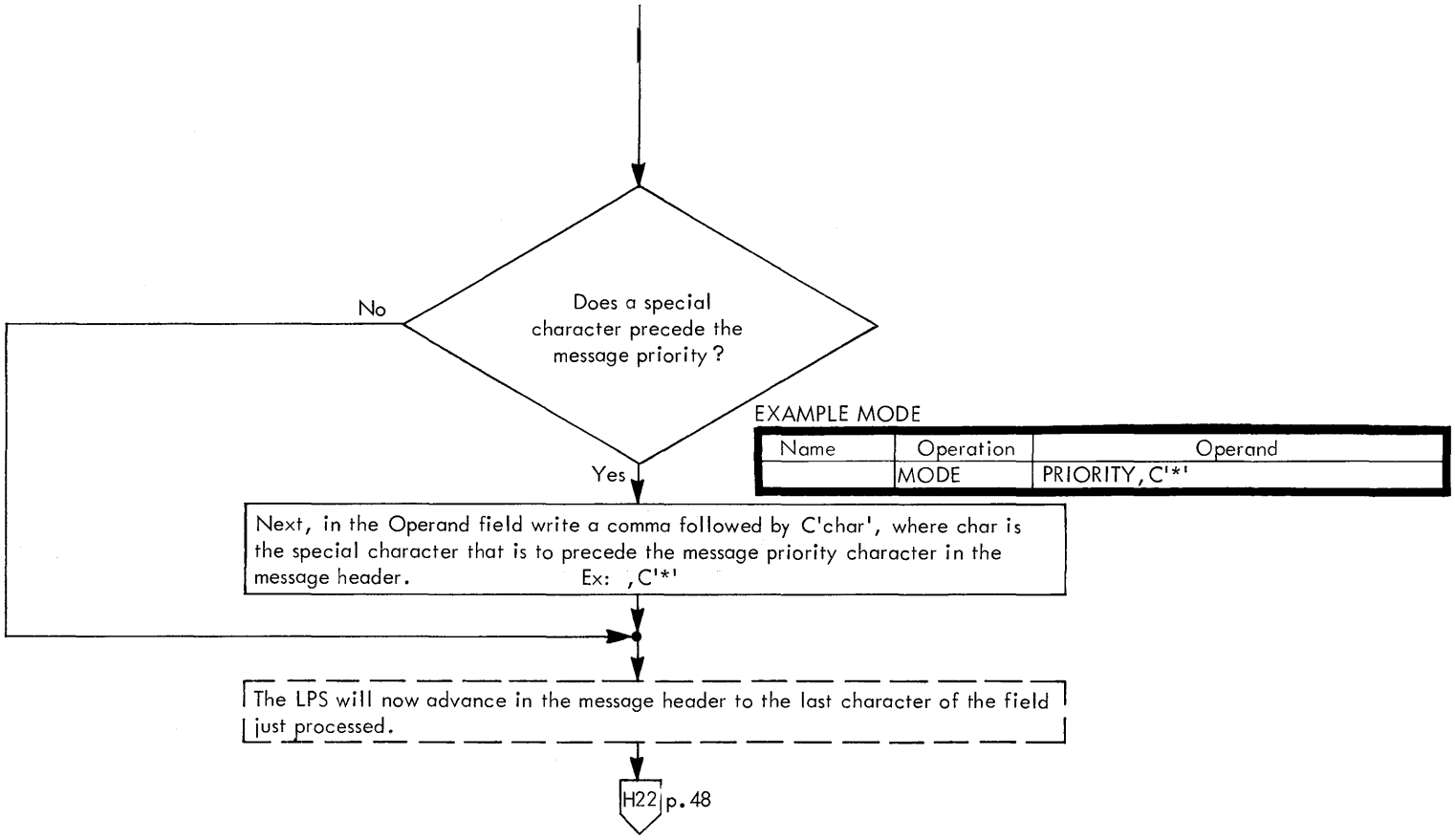
EXAMPLE MODE

Name	Operation	Operand
	MODE	ROUTINE,C'!'



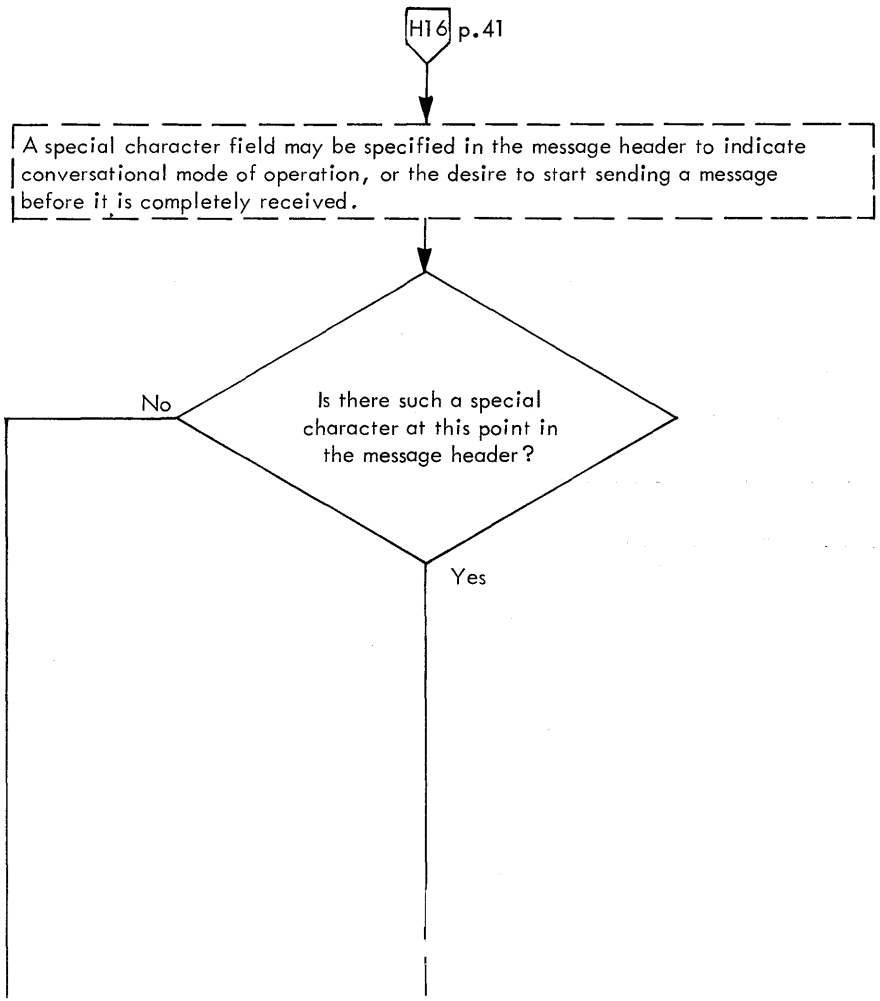
EXAMPLE MODE

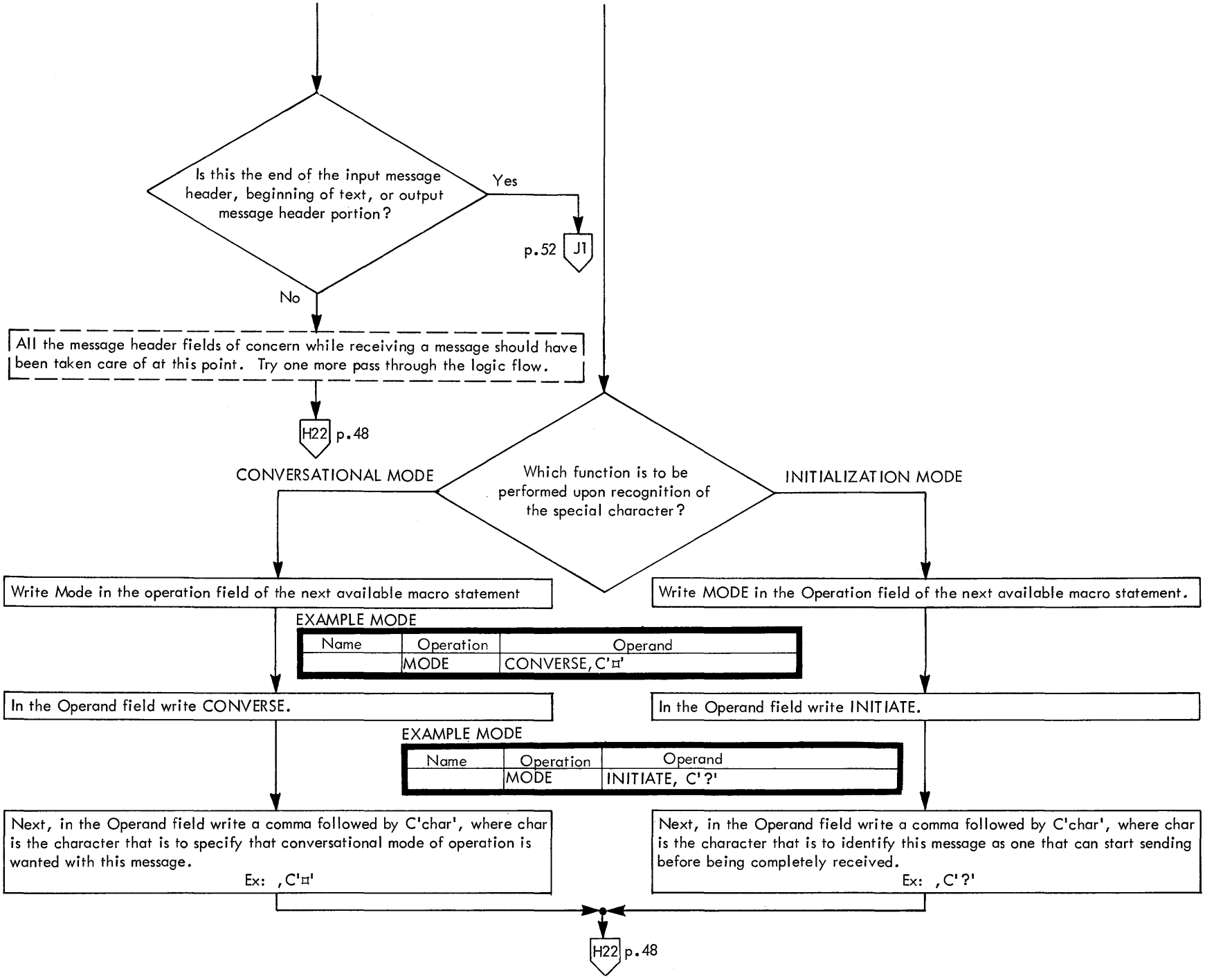
Name	Operation	Operand
	MODE	PRIORITY

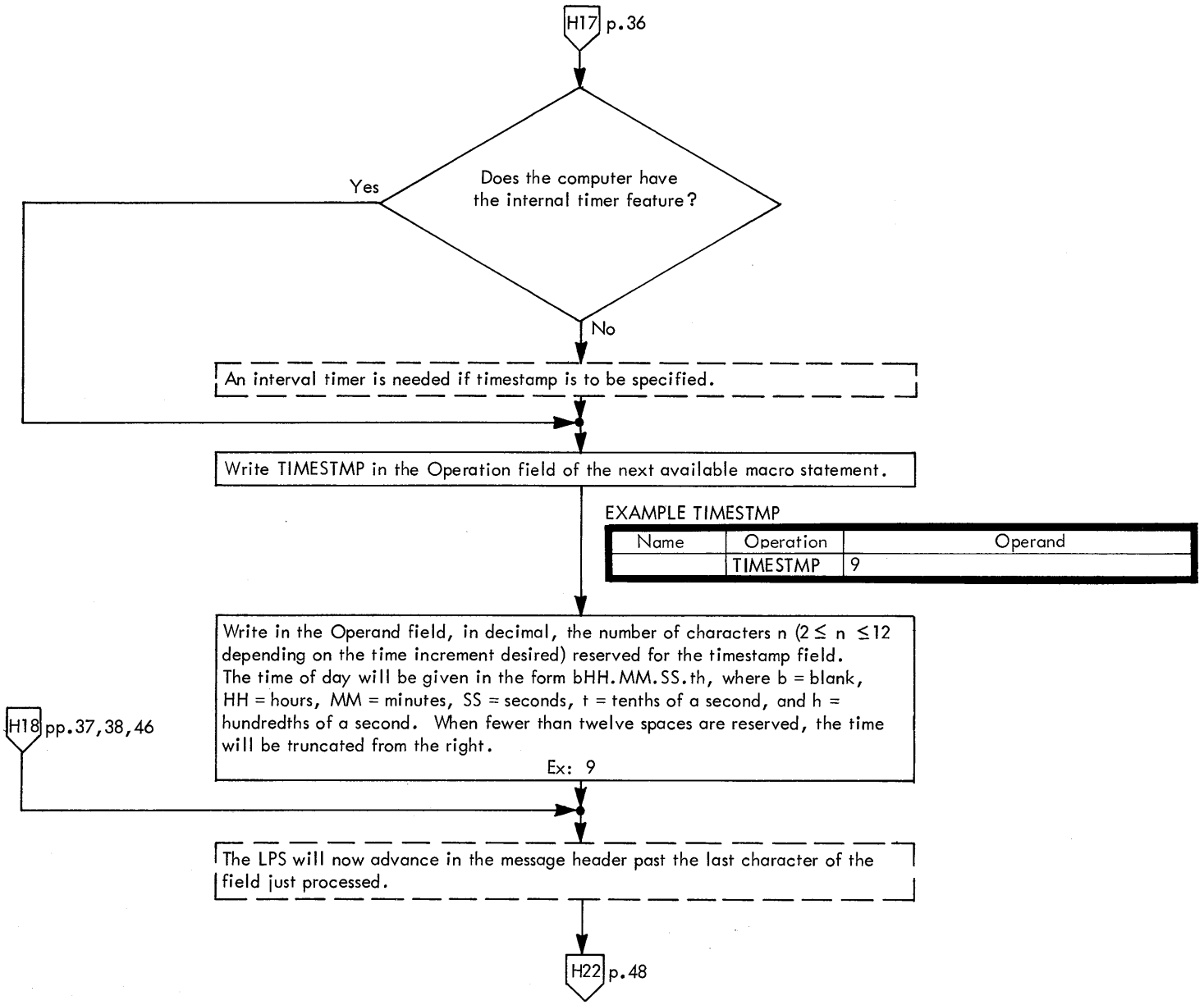


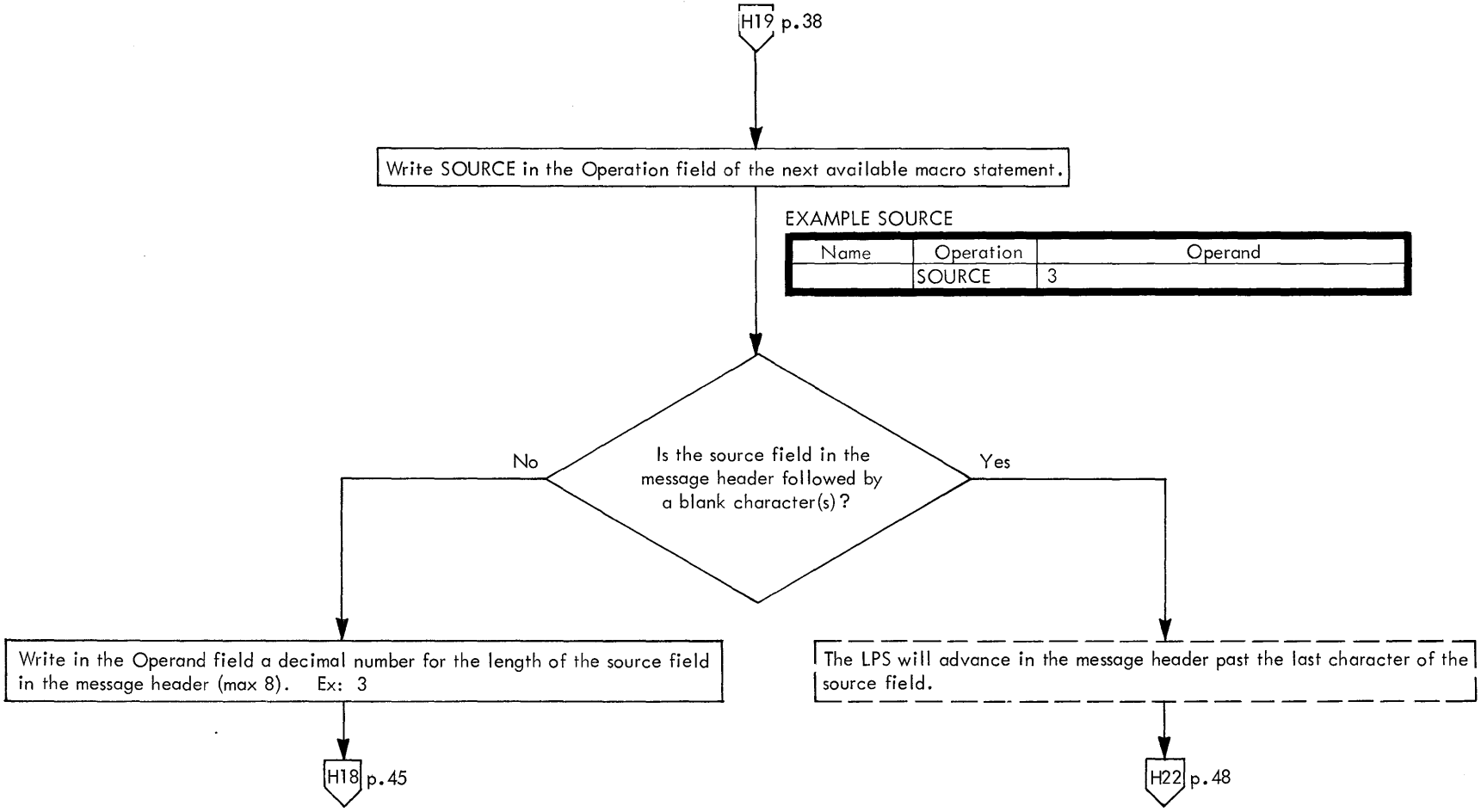
EXAMPLE MODE

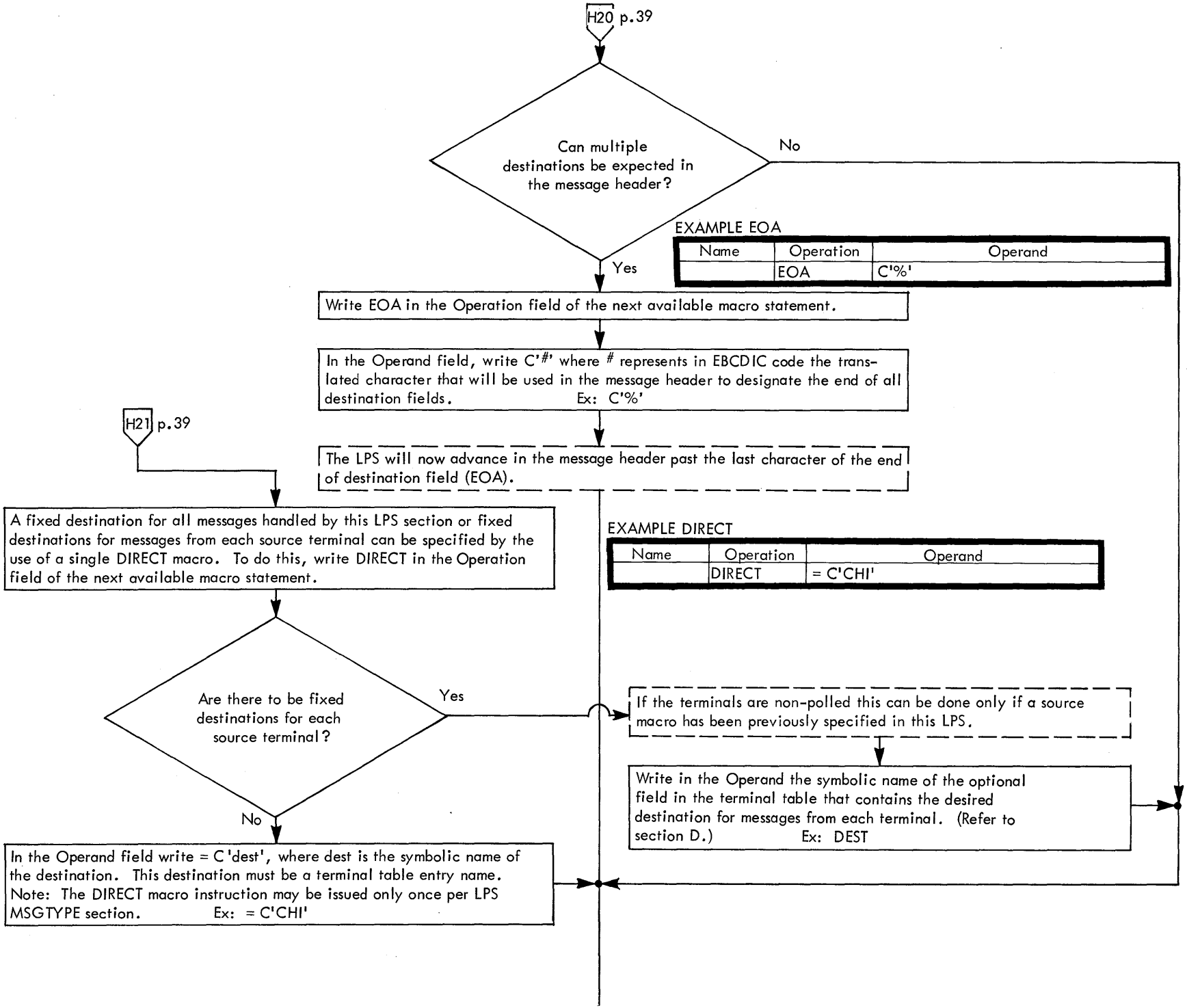
Name	Operation	Operand
	MODE	PRIORITY,C'*'

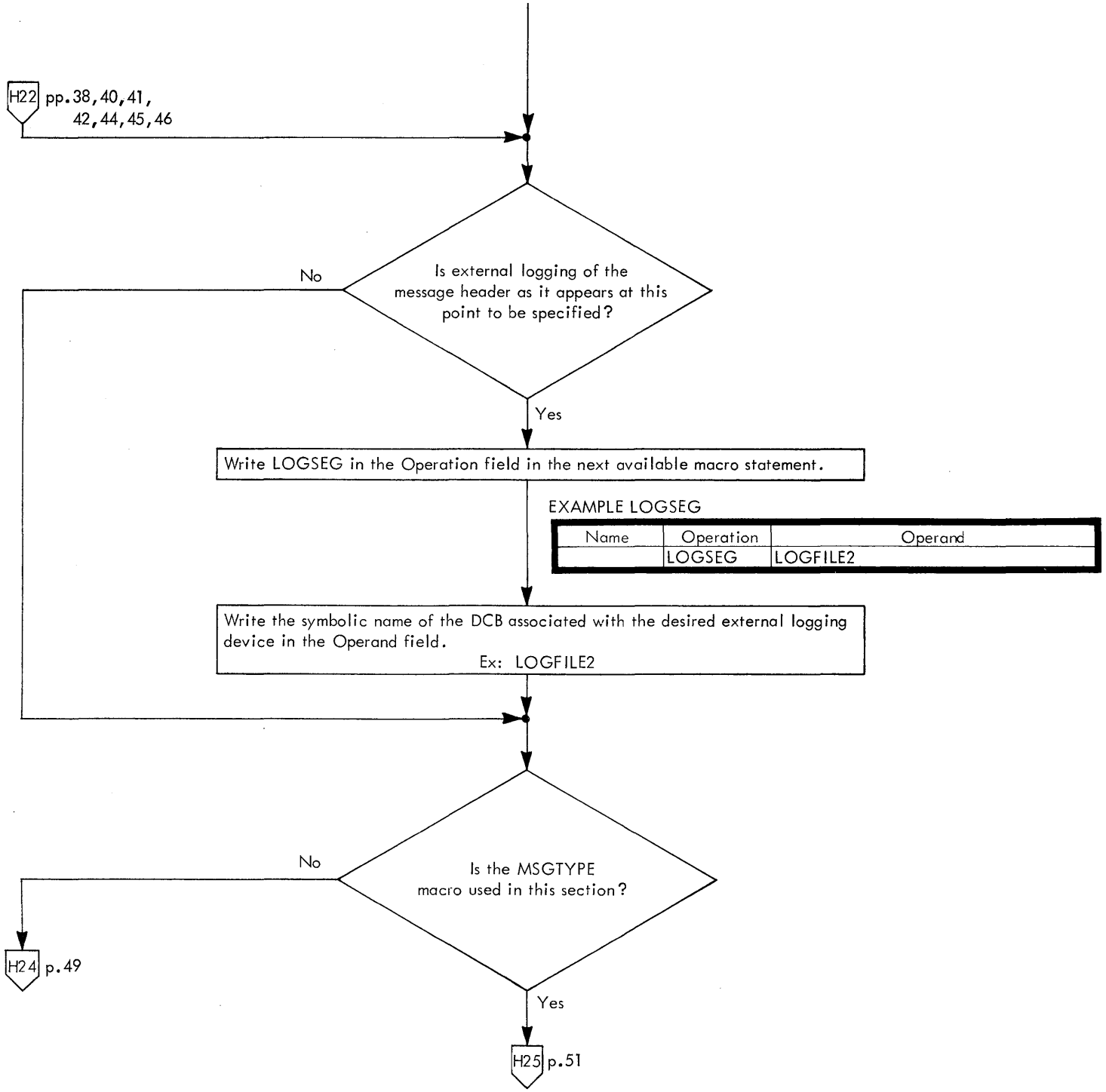

H16 p.41
A special character field may be specified in the message header to indicate conversational mode of operation, or the desire to start sending a message before it is completely received.
Is there such a special character at this point in the message header?
No
Yes

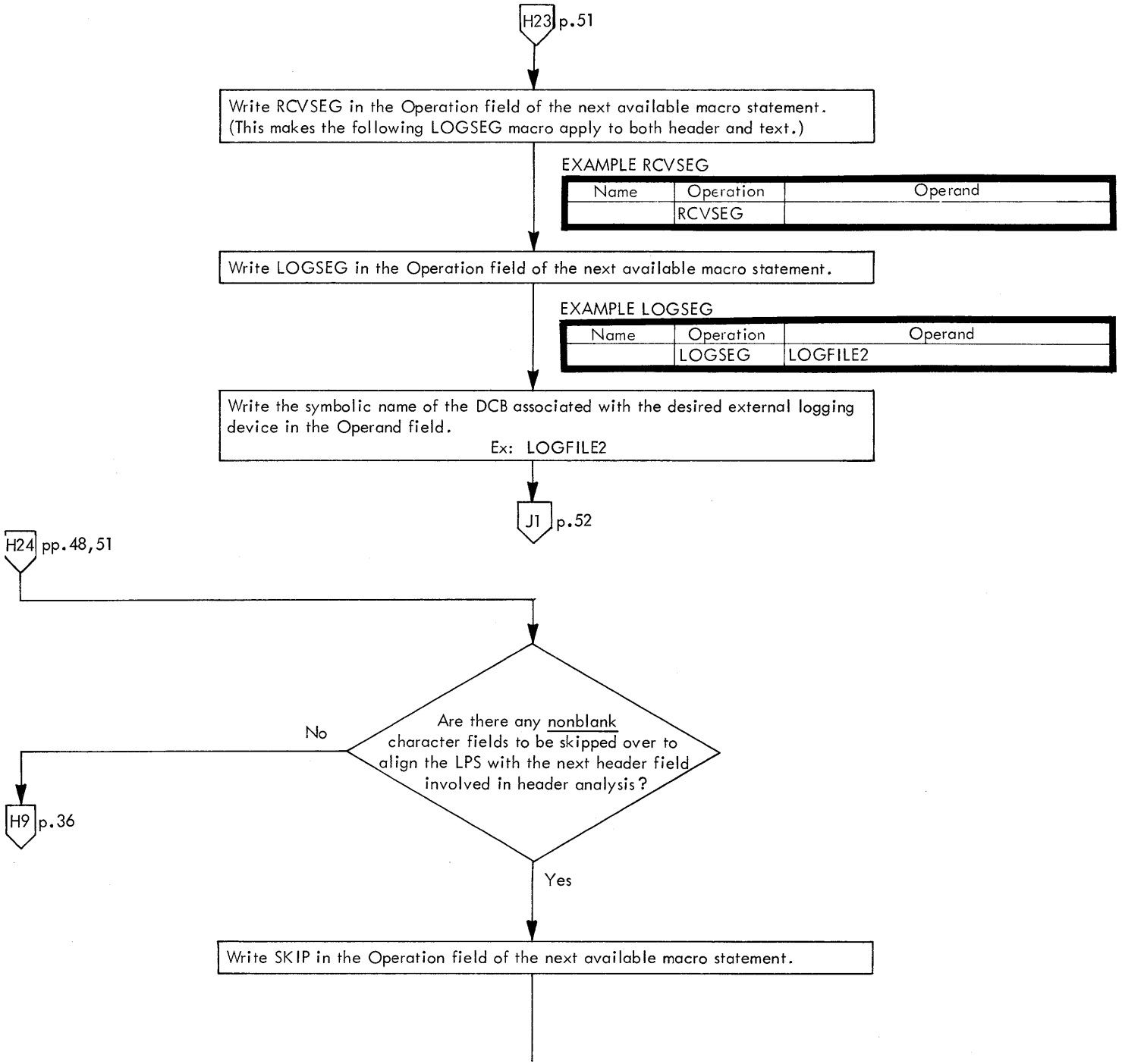

Is this the end of the input message header, beginning of text, or output message header portion?
Yes
p.52 J1
No
All the message header fields of concern while receiving a message should have been taken care of at this point. Try one more pass through the logic flow.
H22 p.48
Which function is to be performed upon recognition of the special character?
CONVERSATIONAL MODE
INITIALIZATION MODE
Write Mode in the operation field of the next available macro statement
Write MODE in the Operation field of the next available macro statement.
EXAMPLE MODE
Name | Operation | Operand
MODE | CONVERSE,C'¤'
In the Operand field write CONVERSE.
In the Operand field write INITIATE.
EXAMPLE MODE
Name | Operation | Operand
MODE | INITIATE, C'?'
Next, in the Operand field write a comma followed by C'char', where char is the character that is to specify that conversational mode of operation is wanted with this message.
Ex: ,C'¤'
Next, in the Operand field write a comma followed by C'char', where char is the character that is to identify this message as one that can start sending before being completely received.
Ex: ,C'?'
H22 p.48


H17 p.36
Does the computer have the internal timer feature?
Yes
No
An interval timer is needed if timestamp is to be specified.
Write TIMESTMP in the Operation field of the next available macro statement.
EXAMPLE TIMESTMP
Name | Operation | Operand
TIMESTMP | 9
Write in the Operand field, in decimal, the number of characters n (2 ≤ n ≤12 depending on the time increment desired) reserved for the timestamp field. The time of day will be given in the form bHH.MM.SS.th, where b = blank, HH = hours, MM = minutes, SS = seconds, t = tenths of a second, and h = hundredths of a second. When fewer than twelve spaces are reserved, the time will be truncated from the right.
Ex: 9
H18 pp.37,38,46
The LPS will now advance in the message header past the last character of the field just processed.
H22 p.48


H19 p.38
Write SOURCE in the Operation field of the next available macro statement.
EXAMPLE SOURCE
Name
Operation
Operand
SOURCE
3
Is the source field in the message header followed by a blank character(s)?
No
Yes
Write in the Operand field a decimal number for the length of the source field in the message header (max 8). Ex: 3
The LPS will advance in the message header past the last character of the source field.
H18 p.45
H22 p.48


H20 p.39
Can multiple destinations be expected in the message header?
No
Yes
EXAMPLE EOA
Name | Operation | Operand
EOA | C'%'
Write EOA in the Operation field of the next available macro statement.
In the Operand field, write C'#' where # represents in EBCDIC code the translated character that will be used in the message header to designate the end of all destination fields. Ex: C'%'
H21 p.39
The LPS will now advance in the message header past the last character of the end of destination field (EOA).
A fixed destination for all messages handled by this LPS section or fixed destinations for messages from each source terminal can be specified by the use of a single DIRECT macro. To do this, write DIRECT in the Operation field of the next available macro statement.
EXAMPLE DIRECT
Name | Operation | Operand
DIRECT | = C'CHI'
Are there to be fixed destinations for each source terminal?
Yes
No
If the terminals are non-polled this can be done only if a source macro has been previously specified in this LPS.
Write in the Operand the symbolic name of the optional field in the terminal table that contains the desired destination for messages from each terminal. (Refer to section D.) Ex: DEST
In the Operand field write = C'dest', where dest is the symbolic name of the destination. This destination must be a terminal table entry name. Note: The DIRECT macro instruction may be issued only once per LPS MSGTYPE section. Ex: = C'CHI'


H22 pp.38,40,41, 42,44,45,46
Is external logging of the message header as it appears at this point to be specified?
No
Yes
Write LOGSEG in the Operation field in the next available macro statement.
EXAMPLE LOGSEG
Name | Operation | Operand
LOGSEG | LOGFILE2
Write the symbolic name of the DCB associated with the desired external logging device in the Operand field.
Ex: LOGFILE2
Is the MSGTYPE macro used in this section?
No
Yes
H24 p.49
H25 p.51


H23 p.51
Write RCVSEG in the Operation field of the next available macro statement. (This makes the following LOGSEG macro apply to both header and text.)
EXAMPLE RCVSEG
Name | Operation | Operand
RCVSEG
Write LOGSEG in the Operation field of the next available macro statement.
EXAMPLE LOGSEG
Name | Operation | Operand
LOGSEG | LOGFILE2
Write the symbolic name of the DCB associated with the desired external logging device in the Operand field.
Ex: LOGFILE2
J1 p.52
H24 pp.48,51
Are there any nonblank character fields to be skipped over to align the LPS with the next header field involved in header analysis?
No
H9 p.36
Yes
Write SKIP in the Operation field of the next available macro statement.

EXAMPLE SKIP

Name	Operation	Operand
	SKIP	,C'#'

A field can be skipped by either specifying a given number of nonblank characters to be skipped or by specifying a particular character configuration that will indicate the end of the skipped field.

Is there a specified number of nonblank characters to be skipped?

No

Write C'chars' in the Operand field of the SKIP statement, where chars represents the particular character configuration in the internal system code.

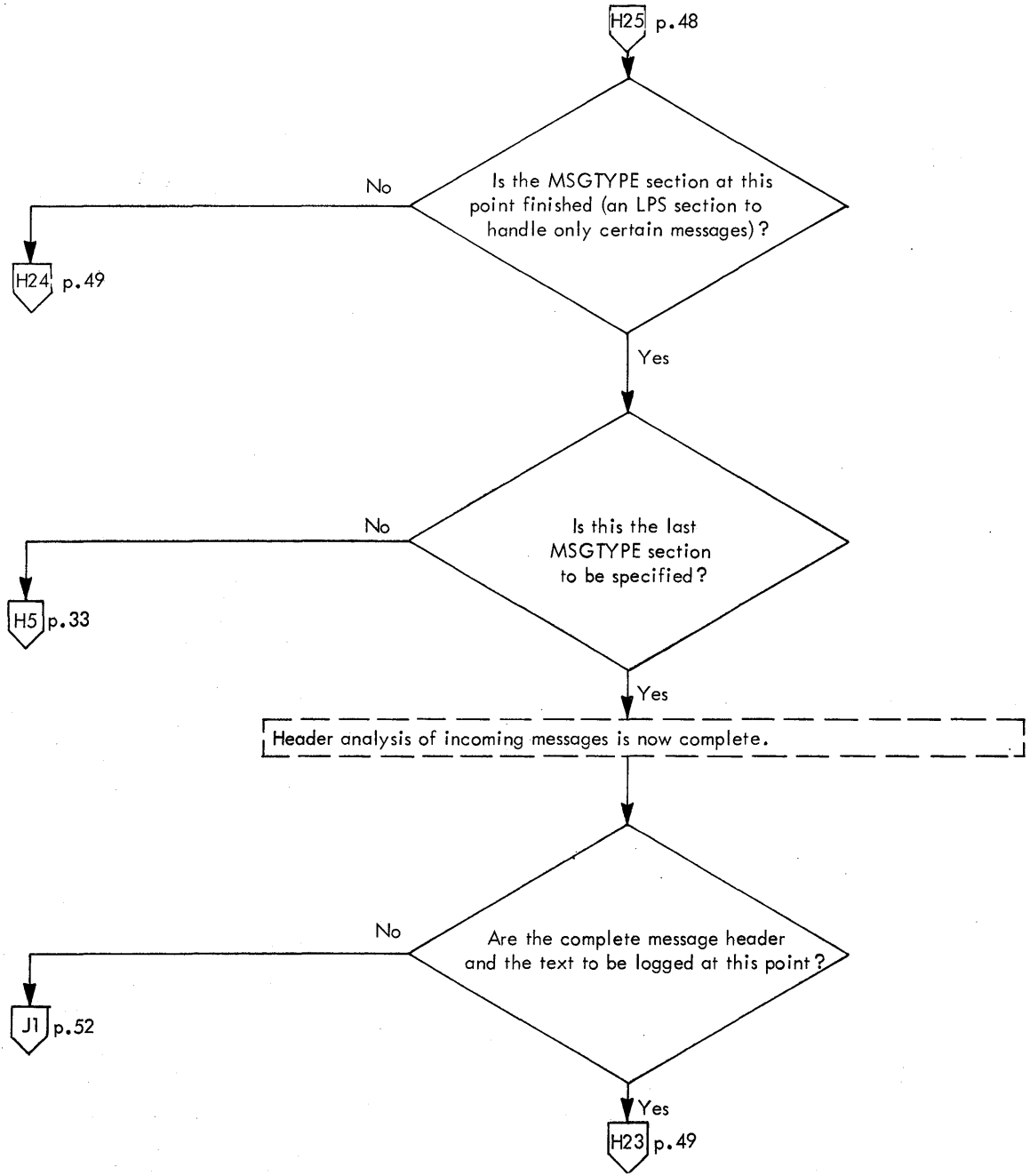
Yes

Write in the Operand field of the SKIP statement the actual number of nonblank characters to be skipped. The number cannot be greater than the number of character positions remaining in the buffer, past the present position of the LPS.
Ex: 10

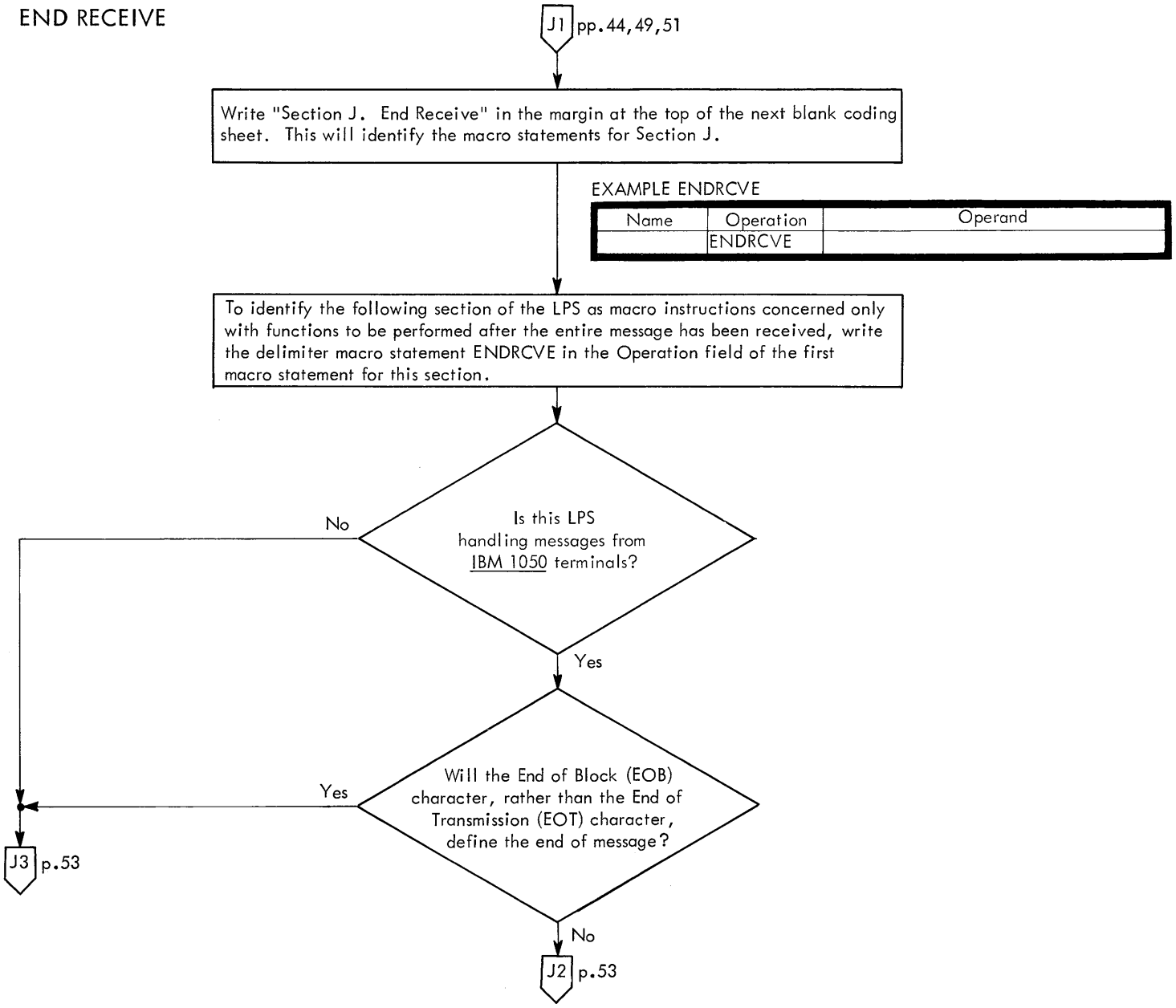
p.36 H9

EXAMPLE SKIP

Name	Operation	Operand
	SKIP	10


H25 p.48
Is the MSGTYPE section at this point finished (an LPS section to handle only certain messages)?
No
H24 p.49
Yes
Is this the last MSGTYPE section to be specified?
No
H5 p.33
Yes
Header analysis of incoming messages is now complete.
Are the complete message header and the text to be logged at this point?
No
J1 p.52
Yes
H23 p.49

SECTION J. END RECEIVE



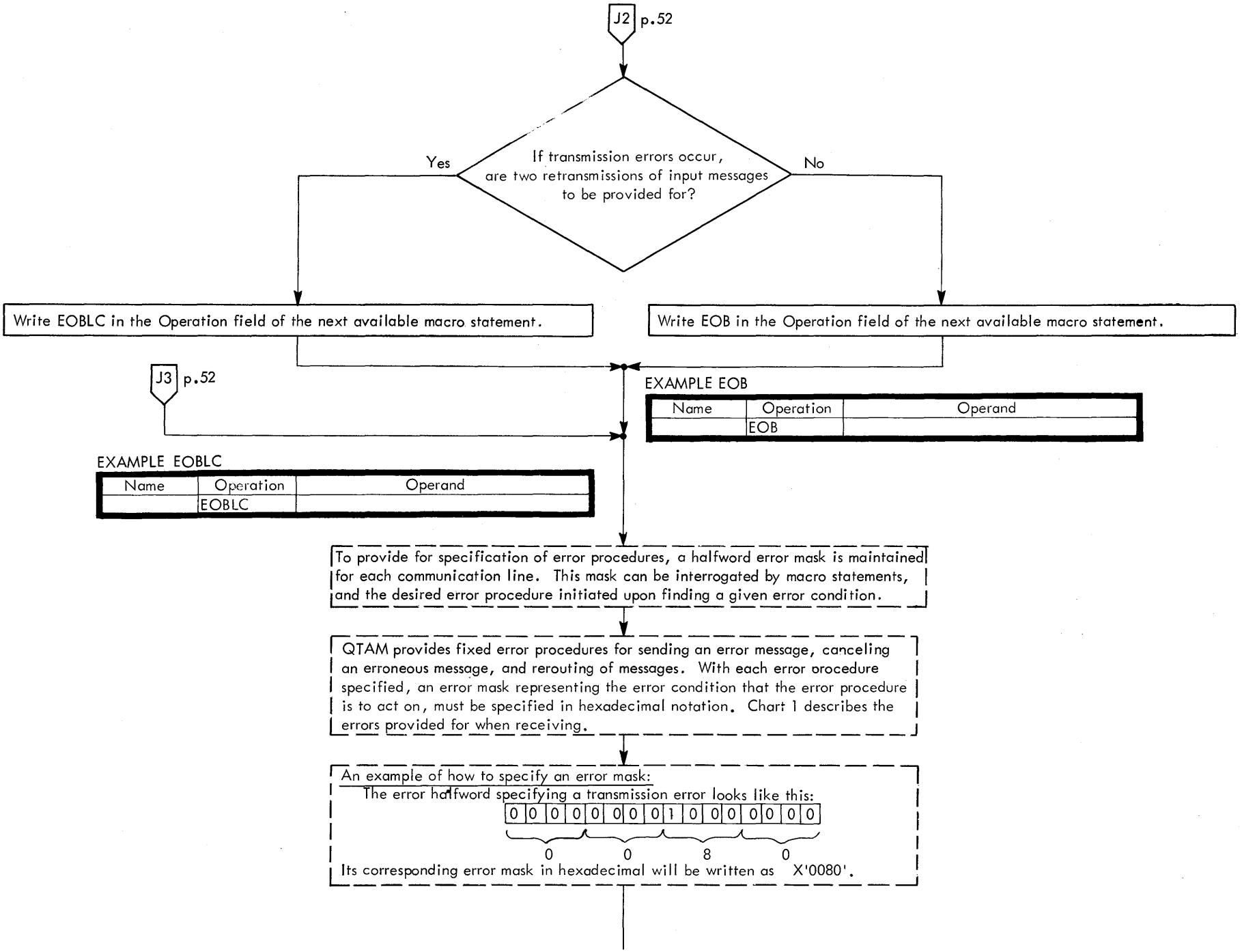

J2 p.52
If transmission errors occur, are two retransmissions of input messages to be provided for?
Yes
No
Write EOBLC in the Operation field of the next available macro statement.
Write EOB in the Operation field of the next available macro statement.
J3 p.52
EXAMPLE EOB
Name | Operation | Operand
EOB
EXAMPLE EOBLC
Name | Operation | Operand
EOBLC
To provide for specification of error procedures, a halfword error mask is maintained for each communication line. This mask can be interrogated by macro statements, and the desired error procedure initiated upon finding a given error condition.
QTAM provides fixed error procedures for sending an error message, canceling an erroneous message, and rerouting of messages. With each error procedure specified, an error mask representing the error condition that the error procedure is to act on, must be specified in hexadecimal notation. Chart 1 describes the errors provided for when receiving.
An example of how to specify an error mask:
The error halfword specifying a transmission error looks like this:
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 8 0
Its corresponding error mask in hexadecimal will be written as X'0080'.

Chart 1: Halfword Error Mask

Hexadecimal Error Mask Positions

Bit Positions	Hex 1				Hex 2				Hex 3				Hex 4			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Error Condition	Illegal Destination Code		Sequence Number High	Sequence Number Low	Incomplete Header	Illegal Source Code			Transmission Error	Time-Out	Breakoff	Insufficient Buffers			(Internally used by QTAM)	(Internally used by QTAM)

A 1-bit in any position indicates the presence of the error condition associated with that position.

Can an error condition be expected that will require sending an error message?

No → J9 p.57

Yes

J4 p.57 →

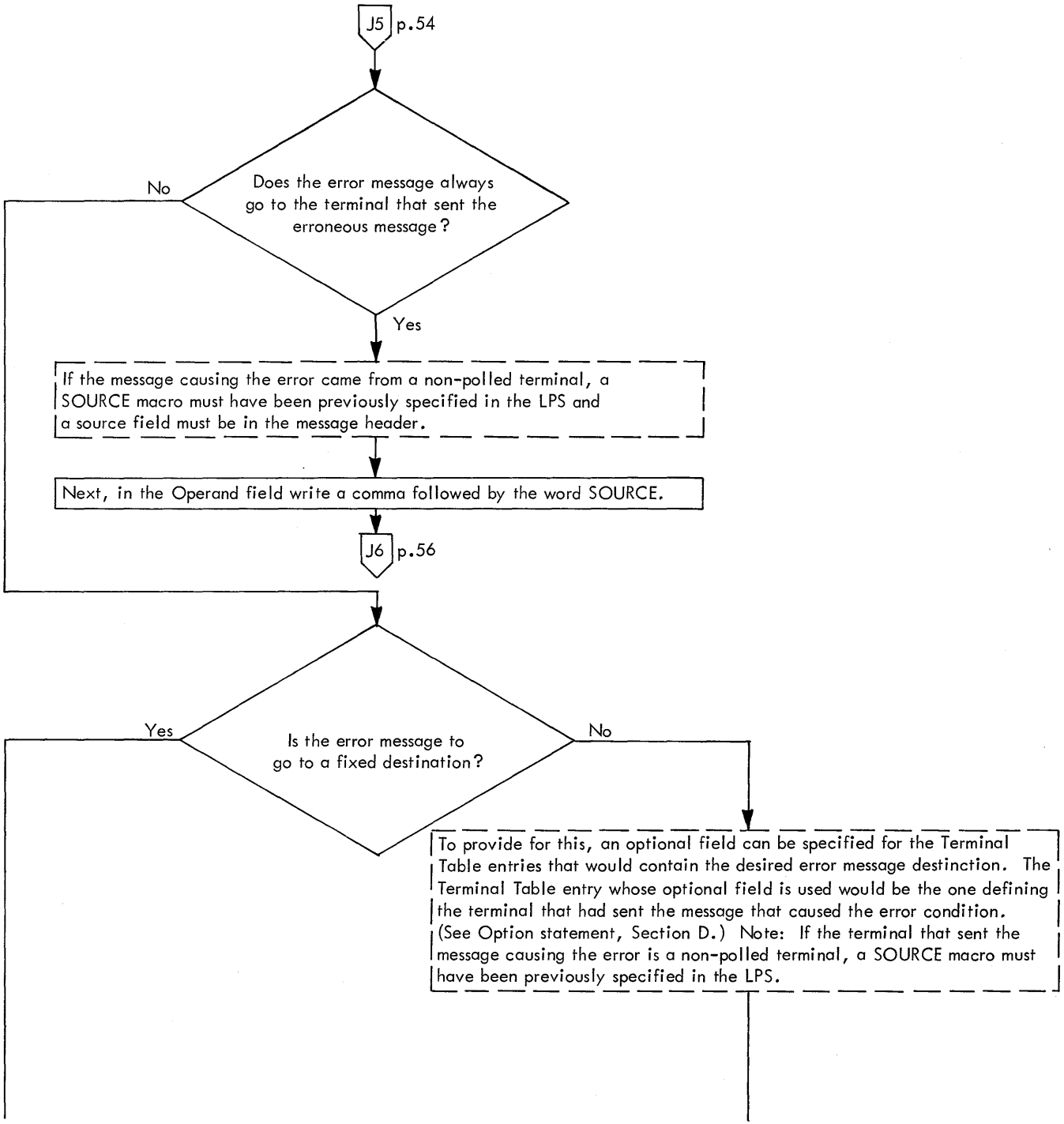
Write ERRMSG in the Operation field of the next available macro statement in this section.

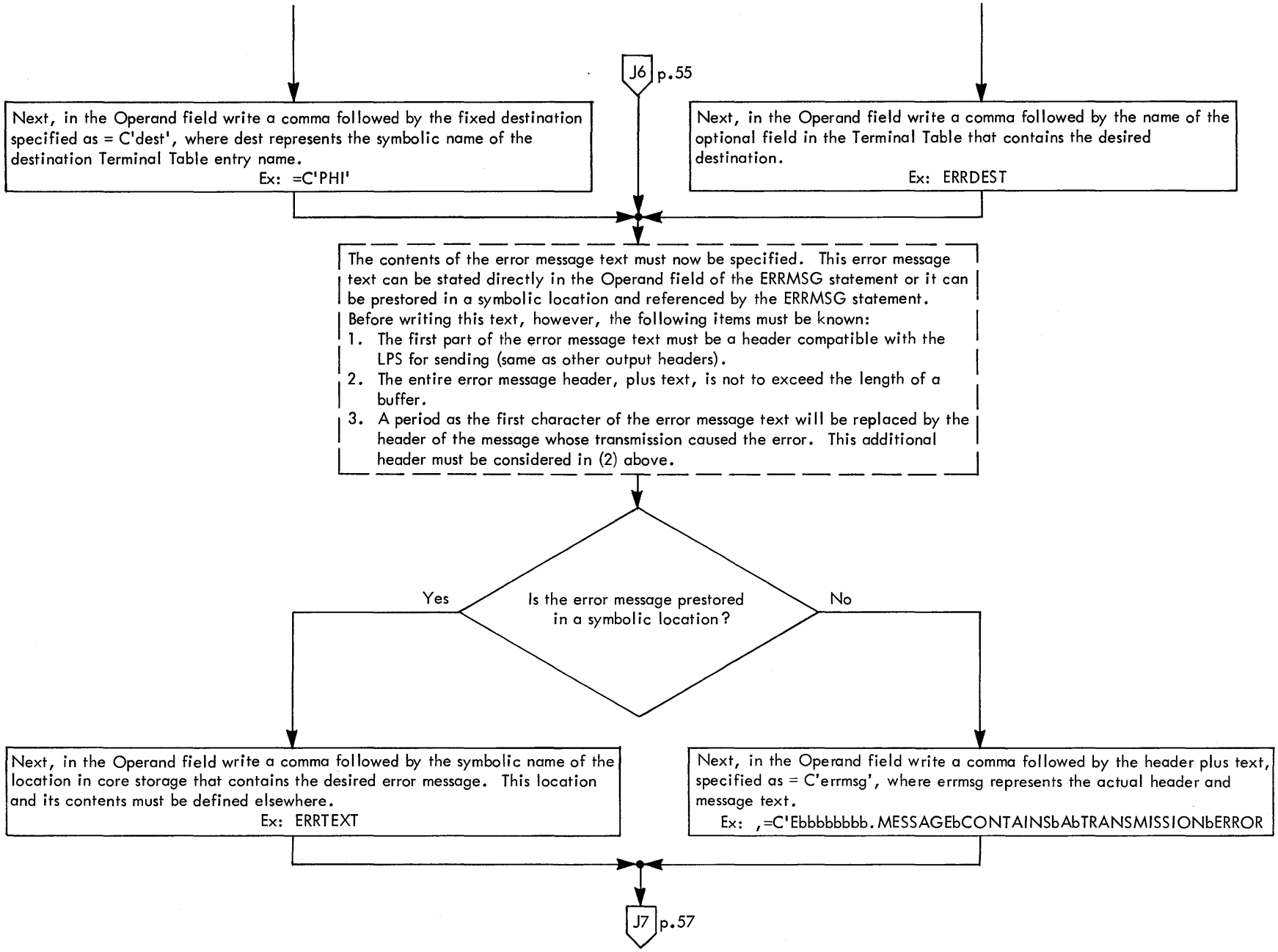
EXAMPLE ERRMSG

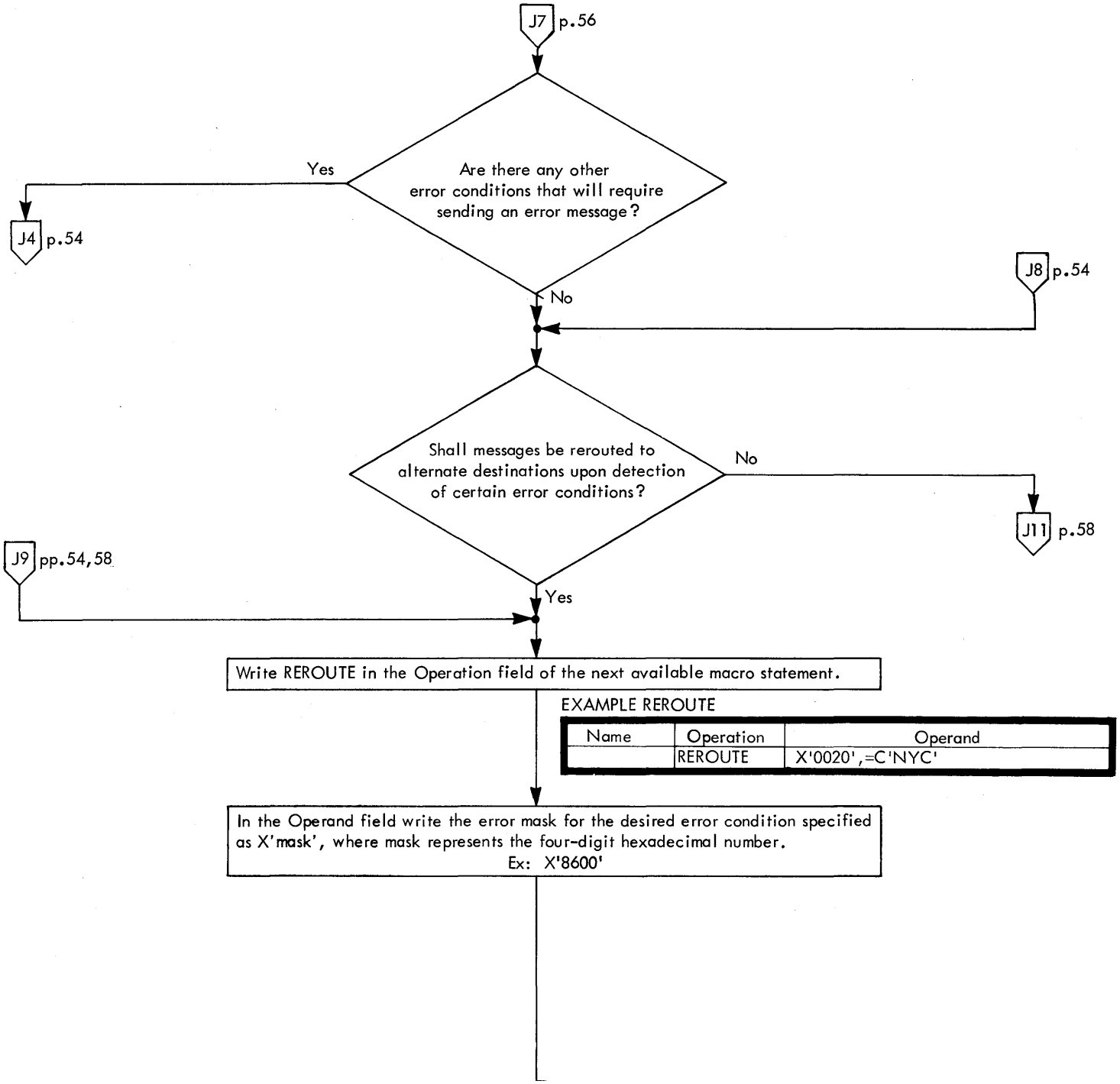
Name	Operation	Operand
	ERRMSG	X'0080', ERRDEST, =C'Ebbbbbbbbb.MESS
		AGEbCONTAINSbAbTRANSMISSIONb
		ERROR

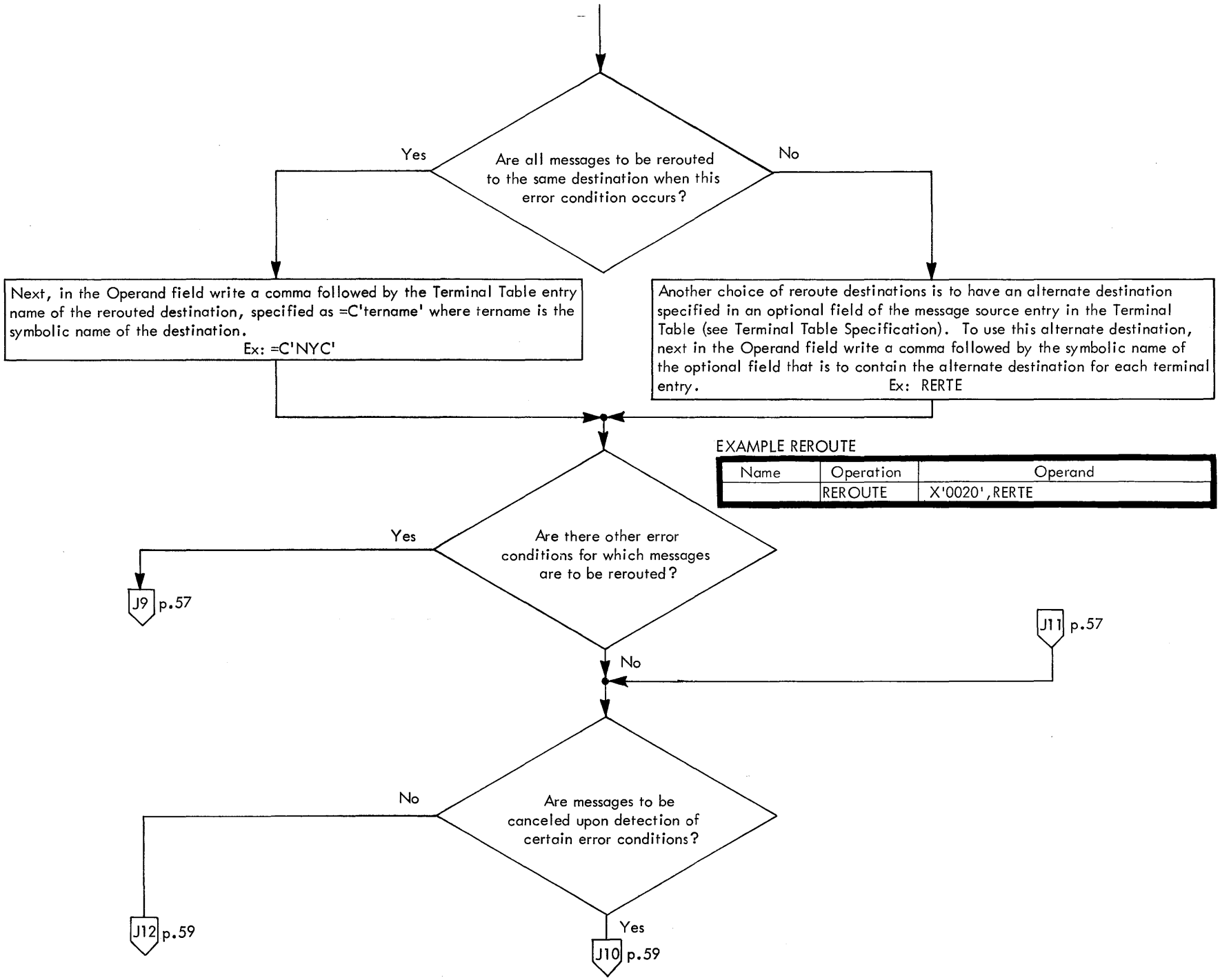
In the Operand field write the error mask of the error condition for which this message is to be sent, specified as X'mask' where mask is a four-digit hexadecimal number. Ex: X'0080'

J5 p.55


J5 p.54
Does the error message always go to the terminal that sent the erroneous message?
No
Yes
If the message causing the error came from a non-polled terminal, a SOURCE macro must have been previously specified in the LPS and a source field must be in the message header.
Next, in the Operand field write a comma followed by the word SOURCE.
J6 p.56
Is the error message to go to a fixed destination?
Yes
No
To provide for this, an optional field can be specified for the Terminal Table entries that would contain the desired error message destinction. The Terminal Table entry whose optional field is used would be the one defining the terminal that had sent the message that caused the error condition. (See Option statement, Section D.) Note: If the terminal that sent the message causing the error is a non-polled terminal, a SOURCE macro must have been previously specified in the LPS.

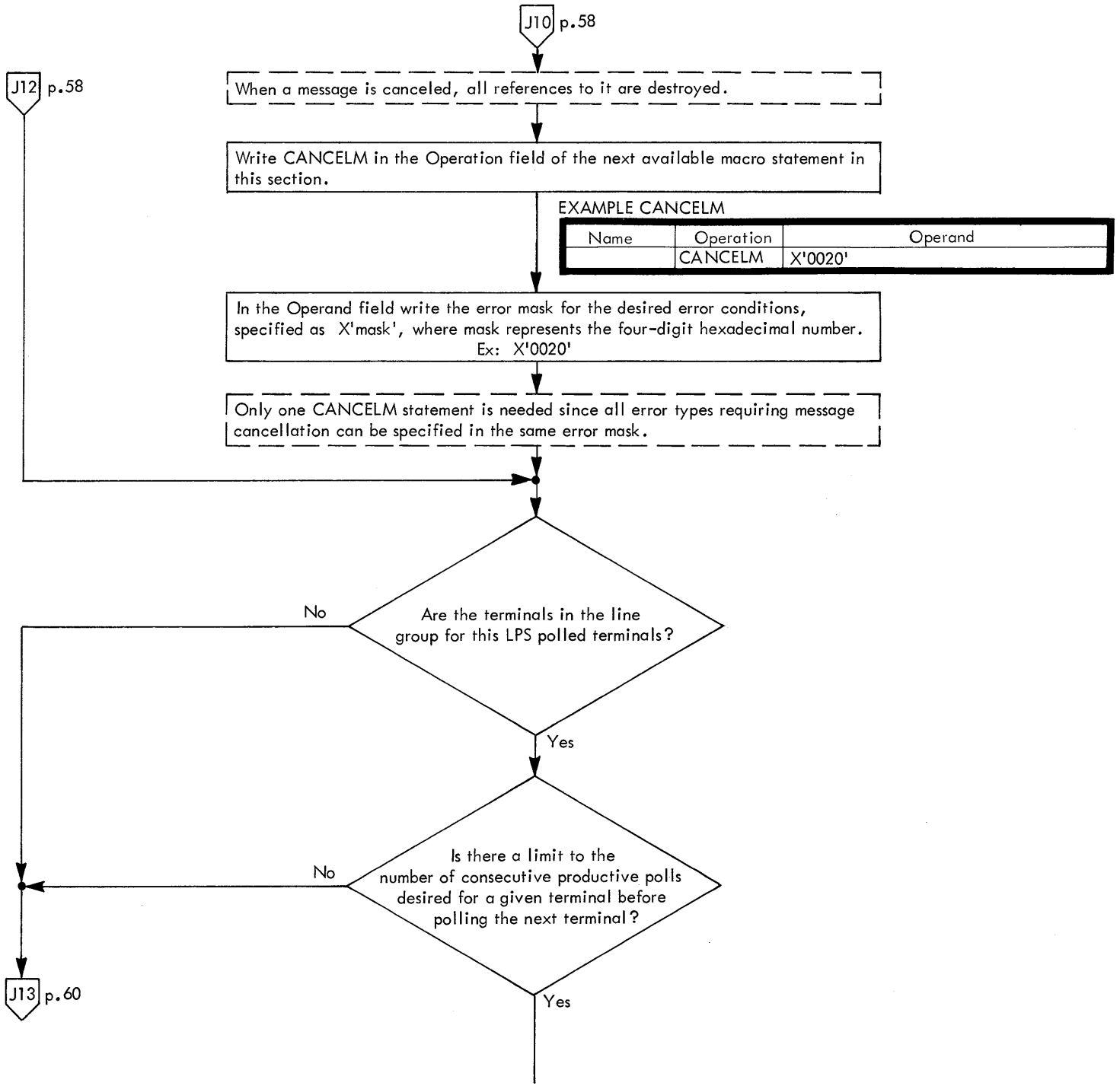

J6 p.55
Next, in the Operand field write a comma followed by the fixed destination specified as = C'dest', where dest represents the symbolic name of the destination Terminal Table entry name.
Ex: =C'PHI'
Next, in the Operand field write a comma followed by the name of the optional field in the Terminal Table that contains the desired destination.
Ex: ERRDEST
The contents of the error message text must now be specified. This error message text can be stated directly in the Operand field of the ERRMSG statement or it can be prestored in a symbolic location and referenced by the ERRMSG statement.
Before writing this text, however, the following items must be known:
1. The first part of the error message text must be a header compatible with the LPS for sending (same as other output headers).
2. The entire error message header, plus text, is not to exceed the length of a buffer.
3. A period as the first character of the error message text will be replaced by the header of the message whose transmission caused the error. This additional header must be considered in (2) above.
Is the error message prestored in a symbolic location?
Yes
No
Next, in the Operand field write a comma followed by the symbolic name of the location in core storage that contains the desired error message. This location and its contents must be defined elsewhere.
Ex: ERRTEXT
Next, in the Operand field write a comma followed by the header plus text, specified as = C'errmsg', where errmsg represents the actual header and message text.
Ex: ,=C'Ebbbbbbbb.MESSAGEbCONTAINSbAbTRANSMISSIONbERROR
J7 p.57

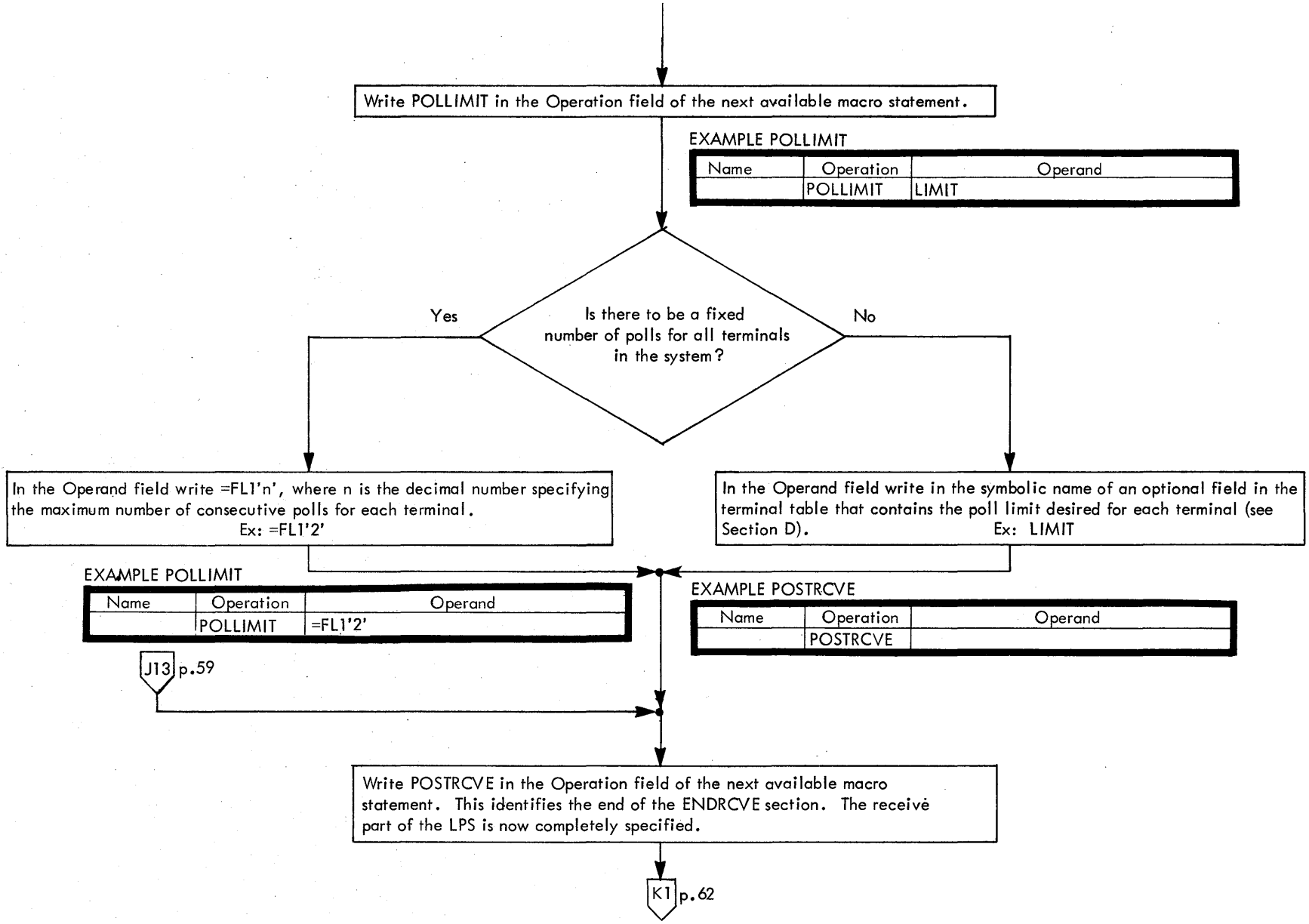

J7 p.56
Are there any other error conditions that will require sending an error message?
Yes
J4 p.54
No
J8 p.54
Shall messages be rerouted to alternate destinations upon detection of certain error conditions?
No
J11 p.58
J9 pp.54,58
Yes
Write REROUTE in the Operation field of the next available macro statement.
EXAMPLE REROUTE
Name | Operation | Operand
REROUTE | X'0020',=C'NYC'
In the Operand field write the error mask for the desired error condition specified as X'mask', where mask represents the four-digit hexadecimal number.
Ex: X'8600'



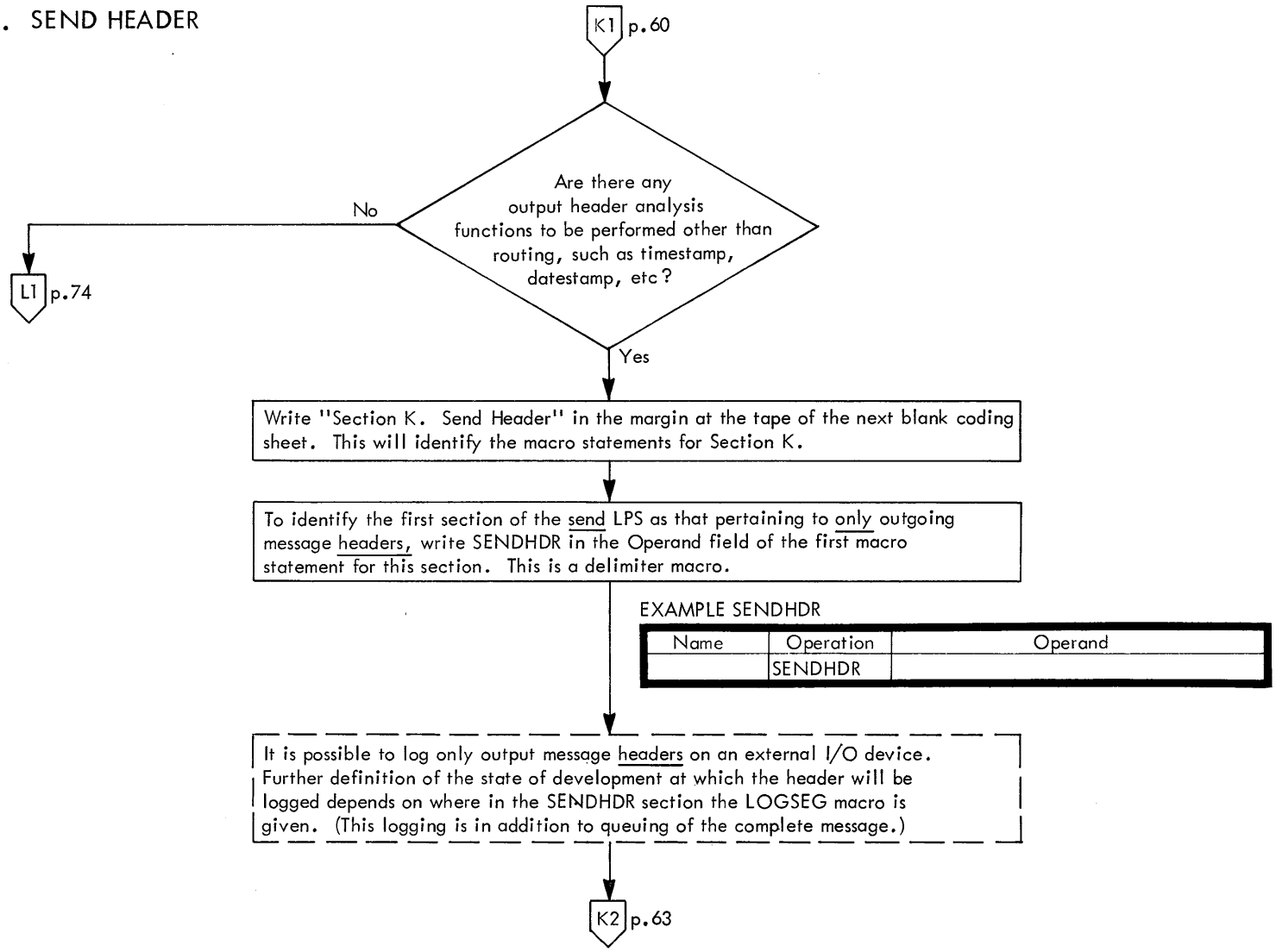
EXAMPLE REROUTE

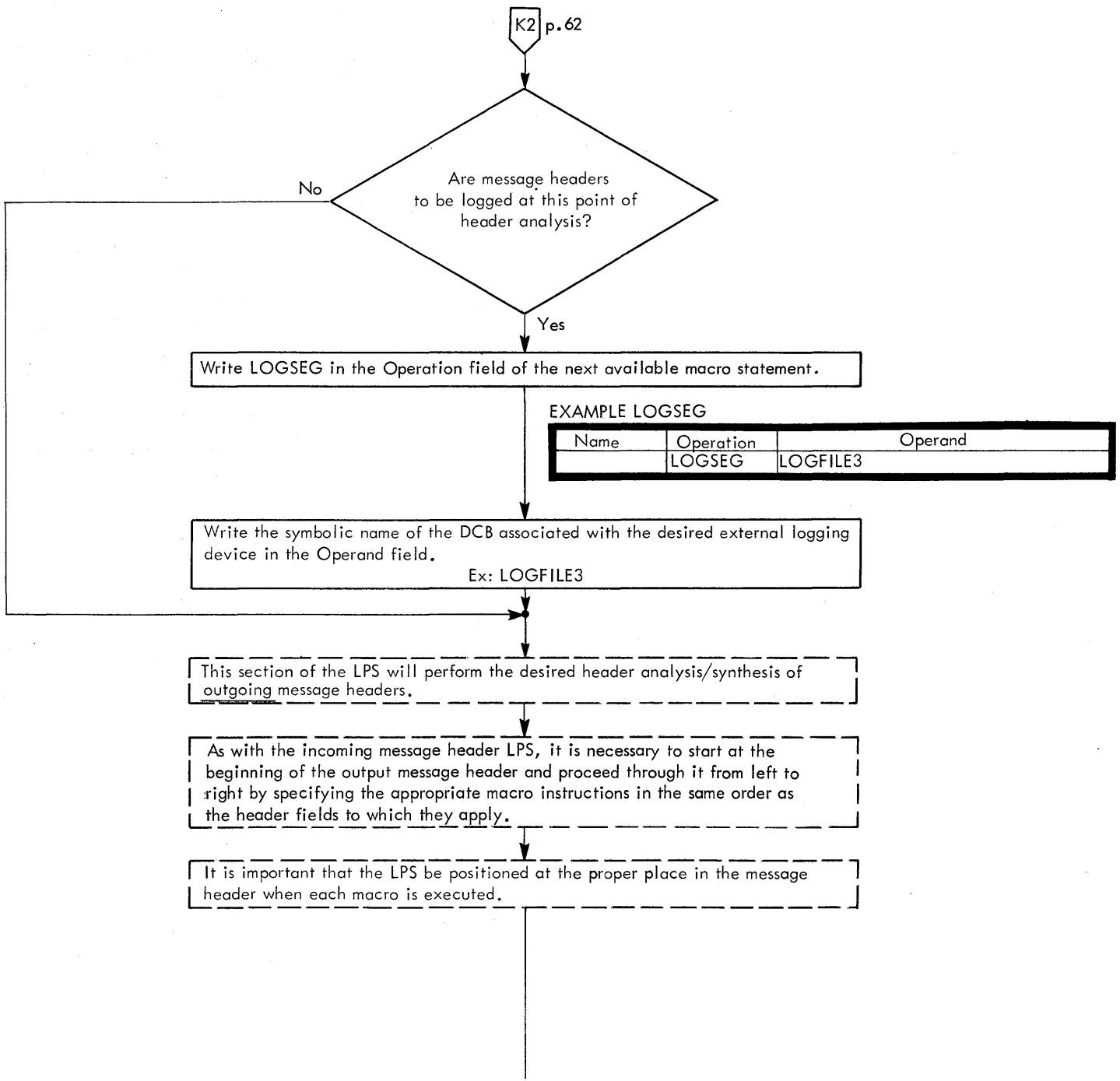
Name	Operation	Operand
	REROUTE	X'0020',RERTE

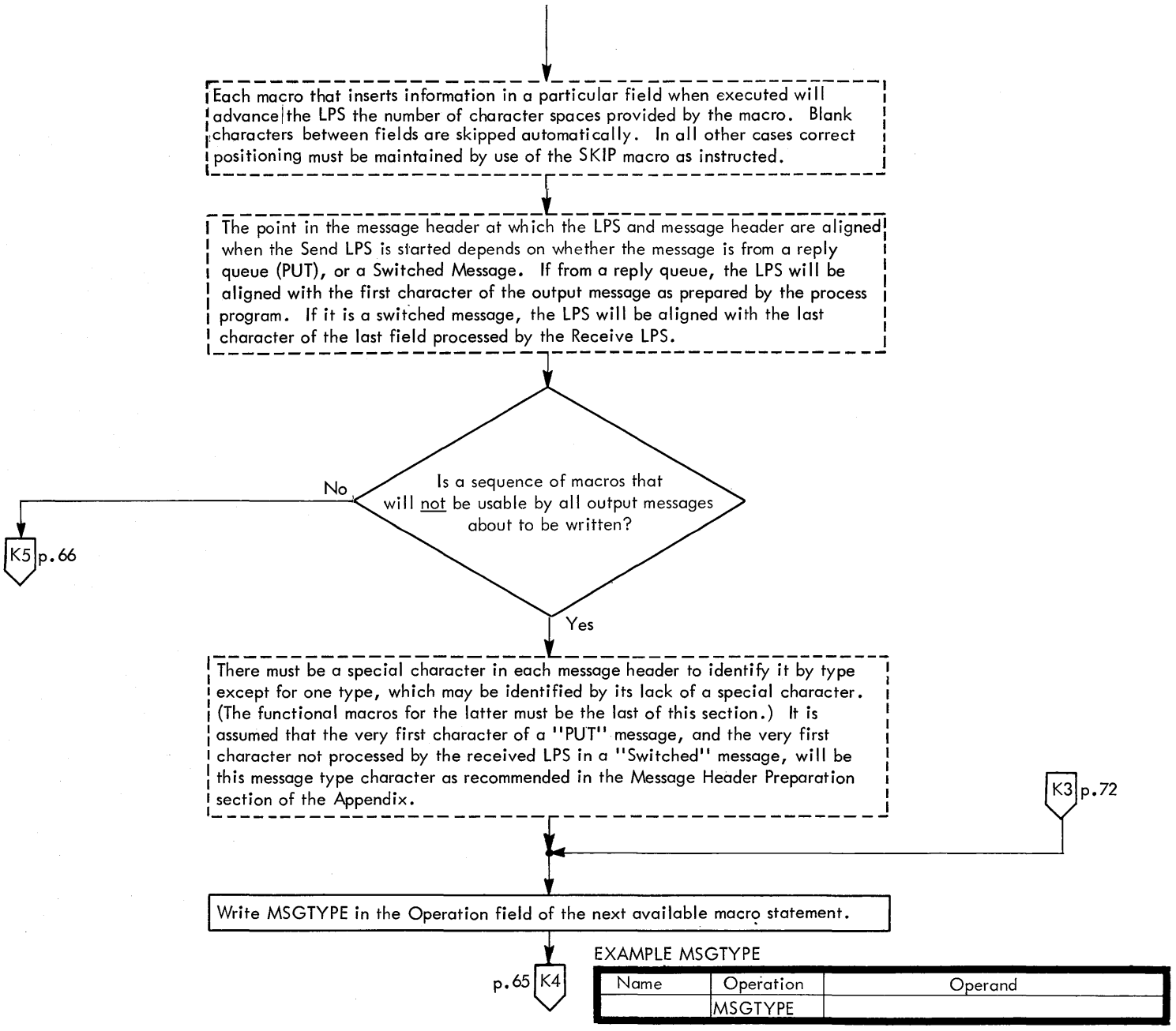

J10 p.58
J12 p.58
When a message is canceled, all references to it are destroyed.
Write CANCELM in the Operation field of the next available macro statement in this section.
EXAMPLE CANCELM
Name
Operation
Operand
CANCELM
X'0020'
In the Operand field write the error mask for the desired error conditions, specified as X'mask', where mask represents the four-digit hexadecimal number.
Ex: X'0020'
Only one CANCELM statement is needed since all error types requiring message cancellation can be specified in the same error mask.
Are the terminals in the line group for this LPS polled terminals?
No
Yes
Is there a limit to the number of consecutive productive polls desired for a given terminal before polling the next terminal?
No
Yes
J13 p.60

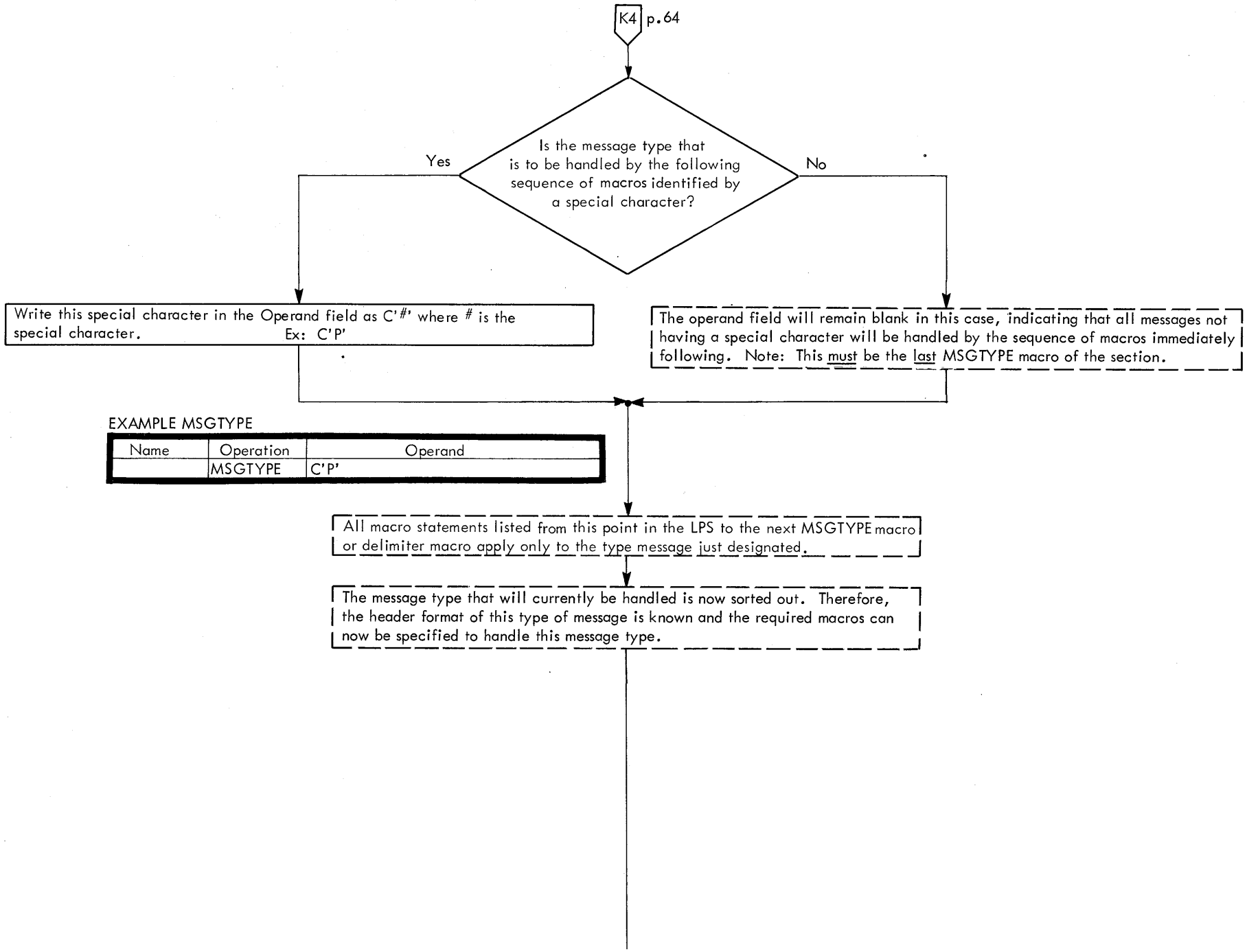

Write POLLIMIT in the Operation field of the next available macro statement.
EXAMPLE POLLIMIT
Name | Operation | Operand
POLLIMIT | LIMIT
Is there to be a fixed number of polls for all terminals in the system?
Yes
No
In the Operand field write =FL1'n', where n is the decimal number specifying the maximum number of consecutive polls for each terminal.
Ex: =FL1'2'
In the Operand field write in the symbolic name of an optional field in the terminal table that contains the poll limit desired for each terminal (see Section D).
Ex: LIMIT
EXAMPLE POLLIMIT
Name | Operation | Operand
POLLIMIT | =FL1'2'
EXAMPLE POSTRCVE
Name | Operation | Operand
POSTRCVE
J13 p.59
Write POSTRCVE in the Operation field of the next available macro statement. This identifies the end of the ENDRCVE section. The receive part of the LPS is now completely specified.
K1 p.62

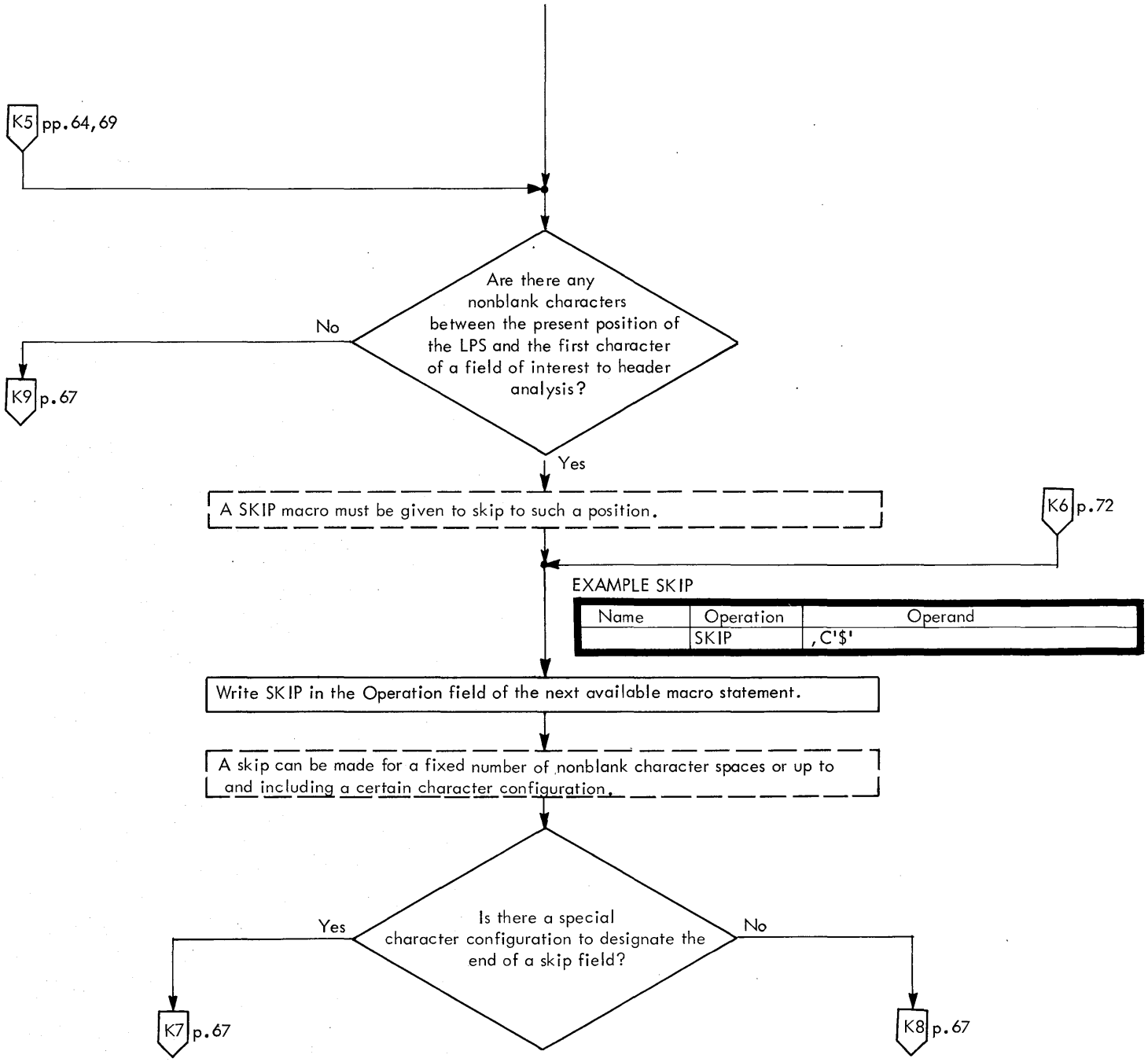
SECTION K. SEND HEADER

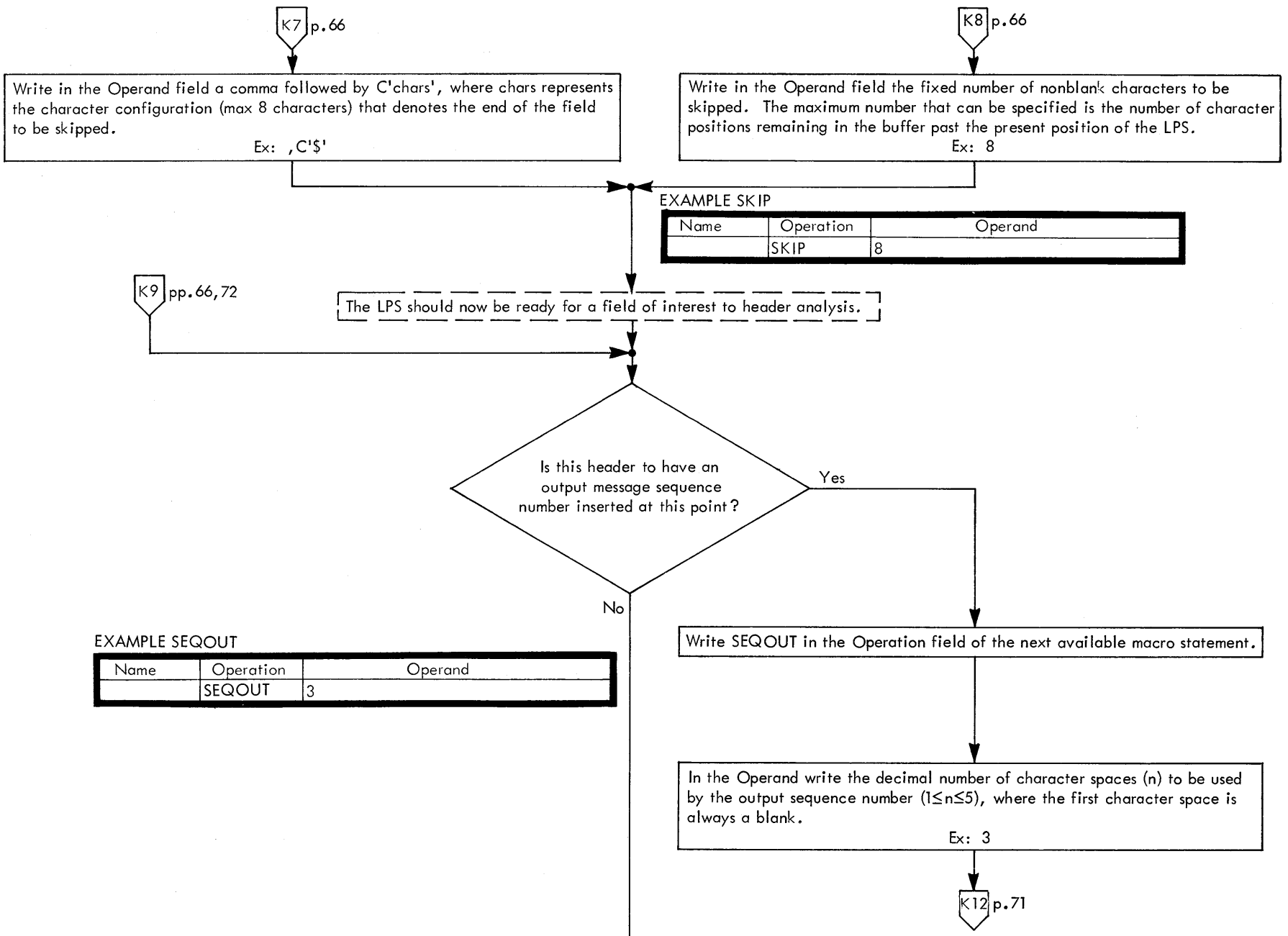


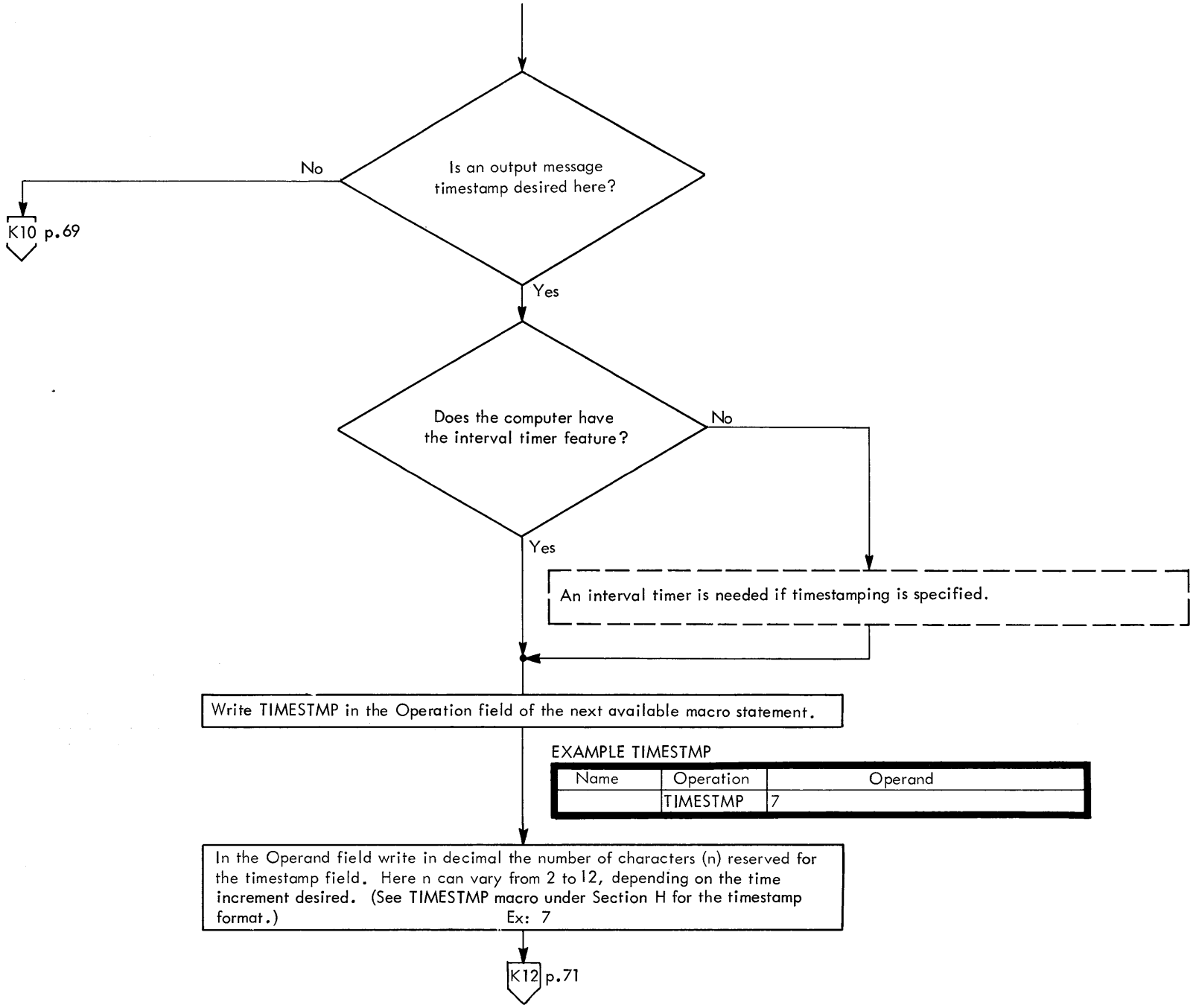

K2 p.62
Are message headers to be logged at this point of header analysis?
No
Yes
Write LOGSEG in the Operation field of the next available macro statement.
EXAMPLE LOGSEG
Name | Operation | Operand
LOGSEG | LOGFILE3
Write the symbolic name of the DCB associated with the desired external logging device in the Operand field.
Ex: LOGFILE3
This section of the LPS will perform the desired header analysis/synthesis of outgoing message headers.
As with the incoming message header LPS, it is necessary to start at the beginning of the output message header and proceed through it from left to right by specifying the appropriate macro instructions in the same order as the header fields to which they apply.
It is important that the LPS be positioned at the proper place in the message header when each macro is executed.


Each macro that inserts information in a particular field when executed will advance the LPS the number of character spaces provided by the macro. Blank characters between fields are skipped automatically. In all other cases correct positioning must be maintained by use of the SKIP macro as instructed.
The point in the message header at which the LPS and message header are aligned when the Send LPS is started depends on whether the message is from a reply queue (PUT), or a Switched Message. If from a reply queue, the LPS will be aligned with the first character of the output message as prepared by the process program. If it is a switched message, the LPS will be aligned with the last character of the last field processed by the Receive LPS.
Is a sequence of macros that will not be usable by all output messages about to be written?
No
K5 p.66
Yes
There must be a special character in each message header to identify it by type except for one type, which may be identified by its lack of a special character. (The functional macros for the latter must be the last of this section.) It is assumed that the very first character of a "PUT" message, and the very first character not processed by the received LPS in a "Switched" message, will be this message type character as recommended in the Message Header Preparation section of the Appendix.
K3 p.72
Write MSGTYPE in the Operation field of the next available macro statement.
p.65 K4
EXAMPLE MSGTYPE
Name
Operation
Operand
MSGTYPE


K4 p.64
Is the message type that is to be handled by the following sequence of macros identified by a special character?
Yes
No
Write this special character in the Operand field as C'#' where # is the special character. Ex: C'P'
The operand field will remain blank in this case, indicating that all messages not having a special character will be handled by the sequence of macros immediately following. Note: This must be the last MSGTYPE macro of the section.
EXAMPLE MSGTYPE
Name
Operation
Operand
MSGTYPE
C'P'
All macro statements listed from this point in the LPS to the next MSGTYPE macro or delimiter macro apply only to the type message just designated.
The message type that will currently be handled is now sorted out. Therefore, the header format of this type of message is known and the required macros can now be specified to handle this message type.

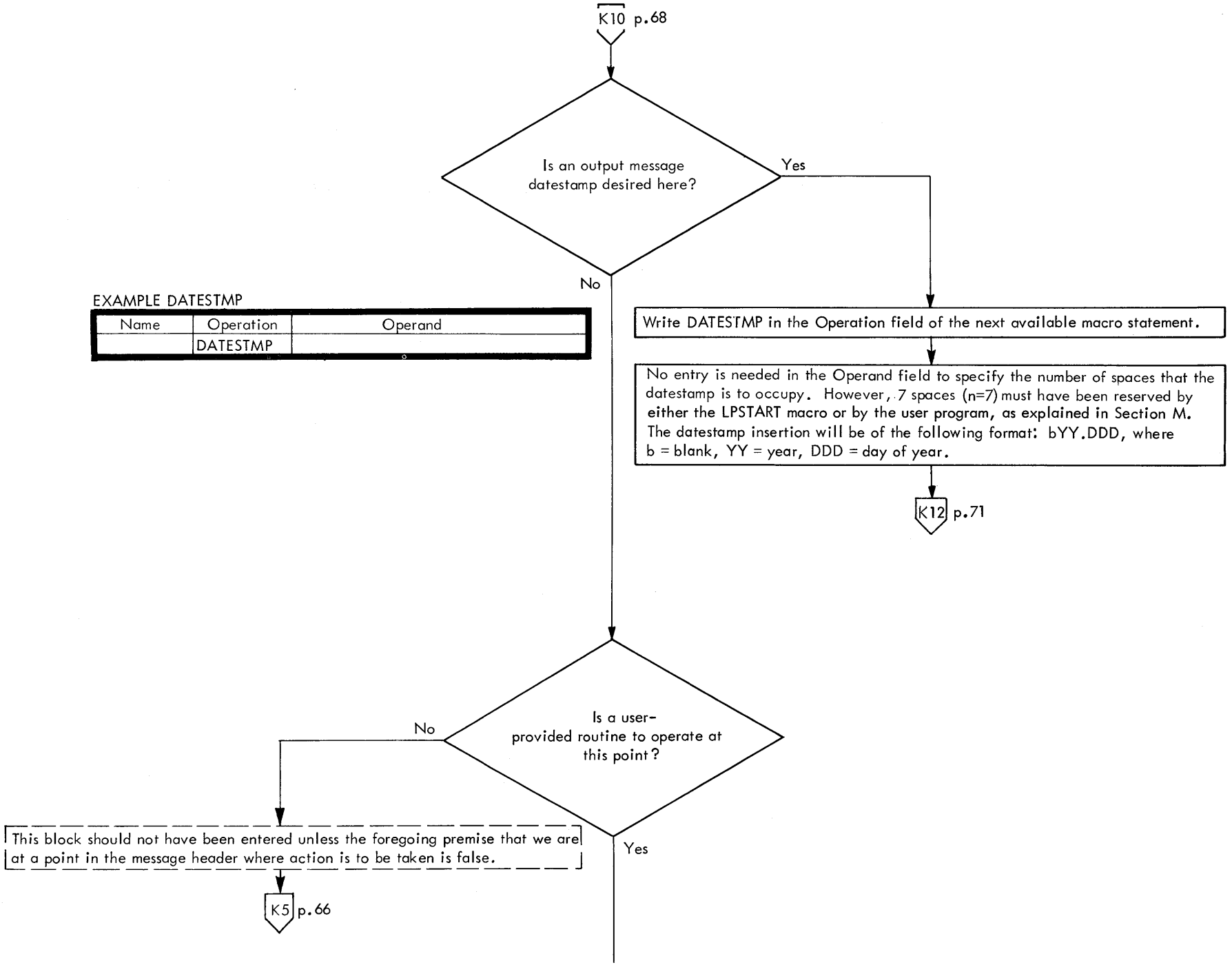

K5 pp.64,69
Are there any nonblank characters between the present position of the LPS and the first character of a field of interest to header analysis?
No
K9 p.67
Yes
A SKIP macro must be given to skip to such a position.
K6 p.72
EXAMPLE SKIP
Name | Operation | Operand
SKIP | ,C'$'
Write SKIP in the Operation field of the next available macro statement.
A skip can be made for a fixed number of nonblank character spaces or up to and including a certain character configuration.
Is there a special character configuration to designate the end of a skip field?
Yes
No
K7 p.67
K8 p.67


K7 p.66
Write in the Operand field a comma followed by C'chars', where chars represents the character configuration (max 8 characters) that denotes the end of the field to be skipped.
Ex: ,C'$'
K8 p.66
Write in the Operand field the fixed number of nonblank characters to be skipped. The maximum number that can be specified is the number of character positions remaining in the buffer past the present position of the LPS.
Ex: 8
EXAMPLE SKIP
Name | Operation | Operand
SKIP | 8
K9 pp.66,72
The LPS should now be ready for a field of interest to header analysis.
Is this header to have an output message sequence number inserted at this point?
Yes
No
EXAMPLE SEQOUT
Name | Operation | Operand
SEQOUT | 3
Write SEQOUT in the Operation field of the next available macro statement.
In the Operand write the decimal number of character spaces (n) to be used by the output sequence number (1≤n≤5), where the first character space is always a blank.
Ex: 3
K12 p.71



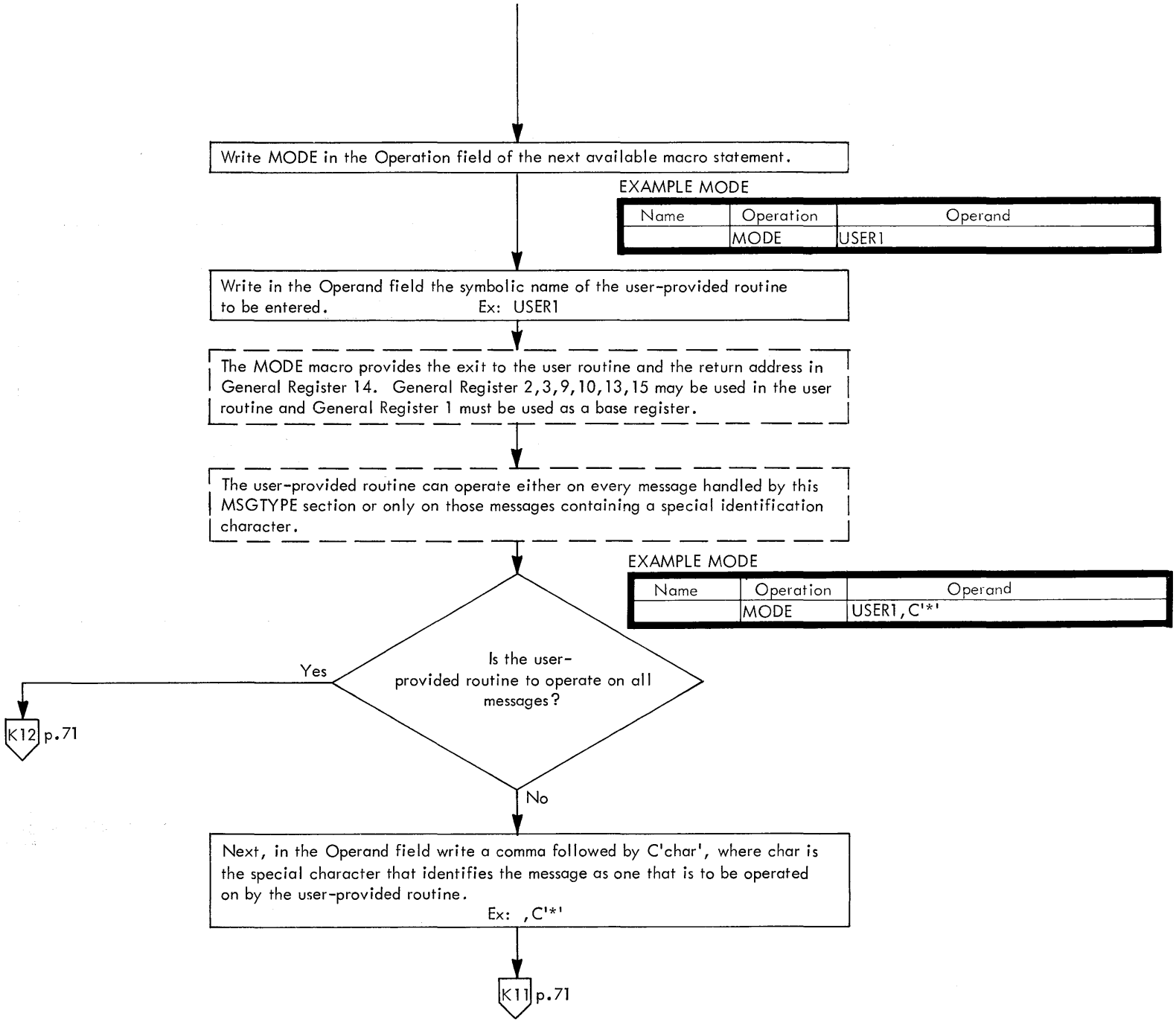
EXAMPLE TIMESTMP

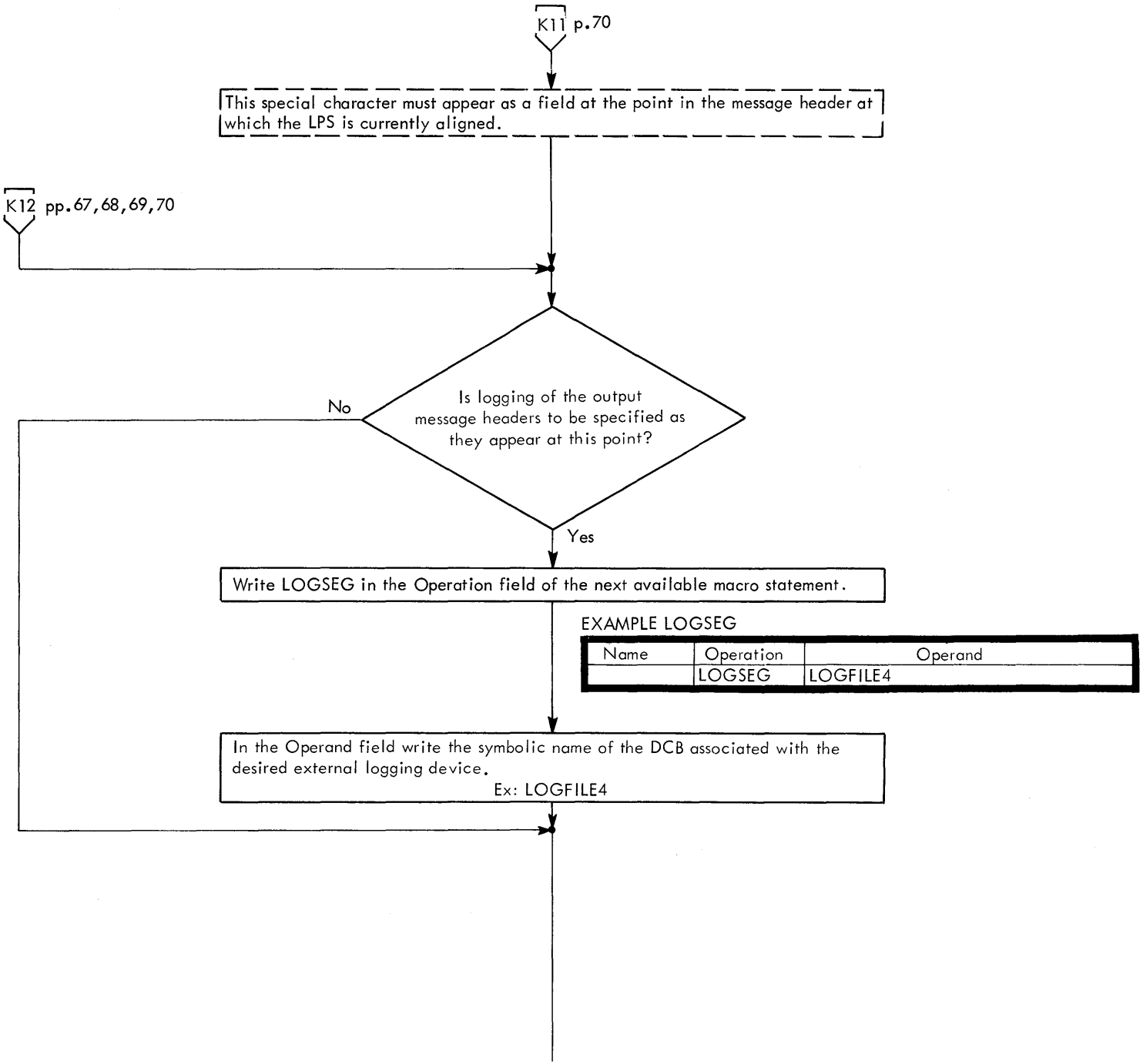
Name	Operation	Operand
	TIMESTMP	7

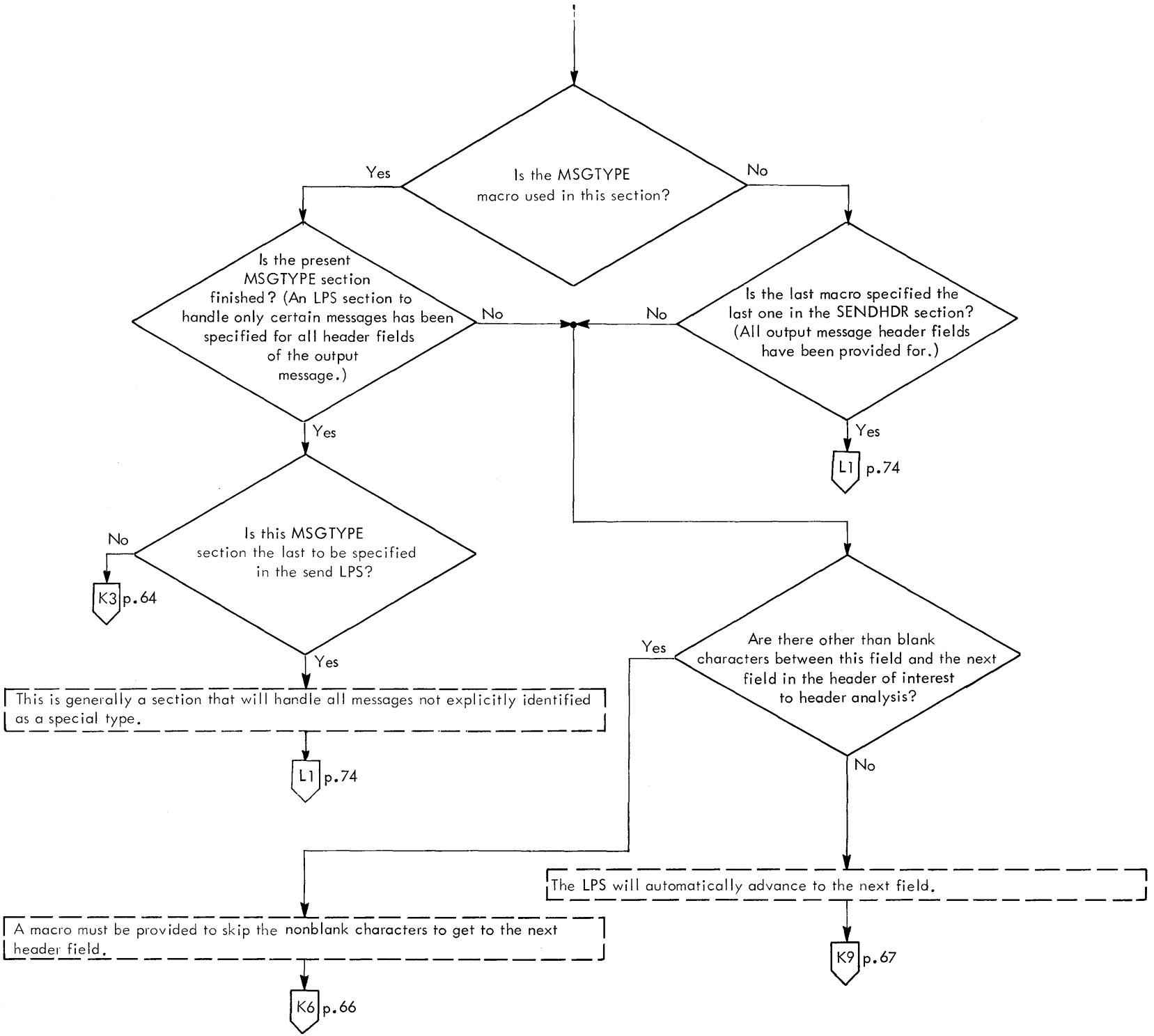


EXAMPLE DATESTMP

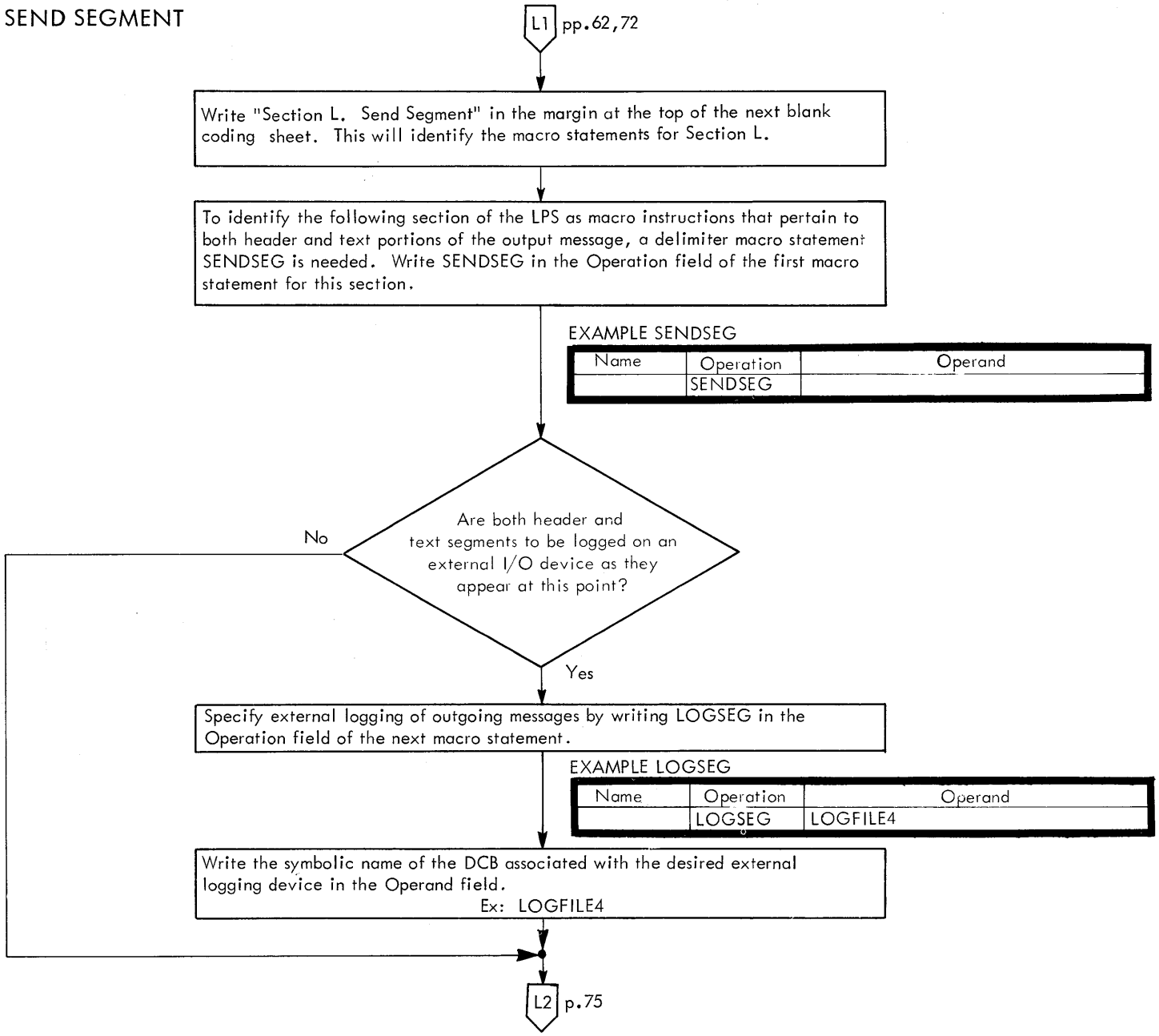
Name	Operation	Operand
	DATESTMP	

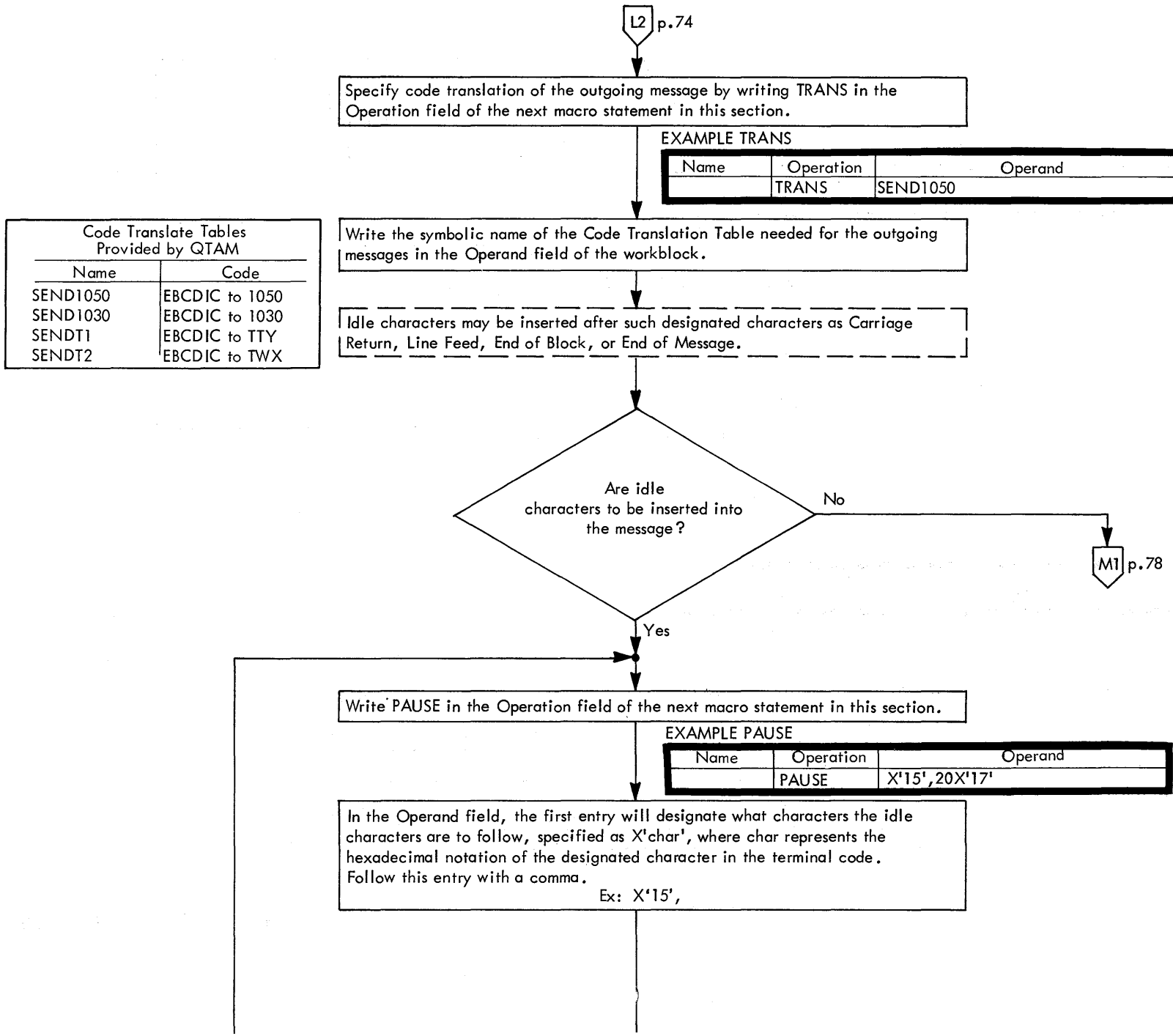

Write MODE in the Operation field of the next available macro statement.
EXAMPLE MODE
Name | Operation | Operand
MODE | USER1
Write in the Operand field the symbolic name of the user-provided routine to be entered. Ex: USER1
The MODE macro provides the exit to the user routine and the return address in General Register 14. General Register 2,3,9,10,13,15 may be used in the user routine and General Register 1 must be used as a base register.
The user-provided routine can operate either on every message handled by this MSGTYPE section or only on those messages containing a special identification character.
EXAMPLE MODE
Name | Operation | Operand
MODE | USER1,C'*'
Is the user-provided routine to operate on all messages?
Yes
K12 p.71
No
Next, in the Operand field write a comma followed by C'char', where char is the special character that identifies the message as one that is to be operated on by the user-provided routine.
Ex: ,C'*'
K11 p.71

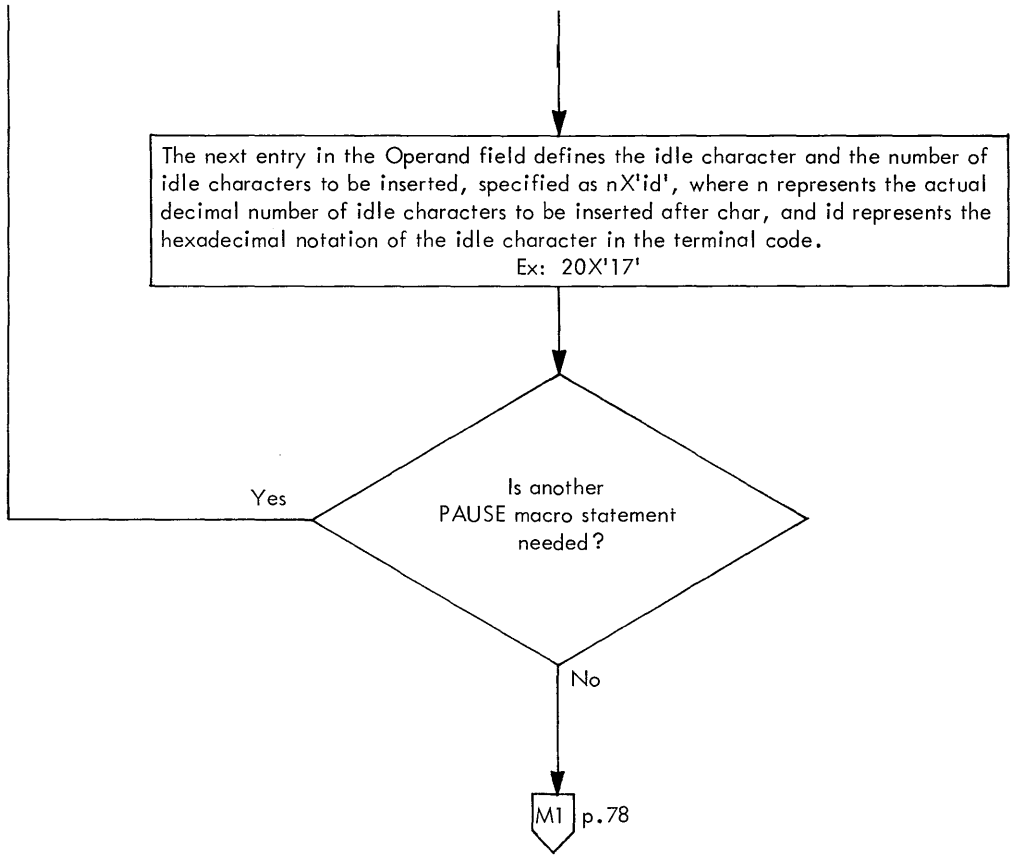

K11 p.70
This special character must appear as a field at the point in the message header at which the LPS is currently aligned.
K12 pp.67,68,69,70
Is logging of the output message headers to be specified as they appear at this point?
No
Yes
Write LOGSEG in the Operation field of the next available macro statement.
EXAMPLE LOGSEG
Name
Operation
Operand
LOGSEG
LOGFILE4
In the Operand field write the symbolic name of the DCB associated with the desired external logging device.
Ex: LOGFILE4


Is the MSGTYPE macro used in this section?
Yes
No
Is the present MSGTYPE section finished? (An LPS section to handle only certain messages has been specified for all header fields of the output message.)
No
Yes
Is this MSGTYPE section the last to be specified in the send LPS?
No
K3 p.64
Yes
This is generally a section that will handle all messages not explicitly identified as a special type.
L1 p.74
Is the last macro specified the last one in the SENDHDR section? (All output message header fields have been provided for.)
No
Yes
L1 p.74
Are there other than blank characters between this field and the next field in the header of interest to header analysis?
Yes
No
A macro must be provided to skip the nonblank characters to get to the next header field.
K6 p.66
The LPS will automatically advance to the next field.
K9 p.67

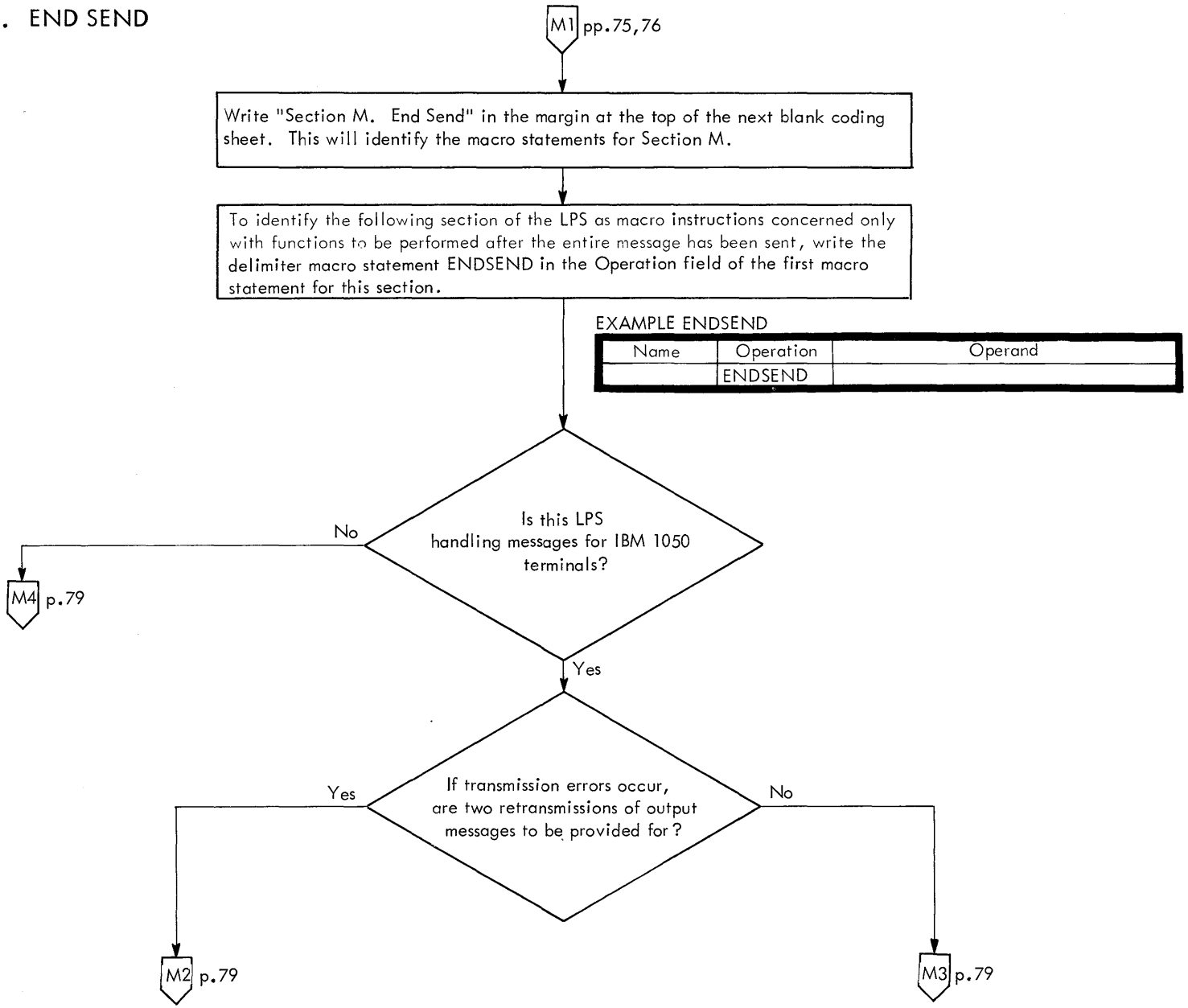
SECTION L. SEND SEGMENT




L2 p.74
Specify code translation of the outgoing message by writing TRANS in the Operation field of the next macro statement in this section.
EXAMPLE TRANS
Name | Operation | Operand
TRANS | SEND1050
Code Translate Tables Provided by QTAM
Name | Code
SEND1050 | EBCDIC to 1050
SEND1030 | EBCDIC to 1030
SENDT1 | EBCDIC to TTY
SENDT2 | EBCDIC to TWX
Write the symbolic name of the Code Translation Table needed for the outgoing messages in the Operand field of the workblock.
Idle characters may be inserted after such designated characters as Carriage Return, Line Feed, End of Block, or End of Message.
Are idle characters to be inserted into the message?
No
M1 p.78
Yes
Write PAUSE in the Operation field of the next macro statement in this section.
EXAMPLE PAUSE
Name | Operation | Operand
PAUSE | X'15',20X'17'
In the Operand field, the first entry will designate what characters the idle characters are to follow, specified as X'char', where char represents the hexadecimal notation of the designated character in the terminal code. Follow this entry with a comma.
Ex: X'15',


The next entry in the Operand field defines the idle character and the number of idle characters to be inserted, specified as nX'id', where n represents the actual decimal number of idle characters to be inserted after char, and id represents the hexadecimal notation of the idle character in the terminal code.
Ex: 20X'17'
Is another PAUSE macro statement needed?
Yes
No
M1
p.78

SECTION M. END SEND



M2 p.78

Write EOBLC in the Operation field of the next available macro statement.

EXAMPLE EOBLC

Name	Operation	Operand
	EOBLC	

M3 p.78

Write EOB in the Operation field of the next available macro statement.

EXAMPLE EOB

Name	Operation	Operand
	EOB	

M4 p.78

To provide for specification of error procedures, a halfword error mask is maintained for each communication line. This mask can be interrogated by macro statements, and the desired error procedure initiated upon finding the occurrence of a given error condition.

QTAM provides fixed error procedures for sending an error message, canceling an erroneous message, and rerouting messages. With each error procedure specified, an error mask, representing the error procedure it is to act on, must be specified in hexadecimal notation. Chart 2 describes the errors provided for when sending.

An example of how to specify an error mask

The error halfword specifying a transmission error looks like this:

0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

0 0 8 0

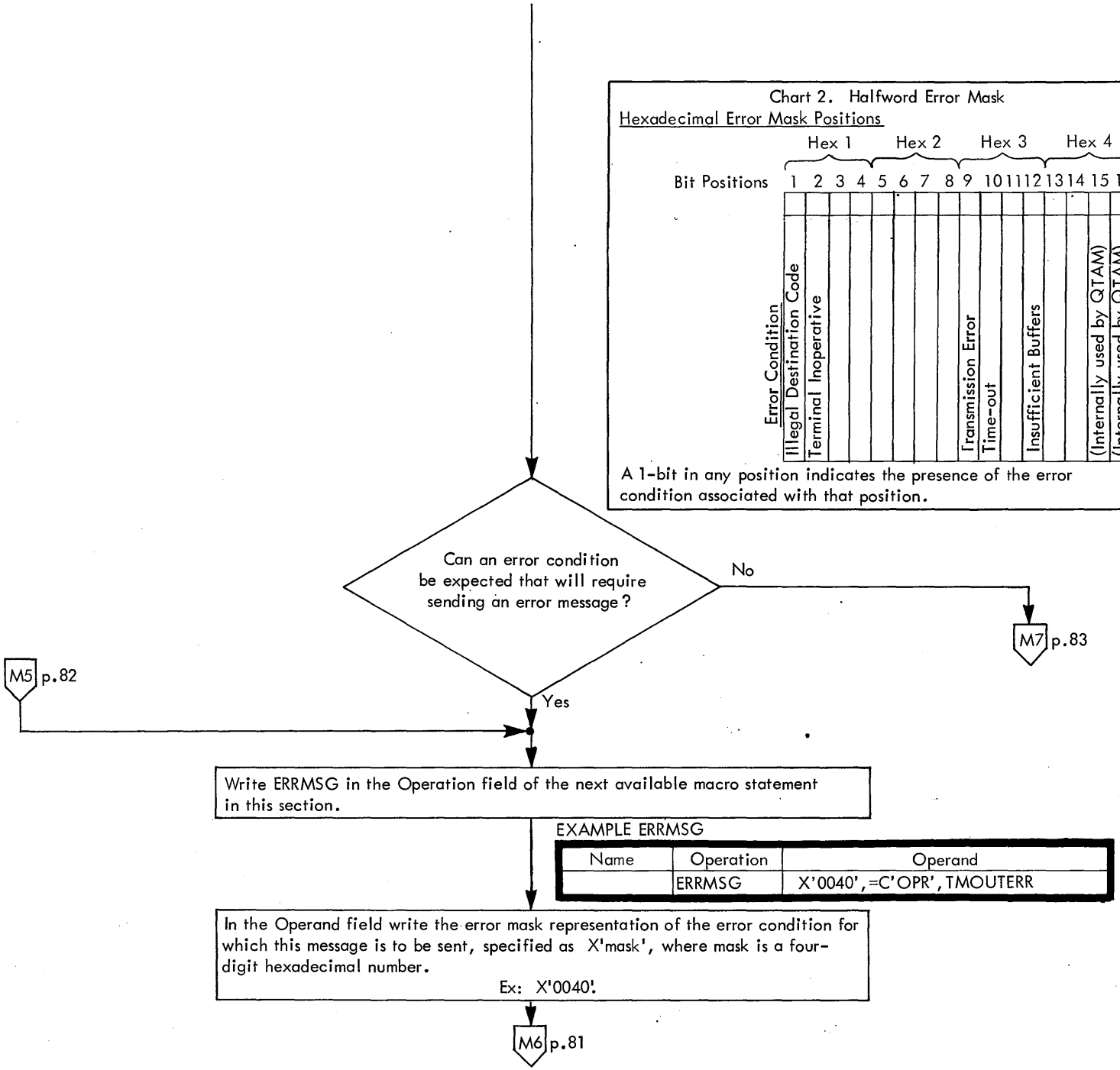
Its corresponding error mask in hexadecimal will be written as: X'0080'.

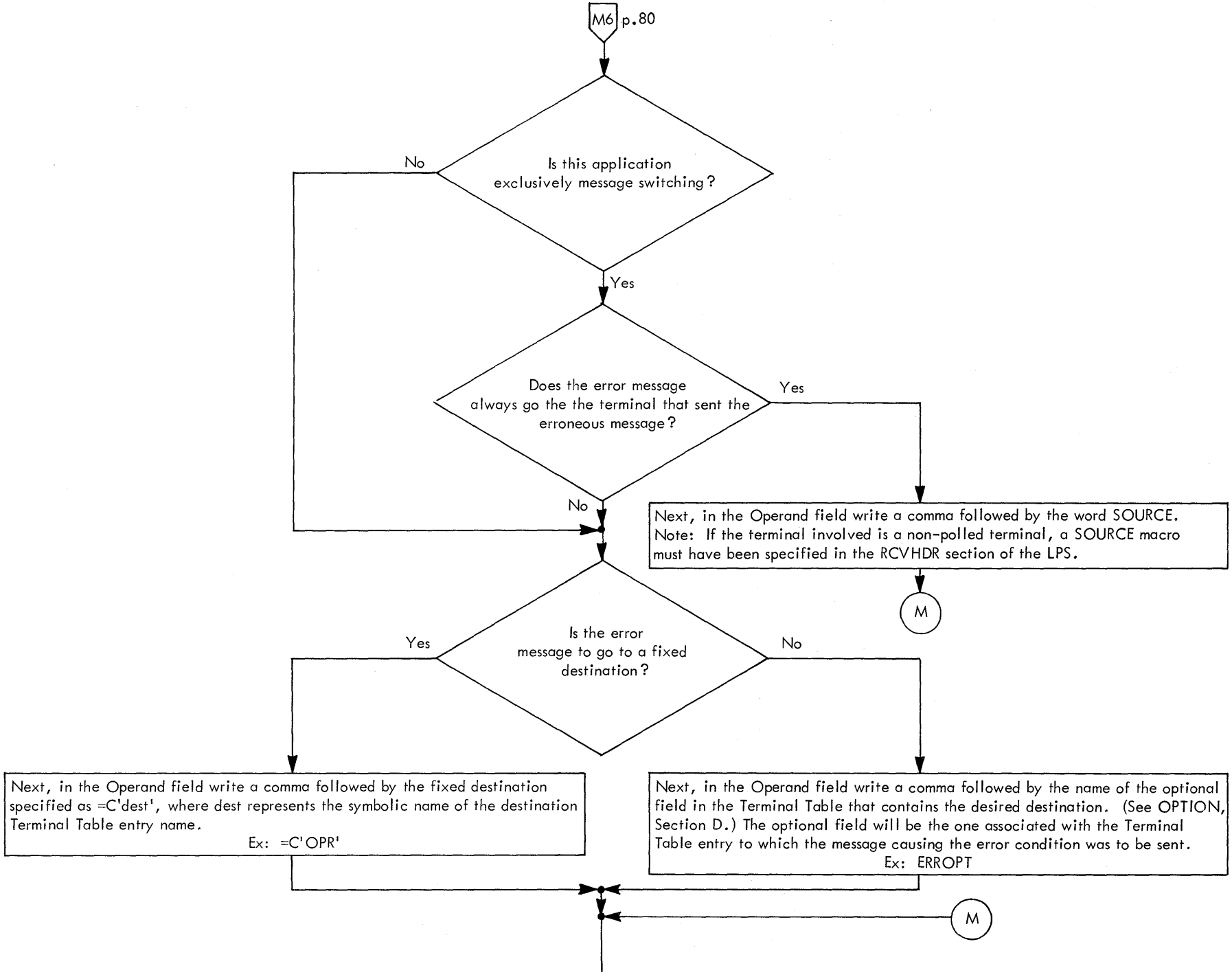
Chart 2. Halfword Error Mask

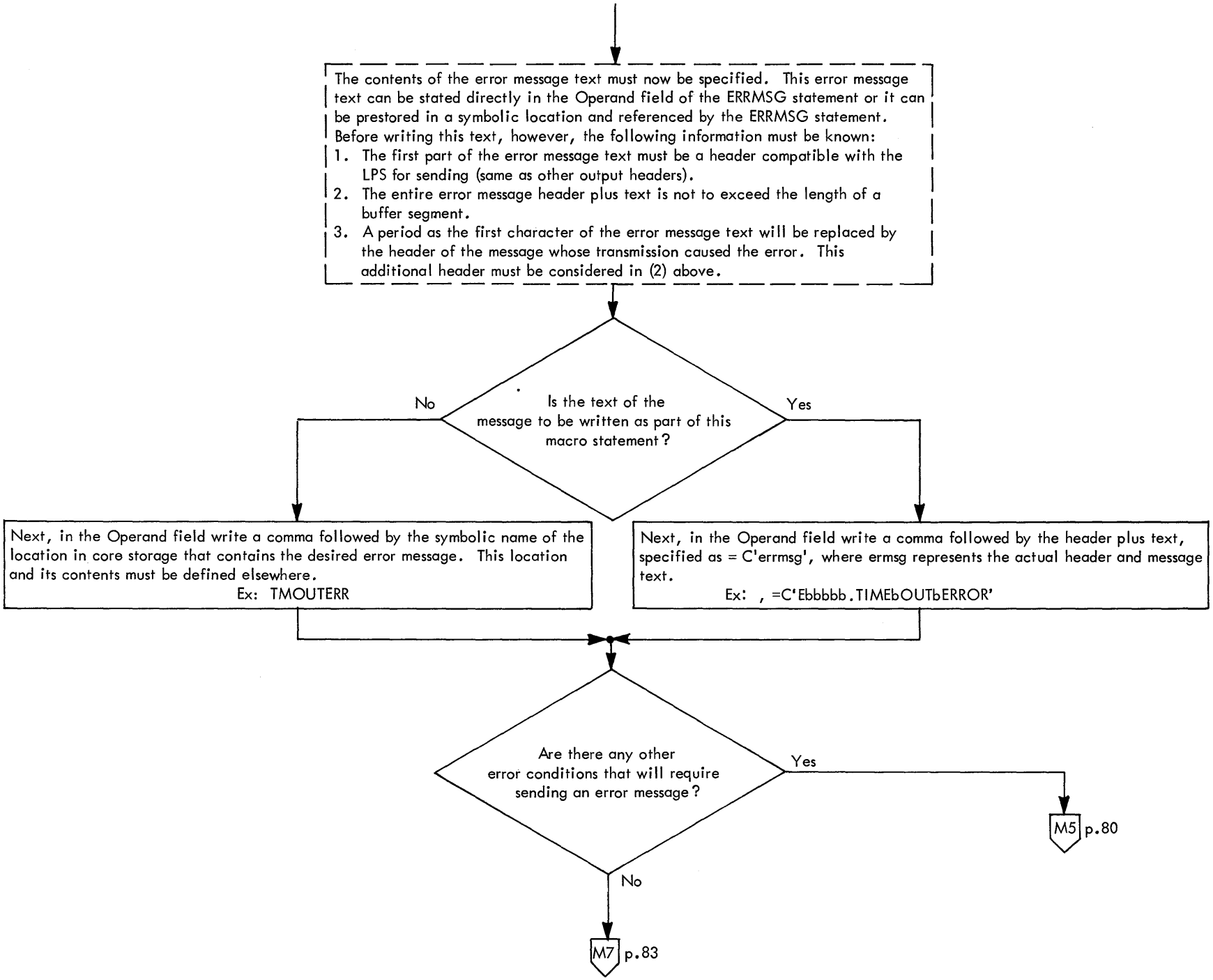
Hexadecimal Error Mask Positions

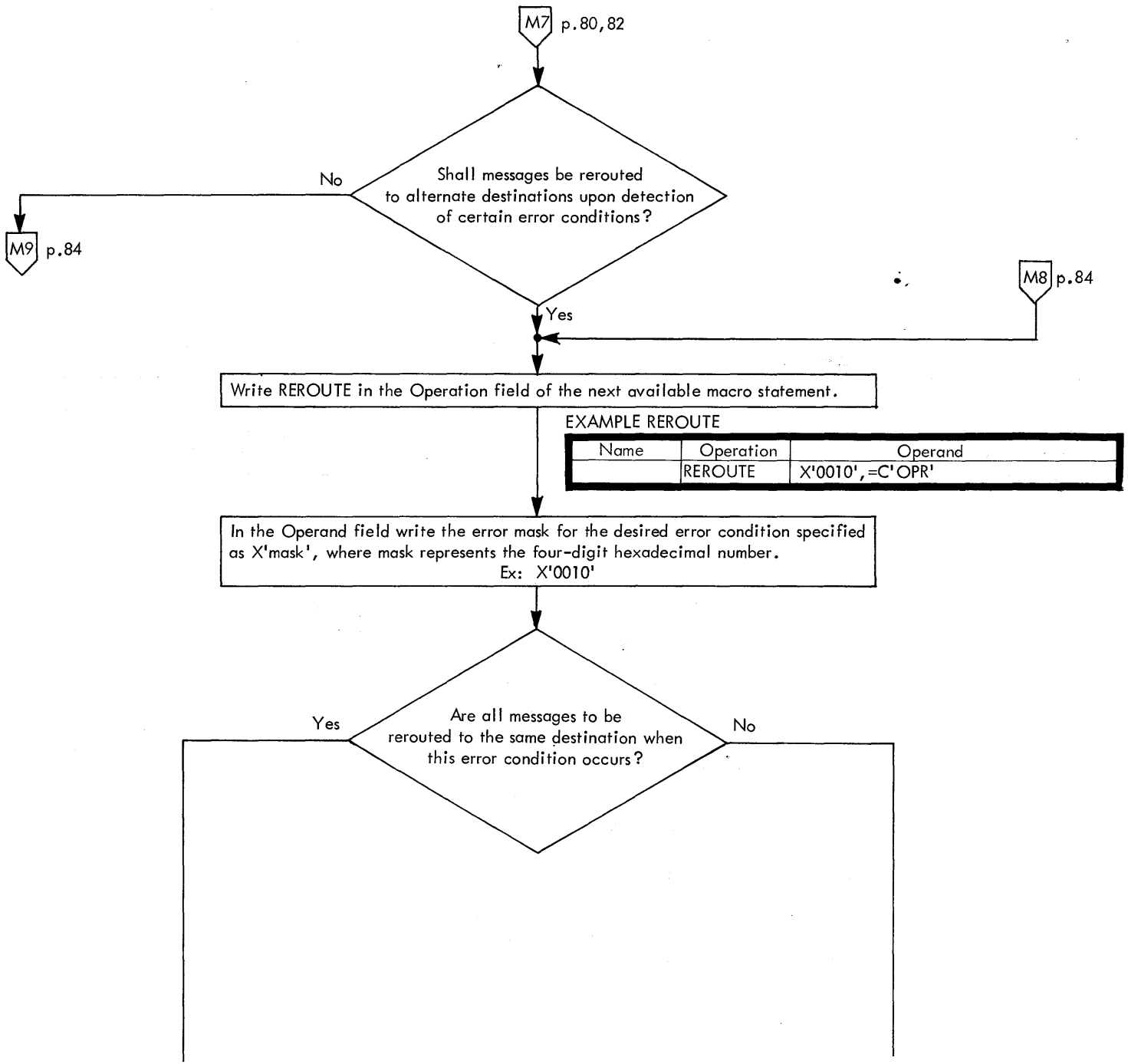
Bit Positions	Hex 1				Hex 2				Hex 3				Hex 4			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Error Condition																
Illegal Destination Code																
Terminal Inoperative																
Transmission Error																
Time-out																
Insufficient Buffers																
(Internally used by QTAM)																
(Internally used by QTAM)																

A 1-bit in any position indicates the presence of the error condition associated with that position.




M6 p.80
Is this application exclusively message switching?
No
Yes
Does the error message always go the the terminal that sent the erroneous message?
Yes
No
Next, in the Operand field write a comma followed by the word SOURCE. Note: If the terminal involved is a non-polled terminal, a SOURCE macro must have been specified in the RCVHDR section of the LPS.
M
Is the error message to go to a fixed destination?
Yes
No
Next, in the Operand field write a comma followed by the fixed destination specified as =C'dest', where dest represents the symbolic name of the destination Terminal Table entry name.
Ex: =C'OPR'
Next, in the Operand field write a comma followed by the name of the optional field in the Terminal Table that contains the desired destination. (See OPTION, Section D.) The optional field will be the one associated with the Terminal Table entry to which the message causing the error condition was to be sent.
Ex: ERROPT
M


The contents of the error message text must now be specified. This error message text can be stated directly in the Operand field of the ERRMSG statement or it can be prestored in a symbolic location and referenced by the ERRMSG statement. Before writing this text, however, the following information must be known:
1. The first part of the error message text must be a header compatible with the LPS for sending (same as other output headers).
2. The entire error message header plus text is not to exceed the length of a buffer segment.
3. A period as the first character of the error message text will be replaced by the header of the message whose transmission caused the error. This additional header must be considered in (2) above.
Is the text of the message to be written as part of this macro statement?
No
Yes
Next, in the Operand field write a comma followed by the symbolic name of the location in core storage that contains the desired error message. This location and its contents must be defined elsewhere.
Ex: TMOUTERR
Next, in the Operand field write a comma followed by the header plus text, specified as = C'errmsg', where ermsg represents the actual header and message text.
Ex: , =C'Ebbbbb.TIMEbOUTbERROR'
Are there any other error conditions that will require sending an error message?
Yes
M5 p.80
No
M7 p.83



EXAMPLE REROUTE

Name	Operation	Operand
	REROUTE	X'0010',=C'OPR'

Next, in the Operand field write a comma followed by the Terminal Table entry name of the rerouted destination, specified as =C'ternam e', where ternam e is the symbolic name of the destination.

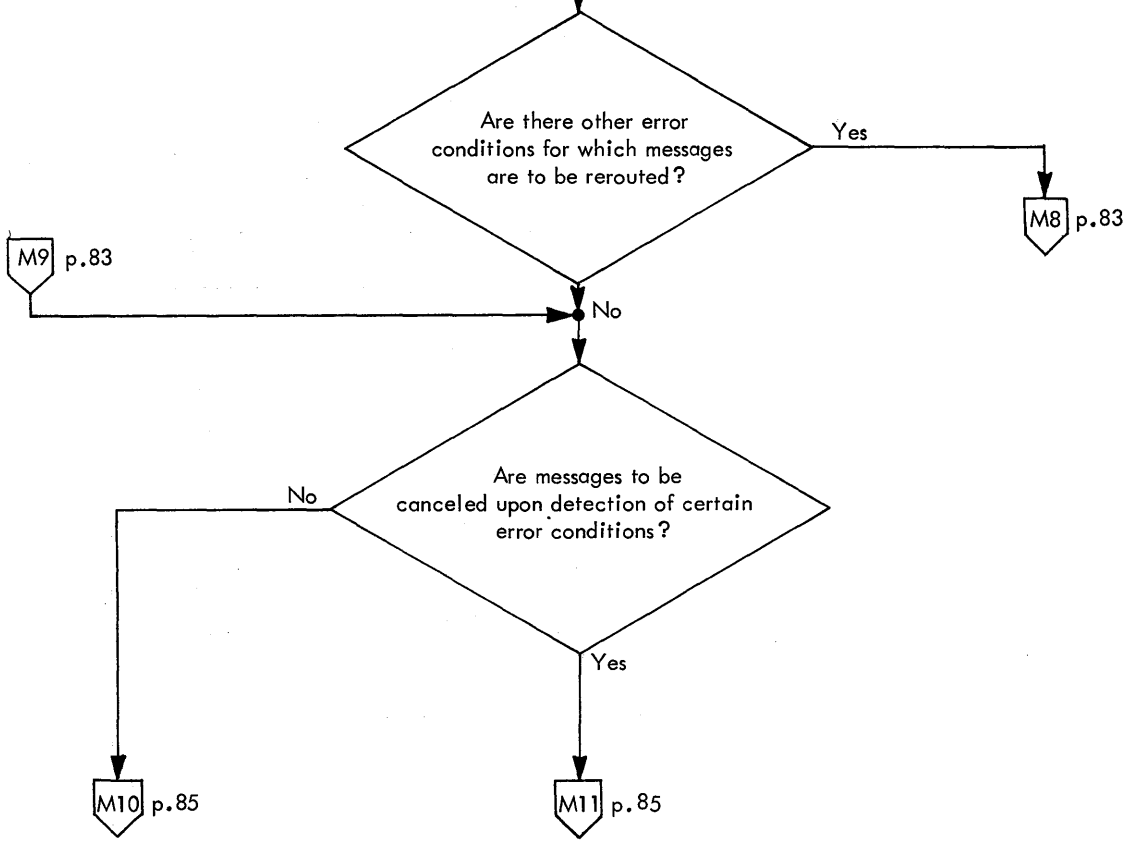
Ex: =C'OPR'

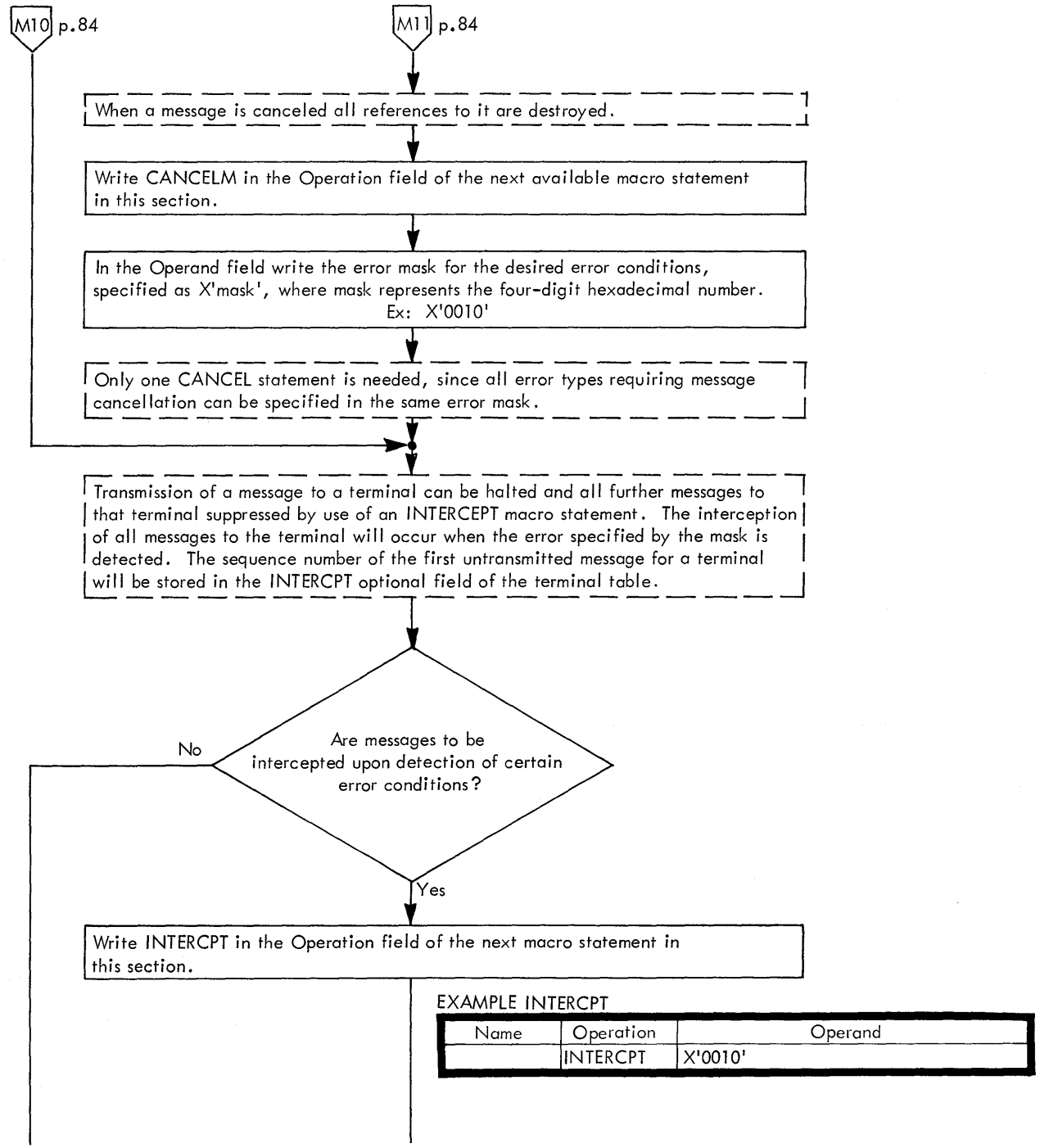
Another choice of reroute destinations is to have an alternate destination specified in an optional field of the message destination entry in the Terminal Table (see Terminal Table Specification). To use this alternate destination, next in the Operand field write a comma followed by the symbolic name of the optional field that is to contain the alternate destination for each terminal entry. If the terminals involved are non-polled terminals, a source macro must have been previously specified in the RCVHDR section of the LPS.

Ex: ERRDEST

EXAMPLE REROUTE

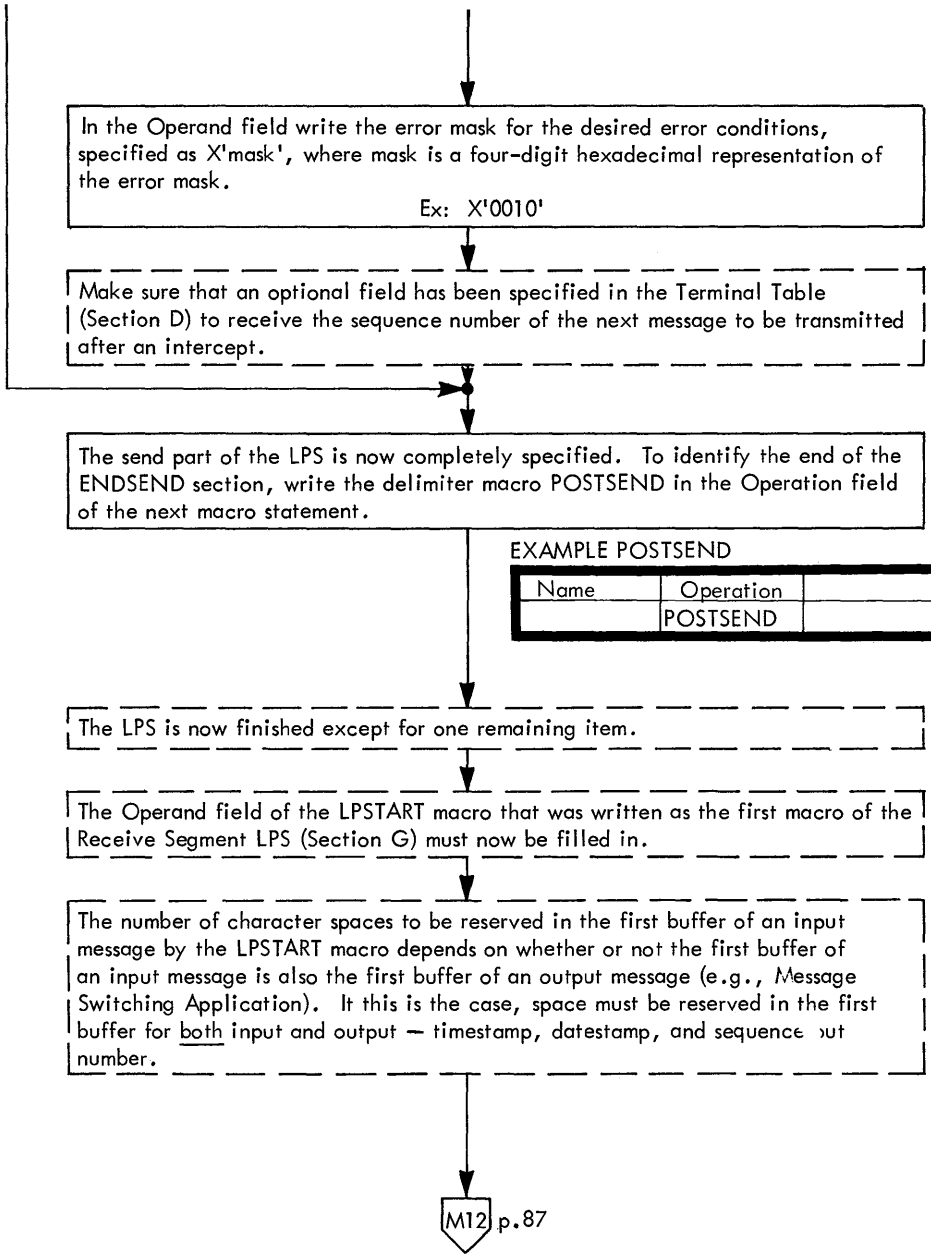
Name	Operation	Operand
	REROUTE	X'0010', ERRDEST

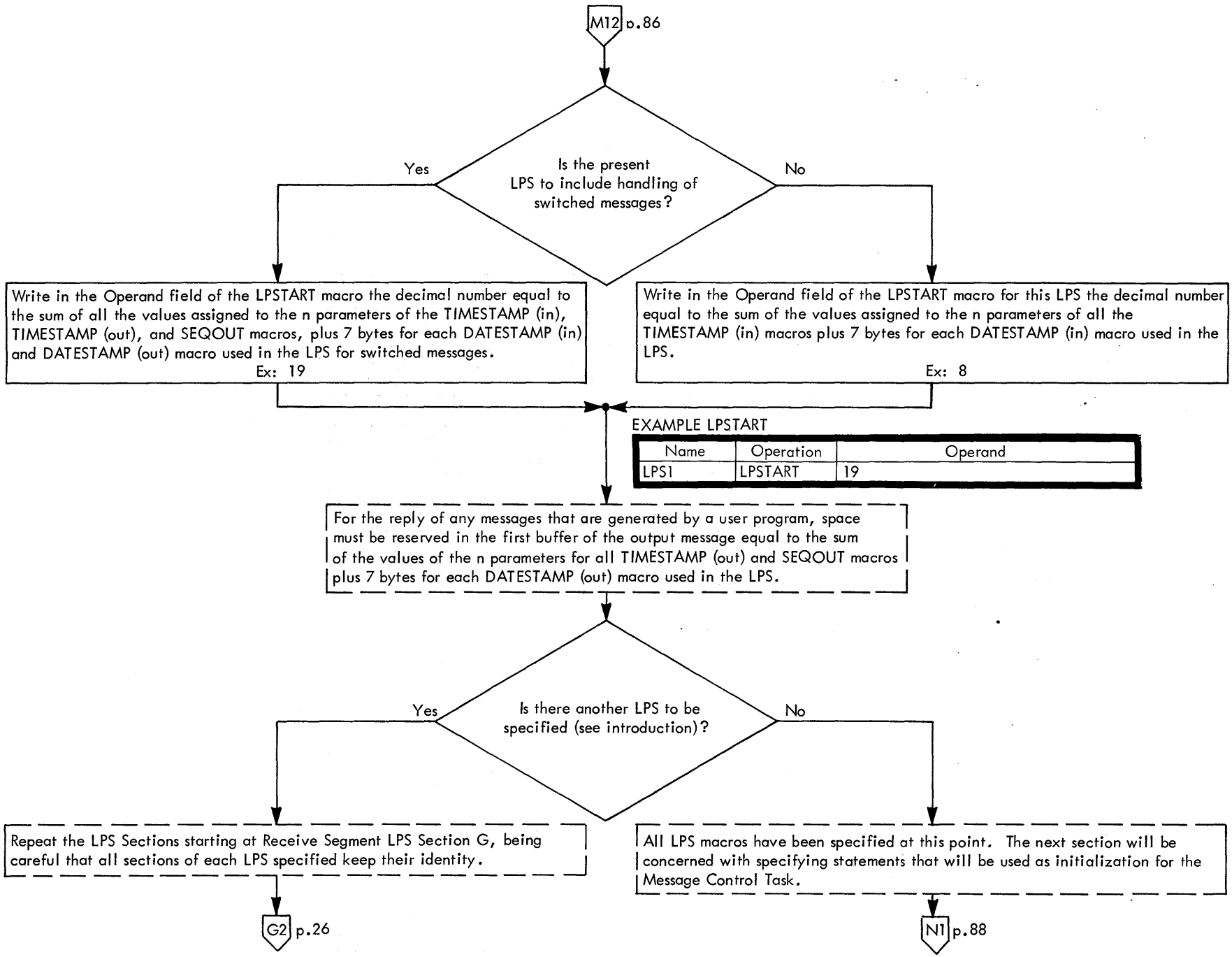




EXAMPLE INTERCPT

Name	Operation	Operand
	INTERCPT	X'0010'


In the Operand field write the error mask for the desired error conditions, specified as X'mask', where mask is a four-digit hexadecimal representation of the error mask.
Ex: X'0010'
Make sure that an optional field has been specified in the Terminal Table (Section D) to receive the sequence number of the next message to be transmitted after an intercept.
The send part of the LPS is now completely specified. To identify the end of the ENDSEND section, write the delimiter macro POSTSEND in the Operation field of the next macro statement.
EXAMPLE POSTSEND
Name
Operation
Operand
POSTSEND
The LPS is now finished except for one remaining item.
The Operand field of the LPSTART macro that was written as the first macro of the Receive Segment LPS (Section G) must now be filled in.
The number of character spaces to be reserved in the first buffer of an input message by the LPSTART macro depends on whether or not the first buffer of an input message is also the first buffer of an output message (e.g., Message Switching Application). It this is the case, space must be reserved in the first buffer for both input and output — timestamp, datestamp, and sequence out number.
M12 p.87


M12 p.86
Is the present LPS to include handling of switched messages?
Yes
No
Write in the Operand field of the LPSTART macro the decimal number equal to the sum of all the values assigned to the n parameters of the TIMESTAMP (in), TIMESTAMP (out), and SEQOUT macros, plus 7 bytes for each DATESTAMP (in) and DATESTAMP (out) macro used in the LPS for switched messages.
Ex: 19
Write in the Operand field of the LPSTART macro for this LPS the decimal number equal to the sum of the values assigned to the n parameters of all the TIMESTAMP (in) macros plus 7 bytes for each DATESTAMP (in) macro used in the LPS.
Ex: 8
EXAMPLE LPSTART
Name | Operation | Operand
LPS1 | LPSTART | 19
For the reply of any messages that are generated by a user program, space must be reserved in the first buffer of the output message equal to the sum of the values of the n parameters for all TIMESTAMP (out) and SEQOUT macros plus 7 bytes for each DATESTAMP (out) macro used in the LPS.
Is there another LPS to be specified (see introduction)?
Yes
No
Repeat the LPS Sections starting at Receive Segment LPS Section G, being careful that all sections of each LPS specified keep their identity.
All LPS macros have been specified at this point. The next section will be concerned with specifying statements that will be used as initialization for the Message Control Task.
G2 p.26
N1 p.88

SECTION N. DATA SET INITIALIZATION

N1 p.87

Write "Section N. Data Set Initialization" in the margin at the top of the next blank coding sheet. This will identify the macro statements for Section N.

The purpose of this section is to provide for specification of macros that will initialize all Data Sets that will be referenced by the Message Control Task and by use of OPEN and ENDREADY macros.

When a Data Set is "opened", the following functions are performed:
1. Control blocks are initialized.
2. Subroutines are acquired for the access method.
3. Control blocks are assigned core storage locations.
4. In the case of a communication line, the line will be made operative for sending or receiving provided its polling list and terminal entries are in an active status.*

Data Sets which will be referenced by the Message Control Task, and which therefore must be "opened", are:
1. Communication Line Groups.
2. Direct Access Storage Device used for queuing messages.
3. External I/O device (s) used for logging messages.

Each such Data Set has a Data Control Block (DCB) by which it is referenced.

Opening of the Direct Access Storage Device, Logging Devices, and Communication Line Groups will now be provided for.

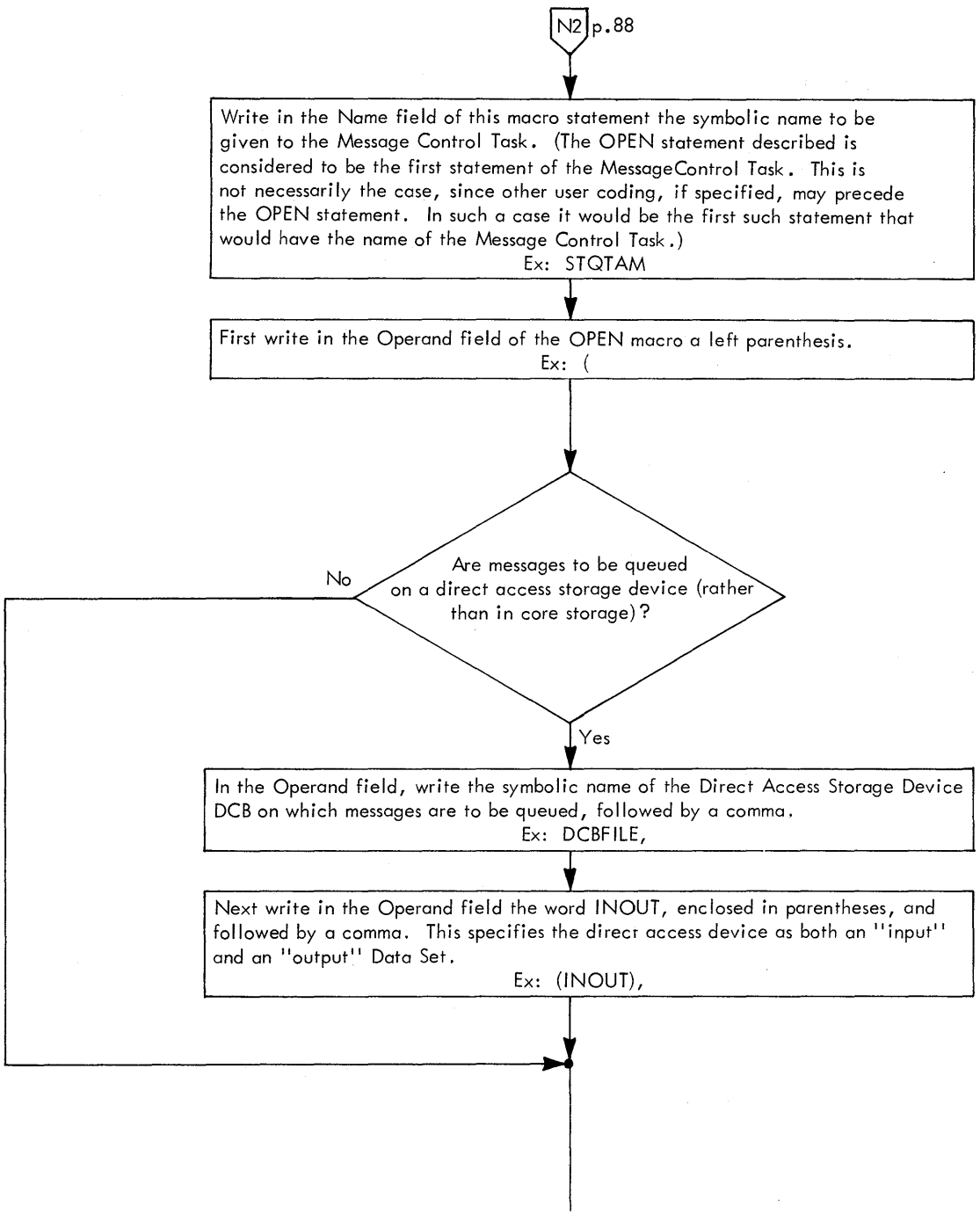
Write OPEN in the Operation field of the first macro statement for this section.

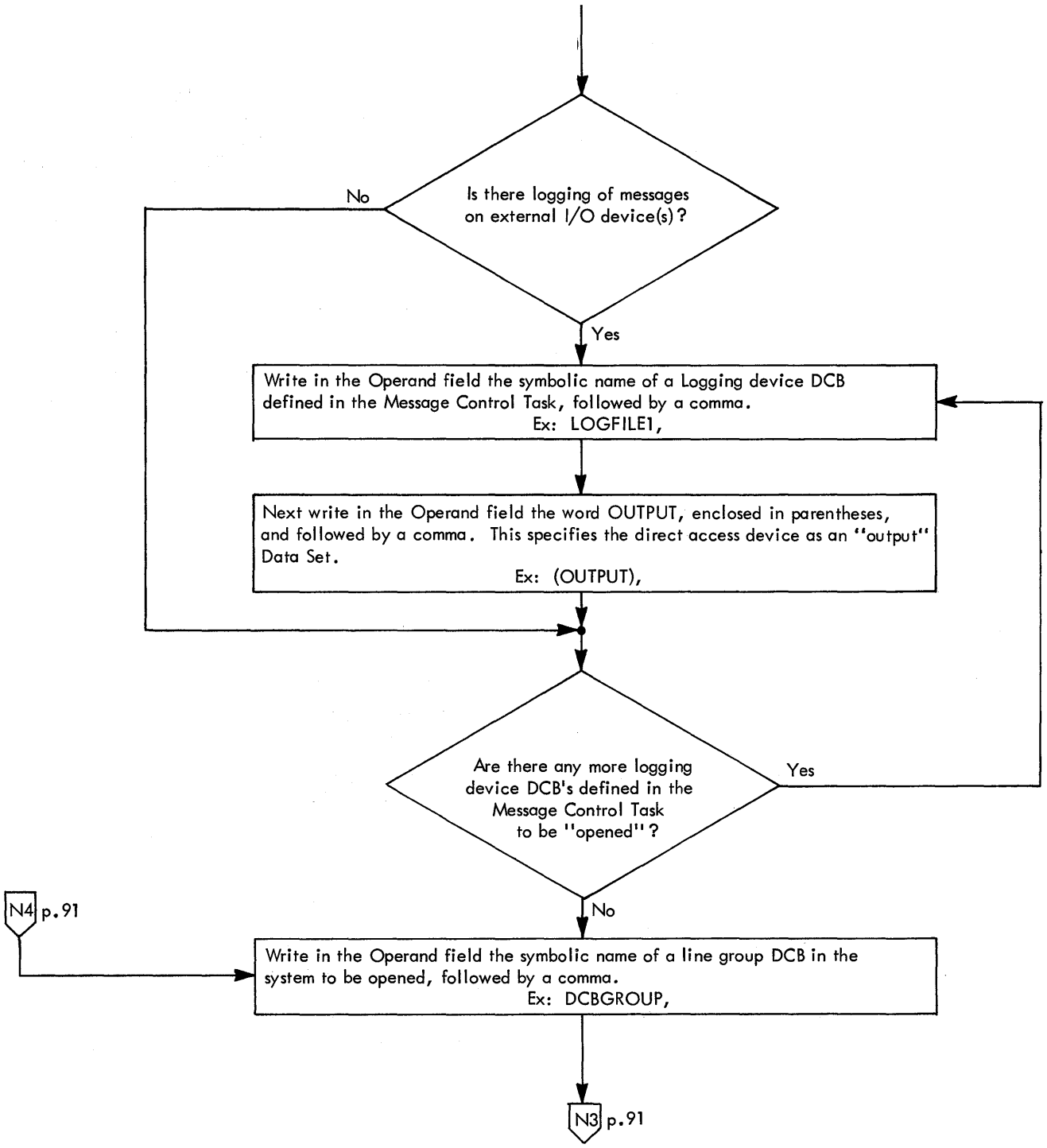
EXAMPLE OPEN

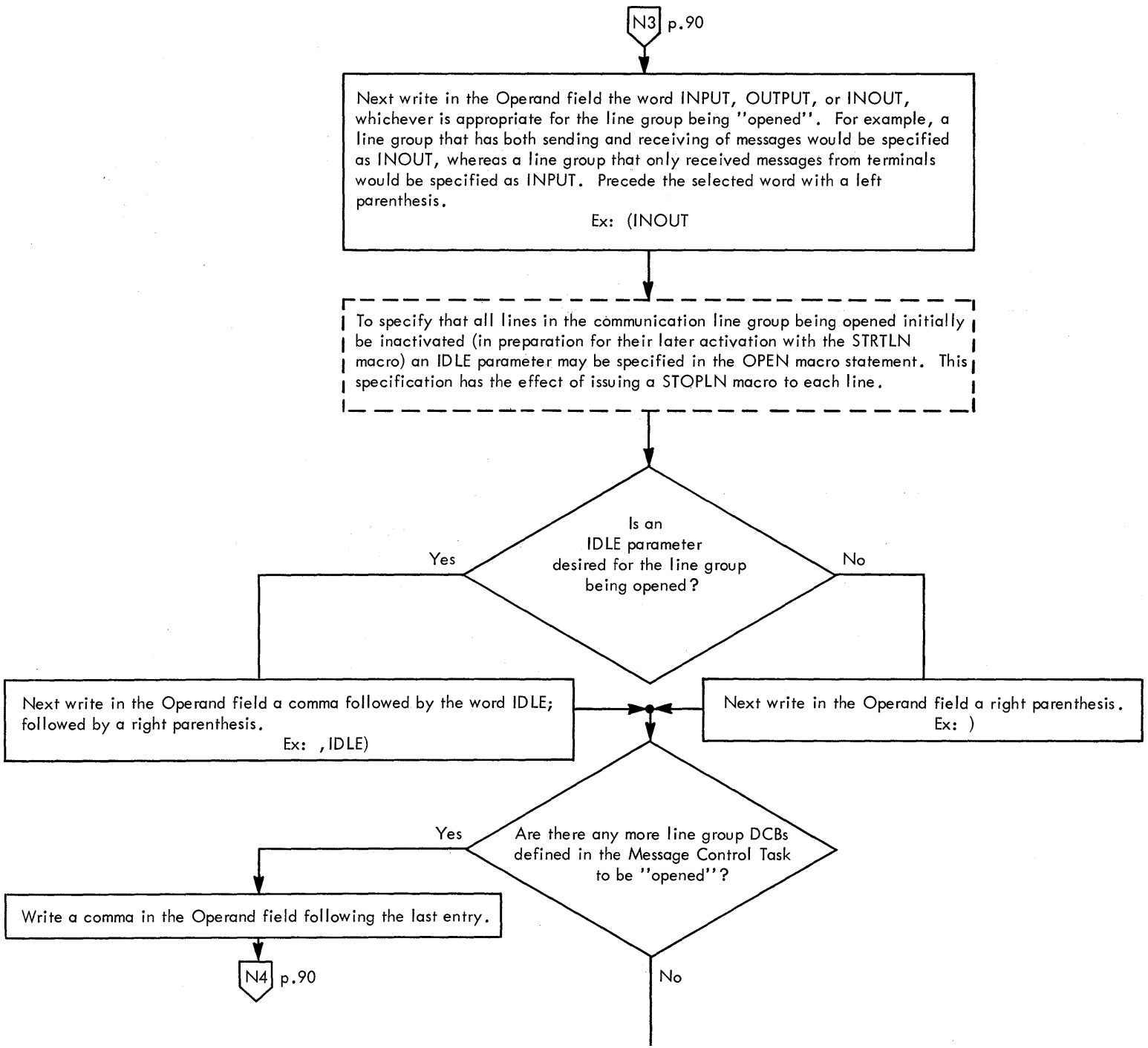
Name	Operation	Operand
STQTAM	OPEN	(DCBFILE,(INOUT),LOGFILE1,(OUTPUT),
		DCBGROUP,(INOUT))

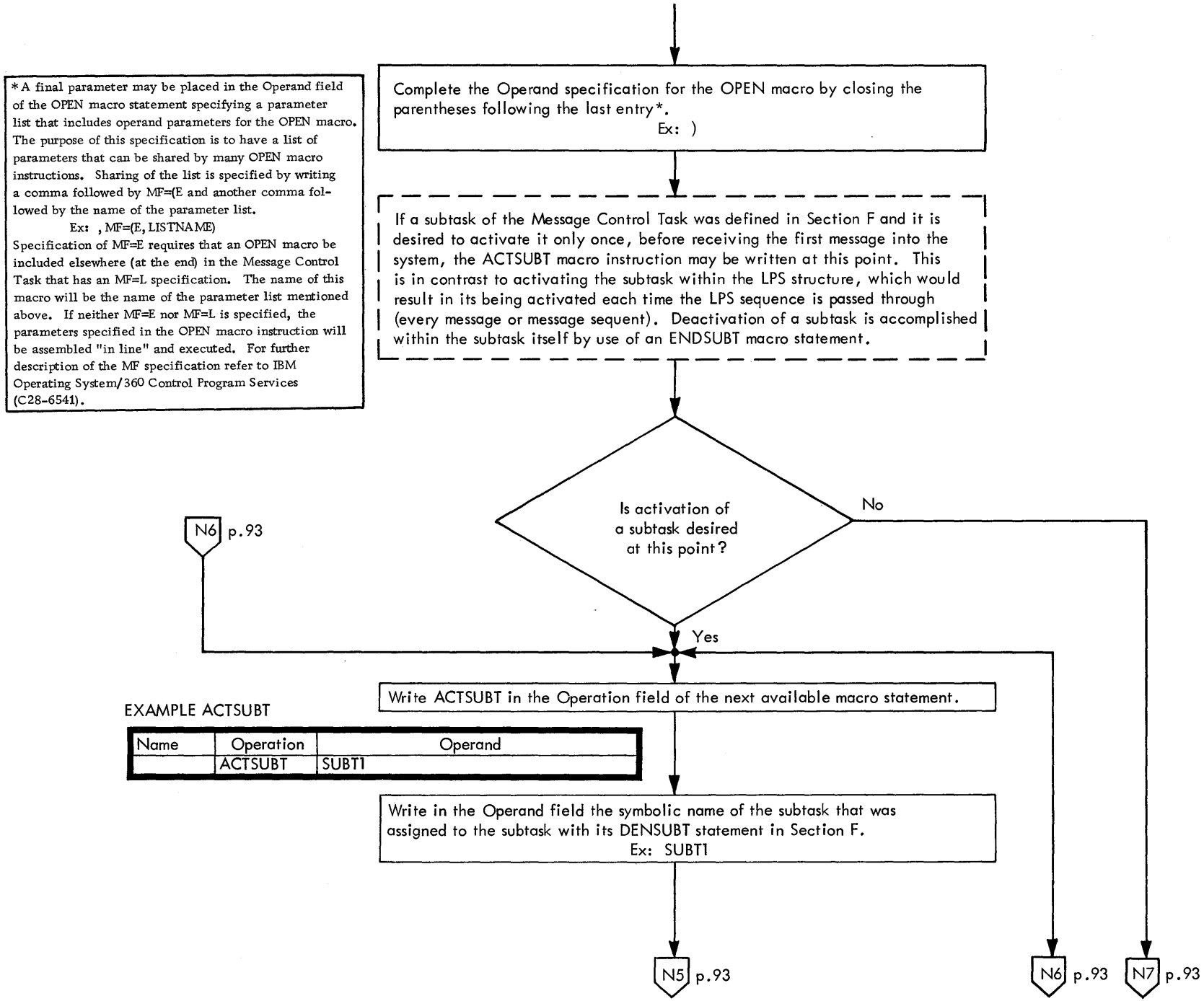
p.89 N2

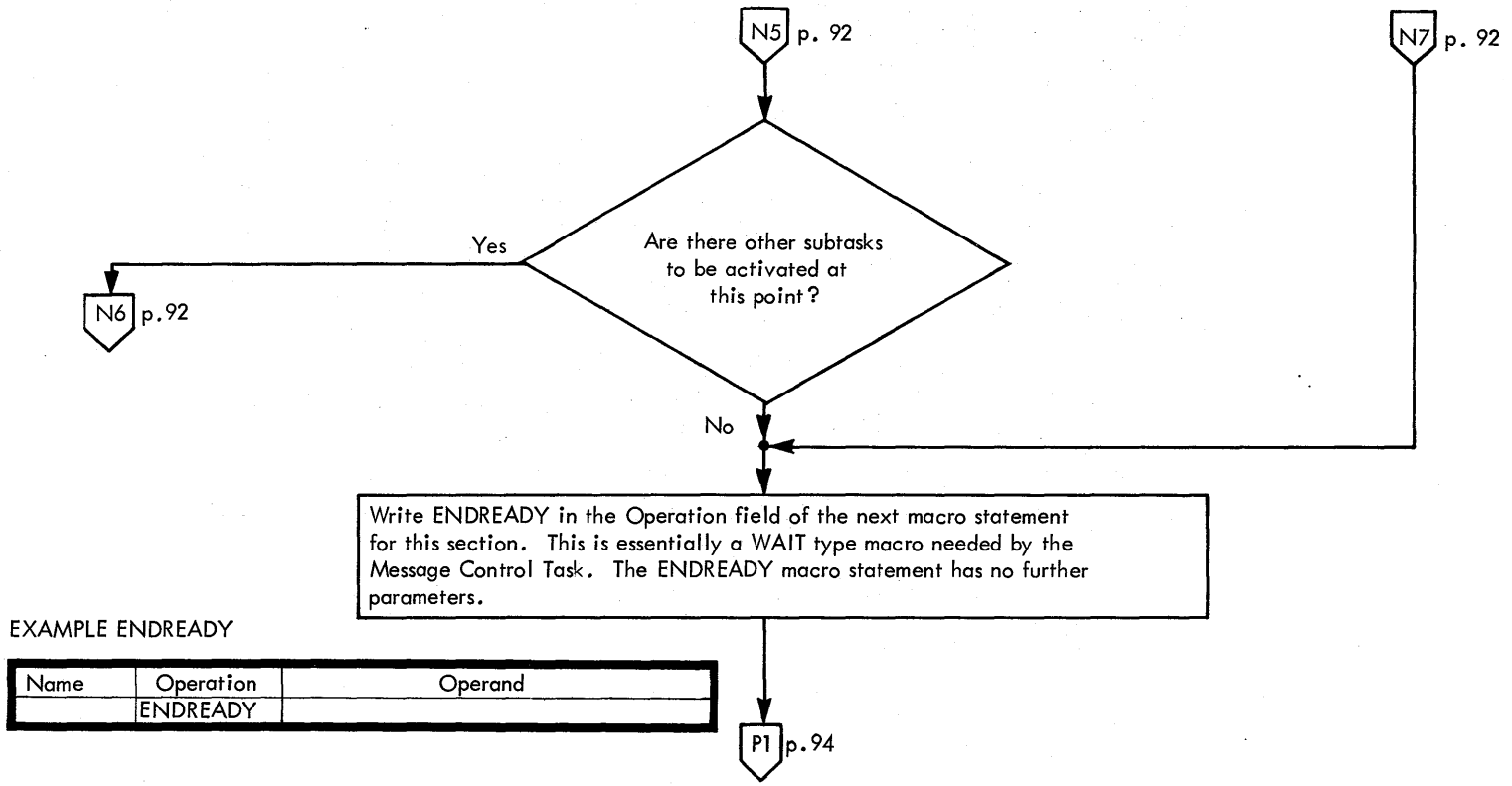
*Terminals may be activated on a per-line or per-terminal basis, rather than by line group, by use of the CPYRL/CHNGPL or STRTLN/STOPLN macro instructions respectively. These macros may be included in the present section prior to, and immediately following, the OPEN macro instruction(s) as required to provide the necessary control (see C28-6553).


N2 p.88
Write in the Name field of this macro statement the symbolic name to be given to the Message Control Task. (The OPEN statement described is considered to be the first statement of the MessageControl Task. This is not necessarily the case, since other user coding, if specified, may precede the OPEN statement. In such a case it would be the first such statement that would have the name of the Message Control Task.)
Ex: STQTAM
First write in the Operand field of the OPEN macro a left parenthesis.
Ex: (
Are messages to be queued on a direct access storage device (rather than in core storage)?
No
Yes
In the Operand field, write the symbolic name of the Direct Access Storage Device DCB on which messages are to be queued, followed by a comma.
Ex: DCBFILE,
Next write in the Operand field the word INOUT, enclosed in parentheses, and followed by a comma. This specifies the direct access device as both an "input" and an "output" Data Set.
Ex: (INOUT),


Is there logging of messages on external I/O device(s)?
No
Yes
Write in the Operand field the symbolic name of a Logging device DCB defined in the Message Control Task, followed by a comma.
Ex: LOGFILE1,
Next write in the Operand field the word OUTPUT, enclosed in parentheses, and followed by a comma. This specifies the direct access device as an "output" Data Set.
Ex: (OUTPUT),
Are there any more logging device DCB's defined in the Message Control Task to be "opened"?
Yes
No
N4 p.91
Write in the Operand field the symbolic name of a line group DCB in the system to be opened, followed by a comma.
Ex: DCBGROUP,
N3 p.91

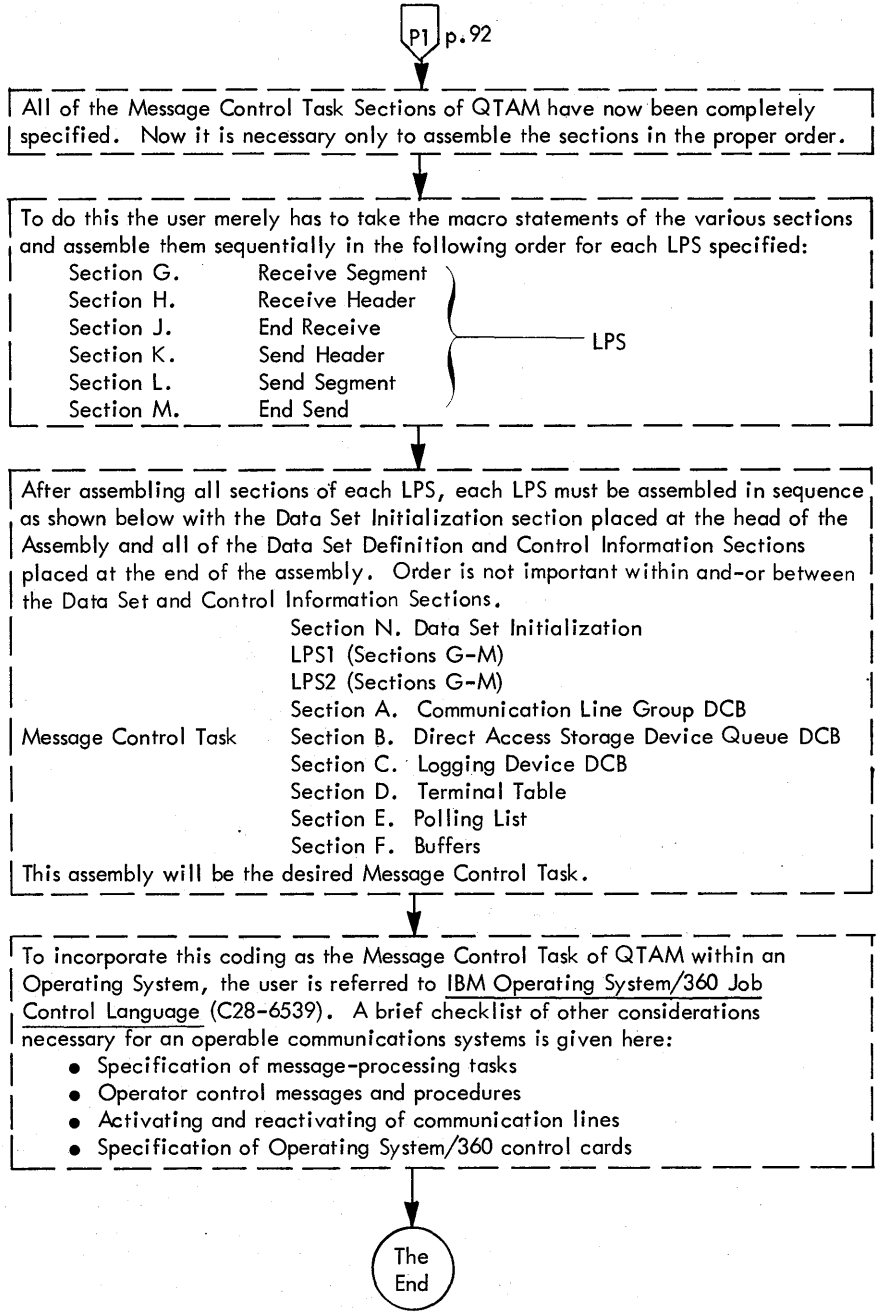

N3 p.90
Next write in the Operand field the word INPUT, OUTPUT, or INOUT, whichever is appropriate for the line group being "opened". For example, a line group that has both sending and receiving of messages would be specified as INOUT, whereas a line group that only received messages from terminals would be specified as INPUT. Precede the selected word with a left parenthesis.
Ex: (INOUT
To specify that all lines in the communication line group being opened initially be inactivated (in preparation for their later activation with the STRTLN macro) an IDLE parameter may be specified in the OPEN macro statement. This specification has the effect of issuing a STOPLN macro to each line.
Is an IDLE parameter desired for the line group being opened?
Yes
No
Next write in the Operand field a comma followed by the word IDLE; followed by a right parenthesis.
Ex: ,IDLE)
Next write in the Operand field a right parenthesis.
Ex:)
Are there any more line group DCBs defined in the Message Control Task to be "opened"?
Yes
No
Write a comma in the Operand field following the last entry.
N4 p.90


*A final parameter may be placed in the Operand field of the OPEN macro statement specifying a parameter list that includes operand parameters for the OPEN macro. The purpose of this specification is to have a list of parameters that can be shared by many OPEN macro instructions. Sharing of the list is specified by writing a comma followed by MF=(E and another comma followed by the name of the parameter list.
Ex: , MF=(E, LISTNAME)
Specification of MF=E requires that an OPEN macro be included elsewhere (at the end) in the Message Control Task that has an MF=L specification. The name of this macro will be the name of the parameter list mentioned above. If neither MF=E nor MF=L is specified, the parameters specified in the OPEN macro instruction will be assembled "in line" and executed. For further description of the MF specification refer to IBM Operating System/360 Control Program Services (C28-6541).
Complete the Operand specification for the OPEN macro by closing the parentheses following the last entry*.
Ex:)
If a subtask of the Message Control Task was defined in Section F and it is desired to activate it only once, before receiving the first message into the system, the ACTSUBT macro instruction may be written at this point. This is in contrast to activating the subtask within the LPS structure, which would result in its being activated each time the LPS sequence is passed through (every message or message sequent). Deactivation of a subtask is accomplished within the subtask itself by use of an ENDSUBT macro statement.
Is activation of a subtask desired at this point?
No
Yes
N6 p.93
Write ACTSUBT in the Operation field of the next available macro statement.
EXAMPLE ACTSUBT
Name
Operation
Operand
ACTSUBT
SUBT1
Write in the Operand field the symbolic name of the subtask that was assigned to the subtask with its DENSUBT statement in Section F.
Ex: SUBT1
N5 p.93
N6 p.93
N7 p.93



EXAMPLE ENDREADY

Name	Operation	Operand
	ENDREADY	


P1
p.92
All of the Message Control Task Sections of QTAM have now been completely specified. Now it is necessary only to assemble the sections in the proper order.
To do this the user merely has to take the macro statements of the various sections and assemble them sequentially in the following order for each LPS specified:
Section G. Receive Segment
Section H. Receive Header
Section J. End Receive
Section K. Send Header
Section L. Send Segment
Section M. End Send
LPS
After assembling all sections of each LPS, each LPS must be assembled in sequence as shown below with the Data Set Initialization section placed at the head of the Assembly and all of the Data Set Definition and Control Information Sections placed at the end of the assembly. Order is not important within and-or between the Data Set and Control Information Sections.
Message Control Task
Section N. Data Set Initialization
LPS1 (Sections G-M)
LPS2 (Sections G-M)
Section A. Communication Line Group DCB
Section B. Direct Access Storage Device Queue DCB
Section C. Logging Device DCB
Section D. Terminal Table
Section E. Polling List
Section F. Buffers
This assembly will be the desired Message Control Task.
To incorporate this coding as the Message Control Task of QTAM within an Operating System, the user is referred to IBM Operating System/360 Job Control Language (C28-6539). A brief checklist of other considerations necessary for an operable communications systems is given here:
• Specification of message-processing tasks
• Operator control messages and procedures
• Activating and reactivating of communication lines
• Specification of Operating System/360 control cards
The End

APPENDIX A: SAMPLE PROGRAM

The sample problem used here to illustrate the Queued Telecommunication Access Method will consist of three areas:

1. Message Switching Application. A message switching application involves messages sent from a remote terminal that have as their destination another terminal or a group of terminals and require no intermediate processing. In this application, the following functions will be provided for:

Receive

a. Control 1050 communication terminals and lines
b. Assemble messages received over communication lines
c. Code convert messages from line code to internal EBCDIC code.
d. Perform message-editing functions, such as time and date stamping, sequence number and source checking
e. Route messages according to destination code to either single or multiple destinations
f. Check for errors in messages
g. Perform corrective action when errors are detected
h. Perform queuing and logging of messages on a 2311 Disk Storage Drive.

Send

a. Insert sequence-out number of message
b. Format message for transmission to terminal
c. Code-convert message from the system EBCDIC code to the line code
d. Address terminal and transmit message

2. Inquiry Application. In the inquiry application described here, messages are sent from remote terminals that contain data to be processed, and a reply is sent back to the source terminal. A system data file is accessed by the processing program using an Operating System/360-supported access method. This file is on a 2311 Disk Storage Drive and is separate from the one used for the queuing of messages.

In addition to those functions listed under message switching, an inquiry application must also provide for the following:

a. Get message from the inquiry Process Queue
b. Access file record
c. Extract required information as indicated by message type
d. Compute value specified by inquiry message
e. Format reply message
f. Put message into the message source destination queue

3. Operator Control Program. In the operator control program described here, the control messages are entered into the system through an in-house system terminal. These messages are sent either to modify the polling list or to inquire about the line or process queue. This operator control program consists of one main program and two subroutines. The main program will be permanently resident in core storage. The two subprograms will be linked by the main program. They will not necessarily be resident in core storage.

In addition to functions a - g under message switching, an operator control program must also provide for the following:

a. Get message from the Control Process Queue
b. Examine message to determine type
c. Link to the program that will handle the message type
d. Operate on message request
e. Format reply message
f. Return to initial program
g. Put message to the in-house terminal destination queue

APPLICATION IMPLEMENTATION

For these applications, certain functions are supplied by QTAM while other functions must be provided by the user. The functions listed under message switching and used by the other applications will be completely specified through the use of this manual and will be handled by QTAM.

The remaining functions listed under inquiry application and operator control program will be programmed as separate tasks. These tasks will be programmed like other processing programs in the system and are completely the responsibility of the user. QTAM does, however, supply certain macro instructions so that the needed programs can communicate with the main QTAM Task (Message Control Task).

The QTAM Task, the Inquiry Task, and the Operator Control Task will all be assembled and executed through a normal job stream. Job control cards required by the job scheduler, such as JOB, EXEC, and DD statements, are not discussed here (see IBM Operating System/360 Job Control Language, C28-6539).

SYSTEM CONFIGURATION

The configuration of the system for the sample program is shown in Figure 1. Each device shown may be categorized into one of two groups.

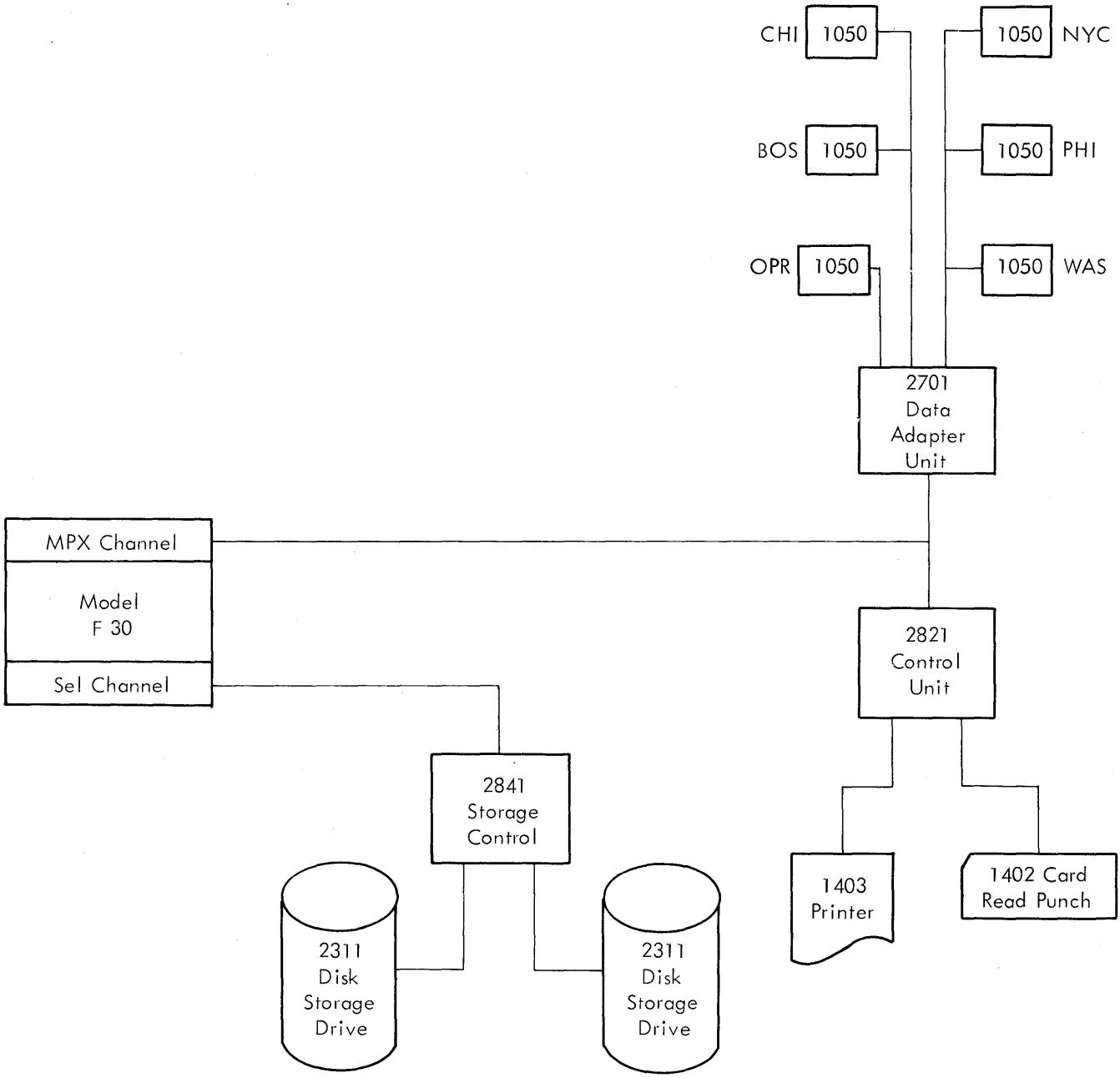


Figure 1. System configuration for sample problem

For Communications:

6 1050 Data Communication Systems
2 Half-Duplex Communication Lines
1 Data Adapter Unit
1 Multiplexor Channel
1 2311 Disk Storage Drive

For the Operating System:
System/360, Model F30 (64K)

1 Selector Channel
1 1052 Console
1 2311 Disk Storage Drive
1 Printer
1 Card Read Punch

JOB DEFINITION

Each task difference and peculiarity is defined in the following paragraphs.

Message Switching Application

Switched messages (which require no processing of message text) are to be routed to their destinations. Destinations specified in the input header may be any of the following.

- Single destination specified in the destination field of the header — for example, NYC.
- Multiple destinations specified in sequence in the header — for example, NYC PHI. . . .
- Distribution list specified in the header. For example, PDW would specify destinations contained in the terminal table list for PDW, that is, Philadelphia, Boston, Washington.

The switched message, when sent to its destination, will have inserted in its message header the in-time stamp and in-date stamp, and a sequence-out number. Output messages will be given priority over input, or received, messages.

Inquiry Application

Inquiry messages require processing of the message by a problem program resident in the central processing system. A reply message must be generated for transmission back to the sending terminal. The input message to be processed by the Inquiry routine must have the destination code INQ. The message entering the process queue (INQ) will have the date and time stamp inserted in the message header. The reply message generated by the inquiry processing program must contain the message type code P in the outgoing message header format. This outgoing message type will have inserted within the message header an out-date and time stamp and a sequence-out number. To allow for these insertions, 19 blank spaces must precede the first character of the reply message.

Operator Control Program

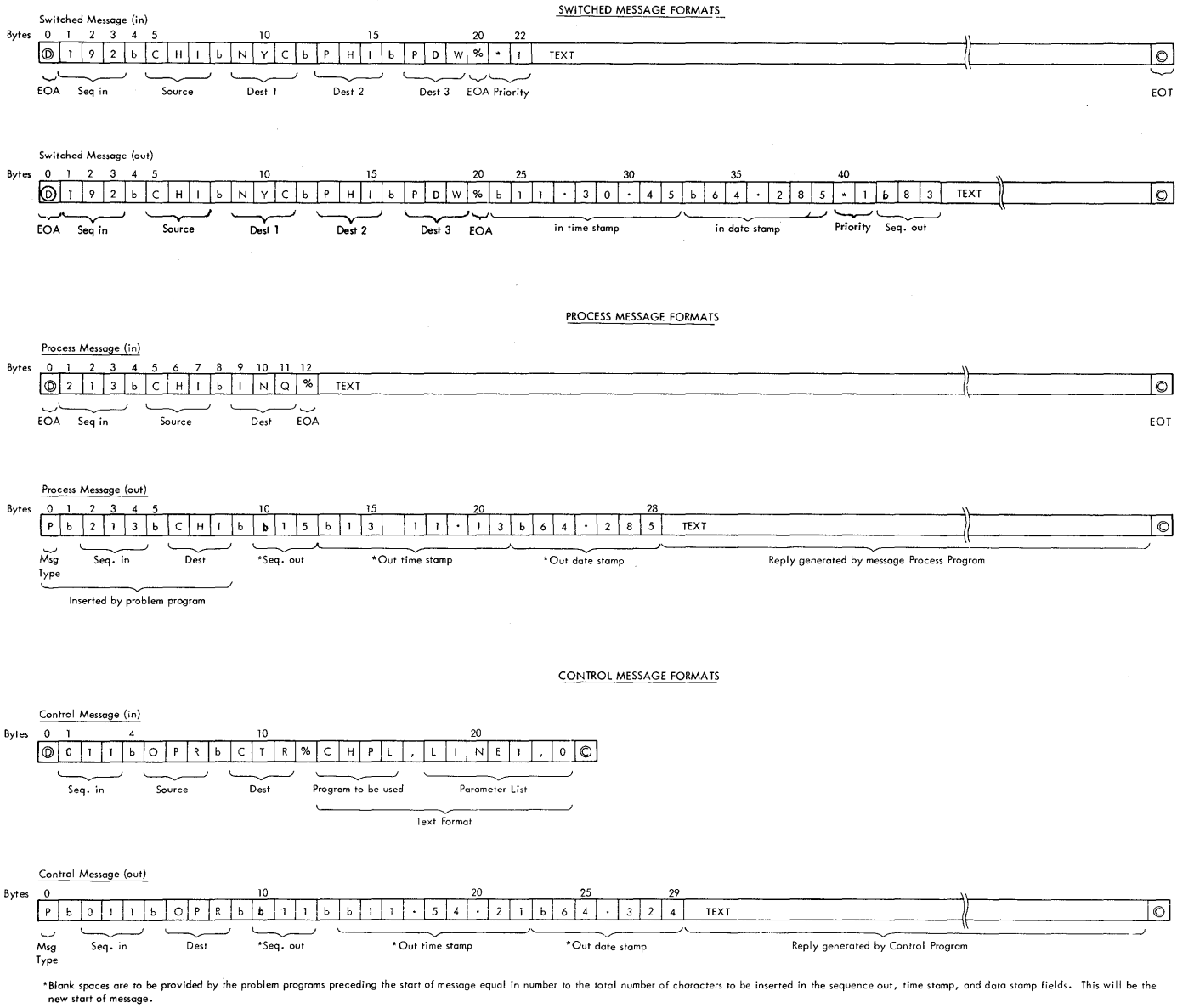
Control messages require processing of the message by a problem program resident in the central processing system. A reply message must be generated for transmission back to the in-house terminal (OPR). The message generated by the Operator Control program will have the destination code (OPR). The message entering the process queue (CTR) will have the date and time stamp inserted in the message header. These messages will be queued in main storage rather than on disk. The reply message is handled in the same manner as the Inquiry reply message.

The message text format of a control message might be like one of the following:

Text	Function
CHPL, LINE1, 0	Stop polling line 1
CHPL, LINE1, 1	Start polling line 1
CHPL, LINE2, 0	Stop polling line 2
CHPL, LINE2, 1	Start polling line 2
CPYQ, LINE1	Get line one queuing information
CPYQ, LINE2	Get line two queuing information
CPYQ, INQ	Get INQ queuing information
CPYQ, CTR	Get CTR queuing information

MESSAGE FORMATS

Figure 2 shows the message formats for the switched, inquiry, and control messages. The format of the messages is shown for both the receiving of the message from the terminal and the sending of the message or the reply to the terminal.

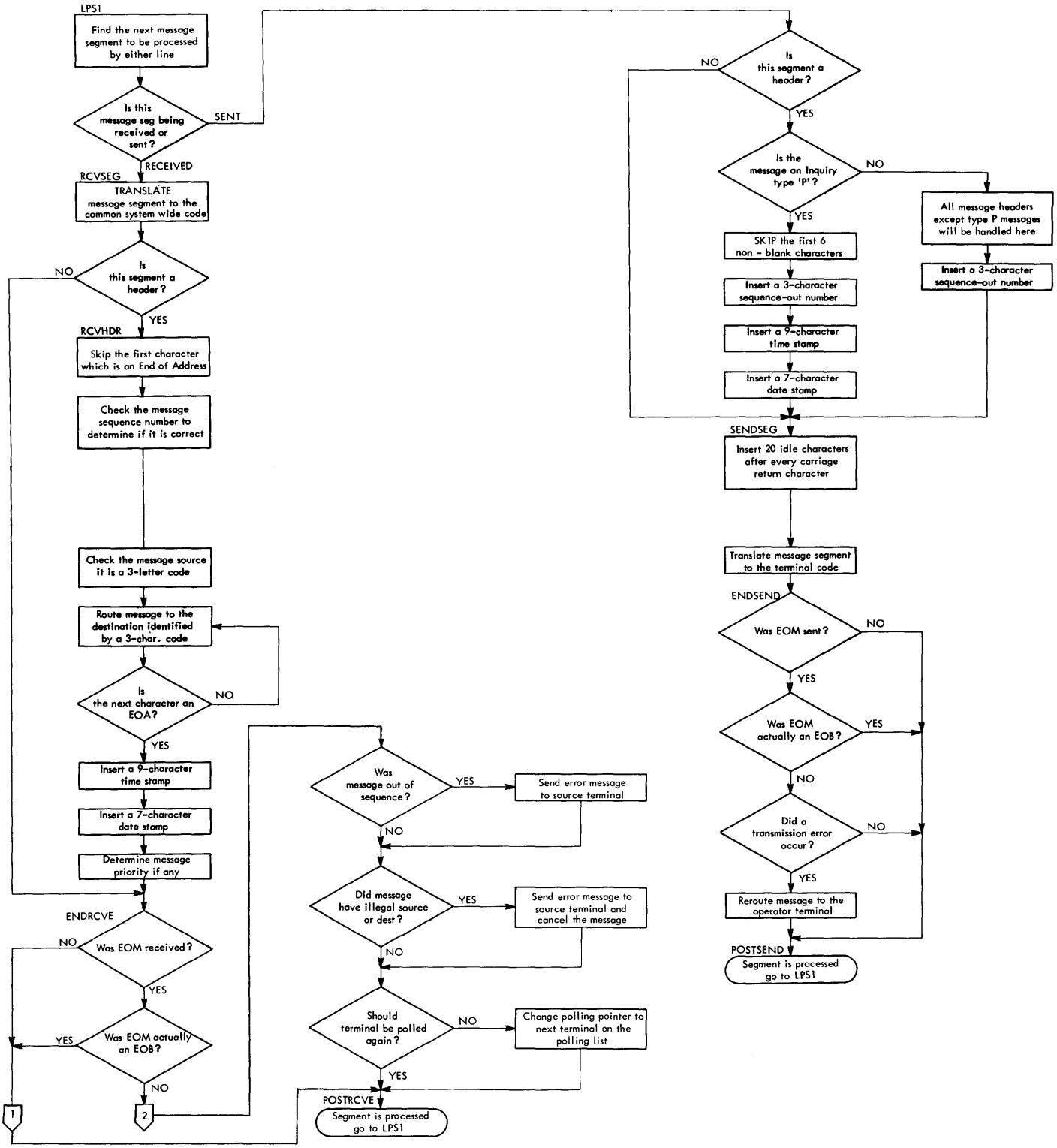


*Blank spaces are to be provided by the problem programs preceding the start of message equal in number to the total number of characters to be inserted in the sequence out, time stamp, and data stamp fields. This will be the new start of message.

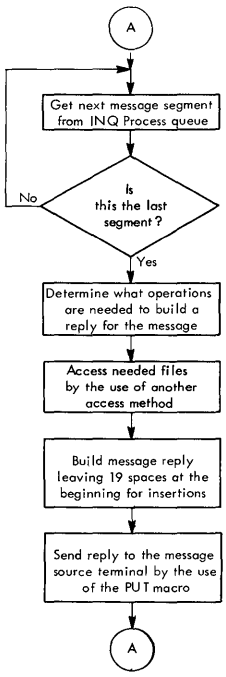
Figure 2. Formats for switched, inquiry, and control messages

PROGRAM FLOWCHART

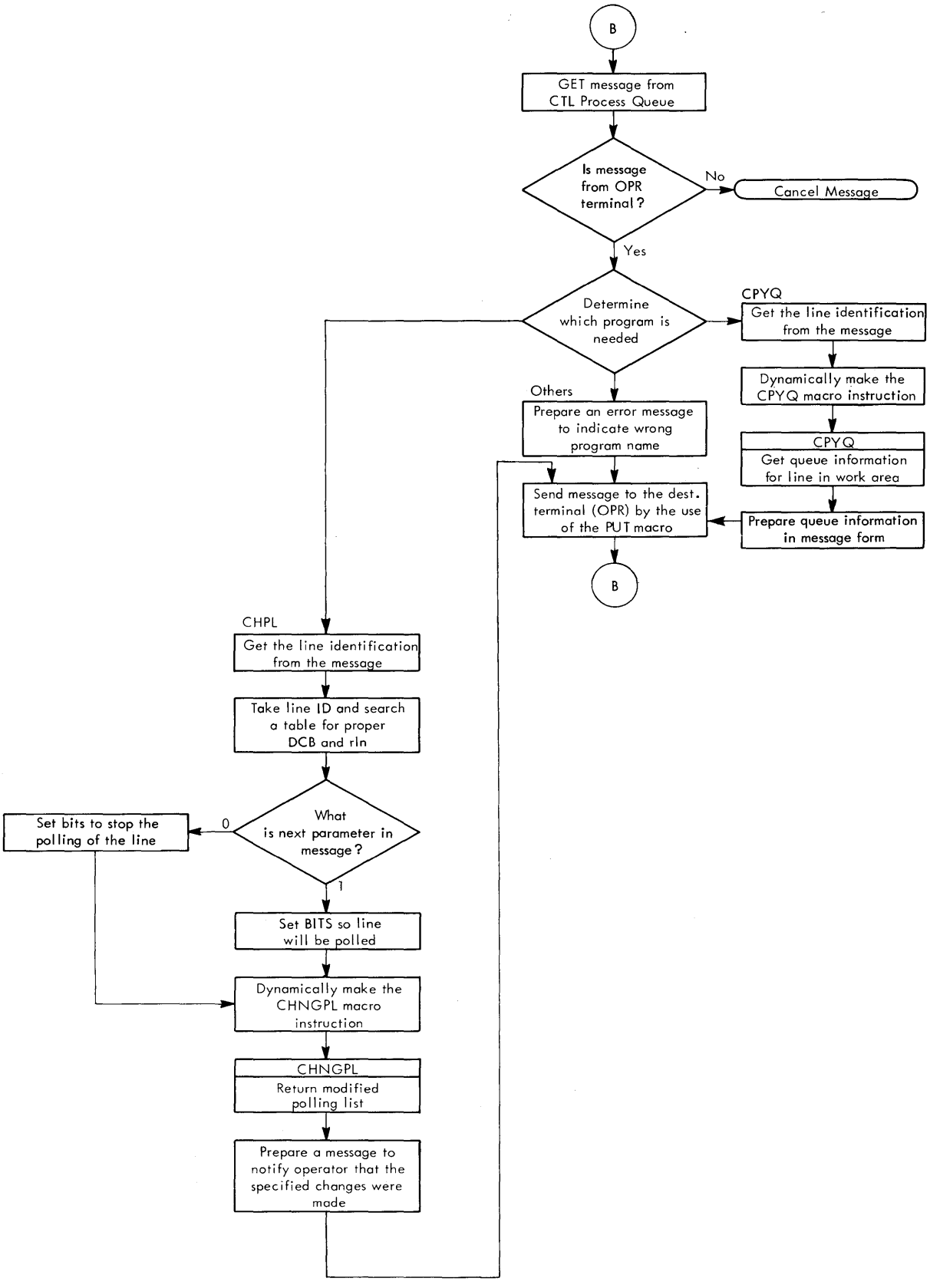
The following charts show the logic flow of the Line Procedure Specification portion of QTAM, the Inquiry Task, and the Operator Control Task. This is included for descriptive information only and does not necessarily represent the actual program logic flow of QTAM.



Message Processing Program



Operator Control Message Processing Program



MACRO CODING

The following pages illustrate the coding needed for each application. The Message Control Task, the coding of which is obtainable with this book, is completely coded. The Inquiry Application and the Operator Control Programs show methods of using the needed QTAM macro instructions and the overall structure of the processing programs but do not show the actual program coding.

MESSAGE CONTROL TASK OF QTAM

Name	Operation	Operand	Comments
STQTAM	OPEN	(DCBFILE, (INOUT), DCBLINE, (INOUT), DCBOPRLG, (INOUT))	Data Set Initialization makes ready for use the Operator in-house terminal communication line and the direct access storage device used for message queuing. It causes polling to be initiated on the line, and updating of queue status tables.
	ENDREADY		
LPS1	LPSTART	19	Receive Segment LPS Section — identifies the start of the LPS and reserves 19 spaces for the date stamp, time stamp, and sequence-out number at the beginning of the header segment.
	RCVSEG		Instructions between this delimiter macro and the next will service both the header and the text segments of the input message.
	TRANS	RCVE1050	Converts 1050 message characters to the common system-wide code EBCDIC.
	RCVHDR		Receive Header LPS Section. Instructions between this delimiter macro and the next will service only the header segment of the input message.
	SKIP	, C'#'	Causes all characters up to and including the # to be skipped, thus allowing the LPS to be in position for the first field (# is a translated (D)).
	SEQIN		Checks sequence of numbered messages for each terminal as they arrive. No operand is needed since the sequence number is ended with a field delimiter character (blank).
	SOURCE		Checks the validity of the source terminal code received in the message header against the code of the terminal that was polled. No operand needs to be specified since the field is ended with a field delimiter character. If the source code is invalid, an error is indicated in the error halfword of the line involved.

Name	Operation	Operand	Comments
	ROUTE		Routes the message to the destination queue(s) specified in the message header and checks the validity of the destination code against the terminal table provided by the user. If the destination code is valid, the message is queued for the specified destination.
	EOA	C'%'	Indicates that multiple destination routing is expected and must be checked for. The operand identifies % as the end-of-address character. This character must appear in the message header after the last destination code.
	TIMESTMP	9	Inserts the time of day in the header field. First character is blank. The operand (9) indicates the number of characters to be inserted.
	DATESTMP		Inserts the date in the header field. The insertion will consist of a blank character followed by a six-character date stamp.
	MODE	PRIORITY, C'*'	If the next character is an *, the character following the * will indicate the message priority.
	ENDRCVE		End Receive LPS Section. Macros between this delimiter macro and the next will service the message after the end of message is received.
	EOBLC		Allows the 1050 terminal to continue receiving after an EOB. It also provides for up to two retransmissions of the message segment if a transmission error is detected. If the error is not corrected, an error is indicated in the error halfword for this line.
	ERRMSG	X'3000', SOURCE, =C'bbb. MESSAGE NOT IN SEQUENCE'	Sends the error text specified to the source terminal, when the error type specified by the mask is detected. The header of the message causing the error will be placed at the beginning of the text. The mask 3000 is the bit configuration (in hexadecimal) used to test the halfword error indicator.
	ERRMSG	X'8600'SOURCE, =C'bbb. MESSAGE HEADER FORMAT ERROR. CORRECT AND RESEND'	Sends the error text specified to the source terminal when the error type specified by the mask 8600 is detected.
	CANCELM	X'8600'	Message containing error indicated by mask 8600 is canceled.
	POLLIMIT	LIMIT	Determines whether the terminal has sent the maximum number of messages allowed on a single polling pass. The operand is the symbolic name of an optional field in the terminal table that contains the limit of consecutive polls for each terminal.

Name	Operation	Operand	Comments
	POSTRCVE		Delimiter macro which indicates the end of the receiving section of the LPS.
	SENDHDR		Send Header LPS Section. Macros between this delimiter macro and the next will service only the header segment of the output message.
	MSGTYPE	C'P'	Message Type P Section. Determine whether the message is type P. If the next character is a P, the macros between this MSGTYPE macro and the next delimiter macro will handle these messages. The problem program must leave 19 spaces at the beginning of the inquiry reply message for the out- time stamp, date stamp, and sequence number.
	SKIP	6	Causes 6 nondelimiter characters (sequence-in number and destination code) to be skipped to position the LPS for the first field.
	SEQOUT	3	Sequentially number outgoing message by terminal. The operand (3) is the number of characters to be inserted (two digits preceded by a blank). Space must be reserved at the beginning of the message by the problem program for the sequence-out number.
	TIMESTMP	9	Inserts a 9-character time-of-day stamp in the outgoing header field. (The first of the 9 characters is a blank.)
	DATESTMP		Inserts a 7-character date stamp in the outgoing message header. The first character is a blank, followed by a 6-character date stamp.
	MSGTYPE		Remaining Message Type Section. All messages except type P messages will be handled by the macros between this MSGTYPE macro and the next delimiter macro (SENDSEG).
	SEQOUT	3	Sequentially number outgoing messages by terminal. The operand (3) indicates that a three-character output sequence number is to be inserted (two digits preceded by a blank). Space must be reserved by the use of LPSTART for message-switching messages.
	SENDSEG		Send Segment LPS Section. Macros between this delimiter macro and the next will service both the header and the text segments of the output message.
	TRANS	SEND1050	Translates output message using code conversion table name SEND1050.
	PAUSE	X'15', 20X'17'	Upon recognition of each carriage return character (X'15'), this routine will insert 20 idle characters (X'17') to provide time for the carriage to return.

Name	Operation	Operand	Comments
	ENDSEND		END SEND LPS SECTION. Macros between this delimiter macro and the next define functions to be performed after the message has been sent.
	EOBLC		Informs QTAM to continue sending upon recognition of an EOB. It also specifies retransmission of a message segment if a transmission error is detected. If the error is not corrected, an error is indicated in the error halfword for this line.
	REROUTE	X'0040', =C'OPR'	Causes a message to be queued for the terminal name OPR when the error type specified is detected.
	POSTSEND		Delimiter macro which identifies the end of the sending portion of the LPS. It also indicates the last instruction of this LPS.
DCBLINE	DCB	DDNAME =DDGROUP1,	Communication Line Group DCB Section — identifies DD statement name associated with this DCB.
		DSORG=CX,	Define DCB as a communication line group type.
		MACRF=(G, P),	I/O access level is GET/PUT.
		CPOLL=(POLLINE1, POLLINE2),	Symbolic names assigned to the polling list of the two lines.
		BUFRQ=2,	Number of buffer requests required by each line.
		ACLOC=13,	Relative position of the selecting address of terminals within its terminal table entry (see terminal table entry format).
		CLPS=LPS1	Identifies LPS1 as the name of the LPS for these two lines.
DCBOPRLG	DCB	DDNAME =DDGROUP2, DSORG =CX, MACRF=(G, P), CPOLL=(POLLINE3), BUFRQ=2, ACLOC=13, CLPS=LPS1	Identifies the operator in-house terminal line group.
DCBFILE	DCB	DDNAME=DDFILE,	Direct Access Device Queue DCB Section — identifies DD statement associated with this DCB.
		DSORG=CQ,	Defines DCB as a direct access device type.
		MACRF=(G, P)	Queue to be accessed at the GET/PUT level.
	TERMTBL	PBW, 3	Terminal Table Section — specifies the extent of the terminal table. PBW is defined as the last entry in the terminal table.

Name	Operation	Operand	Comments
LIMIT	OPTION	FL1	Limit is the symbolic name of the field in this terminal table which contains the limit of consecutive polls for each terminal.
CHI	TERM	L, DCBLINE,1, 6407640D, (3)	The six terminals and their parameters are here defined. L indicates outgoing messages are to be queued by line; DCBLINE is name of associated DCB; 1 is relative line number within the line group; 6407 is 1050 addressing code; 640D is 1050 polling code; the number in parentheses defines the maximum number of consecutive polls for each terminal to be inserted in the LIMIT field defined above.
NYC	TERM	L, DCBLINE, 2, 6207620D, (3)	
PHI	TERM	L, DCBLINE, 2, 6407640D, (2)	
BOS	TERM	L, DCBLINE, 1, 6207620D, (2)	
WAS	TERM	L, DCBLINE, 2, 6707670D, (1)	
OPR	TERM	L, DCBOPRLG, 6207620D, (5)	
INQ	PROCESS		Defines the symbolic names of the process queues in the terminal table, thus allowing INQ and CTR as valid destinations.
CTR	PROCESS	EXPEDITE	The CTR queue will be in main storage since EXPEDITE was used.
PBW	LIST	(PHI, BOS, WAS)	Define a distribution list of message destinations as PHI, BOS, WAS.
POLLINE1	POLL	(CHI, BOS)	Polling List Section — defines order of polling of terminals attached to line 1.
POLLINE2	POLL	(NYC, PHI, NYC, WAS)	Defines order of polling of terminals attached to line 2.
POLLINE3	POLL	(OPR)	Defines polling of the operator in-house terminal line. Polling is initiated upon execution of the open for the line group.
	BUFFER	DCBFILE, 10, 95	Buffer Section — provides internal storage buffering for the DCBFILE used for message queuing. Ten buffers are specified with 95 bytes per buffer.

INQUIRY MESSAGE PROCESSING TASK OF QTAM

STPROCES	OPEN	(PROCESSQ)	Opens the message control queue (INQ) so that messages can be obtained from the queue by the use of a GET macro instruction.
	OPEN ⋮	(REPLYQ)	Opens the output DCB; this causes a linkage to the Message Control Task so that the PUT macro instruction can send messages to their destination.
LOOP	GET	PROCESSQ, WORKA1	Queue Access Section. Get the next sequential segment from the queue (INQ) referenced by the DCB and place it in work area WORKA1. If no segments are available, a WAIT it implied.

(The message is now available for processing by the problem program. References to files for data to be used in preparing the reply, if required, would be under normal procedures of Operating System/360.)

Name	Operation	Operand	Comments
	PUT	REPLYQ, WORKA2	Place the processed message on the appropriate destination queue specified by the reply DCB parameter TRMAD. The terminal table entry name in the location specified by TRMAD will be the destination.
	B	LOOP	Branch to beginning to "GET" next message for processing.
WORKA1	DS	CL100	Define work areas.
WORKA2	DS	CL100	
SOURCE	DS	CL3	Defines a 3-character area referred to by the TRMAD parameter of the DCB. It will contain the message destination Terminal Table entry name.
PROCESSQ	DCB	DDNAME=INQ,	Process Program Queues DCB Section (Type 1) — name of the Process Queue Terminal Table entry name and associated DD statement for this process DCB.
		MACRF=G,	Program at the GET level.
		BUFRQ=2,	Two buffers will be queued ahead in core from the direct access device queue.
		SYMAD=ERROR,	Name of routine that will handle overflow of messages.
		TRMAD=SOURCE,	Specifies the name of the location that will contain where the message came from in the case of a GET macro, and the message destination in the case of a PUT macro.
		DSORG=MQ,	Defines DCB as a process program type.
		SOWA=100,	Work area size (in bytes).
		RECFM=S	Working unit is a segment.
REPLYQ	DCB	DDNAME=DDREPLY, MACRF=P,	Process Program Queues DCB Section (Type 2).
		TRMAD=SOURCE, DSORG=MQ, SOWA=100, RECFM=G	Refer to PROCESSQ DCB for explanation of parameters. RECFM=G means that the complete messages are being PUT to the destination.

OPERATOR CONTROL TASK OF QTAM

Name	Operation	Operand	Comments
STOPRTOR	OPEN	(PRCDCB)	OPENs the message control queue (CTR) so that messages can be obtained from the queue by the use of a GET macro instruction.
	OPEN	(RPLYDCB)	OPENs the output DCB; this causes a linkage to the message control task so that the PUT macro instruction can send messages to their destination.
NEXT	GET	PRCSDCB, WORKA1	Get the next sequential message segment from the queue (CTR) and place it in the work area.

(Determine which subprogram the message requests and LINK to it. The program that is linked will do the compilation required to accomplish the request. The program will execute CPYPL, CHNGPL, and CPYQ macros as required. The linked program will then prepare a reply for the operator and place it in WORKA2 and return to the main program.)

	PUT	RPLYDCB, WORKA2	Sends the reply message to OPR.
	B	NEXT	GET next message from queue, or wait until next message enters the queue.
WORKA1	DS	CL100	Define work areas.
WORKA2	DS	CL200	
SOURCE	DS	CL3	Provide area that will contain the source terminal entry name. The name will be examined by the operator control program to see if it is OPR. If not, the message will be canceled (see program logic flow).
DEST	DS	C'OPR'	Provide the destination terminal entry name. All replies will then go to this terminal (OPR) when a PUT macro references the DCB named REPLYDCB.
PRCSDCB	DCB	DDNAME=CTR, MACRF=G, BUFRQ=2, SYNAD=ERROR, TRMAD=SOURCE, DSORG=MQ, SOWA=100, RECFM=S	
RPLYDCB	DCB	DDNAME=DDRPLY, MACRF=P, TRMAD=DEST, DSORG=MQ, SOWA=200, RECFM=G	

APPENDIX B: MESSAGE HEADERS FOR QTAM

A message header is a prefix to a message received or sent via communication lines containing information for:

Routing of the message
Sequence numbering
Source validity checking
Time stamping
Date stamping
Priority assignment
Identification of message type
Specification of special user functions

The message header may contain other information not of immediate use for handling of the message — for example, various device-required characters, identification fields for operator use, or fields required for later processing of the message.

A particular telecommunications system application using QTAM may have all or none of this information in the message headers, depending on the desired flexibility of the system and the functions to be performed.

QTAM provides a flexible, high-level macro language that can be used to specify header analysis procedures for nearly all reasonable message headers and supported communications equipment. This document describes desirable and necessary features for message headers being analyzed by QTAM.

In general, message headers are for input messages (messages received from terminals via communication lines), output messages (messages sent to terminals via communication lines), or both, as in the case of a message-switching application where the same header (and text) is sent as is received, except for additions made by the QTAM program.

Message headers change in appearance as they proceed through, and are operated on by, various parts of the communications system. The following discussion is concerned only with the message headers as they are prepared at the terminal for sending.

A number of considerations should be made when preparing message header formats designed for easy and efficient use of QTAM:

- Line control characters. Depending on the particular terminal device and communication line control discipline used (for example, Teletype 28ASR terminals in an 83B2 control system), particular control characters or sequences of characters will be needed in the message header. The actual characters used are very device-dependent and will not be covered here. To determine what these characters are for a particular system, it is best to consult manuals describing the particular devices used.

  Just who is responsible for placing specific control characters in a message header also depends on the devices used. Usually it is a shared responsibility of both the person preparing the message and the terminal device used to transmit it.* This information is also to be derived from the manual for the particular devices involved.

- Header field definition. The content of a message header is contained within "fields" consisting of one or more characters each (Figure 3). Each such field contains information for a particular operation or function to be performed on that message (header and text).

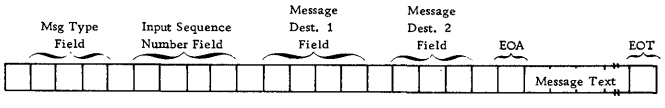


Figure 3. Example of message header fields

- Fixed or variable length. Message header fields must in general be of a fixed length for each particular field. Exceptions to this are (1) source and destination fields of a message that may consist of any number of characters up to a maximum of eight bytes (characters), and (2) input sequence number fields, which may be any length up to five bytes containing four digits (the first byte is always blank).
- Field separation. Header fields may be separated by any number of blank characters. If blank characters are used to separate the fields, the length of the fields need not be specified in the QTAM macros. For variable-length fields (as discussed above) blank characters must be used to delineate the field, since a fixed length cannot be specified. If fields are not separated by blank characters, the exact length of each field must be specified.
- Order of fields within header. In message headers for a message-switching application using QTAM, all fields concerned with receiving the message must precede those concerned with sending it (Figure 4).

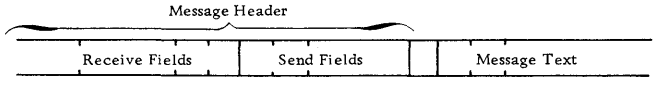


Figure 4. Order of header fields for message switching

*The same is true for reception of the message by the CPU. Certain control characters will be deleted by the control unit and others will be passed through to be handled by the QTAM program.

- Header length. The length of the message header must be less than or equal to that of the buffer length specified in the QTAM macro program, minus 32 bytes used for the header prefix. Conversely, when specifying the macro program for QTAM, the length of the buffers specified must be sufficient to contain the maximum message header plus 32 bytes for the prefix.
- Message type identification. Message headers that require different handling procedures or have different formats from other messages on a line may be identified by a special character or sequence of characters. It is desirable that this "message type" identification be the first field of the message header. This makes possible early separation of the various types of messages involved so that the proper procedure can be followed by each.
- Skipping unwanted fields. Fields that are not checked or used by header analysis may be skipped by either specifying within the QTAM macros the number of nonblank characters in the field to be skipped or by identifying the end of the field by a special character configuration in the message header. This configuration can be from one to eight nonblank characters in length. (See Figure 5.)

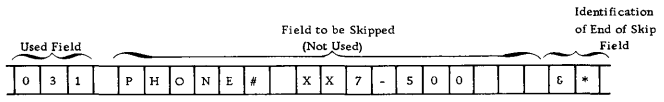


Figure 5. Skipping a variable-length field

- Message priorities. Priority of a message can be identified by either a special character in the message header followed by the message priority or by just specifying the message priority. A special character should be used when all messages are not necessarily given a priority, while the second case can be used when a priority is inserted in every message. Message priority levels range from 1, 2, ---, 9, A, B, ---, Y, Z, where Z is the highest priority and 1 is the lowest.
- Destination fields. When multiple destinations are desired in the message header, a special character or sequence of characters must be reserved to identify the last destination (address) listed. This character(s) is termed the end of address (EOA) character(s). Each destination may be separated by a blank character(s) from the others. If this is done, the length of each destination will not have to be specified within the QTAM macro program, and the destinations may then be variable in length (Figure 6).

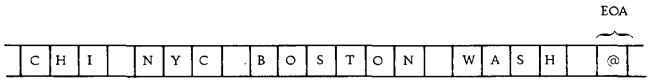


Figure 6. Multiple destination fields of variable length

If there is never more than one destination, the End of Address field is not needed, and if the destination is fixed it does not even have to be included in the message header.

- Special handling. A special character may be placed in the message header to specify that the message is to be handled in the "initiate" mode. When a message is handled in the "initiate" mode, message segments may be either processed or sent to a destination before the entire message is received. The special character is needed only when not all of the messages of a given type are to be handled in this mode.
- Conversational mode. A special character may also be placed in the message header to identify that the message is to be handled in "conversational" mode. Receipt of a message in this mode will imply that the next terminal on the line will not be "polled" until a further exchange of messages has occurred. The special character is needed only when not all of the messages operate in this mode.
- User routines. Just as for the "initiate" and "conversational" modes, a user routine can be specified to operate only on certain messages containing a special character. Again, this special character is needed in the message header only if not all messages require the particular user routine.

Printed in U.S.A. C20-1640-0

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York 10601