Advanced Communications Function for TCAM, Version 3
Program Number 5665-314

IBM

TCAM

Installation, Resource Definition, and Customization Reference (MVS)

SC30-3236-1
File No. S370/4300/30XX-50

Second Edition (June 1985)

This edition applies to Version 3 of the Advanced Communications Function for TCAM, Program Product 5665-314. This edition also applies to all subsequent releases and modifications unless otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30XX and 4300 Processors Bibliography,* GC20-0001, for the editions that are applicable and current.

Preface

This publication is a reference publication that contains detailed information on the macro instructions used with the Advanced Communications Function for the Telecommunications Access Method (TCAM). This program product supports IBM's Systems Network Architecture (SNA) and operates with OS/VS2 Multiple Virtual Storage (MVS). This manual is for a system programmer who is thoroughly familiar with the *TCAM Installation Guide* and is ready to code an MCP. It is essential that the user refer to the *TCAM Installation Guide* for a functional explanation of TCAM; this book is not tutorial and contains only an explanation of the macros with specific operand-coding considerations.

Before using this book, the reader should also be familiar with the contents of the *TCAM General Information*, and the *TCAM Application Programming* manuals. The reader is also assumed to have a basic understanding of SNA.

The first chapter of this book explains how the book is organized and the macro format used throughout. The remaining chapters and appendixes cover TCAM reserved option fields, the message error record, internal and transmission codes, internal TCAM macros available to the user, and aids for TCS-Brokerage users.

Related Publications

The following list includes the complete title and order number of manuals referred to in this publication:

Short Title	Full Title	Order No.
TCAM Utilities	*Advanced Communications Function for TCAM, Version 3 Service Facilities, System Service Programs, and Utilities*	SC30-3138
Data Management Macro Instructions	*OS/VS2 MVS Data Management Macro Instructions*	GC26-3873
Data Management Services	*OS/VS2 Data Management Services Guide*	GC26-3875
SNA Format and Protocol Reference Manual	*Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic*	SC30-3112
SNA Reference Summary	*Systems Network Architecture Reference Summary*	GA27-3136

Short Title	**Full Title**	**Order No.**
TCAM Application Programming	*Advanced Communications Function for TCAM, Version 3, Application Programming*	SC30-3233
TCAM Diagnosis Guide	*Advanced Communications Function for TCAM, Version 3, Diagnosis Guide*	SC30-3234
TCAM Diagnosis Reference	*Advanced Communications Function for TCAM, Version 3, Diagnosis Reference*	LY30-5560
TCAM General Information	*Advanced Communications Function for TCAM, Version 3, General Information*	GC30-3235
TCAM Installation Guide	*Advanced Communications Function for TCAM, Version 3, Installation, Resource Definition, and Customization Guide*	SC30-3237
TCAM Messages	*Advanced Communications Function for TCAM, Version 3, Messages*	SC30-3238
TCAM Migration	*Advanced Communications Function for TCAM, Version 3, Migration*	SC30-3251
TCAM Operation	*Advanced Communications Function for TCAM, Version 3, Operation*	SC30-3239
TCAM Planning Guide	*Advanced Communications Function for TCAM, Version 3, Planning Guide*	SC30-3240
TCAM Program Reference Summary	*Advanced Communications Function for TCAM, Version 3, Program Reference Summary*	LY30-5561
VTAM Planning and Installation Reference	*Advanced Communications Function for VTAM, Planning and Installation Reference*	SC27-0584
VTAM Programming	*VTAM Programming*	SC27-0611
IBM System 370 Principles of Operation	*IBM System 370 Principles of Operation*	GA22-7000

Contents

Figures

Chapter 1. Organization of This Book

This chapter explains the organization of this book and the macro format used throughout. The remaining chapters and appendixes contain the following information:

- Chapter 2, "TCAM Macro Instructions," contains all the general-purpose TCAM macros in alphabetic order and supplies information on the macro function, possible default value, coding format, notes on specific operands, and return codes.
- Chapter 3, "Option Fields Reserved for Use by TCAM Functions," describes the functions and initialization requirements of certain option fields that are reserved for TCAM functions.
- Appendix A, "Message Error Record," explains the testing of the message error record bits and lists the message error bit definitions.
- Appendix B, "Internal and Transmission Code Charts," contains two sets of charts that include character sets and hexadecimal code for extended binary-coded decimal interchange code (EBCDIC), USASCII 7-bit and 8-bit ASCII codes and the valid hexadecimal representations of graphic and control characters for these in collating sequence.
- Appendix C, "Internal TCAM Macros Available to the User," describes specialized macro instructions that are normally issued by TCAM code, but that may also be coded by the user.
- Appendix D, "Aids for TCS-Brokerage Users," describes an option field and macros that are supported to facilitate migration from TCS-Brokerage to TCAM.

TCAM Macro Descriptions

Macro instructions included in this publication are described in a standardized way. Following the name of each macro is a list of bulleted items briefly setting forth the functions of the macro and giving information on its use. This list is followed by a more detailed description of the use of the macro and any restrictions on its use. The prose description is followed in turn by two lists that set forth requirements for the use of the macros. One of these, titled "Supported Resources and General Requirements," describes certain prerequisites for using message-handler and resource-definition macros, based upon type of station and message content. The second list, titled "Valid Subgroups," tells which message-handler subgroups each message-handler macro is valid in. These two lists are described in more detail later in this chapter.

The two lists of requirements are followed by a formatted illustration of each macro instruction that includes for each operand a description of the function, format, and default value for that operand and a series of notes pertaining to that operand. The formatted illustration is described in more detail later in this chapter.

Supported Resources and General Requirements

Each macro in this publication has included in its description a list of restrictions under the category of "Supported Resources and General Requirements". For message-handler macros and resource-definition and table-definition macros, this list will include one or more character strings (described below) that are meant to indicate the environment in which the macro will be used. For other macros, the heading for this category will be followed by the phrase "not applicable," indicating that this macro is not a message-handler, resource-definition or table-definition macro.

Character strings that may be coded following the "Supported Resources and General Requirements" heading are as follows:

SNA	For a resource-definition macro, *SNA* means that the macro may be used to define a SNA station. For a table-definition macro, *SNA* means that the table is used to support SNA resources. For a message-handler macro, *SNA* means that the macro may be coded in a DMH to process messages being received from a resource in the same TCAM system or from a resource in another TCAM system (if coded in the incoming group) or sent to such a resource (if coded in the outgoing group).
APP	For a resource-definition macro, *APP* means that the macro may be used to define an TCAM application program. For a message-handler macro, *APP* means that the macro may be coded in an application message handler (AMH) to process messages being received from an application (if coded in the incoming group) or sent to an application (if coded in the outgoing group).
ALL	*ALL* is equivalent to SNA and APP and means that the macro may be used for all external LUs and applications supported by TCAM. The designation ALL may appear alone or with FHP.
FHP	*FHP* means that the message-handler macro will execute correctly only if the message being processed contains a fixed header prefix (FHP).
IEDTCSD	*IEDTCSD* means that this macro requires certain DSECTs and/or labels in order to be assembled correctly. These DSECTs and labels are included automatically with the model MCPs. If this macro is to be assembled as part of an MCP which is not a model MCP, the IEDTCSD macro should be coded after the executable code in the MCP to generate the required DSECTs and labels.
Not Applicable	This macro is not a resource-definition or message-handler macro and does not require the labels generated by the IEDTCSD macro in order to be executed.

In addition to the restrictions indicated under this category, further restrictions on usage may be indicated in the description of the macro and its operands.

Valid Subgroups

Some TCAM functional macros are restricted to being coded in certain subgroups of a message handler. An TCAM message handler is divided into an incoming and outgoing group of macro instructions. These two groups are further divided into subgroups as follows:

Incoming Group

- *Inblock subgroup:* Processes all incoming message segments.
- *Inheader subgroup:* Handles only those incoming message segments that include all or part of a message header.
- *Inbuffer subgroup:* Processes all incoming message segments.
- *Inmessage subgroup:* Executes after a complete message has arrived at the MCP.

Outgoing Group

- *Outheader subgroup:* Handles only those outgoing message segments that include all or part of a message header.
- *Outbuffer subgroup:* Processes all outgoing message segments.
- *Outmessage subgroup:* Executes after a complete message has been sent or when contact cannot be established with a station.

These subgroups are controlled by delimiter macros that must be coded in a specific sequence. See "Coding the Message Handler" in the *TCAM Installation Guide* for detailed instructions on coding the message handler.

TCAM Macro Formats

A format illustration accompanies each macro instruction in this publication. The illustrations indicate which operands must be coded exactly as shown, which are required, which are variable, etc. The positional operands are listed first, followed by the required operands in alphabetic order, and then the optional operands in alphabetic order. The operand descriptions following the format are combined in alphabetic order for ease of reference. The conventions stated to describe the operand are:

1. Keyword operands are described by a three-part structure that consists of the (uppercase) keyword, followed by an equal sign (both of which must be coded), followed by a lowercase variable or an uppercase fixed value to be specified by the user.

   Examples: KEYWORD=value, METHOD=NORMAL

2. Positional operands are described by either lowercase names, which are merely convenient references to the operand and are never coded by the programmer, or by an uppercase operand that is used exactly as shown. The programmer replaces the lowercase operand by an allowable expression as defined in the macro description.

   Examples: *qtype, destname, mask,* MESSAGE

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

4. Lowercase letters and terms represent information that must be supplied by the programmer. Restrictions (such as the maximum value that may be specified) are stated in the description of the operand under the macro description.

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

6. {A} Information contained within a vertical stack of braces represents alternatives,
   {B} one of which must be chosen by you when coding the operand in which the braces appear.

7. [...A] Information contained within brackets represents an option that can be included or omitted, depending on the requirements of the program. In accordance with the rules for the assembler language, a comma must be coded if a positional operand is omitted.

   [...A] If more than one alternative is included within a single set

   ...B of brackets, either of the alternatives may be chosen, or the operand may be omitted (that is, none of the alternatives need be chosen).

8. Operands that are not enclosed within brackets are required.

9. [{A}] Underlined elements represent default values.
   {B}
   {*C*}

In describing and illustrating the macro instructions, the following conventions are used:

- *Register Notation:* Unless otherwise specified, a register (2 through 11) may be used. The number (or name if allowed) of the register must be enclosed in parentheses.

- *Error Returns:* Error return codes are returned in register 15.

Other Conventions Used in This Book

The TCAM message control program is coded using the basic assembler language. All coding examples included in this book use assembler-language coding conventions.

Chapter 2. TCAM Macro Instructions

This chapter contains all the general purpose TCAM macros in alphabetic sequence. For an explanation of the macro format and organization of this chapter, see "Chapter 1. Organization of This Book."

The macro descriptions in this chapter contain coding details. Before attempting to code an MCP, read the conceptual information about these macros in the *TCAM Installation Guide*.

ACCTING Macro

The ACCTING macro:

- Provides the TCAM user with a method for gathering detailed accounting information and sending the collected data to a user-written application program in the same node or in another node (as described in the chapter of the *TCAM Utilities* manual called "Dynamic Accounting Service Facility").
- Causes a special action flag to be set signaling that a half-section of the system accounting area is filled.
- Is required once in the MCP initialization section, after an INTRO macro and before a READY macro, to create the accounting table, if the dynamic accounting service facility is desired.
- Is required in the appropriate message handler subgroups where accounting data is needed to create accounting records.
- Is used in conjunction with the TESTBR, SPECACT, and SENDMSG macros.

The KEY=, QNAME=, NODE=, RESOURC=, and TC= operands specify the destination of the storage areas when they are full. You may specify a key, an external LU, or application program name, or—if the extended networking facility is being used—TCAM address (node identifier and resource identifier). If NODE= and RESOURC= are coded, specify the transmission category. KEY=, QNAME=, and the NODE= and RESOURC= operands are mutually exclusive.

Supported Resources and General Requirements: ALL

Valid subgroups: Inblock, inheader, inbuffer, outheader, and outbuffer.

When Used in the MCP Initialization Section

NAME	OPERATION	OPERANDS
[symbol]	ACCTING	KEY=chars
		,QNAME=chars
		,NODE=n
		,RESOURC=n
		[,AREASZE={2000}] {n }
		[,DSECT={NO }] {YES}
		[,TC={n}] {2}
		[,USERFHP=name]

When Used in the Inheader, Inbuffer, Outheader, and Outbuffer Subgroups

NAME	OPERATION	OPERANDS

[symbol]	ACCTING	ACCTNUM=n
		[,INCLFHP={NO }]
		{YES}
		[,OPTION=opfield]
		[,USERDAT={(reg)}]
		{name }

```
ACCTNUM=n
```

Function: Provides an identifying account number to be used as a prefix to all accounting records gathered through execution of the particular ACCTING macro in which this operand is coded.

Format: *n* should be either an explicit decimal number from 1 to 65,535 or a symbol that has previously been equated to a decimal value from 1 to 65,535. It is your responsibility to assign to this operand a value that is within the prescribed limits.

Default: None. Specification is required.

```
AREASZE={2000   }
        {integer}
```

Function: Specifies the number of bytes of main storage to use to gather the accounting records.

Format: *integer* is a decimal integer from 50 to 20,000.

Default: AREASZE=2000.

Note: Since half of the storage area may be scheduled for routing while the other half receives accounting information, the size of the area specified is divided into two storage areas.

```
DSECT={NO }
      {YES}
```

Function: Generates the dummy sections describing an accounting record and an accounting control block.

Format: YES or NO.

Default: DSECT=NO.

Note: If YES is specified, all other operands are ignored and accounting record and accounting control block dummy sections are generated.

```
INCLFHP={NO }
        {YES}
```

Function: Specifies whether the size of the fixed prefix header should be included in the character count placed in the accounting record being built.

Format: YES or NO.

Default: INCLFHP=NO.

Note: If a user FHP is used, the FHP is built when a message arrives in the message handler from an external LU or an application program. ACCTING macros in external LU and application program message handlers would specify NO because the FHP was not transmitted. If the extended networking facility is not being used, specify INCLFHP=NO. However, in internodal message handlers (used when doing extended networking), the FHP is often transmitted; in this case specify YES.

```
KEY=chars
```

Function: Specifies a key to be used to route accounting data. The key must also appear as the KEY= operand of a KEYDEF macro.

Format: chars is a character or hexadecimal data string 8 bytes long, with framing CL8'' or XL8'' characters specified.

Default: None. Specification of *either* KEY= or QNAME=, or of both NODE= and RESOURC= is required.

Note: KEY, QNAME, and NODE/RESOURCE are mutually exclusive.

```
NODE=n
```

Function: Specifies that the ACCTING macro should use the node table to route the data. This operand provides the node identifier portion of the TCAM address of the destination to which accounting data should be sent.

Format: n should either be a decimal integer from 1 to 245 or a symbol that has been previously equated to a decimal value from 1 to 245. You must assign to this operand a value that falls within the prescribed limits.

Default: None. Specification of *either* KEY= or QNAME=, or of both NODE= and RESOURC= is required.

Note: If NODE= is specified, specify RESOURC=.

OPTION=opfield

Function: Specifies the name of an option field defined for the origin (if executed in the incoming group) or destination (if executed in the outgoing group) of the message being processed, the contents of which are to be included in the accounting record being built.

Format: opfield must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Note: The maximum length of the accounting record is 255 bytes. The fixed portion of the accounting record is 20 bytes long. Therefore, the combined length of data stored in the accounting record as a record of coding the OPTION= and USERDAT= operands cannot exceed 235 bytes.

QNAME=name

Function: Specifies the TCAM name of the destination station or application program to which accounting data is to be sent.

Format: name must conform to the rules for assembler language symbols. Framing characters are not used.

Default: None. Specification of *either* KEY=or QNAME=, or of both NODE= and RESOURC= is required.

Note: name is is the name of a single, cascade, or process entry in the terminal table.

RESOURC=n

Function: Specifies the resource identifier of the TCAM address.

Format: n should either be an explicit decimal integer from 100 to 65,535 or a symbol that has previously been equated to a decimal value from 100 to 63,535.

Default: None. Specification is required if NODE= is coded.

Note: If the NODE= operand specifies the local MCP's node identifier, the accounting data is routed within the node on the basis of the RESOURC= operand.

symbol

Function: Name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

```
TC={n}
   {2}
```

Function: If NODE= and RESOURC are coded, this operand specifies the transmission category (and associated queue) to be used for data destined for a remote node.

Format: n is a decimal integer not greater than the number of transmission categories defined in this MCP; n may not exceed 16, the maximum number of transmission categories allowed in any MCP.

Default: TC=2.

Note: TC=n indicates that accounting data destined for a remote node should be placed on the transmission category *n* queue to the remote node, as specified by the user in the TC= operand of the NODEDEF macro defining the remote node.

TC=1 indicates that the queue specified by the NODEDEF macro TC1 operand should be used for accounting data.

Refer to the NODEDEF macro description for an explanation of transmission categories.

```
USERDAT={(reg) }
        {(name)}
```

Function: Provides the address of user data to be included in the accounting record.

Format: *(reg)* or *(name)* can be specified either as an explicit decimal integer from 2 through 12 or as a symbol that has previously been equated to a decimal value from 2 through 12; the integer or symbol used must include framing parentheses. *name* must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Note: *(reg)* specifies the general register that contains the address of the user data field. *name* is the symbolic name of a storage area containing the user data field.

The first position of the user data field must be a 1-byte length field containing the length of the data to be inserted (not including the 1-byte length field itself). This length byte is not stored with the data.

The maximum length of the accounting record is 255 bytes. The fixed portion of the accounting record is 20 bytes long. Therefore, the combined length of data stored in the accounting record as a result of coding the OPTION= and USERDAT= operands cannot exceed 235 bytes.

```
USERFHP=name
```

Function: Specifies the address of a user-coded FHP to be added to the front of the data areas before they are queued.

Format: *name* must conform to the rules for assembler language symbols.

Default: None. Specification is optional. If USERFHP= is omitted, the ACCTING macro will provide an FHP.

Note: *name* is the label of an area of addressable storage containing an appropriate FHP.

Return Codes

The following return code is set in register 15:

Code	**Meaning**
X'00000000'	Successful execution
X'00000004'	Both areas are full

Note: No ACCTING records will be created for zero length buffers (error buffers).

ALTNAME Macro

The ALTNAME macro:

- Gives an alternate name, or alias, to the terminal table entry that is specified by the TERMINAL, TLIST, TPROCESS, or LOGTYPE macro that precedes the ALTNAME macro
- Is optional among macros defining the terminal table
- May be issued more than once
- Must be placed immediately following a TERMINAL, TLIST, TPROCESS, or LOGTYPE macro instruction, either alone or together with other ALTNAME macro instructions
- May not be specified as the last macro instruction defining the terminal table

There are many uses for the alternate name facility. Two separate office units might use the same station but identify themselves separately. An ALTNAME macro instruction is the easy solution to this duplication.

An application program may send messages to many different external LUs. This program can be tested—using only a single external LU—by specifying the many different message destinations as alternate names for the same external LU.

Also, a working application that routes traffic to a given destination does not have to be altered if the destination is to be changed. The generated external LU name can be specified in the MCP as an alternate name for the new external LU and the application program can remain unaltered.

If the ALTNAME macro is used to assign an alternate name for any terminal-table entry other than an external LU, you must define the REALNAME option field. For more information on the REALNAME option field, see the description in the chapter titled "Option Fields Reserved for Use by TCAM Functions."

Supported Resources and General Requirements: ALL.

Valid subgroups: Not applicable.

NAME	OPERATION	OPERANDS
[symbol]	ALTNAME	(no operands)

`symbol`

Function: Specifies an additional name of the terminal table entry.

Format: *symbol* must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. The name must be specified.

Return Codes

None.

CANCELMG Macro

The CANCELMG macro:

- Cancels messages either unconditionally or conditionally when certain errors occur
- If specified in the inmessage subgroup, must be the first functional macro executed (one or more CANCELMG macros may be specified in the same subgroup, but only one can be executed)
- Is valid in the inmessage subgroup only if a valid FORWARD macro has been issued.

CANCELMG causes immediate cancellation of an entire message if any errors specified by the error mask operand are also indicated in the message error record. (See the "Appendix A. Message Error Record" for a description of the message error record.) A canceled message is not sent to its destination. If there are multiple destinations, the canceled message is not sent to any of them. However, a single buffer is placed in the destination queue for each canceled message for use in inmessage processing. This buffer remains in the queue until it is dequeued by a send operation to a station or an application program. For a way to avoid overflowing a main-storage queue with canceled header buffers, see the THRESH= operand on the FORWARD macro. The ERRORMSG (HEADER=NO) or MSGGEN macro may notify the origin of the error, or the REDIRECT macro may send the message that is in error to a selected destination.

Note: If an all-zero mask is specified or if the mask is omitted, the message is canceled unconditionally.

CANCELMG may be executed in an outheader subgroup. When CANCELMG is executed in outheader, no attempt is made to send the message to a destination. This message will be sent through the outgoing group of the MH as a zero length buffer.

You must not code a CANCELMG macro preceding any ERRORMSG (HEADER=YES), SENDMSG or REDIRECT macro in an inmessage subgroup that handles messages sent to a main-storage-only queue. A CANCELMG macro coded after these macros is not effective.

If the CANCELMG macro is executed in the inmessage subgroup for a lock-mode message, the lock is not broken.

Supported Resources and General Requirements: ALL (if APP, in inmessage subgroup only).

Valid subgroups: Inmessage, outheader.

CANCELMG

When Used in an Inmessage Subgroup

If CANCELMG is coded in the inmessage subgroup, it must be the first functional macro executed.

NAME	OPERATION	OPERANDS
[symbol]	CANCELMG	[mask] [,CONNECT={AND}] {<u>OR</u> }

Note: If multiple CANCELMG macros are issued in the same message subgroup, no other macros may be issued between CANCELMG macros.

When Used in an Outheader Subgroup

The CANCELMG macro cancels a message in the outheader subgroup. No attempt is made by TCAM to send the message; it is instead dequeued, and bits 24 and 27 of the message error record are set.

NAME	OPERATION	OPERANDS

[symbol]	CANCELMG	[conchars]
		[,BLANK={<u>YES</u> }]
		{NO }
		{char}

```
BLANK={YES }
      {NO  }
      {char}
```

Function: Specifies whether EBCDIC blank characters are to be ignored when encountered in the character string in the message header being compared to the string specified by the *conchars* operand, or whether blanks are to be part of the header string when encountered in it. If EBCDIC blanks are to be counted as part of the header string, this operand also specifies whether some other hexadecimal character is to be ignored when encountered in the header string.

Format: YES, NO, or *char*. *char* is a single character that may be specified in either character or hexadecimal format. If character format is specified, it may be unframed or framed with C'' or CL1'' characters. If hexadecimal format is specified, it must be framed with X' or XL1' characters.

Default: BLANK=YES.

Note: YES specifies that the EBCDIC blank character (X'40') is to be ignored by this macro whenever it is encountered in the header character string being checked against the control character string specified by the *conchars* operand. For example, if BLANK=YES and an eight-byte field in the header is being checked by this macro, a blank appearing in the fifth byte of the field is ignored and the sixth through ninth bytes is considered to be the last four bytes of the field, assuming that no blanks are coded in the sixth through ninth bytes.

NO specifies that the EBCDIC blank character is to be treated in the same way as any other character when it is encountered by this macro in the header string being compared to the string specified by *conchars*.

char specifies that the single character replacing *char* is to be ignored by this macro whenever it is encountered in the header string being compared to the string specified by the *conchars* operand. That is, the macro automatically skips over the character without performing a comparison and goes on to check the next character in the header. If BLANK=*char* is specified and *char* is not the EBCDIC blank character (X′40′), the EBCDIC blank is not ignored by this macro when it is encountered in the header string but is compared to the character in the corresponding space in the *conchars* string, in the same way as any other character.

This operand is meaningless unless the *conchars* operand is also specified.

```
conchars
```

Function: Specifies the character or character string that, if found in the header as the next nonblank field, causes execution of the CANCELMG macro function.

Format: One to eight nonblank characters in character or hexadecimal format. If character format is used, the string may be unframed or framed with C′ or CLn′ characters. If hexadecimal format is used, the string must be framed with X′ or XLn′ characters.

Default: None. Specification is optional.

Note: If this operand is omitted and CANCELMG occurs in an outheadeı subgroup, the CANCELMG function is performed unconditionally. If the next field in the header does not match this operand, the function is not performed.

```
CONNECT={AND}
        {OR }
```

Function: Specifies the type of logical connection to be made between the mask and the message error record.

Format: AND or OR.

Default: CONNECT=OR.

Note: AND specifies that the macro is to be executed only if *all* of the bits specified by *mask* are on in the message error record. OR specifies that the macro is to be executed if *any* bit specified by *mask* is on in the message error record.

Example 1:

```
CANCELMG X'0000080100',CONNECT=AND
```

specifies that the message is to be canceled only if bits 20 *and* 31 of the message error record are on.

Example 2:

```
CANCELMG X'0000080100',CONNECT=OR
```

specifies that the message is to be canceled if either bit 20 *or* bit 31 of the message error record is on.

`mask`

Function: Specifies the 5-byte bit configuration used to test the message error record for the message. (The message error record is described in Appendix A.)

Format: Unframed decimal integer or hexadecimal field. If hexadecimal format is used, framing characters must be specified. If X'' is used, leading zeros must be coded. If XL5'' is used, leading zeros may be omitted.

Default: None. Specification is optional.

Note: If this operand either specifies an all-zero mask or is omitted, CANCELMG executes unconditionally.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Return Codes

None.

CHECKPT Macro

The CHECKPT macro:

- Causes an incident record to be taken of the option fields for the originating external LU or application program (when coded in an incoming group), or of the status and option fields and sequence numbers assigned to the destination (when coded in an outgoing group)
- Cannot be coded in the inblock subgroup of the message handler or in the outheader or outbuffer subgroup of an application-program message handler
- Does not checkpoint an option field if an address-type constant has been specified in the option field.

When coded in an inheader, inbuffer, or inmessage subgroup, the CHECKPT macro causes an incident record to be made of the option fields assigned to the originating external LU or application program. This checkpoint record is taken after the entire incoming group has executed and the message has been queued, so the option fields reflect that a message has been processed by the incoming group.

When coded in an outheader, outbuffer, or outmessage subgroup, CHECKPT causes an incident record to be taken of the status of the option fields and the sequence numbers assigned to the destination. This checkpoint record is taken after the entire outgoing group has been executed and the message has been sent; the option fields reflect that a message has been sent by the outgoing group.

If a message segment goes through any subgroup in which a CHECKPT macro is executed, an incident record is made after that message has been completely handled by the appropriate MH group. Only one record per message is made, even if more than one CHECKPT macro is coded in the group. If no CHECKPT macro is coded in a group, no incident record is made when the message leaves the group.

For more information on TCAM's checkpoint facility, see the chapter titled "Checkpoint/Restart Service Facility" in *TCAM Utilities*.

Supported Resources and General Requirements: ALL.

Valid subgroups: Inbuffer, inheader, inmessage, outbuffer, outheader, and outmessage.

NAME	OPERATION	OPERANDS
[symbol]	CHECKPT	(no operands)

symbol

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

None.

CLOSE Macro

The CLOSE macro:

- Is issued in the message control program to deactivate any log data set, checkpoint data set, and DASD message queue that is open in the MCP
- Must appear following the READY macro or be branched to from instructions following the READY macro.

The CLOSE macros must be coded in the following order:

1. Log data set
2. MCP checkpoint data set
3. Message queue data set.

Supported Resources and General Requirements: Not applicable.

Valid subgroups: Not applicable.

NAME	OPERATION	OPERANDS
[symbol]	CLOSE	(dcbname,,...)

`(dcbname,,...)`

Function: Specifies the names of the data control blocks for the data sets being closed.

Format: Each *dcbname* must conform to the rules for assembler language symbols, and must correspond to the name specified on the DCB macro for the data set being closed.

Default: None. This operand is required.

Note: If register notation is used, the address of the data control blocks must previously have been loaded into the general registers specified. Register numbers must be enclosed in parentheses.

All MCP data sets of the same type (for example, all message queue data sets) can be closed with one CLOSE macro by including the names of their data control blocks as operands. If more than one *dcbname* is coded in a CLOSE macro, the names are separated by double commas. For example,

```
symbol     CLOSE       (dcbname1,,dcbname2,,dcbname3).
```

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

None.

CODE Macro

The CODE macro:

- Translates the data in the buffer being handled
- Optionally tests for basic operator commands
- Tests for SNA DFC commands and function management (FM) headers
- Allows you to bypass or continue inheader and inbuffer subgroup processing for operator control input
- Allows validity and security checks on basic operator control input
- May be issued at any point in the subgroup
- Is optional for stations
- Allows insertion of a 2-byte correlation ID or specification that the ID is already inserted after control characters
- Allows specification of a station or application other than the originator to receive replies to basic operator commands.

CODE causes the message segment being handled to be translated. If specified in an inblock, inheader, or inbuffer subgroup, translation is from transmission code to EBCDIC; if specified in an outheader or outbuffer subgroup, translation is from EBCDIC to the transmission code. Translation should not be performed in the message handler for an application program because both the message handler and the application program work on EBCDIC data. The CODE macro may be issued with no translation or the IEDOPTCL macro may be issued if operator control is desired in the application program. The transmission code to be used is specified by the TRANS= as well as the ALTCD= operand of the GROUP macro or by an operand of the CODE macro, which overrides the table specified in the GROUP macro.

If CODE is included in an inheader or outheader subgroup, and any segments of a message are processed by that subgroup, the *entire message* is translated. (Special considerations for logical messages are discussed below.) Macros issued before CODE in the incoming group act on message characters that are in line code, while macros issued subsequent to CODE act on message characters that are in EBCDIC. The converse is true for the outgoing group. If CODE is not included in the incoming group, incoming messages are not translated; if CODE is not included in the outgoing group, outgoing messages are not translated.

Note: Once the CODE macro has been executed in a subgroup of an incoming or outgoing group, care must be taken that no segment of a message is routed through another subgroup which also contains a CODE macro.

The CODE macro permits flexibility in handling of buffers, with respect to translation, by overriding the translation table specified in the GROUP macro.

If OPCHK=YES is coded or if this operand is omitted, the CODE macro checks the first portion of the message to see if it contains the character string indicating that this message is a basic operator command. If OPCHK=NO is coded, CODE does not check for basic operator commands.

If CODE detects a basic operator command and MHPROC=NO is coded (or the MHPROC operand is omitted), control is passed to the inmessage subgroup, where macros may check the message error record and take appropriate action if errors are detected. Unless a CANCELMG macro is executed in the inmessage subgroup, the basic operator command is routed to the basic operator control system service program following execution of the inmessage subgroup.

If CODE detects a basic operator command and MHPROC=YES is coded, CODE sets a return code of zero and passes control to the next sequential instruction. This gives you the opportunity to cancel the basic operator command, make security and validity checks, or change your own tailored basic operator control input to the TCAM format in the MH before it is sent to operator control. After making whatever changes you desire, you should bypass further message-handling and branch to an INMSG or INEND macro. If the message is not a basic operator control string, you can continue your normal message-handling routines.

CODE tests for basic operator commands and transfers control accordingly unless you specifically code the OPCHK=NO operand to reflect no basic operator command checking. If basic operator commands may be entered by any station, then a CODE macro (or an IEDOPCTL macro) should be issued in the inheader subgroup of the MH handling incoming messages.

Note: One or more characters may be placed in front of the character string identifying your basic operator command as such. This is permissible if, before issuing CODE, you set the scan pointer to the first byte of the basic operator control character string.

The CODE macro or the IEDOPCTL macro must be issued in either the inblock or the inheader subgroup handling messages from an external LU if basic operator commands are to be entered by that external LU. However, you may not wish to translate ordinary messages entered at the external LU. One way to avoid having to translate every message is described below:

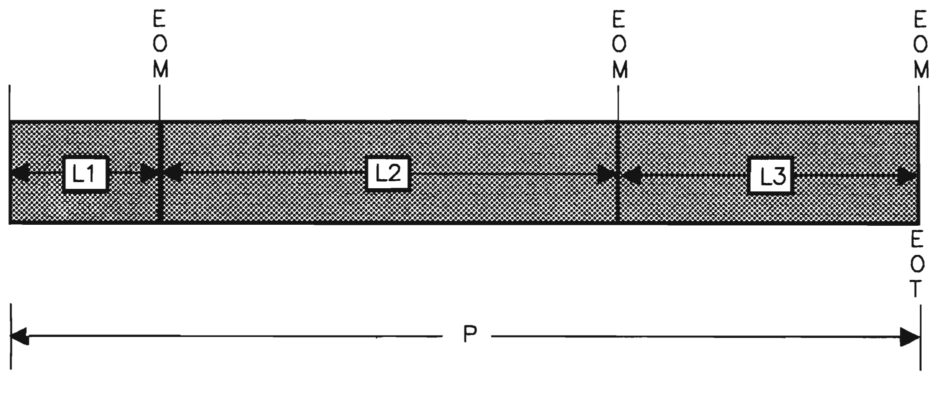
1. Code a special inheader subgroup as the first subgroup of the incoming group; this special subgroup may consist of a MSGTYPE macro followed by a CODE macro.

2. Have the MSGTYPE macro look at the first field in each incoming message in line code, and execute only if this field consists of some specific character.

3. Enter A before the identification sequence of each basic operator command. If the first character of a message is A, the CODE macro is executed and the message is translated; otherwise, control is passed to the next delimiter, which may be another inheader subgroup designed to handle ordinary messages.

The CODE macro instruction provides the same functions for logical messages (translation and checking for basic operator commands) as those described above. However, the following special considerations apply to the CODE macro when it is used in the inblock subgroup handling incoming messages to be deblocked into multiple logical messages. Logical messages formed by blocking incoming physical transmissions require no special considerations.

The format of an incoming physical transmission that is to be deblocked is:



where P is all the data in the entire physical transmission and L1, L2, and L3 are the logical messages that will result from deblocking P.

When the CODE macro is issued in the inblock subgroup *before* the SETEOM functional macro, all the incoming data (P) is translated. If logical messages L1, L2, and L3 are to be checked to determine if they are operator commands, code MHPROC=YES and include an IEDOPCTL macro in the inheader subgroup of this incoming group.

Supported Resources and General Requirements: ALL.

Valid subgroups: Inblock, inbuffer, inheader, outbuffer, and outheader.

NAME	OPERATION	OPERANDS
[symbol]	CODE	`[{tablename }]`
		` {NONE      }`
		` {(register)}`
		`[,MHPROC={YES}`
		`         {NO }`
		`[,OPCHK={YES}`
		`        {NO }`
		`[,RSPDEST={'name'     }]`
		`          {fieldname }`
		`          {(register)}`
		`          {**        }`
		`[,USERID={X'id'      }]`
		`         {fieldname  }`
		`         {(register) }`
		`         {**         }`
		`         {NO         }`

```
MHPROC={YES}
       {NO }
```

Function: Allows continuation of message-handler processing with a return code from CODE processing.

Format: YES or NO.

Default: NO.

Note: NO causes inheader and inbuffer processing to be bypassed; inmessage is executed. YES specifies that MH processing continues with a return code of zero from CODE.

```
OPCHK={NO }
      {YES}
```

Function: Allows validity and security checks on an operator control input.

Format: YES or NO.

Default: YES.

Note: NO specifies that CODE is to do no checking of basic operator commands (translation only). YES specifies that CODE is to be checked for basic operator commands.

```
RSPDEST={**          }
        {'name'      }
        {fieldname   }
        {(register)  }
```

Function: Specifies the name of the external LU or application that is to receive the reply to this request.

Default: RSPDEST=**.

Format: **, 'name' fieldname, or (register).

Note: ** specifies that the origin of the request is the destination of the reply.

'name' specifies the name of the external LU or application to receive the reply.

fieldname specifies the symbolic name of the field containing the name of the external LU or application to receive the reply.

(register) specifies the name or number of a register containing the address of a field that contains the name of the external LU or application to receive the reply.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

```
{tablename }
{NONE      }
{(register)}
```

Function: Specifies the type of translation to be done.

Format: *tablename, (register),* or NONE. *tablename* must be the name of a user-defined table that conforms to the rules for assembler language symbols or an IBM supplied table. *(register)* must specify a decimal integer between 2 and 11.

Default: None. Specification is optional.

Note: If this operand is omitted, the table used for translation is that specified by the TRANS= or ALTCD= operand of the GROUP macro.

If NONE is specified, the message is not translated. NONE can be used to check for operator commands when the station transmits in EBCDIC or when scan pointer adjustment is required after translation prior to checking the operator control character string. For example, for 3270 data streams, position the scan pointer beyond the device dependent characters and then execute the CODE macro with NONE specified. The message is not translated, but it is checked for operator commands. This is the same function performed by the IEDOPCTL macro.

If *(register)* is specified, the following restrictions apply:

- The register must be previously loaded with the address of the table to be used. A user-defined translation table must consist of a fullword on a fullword boundary, followed by a 256-byte table for translating from transmission code into EBCDIC, followed by a 256-byte table for translating from EBCDIC into transmission code. The first word must contain the address of the first byte of the second table. The high-order byte of the first word must be zero.
- The CODE macro with *(register)* specified can be issued only in the inblock, inbuffer, or outbuffer subgroups. It must not be issued in the inheader or outheader subgroups.

```
USERID={X'id'      }
       {fieldname  }
       {(register)}
       {**         }
       {NO         }
```

Function: Specifies whether a user correlation ID is to be accepted on basic operator control commands.

Format: X'*id*', *fieldname*, *(register)*, **, or NO.

NO specifies no correlation ID.

X'*id*' specifies the user ID in hexadecimal.

fieldname specifies the symbolic name of the field containing the four-byte user ID.

(register) specifies the name or number of a register containing the address of a field containing the four-byte user ID.

** specifies that the user correlation ID is in the basic operator control command following the basic operator control identifier (which is specified on the CONTROL= operand of the INTRO macro).

Default: USERID=NO.

Return Codes

One of the following return codes is set in register 15:

Code	Meaning
X'00000000'	1. If translation is specified, the message was translated. 2. If MHPROC=YES is specified, message is basic operator control command (and was successfully translated, if necessary).
X'00000004'	1. The correct translation table could not be found. The message was not translated. 2. When OPCHK=YES is coded, TCAM could not determine if this message is a basic operator control command for one of the following reasons: • The message origin is not correct. • The external LU is not a basic secondary operator control station. • The external LU is in lock mode. • The string was found but this is not a single-buffer message. • The buffer is not a header buffer. 3. Zero length buffer.
X'00000008'	The name of the destination specified by the RSPDEST operand is not a valid TCAM destination.
X'0000000C'	When OPCHK=YES is coded, the operator control string was not found (that is, this message is not a basic operator control command).
X'00000010'	No table was specified and the entry is for an application program. The message was not translated.

CODEBR Macro

The CODEBR macro instruction:

- Interrogates a return code set by another macro and selects the next instruction to be executed
- May be issued more than once.

The register specified via the REG= operand is interrogated. If the contents are all zeros, the next sequential instruction is then executed. If the contents consist of a negative return code, the next instruction executed is the one named in the NEG= operand. If the contents consist of a positive return code, the next instruction executed is the one named for that return code in the POS= operand. If the register value is four, the first POS= operand name is used. If it is eight, the second, and so on. If the value is too high for the last POS= operand, the next sequential instruction is executed.

Before the register is interrogated, the 2 low-order bits are cleared to ensure that the value is a multiple of four. Thus, a value of one, two, or three in the specified register is regarded as zero; a value of five, six, or seven, as four, and so on.

Supported Resources and General Requirements: ALL.

Valid Subgroups: Inheader, inbuffer, outheader, and outbuffer.

NAME	OPERATION	OPERANDS

[symbol]	CODEBR	NEG=name
		,POS=(name1,name2,...)
		[,REG= {integer}]
		{15 }

NEG=name

Function: Specifies the name of the next instruction to be executed if the register contains a negative value. This must be the name of a macro or instruction in this subgroup, of the name of a delimiter macro for another subgroup.

Format: Must conform to the rules for assembler language symbols.

Default: None. Must be specified if POS= operand is omitted. Can be optionally coded with the POS= operand.

POS=(name1,name2...)

Function: Specifies one or more names of the next instruction to be executed if the register contains a positive value. The first name is used if the value is four, the second name if the value is eight, and so on. Each name must be the name of a macro or instruction in this subgroup or the name of a delimiter macro for another subgroup.

Format: Each name must conform to the rules for assembler language symbols. The names are separated by commas and are enclosed in parentheses. (If only one name is used, the parentheses are unnecessary.)

Names may be omitted if the omission is shown by the presence of the surrounding commas. An omitted name is assumed to specify a branch to the next sequential instruction. (Example: CODEBR POS=(,NB,,NC); return codes 4 and 12 cause a branch to the next sequential instruction.)

Default: None. Must be specified if NEG= operand is omitted. Can be optionally coded with the NEG= operand.

Note: If the content of the register is not a multiple of four, it is reduced to the next lowest multiple of four.

If there is no name corresponding to the content of the register (for example, if it is 8 and only one POS=name is specified), the next sequential instruction is then executed.

```
REG={integer}
    {15     }
```

Function: Specifies the register to be interrogated.

Format: *integer* is a decimal integer, minimum 1, maximum 15.

Default: 15.

Note: Register 0 cannot be used to contain a return code.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Return Codes

None.

COMMAND Macro

The COMMAND macro:

- Defines to TCAM the name and specifications of a user-written extended operator control command and specifies the module supplied by the user for processing this command
- May be optionally coded, one or more times, in module DKJUSR.

For a description of the parameter list passed to the user-supplied processing module and a sample processing module illustrating use of the parameter list, see the chapter titled *"Basic and Extended Operator Control System Service Programs"* in the *TCAM Utilities* manual.

Supported Resources and General Requirements: Not applicable.

Valid subgroups: Not applicable.

NAME	OPERATION	OPERANDS

[symbol]	COMMAND	CONSOLE={YES}
		{NO }
		,MODULE=usermod
		,CTLTERM={YES}
		{NO }
		[,ALIAS=name]
		[,LASTONE={YES}]
		{NO }

```
ALIAS=name
```

Function: Provides an alternate (usually a shorter) form of the command verb name.

Format: *name* must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Note: If more than one alias is required, another COMMAND macro must be coded.

```
CONSOLE={YES}
        {NO }
```

Function: Specifies that the command may (YES) or may not (NO) be entered from the system console.

Format: YES or NO.

Default: None. This operand is required.

```
CTLTERM={YES}
        {NO }
```

Function: Specifies whether the external LU or application program entering the command is required to be an extended operator control station.

Format: YES or NO.

Default: None. This operand must be specified.

```
LASTONE={YES}
        {NO }
```

Function: Indicates the end of user-supplied verbs.

Format: YES or NO.

Default: LASTONE = NO.

Note: LASTONE = YES must be specified on the last user-supplied COMMAND macro in module DKJUSR.

```
MODULE=usermod
```

Function: Specifies the load module name of the user-supplied command processor.

Format: *usermod* must conform to the rules for assembler language symbols.

Default: None. This operand is required.

```
symbol
```

Function: Specifies the name of the command verb.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. The name must be specified.

Return Codes

None.

COREDSP Macro

The COREDSP macro:

- Provides assistance for application program restart
- Is normally issued in an application program but may be issued (for main storage display during testing) anywhere in an executable section of the MCP.

The COREDSP macro instruction enables the issuing program to display selected main storage locations and, if so specified, allows the system console operator to indicate whether an application program should attempt to process the current message or should access another message. The issuing program may display any location in its virtual storage area.

The macro instruction can be coded in such a way that, after an initial main-storage display, the system console operator may request another display or decide that the record last accessed by the application program is not to be processed again.

The contents of registers 1, 14, and 15 are destroyed by the COREDSP macro instruction. Register 13 is presumed to contain the address of an 18-fullword register save area. All other registers are saved and restored by the COREDSP macro. When control is returned to the user, register 1 contains the address of the work area used by the macro (see the WORK operand, below), and register 15 contains the return code (0 if the operator entered GO or if no operator conversation occurred, and 4 if the operator entered NO).

Supported Resources and General Requirements: ALL.

Valid subgroups: Inblock, inheader, inbuffer, outheader, and outbuffer.

NAME	OPERATION	OPERANDS

[symbol]	COREDSP	`ADDR={label}`
		`     {(reg)}`
		`[,CONV={NO }]`
		`       {YES}`
		`[,NAME={dataid}]`
		`       {(reg) }`
		`[,WORK={areaname}]`
		`       {(reg)   }`

```
ADDR={label}
     {(reg)}
```

Function: Identifies the first main storage address to be displayed.

Format: *label* is the symbolic name of the field at which main storage display should begin and must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.") *(reg)* is a decimal integer from 1 to 12, enclosed within parentheses.

Default: None. This parameter is required.

Note: The register with the specified number contains the main storage address of the message or other area for which display is being requested.

```
CONV={NO }
     {YES}
```

Function: Specifies whether the system console can request more than one main-storage display.

Format: YES or NO.

Default: CONV=NO.

Note: YES specifies that the system console can receive more than the one main storage display specified in the macro and can bypass the last record accessed. For more information on this capability, see "Optional TCAM Facilities for the Application Programmer" in the *TCAM Application Programming* manual.

NO specifies that only the initial macro-specified display will be provided.

```
NAME={dataid}
     {(reg) }
```

Function: Supplies the identifier that precedes the display characters on the first print line.

Format: *dataid* or (*reg*). *dataid* must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.") (*reg*) is a decimal integer within parentheses.

Default: The name of the current CSECT will be used to identify the message.

Minimum: 1

Maximum: 12

Note: If *(reg)* is used, the specified number is of a register containing the address of an 8-byte field that includes the identifier name.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

```
WORK={areaname}
     {(reg)   }
```

Function: Specifies a 184-byte, doubleword-aligned work area.

Format: *areaname* or (*reg*). *areaname* must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

(reg) is a decimal integer within parentheses: minimum 2, maximum 12.

Note: *areaname* is the name of the work area. If *(reg)* is used, the specified number is that of a register containing the address of the work area.

Default: The work area is internally generated.

Return Codes

The following return code is set in register 15:

Code	**Meaning**
X'00000000'	Indicates that either CONV=NO has been coded or CONV=YES, has been coded and the operator has entered GO in response to COREDSP indicating current message should be processed.
X'00000004'	Indicates the operator responded NO to CORESP which indicates current message should not be processed.

COUNTER Macro

The COUNTER macro:

- Maintains a count of complete messages or message segments received from or sent to an external LU or application program
- Is optional.

COUNTER enables you to maintain four types of count. The position of the COUNTER macro within an MH determines which type of count is maintained. COUNTER is optional and may appear:

- In the inblock subgroup to count incoming physical message segments arriving at this MH or at incoming logical message segments for each origin
- In the inheader subgroup to count incoming messages for each origin
- In the inbuffer subgroup to count incoming message segments for each origin
- In the outheader subgroup to count outgoing messages for each destination
- In the outbuffer subgroup to count outgoing message segments for each destination.

Any one or more of the counts may be maintained by including COUNTER in the appropriate subgroups; within a subgroup, it may appear at any point.

For each COUNTER macro, you must define (by an OPTION macro) a halfword option field for the appropriate external LU or application program. This provides space for maintaining the counters.

If the MH includes the SETEOM macro, the number of times the COUNTER macro executes depends on the PROCESS operand of SETEOM and on the position of the COUNTER macro relative to the SETEOM macro.

If the SETEOM macro specifies PROCESS=NO (any data following an EOM indicator is ignored), the COUNTER macro always adds one to the counter for each resulting blocked logical message.

If the SETEOM macro specifies PROCESS=YES (any data following an EOM indicator is processed in a new buffer) and if the COUNTER macro appears *before* SETEOM, 1 is added to the counter to reflect each buffer in the entire incoming transmission sequence. However, if COUNTER appears *after* SETEOM (that is, after the data is deblocked), each buffer of each resulting logical message causes a counter to be updated by 1. In the latter situation, if a counter is associated with each external LU that is eligible to enter messages to this MH, a count is maintained for incoming logical messages on a external LU basis.

Supported Resources and General Requirements: ALL.

Valid subgroups: Inblock, inbuffer, inheader, outbuffer, and outheader.

NAME	OPERATION	OPERANDS
[symbol]	COUNTER	opfldname

opfldname

Function: Specifies the name of the halfword option field for the external LU or application program in which the count is to be maintained.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary") and must be the name of a halfword option field defined by an OPTION macro.

Default: None. This operand is required.

Note: The field contains a binary count up to a 65,535. When the maximum count has been reached, the count is reset to 0 for the next message or segment counted.

You may gain access to the field at any time to determine or reset the count (by operator commands or by user code including the LOCOPT macro). The count is initially set using the OPDATA operand of the TERMINAL or TPROCESS macro.

If the option field is not found, COUNTER is not executed, and control passes to the next MH instruction.

symbol

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	Successful Return.
X'000000FF'	Zero length buffer.

DAFROUTE Macro

The DAFROUTE macro:

- Routes a message, based on the TCAM address contained in the DAF field of an FHP, to a external LU or an application program located either in the same TCAM node or in another TCAM node
- Is part of TCAM extended networking
- When the destination resource id refers to a cascade list, selects that entry of the cascade list for which the fewest unsent messages are currently queued
- Can alternately be specified to perform only availability validation, deferring the routing functions
- May be coded more than once; if a single appearance of the macro is executed more than once during an execution of the MH, only the last execution of the macro is effective.

The DAFROUTE macro routes a message based on a TCAM address (node identifier and resource identifier) passed to the macro. The macro uses the node table and resource table to determine where the message should be sent. If the destination node and resource id represents a local cascade list or if the destination is a resource in another TCAM system for which the node-table entry refers to a cascade list (for dynamic internodal load balancing), DAFROUTE selects that destination queue of the cascade list for which the smallest number of unsent messages is currently queued. (Any queues of the cascade list that are held, transferred, or in purge mode are not used.)

Additionally, any main-storage queue associated with a destination is tested to see if the number of unsent messages is greater than the value specified in the THRESH option field associated with the queue. If so, the entry is not used.

The NODEBAD, NODEDWN, RESBAD, RESDOWN, and ANYBAD operands specify the name of the next instruction to be executed if a particular condition is detected during DAFROUTE processing. If none of these conditions are encountered, the next sequential instruction is executed.

The IEDTCSD macro should be included in the MCP to provide dummy sections (DSECTS) for control blocks referenced by the code generated by DAFROUTE.

Supported Resources and General Requirements: ALL, FHP.

Valid subgroups: Inheader.

NAME	OPERATION	OPERANDS
[symbol]	DAFROUTE	{ANYBAD=label }
		{NODEBAD=label }
		{,NODEDWN=label}
		[,RESBAD=label]
		[,RESDOWN=label]
		[,DAF= {<u>FHPDAF</u>}]
		{(reg) }
		{FHPOAF}
		[,DSECT={<u>NO</u> }]
		{YES}
		[,TC={<u>FHP</u> }]
		{(reg) }
		{integer}
		[,TNT={label}]
		{(reg)}

```
ANYBAD=label
```

Function: Specifies the label of the next instruction to be executed if any of the error conditions are detected. If ANYBAD is specified, all other error operands (NODEBAD, RESBAD, NODEDWN, RESDOWN) are ignored.

Format: label must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional. If omitted, the NODEBAD, RESBAD, NODEDWN, and RESDOWN operands must be specified.

```
DAF={FHPDAF}
    {(reg) }
    {FHPOAF}
```

Function: Specifies the location of the network address to be used by the macro to route the message.

Format: (*reg*), FHPDAF, or FHPOAF. *(reg)* can be specified as an explicit decimal integer from 2 through 12 within framing parentheses or as a symbol within framing parentheses that has previously been equated to a decimal value from 2 through 12. It is your responsibility to assign to this operand a value that is within the prescribed limits.

Default: DAF=FHPDAF.

Note: *(reg)* is a general purpose register (2-12) containing the address of a 3-byte field containing the TCAM address (node and resource identifiers) to be used to route the message.

FHPDAF specifies that the TCAM destination address field (FHPDAFLD) in the fixed header prefix is to be used to route the message.

FHPOAF specifies that the TCAM origin address field (FHPOAFLD) in the fixed header prefix is to be used to route the message.

```
DSECT={NO }
      {YES}
```

Function: Generates a dummy section describing the formats of THRESH option fields and the parameter list passed to the DAFROUTE functional routine.

Format: NO or YES.

Default: DSECT=NO.

Note: If YES is specified, all other operands are ignored and the dummy section describing the THRESH option field and the parameter list to the functional routine are generated.

`NODEBAD=label`

Function: Specifies the label of the next instruction to be executed if either of the following conditions occurs: the node identifier of the TCAM address for the destination is invalid (that is, there is no entry in the node table corresponding to the passed node identifier) or the terminal-table entry named in the TC*n* operand of the NODEDEF macro for the destination host node is invalid.

Format: label must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is required unless the ANYBAD operand is coded.

`NODEDWN=label`

Function: Specifies the label of the next instruction to be executed if either of the following conditions occurs:

- The node table entry status information indicates that the target host node is inactive.
- The internodal destination queue for the host node in another TCAM system is located in main storage without disk backup, and the queue of unsent messages exceeds the threshold value specified in the THRESH option field associated with the internodal destination queue.

Format: *label* must conform to the rules for assembler language symbols.

Default: None. Specification is required unless the ANYBAD operand is coded.

`RESBAD=label`

Function: Specifies the label of the next instruction to be executed if the resource identifier of the passed TCAM address is invalid. (The node identifier portion of the TCAM address indicates that the destination resource is owned by this TCAM node but there is no entry in the resource table corresponding to the resource identifier.)

Format: label must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is required unless the ANYBAD operand is coded.

`RESDOWN=label`

Function: Specifies the label of the next instruction to be executed if either of the following conditions occurs:

- The resource identifier in the Key entry reflects a destination station or application in this TCAM system with a main-storage queue of unsent messages greater than the threshold value specified in its THRESH option field.
- The resource identifier in the Key entry reflects a destination station or application in this TCAM system that uses main-storage-only queueing and that is not currently active.

Format: label must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is required unless the ANYBAD operand is coded.

```
symbol
```

Function: Specifies the name of the macro. (See the *symbol* entry in the "Glossary.")

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

```
TC={FHP     }
   {(reg)   }
   {integer}
```

Function: Specifies which transmission category and associated internodal destination queue specified in the node-table entry is to be used if the message requires routing to another host node.

Format: integer or *(reg)* or FHP. *integer* is an unframed decimal integer; *(reg)* can be specified as an explicit decimal integer from 2 through 12 within framing parentheses, or a symbol that has previously been equated to a decimal value from 2 through 12.

Default: TC=FHP.

Note: integer is the transmission category specified as a decimal integer not greater than the number of transmission categories defined in the MCP; *integer* may not exceed 16, the maximum number of transmission categories allowed in any MCP.

(reg) is a general register from 2 to 12 that contains the address of a 1-byte field that contains the transmission category.

FHP indicates that the transmission category is to be found in the FHP.

```
TNT={label}
    {(reg)}
```

Function: Specifies that the macro is to analyze and resolve the TCAM address but is not to route the message. The terminal-name-table (TNT) index of the terminal-table entry represented by the destination TCAM address is returned. (The TTCIN operand of the FORWARD macro can be used subsequently to complete the routing.)

Format: *label* or *(reg)* *label* must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.") *(reg)* is a general purpose register specified by a decimal number from 2-12 within framing parentheses, or by a symbol, (within parentheses) that has previously been equated to a value from 2 through 12. The selected TNT is placed in the low-order two bytes of the register.

Default: None. Specification is optional.

Return Codes

None.

DATETIME Macro

The DATETIME macro:

- Inserts the date, the time, or both in an incoming or outgoing message header at the current position of the scan pointer.

The DATETIME macro inserts either the date or the time or both in the header of an incoming or outgoing message. Seven characters are inserted for the date, if specified: a blank, the last two digits of the year, a period, and the 3-digit day number. Nine characters are inserted for the time, if specified: a blank, two digits for the hour, a period, two digits for the minute, a period, and two digits for the second.

Note: If both date and time are specified, the date is inserted first. If neither operand is coded, both the date and the time are inserted with the date being inserted first.

Space in the header for these insertions, seven characters for the date and nine characters for the time, must be reserved by the RESERVE operand of the GROUP macro. If the insertions are required for an application program, then the space must be reserved by the RESERVE operand of the PCB macro. After DATETIME has executed, the scan pointer is positioned at the last inserted character.

When the DATETIME macro is coded in an outheader subgroup, the macro may operate upon the first message segment only. This is because TCAM does not maintain reserve bytes for any segment of an outgoing message except the first.

To avoid having to specify a large first buffer, you may design your header so that it occupies two buffers, and then insert the incoming date and time in that portion of the header contained in the second buffer, and the outgoing date and time in that portion of the header contained in the first buffer. The characters inserted by DATETIME are in EBCDIC. Therefore, the DATETIME macro should not be issued *before* a CODE macro in an inheader subgroup, or *after* a CODE macro in an outheader subgroup.

Supported Resources and General Requirements: ALL.

Valid subgroups: Inheader, outheader.

NAME	OPERATION	OPERANDS

[symbol]	DATETIME	[DATE={<u>YES</u>}]
		{NO }
		[,TIME={<u>YES</u>}
		{NO }

```
DATE={YES}
     {NO }
```

Function: Specifies whether the date is to be inserted in a message header.

Format: YES or NO.

Default: DATE=YES

Note: YES specifies that the date is to be inserted in the message header. NO specifies that the date is to be omitted from the header.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

```
TIME={YES}
     {NO }
```

Function: Specifies whether the time is to be inserted in a message header.

Format: YES or NO.

Default: TIME=YES

Note: YES specifies that the time is to be inserted in the message header. NO specifies that the time is to be omitted from the header.

The time inserted is the time at which this DATETIME macro is executed. TCAM determines this by examining the system timer.

Return Codes

One of the following return codes is set in register 15:

Code	Meaning
X'00000000'	Successful execution.
X'00000004'	Macro does not execute; insufficient reserve space is available.
X'FFFFFFFC'	Macro does not execute; zero-length buffer.

DCB Macro

The DCB macro defines the data control block for:

- The MCP checkpoint data set
- The log data set
- The message queue data set.

The specific DCB macros required for the applicable data sets are described below. In each case, the operands may be specified in any order and are separated by commas with no intervening blanks. When a parameter can be provided by an alternate source, a symbol appears in the *Alternate Source* entry for the operand. When there is no alternate source (that is, the parameter must be specified by the operand), the *Alternate Source* entry specifies "None." They have the following meanings.

Symbol	Explanation
DD	The value of the operand can be omitted from the DCB macro and provided at execution by the data definition (DD) card for the data set.
PP	The value of the operand can be provided by the user's problem program any time before the data control block exit at open.
OE	The value of the operand can be provided by the problem program any time up to and including the data control block open exit at open.

Supported Resources and General Requirements: Not applicable.

Valid subgroups: Not applicable.

MCP Checkpoint DCB Macro

The checkpoint DCB macro:

- Defines a single checkpoint data set to provide checkpoint records of changes in the MCP.

NAME	OPERATION	OPERANDS
[dcbname]	DCB	DDNAME=ddname ,DSORG=TQ ,MACRF=(G,P) ,OPTCD=c [,BLKSIZE={integer}] {300 } [,EXLST=address]

```
BLKSIZE={integer}
        {300    }
```

Alternate Source: DD.

Function: Specifies the size (in bytes) of the environment checkpoint records on disk.

Format: Unframed decimal integer.

Default: BLKSIZE = 300.

Note: If a number less than 300 or greater than the maximum blocksize allowed by the appropriate secondary-storage device is specified, 300 or the maximum blocksize allowed respectively is used, and a message notifying the operator is generated on the system output device. A larger record size may use disk space more efficiently, but the work area would take up more main storage. These factors must be considered to arrive at the best block size for a given application. See "Checkpoint/Restart Service Facility" in *TCAM Utilities* for information on determining the size of environment checkpoint records.

`dcbname`

Function: Specifies the name of the macro and the name the data control block generated by the expansion of the macro.

Format: dcbname must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. This name is required.

`DDNAME=ddname`

Alternate Source: The value can be provided by the user before the OPEN macro is issued.

Function: Specifies the name that appears in the DD statement associated with the data control block.

Format: ddname must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. This operand is required.

`DSORG=TQ`

Alternate Source: None.

Function: Specifies that the data set organization is for the message queue or checkpoint data set.

Format: TQ.

Default: None. This operand is required.

`EXLST=address`

Alternate Source: PP.

Function: Specifies the address of the problem-program exit list.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Note: This list must be provided if user label, data control block, or user ABEND exits are required. The list must start on a fullword boundary. Its format and contents are shown in the *Data Management* publications. User ABEND exits are described in the chapter titled "Defining Data Sets" in the *TCAM Installation Guide.*

`MACRF=(G,P)`

Alternate Source: None.

Function: Specifies that access to the data set is gained with GET and PUT macro instructions.

Format: (G,P).

Default: None. This operand is required.

`OPTCD=C`

Alternate Source: PP, OE, DD.

Function: Specifies that the data set is for checkpoint records.

Format: C.

Default: None. This operand is required.

Return Codes

None.

Log Data Set DCB Macro (BSAM)

The log data set:

- Is defined by a BSAM DCB macro that is issued with the DCBs defining the message queue and checkpoint data sets
- Should be defined for each secondary-storage device on which messages or message segments may be logged
- Has UNIT parameter of the DD statement associated with each log DCB macro to specify the address of the appropriate secondary-storage device.

The BSAM DCB macro is described in the *OS/VS Data Management Macro Instructions* publication. In the BSAM DCB macro for a log data set, you should code the following operands:

NAME	OPERATION	OPERANDS
[logdcb]	DCB	BLKSIZE=unitsz ,DDNAME=ddname ,DSORG=PS ,MACRF=W ,NCP=integer ,RECFM=F ,SYNAD=address

BLKSIZE=unitsz

Alternate Source: DD, PP.

Function: Specifies the size of a buffer unit.

Format: unitsz is an unframed decimal integer.

Default: None. This operand is required.

Note: unitsz must be the same as the value specified in the UNITSZ operand of the INTRO macro.

DDNAME=ddname

Alternate Source: PP.

Function: Specifies the name that appears in the DD statement associated with the data control block.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. This operand is required.

DSORG=PS

Alternate Source: None.

Function: Specifies that the data set identified by this data control block is organized in the physical-sequential mode.

Format: PS.

Default: None. This operand is required.

logdcb

Function: Specifies the name of the macro instruction and the name of the data control block generated by the expansion of the macro.

Format: logdcb must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. This name is required.

MACRF=W

Alternate Source: None.

Function: Specifies that access to the data set is gained with WRITE macro instructions.

Format: W.

Default: None. This operand is required.

NCP=integer

Alternate Source: None.

Function: Specifies the maximum number of buffer units that may appear in a buffer.

Format: integer is an unframed decimal integer from 1 to 99.

Default: None. This operand is required.

RECFM=F

Alternate Source: DD.

Function: Specifies format characteristics of the data set being created.

Format: F.

Default: None. This operand is required.

Note: F specifies that the data set contains fixed-length records.

SYNAD=address

Alternate Source: None.

Function: Specifies the name of a user-specified error-analysis routine to be given control when a permanent I/O error is detected.

Format: address must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional. If the SYNAD operand is not specified, the MCP terminates abnormally when a permanent I/O error occurs during the logging operation.

Note: The error routine must conform to the standards set forth in the discussion of this operand in the *Data Management Macro Instruction* publication. Upon return from the error-analysis routine, the log function continues as if no error had been encountered. All indications of the I/O error must be removed from the DCB (that is, reset the first two bits of the IFLGS field of the data control block to zeros. See the publication *OS/VS Data Management for System Programmers*). If an I/O error is still indicated, the MCP terminates abnormally when the next WRITE is issued.

Return Codes

None.

Message Queue DCB Macro

The message queue DCB macro:

- Defines a message queue data set residing on reusable or nonreusable disk
- Specifies the location of the data set
- Specifies the percentage of records in a nonreusable data set to be filled before a flush closedown is initiated
- Is not issued for a message queue residing in main storage.

NAME	OPERATION	OPERANDS
`[diskdcb]`	`DCB`	`DDNAME=ddname`
		`,DSORG=TQ`
		`,MACRF=(G,P)`
		`,OPTCD={L}`
		`       {R}`
		`[,EXLST=address]`
		`[,THRESH={95      }]`
		`          {percent}`

`DDNAME=ddname`

Alternate Source: PP.

Function: Specifies the name that appears in the DD statement associated with the data control block.

Format: ddname must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. This operand is required.

`diskdcb`

Function: Specifies the name of the macro instruction and the name of the data control block generated by the expansion of the macro.

Format: diskdcb must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. This name is required.

```
DSORG=TQ
```

Alternate Source: None.

Function: Specifies that the data set organization is that for the message queue or checkpoint data set.

Format: TQ.

Default: None. This operand is required.

```
EXLST=address
```

Alternate Source: PP.

Function: Specifies the address of the problem-program exit list.

Format: address must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Note: This operand is required if user label, data control block, user ABEND, or block count exits are required. The list must start on a fullword boundary. Its format and contents are shown in the *OS/VS Data Management* publications. User ABEND exits are described in the chapter titled "Defining Data Sets" in the *TCAM Installation Guide*.

```
MACRF=(G,P)
```

Alternate Source: None.

Function: Specifies that access to the data set is gained with GET and PUT macro instructions.

Format: (G,P).

Default: None. This operand is required.

```
OPTCD={L}
      {R}
```

Alternate Source: DD, PP, OE.

Function: Specifies the location of the data set.

Format: L or R.

Default: None. This operand is required.

Note: L specifies that the data set is to be on nonreusable disk. R specifies that the data set is to be on reusable disk.

```
THRESH={95     }
       {percent}
```

Alternate Source: DD, OE, PP.

Function: Specifies the percentage of records in the non-reusable disk message queue to be used before a flush closedown of the system is initiated.

Format: integer is an unframed decimal integer from 1 to 95.

Default: 95.

Note: This operand is meaningful for nonreusable disk queues only.

Return Codes

None.

DESTFLD Macro

The DESTFLD macro:

- Validates destination names in the message header
- Queues messages for one or more destinations
- May be used more than once
- Requires the presence of a TCSOPTS option field
- Requires that an FHP be present in the message being processed if destinations are specified in the message header
- Does destination scanning, validity checking, and queue scheduling for destinations in the same TCAM system and for level 3 data flow routing to destinations in another TCAM system--the KEYPROC or DAFROUTE macro should be used for level 2 or level 2+ data flow routing to destinations in another TCAM system. See the section titled "Data Flow Levels in a Network with Multiple TCAM Systems" in the "TCAM Extended Networking" section of *TCAM Planning Guide* for an explanation of the different data flow levels.
- Requires that DLQ= be specified on INTRO macro or overridden at startup.

If destinations are specified in the message header, DESTFLD requires that all such destinations be located in the first buffer unit.

If a destination uses main-storage-only queuing and the count of the messages on the queue exceeds the value specified by the DESTFLD THRESH= operand, the message error record (MER) threshold-exceeded bit (bit 22) is set on and the message is scheduled to be sent to the dead-letter queue. Scheduling to other valid destinations specified in the header continues in this case.

Care should be taken when specifying multiple destinations. If any one of the destinations has disk queuing, it should be specified ahead of all destinations with main—storage queuing. If the first (or primary) destination has main—storage queuing, the message may be serviced and freed from the queue before it can be copied to all secondary destinations. This could cause an ABEND (045-2). Having disk queuing for the primary destination ensures that the message will be available for a longer period of time, even after it is marked serviced.

If message priorities are enforced via a PRIORITY macro coded after DESTFLD, the PRISAVE= operand of PRIORITY must be coded to ensure that message priority will be maintained for all destinations in a multiple-destination message and for all destinations that are part of a distribution or cascade list.

The DESTFLD macro expects the scan pointer to be set to either the last byte of a field preceding the first destination name in the message header, or to a blank or series of blanks preceding the first destination name.

After successful execution of the DESTFLD macro, register 15 is set to a return code of 0 and:

- If the DESTFLD macro was coded with no DEST= operand and no EOA= operand, the scan pointer is set to the last character of the single destination name in the message header.
- If the series of one or more destination names in the message header is delimited by a numeric field, the scan pointer is set to the position preceding the numeric field.
- If the series of one or more destination names in the message header is delimited by a specific character string, the scan pointer is set to the last byte of the specified string.
- If the series of one or more destination names in the message header is delimited by the end-of-header pointer (FHPTEXTP) in the FHP, the scan pointer is set to the last end-of-header character.

- If the series of one or more destination names in the message header is not to be retained in the outgoing message, the start-of-header pointer in the FHP (FHPHEADP) is altered to point to the position beyond the last destination (including any delimiting characters), effectively deleting the destination(s) from the message.

After unsuccessful completion of DESTFLD processing, a return code is set in register 15. The location of the scan pointer is unpredictable if the return code is a value of minus four or plus 4 through 16.

Supported Resources and General Requirements: ALL, FHP.

Valid Subgroups: Inheader.

NAME	OPERATION	OPERANDS

[symbol]	DESTFLD	[DEST=name]
		[,EOA={EOH }]
		{NUMERICS}
		{chars }
		[,KEEP={NO }]
		{YES}
		[,MAXDEST={number}]
		{5 }
		[,PROC={OPENONL}]
		{YES }
		{NO }
		[,THRESH=qmaximum]
		[,TLIST={NO }
		{YES}

```
DEST=name
```

Function: Indicates that the message is to be unconditionally routed to the specified destination.

Format: The name of a single, destination list, cascade list, or process entry in the terminal table. Framing characters are not used.

Default: None. Specification is optional.

Note: When this operand is used, all others except PROC=OPENONL (if coded) are ignored.

No FHP is needed for a message handled by a DESTFLD macro with this operand.

```
EOA={EOH     }
    {NUMERICS}
    {chars   }
```

Function: Specifies the delimiter for the series of one or more destination names in each message header.

Format: EOH, NUMERICS, or *chars*. *chars* is a character string, composed of one to eight nonblank characters in either character or hexadecimal format, which delimits the series of destinations. If character format is used, the string must be framed with C″ character. If hexadecimal format is used, the string must be framed with X″ character.

Default: None. Omission implies one destination delimited by a blank.

Note: EOH specifies that the delimiter for the series of one or more destination names in each message header is the end of header as determined by the end-of-header pointer (FHPTEXTP) in the FHP.

NUMERICS specifies that the delimiter for the series of one or more destination message header is a numeric character preceded by a blank.

When EOA = NUMERICS is used, the delimiter is usually the input sequence number.

The delimiter of a single destination within the message header is a blank character or the number of characters as specified in the MAXLEN operand of the TTABLE macro.

```
KEEP={YES}
     {NO }
```

Function: Specifies that you wish to have the series of one or more destination names MAXDEST = {maximum} removed from the message on output transmissions.

Format: YES or NO.

Default: KEEP = YES.

Note: If KEEP = NO is specified, the series of one or more destination names must be at the beginning of the message header and EOA must be coded.

KEEP = NO is usually coded in an inheader subgroup of an AMH.

```
MAXDEST={maximum}
        {5      }
```

Function: Specifies the greatest number of destination names allowed in the message header.

Format: maximum is an unframed decimal integer from 1 through 64.

Default: MAXDEST = 5.

Note: If the default (5) is taken or if the value specified is greater than 1, the EOA operand must be coded.

```
PROC={YES     }
     {NO      }
     {OPENONL}
```

Function: Specifies that a destination named in the message header may be an application program.

Format: YES, NO, or OPENONL.

Default: None. Specification is optional.

Note: If this operand is omitted, a destination specified in the message header that is the name of a TPROCESS macro causes the message to be rejected; however, if that destination is specified via the DEST operand rather than in the message header, omitting this operand does not cause a message rejection.

PROC=OPENONL indicates that a TPROCESS entry will be accepted as a valid destination if the associated application program's DCB has already been opened.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

```
THRESH=qmaximum
```

Function: Permits you to set a threshold value that indicates the maximum number of unsent messages that may be waiting on any main-storage-only queued destination to which DESTFLD is routing. When DESTFLD processing finds this threshold is exceeded for a destination, it sets a message error record bit for eventual user action.

Format: qmaximum is an unframed decimal integer from 1 through 127.

Default: None. Specification is optional.

Note: This operand is ignored when the destination is not using a main-storage-only queue.

qmaximum indicates the maximum number of messages expected to be queued at any one time, under normal traffic conditions, for a destination that uses main-storage-only queuing. This operand causes the MCP to set bit 22 in the message error record when the number of messages queued exceeds the user-specified number; the message being processed is then queued on the dead-letter queue.

If the destination is an application program, this operand permits you to delay activation of an application program until a certain number of messages are queued for that destination. You can also use this operand as an aid in controlling the efficient use of system resources when one or more destinations use main-storage-only queuing. For instance, to prevent main storage from being tied up with unsent messages, you can use it to warn of an excessive number of messages queued for a particular destination and then to issue a conditional macro such as CANCELMG, REDIRECT, SENDMSG to cause alternative processing of the message in your inmessage subgroup. If there are multiple destinations, the macros (except CANCELMG) are effective only if the number specified is exceeded for the first destination.

In order for this function to be provided for a destination, the destination must not be a cascade list or a distribution list.

```
TLIST={NO }
      {YES}
```

Function: Indicates that any destination field specified in the header may be the name of a distribution type TLIST entry.

Format: YES or NO.

Default: None. Specification is optional.

If this operand is omitted, a destination in the message header that is the name of a distribution list causes the message to be rejected; however, if that destination is not specified in the message header, omitting this operand does not cause a message rejection.

If this operand is omitted, a cascade-type TLIST entry may be specified as a destination in the message header without causing message rejection.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	Successful execution.
X'00000004'	Zero length buffer.
X'00000008'	Either: • No valid end-of-header pointer in the FHP. • The scan pointer is located within the data in the buffer but not within the header or the first buffer unit.
X'0000000C'	Is not a first buffer of a message.
X'00000010'	No FHP in the message buffer.
X'00000014'	The destination is an inactive application program and the user defined PROC=OPENONL.
X'00000018'	The end of the series of destination names was not found within the first buffer unit of the message.
X'0000001C'	The destination name is invalid.
X'00000020'	The destination does not have a TCSOPTS option field associated with it.

X'00000024'	A number of destination names greater than that specified by MAXDEST operand was specified in the message header.
X'00000028'	No destination name was found before the destination field delimiter.
X'FFFFFFFC'	The scan pointer is located beyond the data in the buffer.

DEVCON Macro

The DEVCON macro:

- Permits application programs that were originally written for IBM 2260 Display Stations to be used, without alteration, by display stations of the IBM 3270 Information Display System
- Converts outgoing 2260 data streams to 3270 format
- Converts incoming 3270 data streams to 2260 format
- Is required in a DMH which may process extended operator commands entered by IBM 3270 stations
- May be coded more than once.

The DEVCON macro allows you to convert your station hardware from 2260s to 3270s without altering or reassembling your existing 2260-based application programs. In addition, messages from 3270 devices that are processed by DEVCON may be sent to the extended operator control program, thereby providing you with the option of specifying a 3270 device as an TCAM operator control station. The DEVCON macro may be issued in the outheader subgroup of the message handler associated with the 3270 devices to convert data streams constructed for 2260s to a format that can be sent to a 3270. When the DEVCON macro is issued in the inheader subgroup of the message handler, it converts the 3270 input data stream to a 2260 equivalent.

In order for DEVCON to operate accurately, you must meet the following requirements:

1. The BUFSIZE of the GROUP macro for a 3270 station must specify a size large enough to handle the largest screen to be displayed or received.
2. The 3270 station operator must delimit each line of input with a displayable end-of-line character before he depresses the 3270 new-line key to advance the display cursor for the next line of input. You can specify any 3270 keyable character as the displayable end-of-line symbol, or you can use the default value established by DEVCON, which is the single quote character(′). This end-of-line character is necessary on input because the 3270 data stream format does not include a transmitted new line character equivalent to the 2260 data stream use of X′15′ to indicate that the New Line key was depressed. On output, wherever the 2260 application program has inserted a new-line symbol (X′15′), DEVCON processing will replace it with the user-selected end-of-line character.

As a result, assuming the default end-of-line symbol is used, outgoing 2260 New Line characters (X′15′) are converted to single quotes; on the incoming side, single quotes are converted to X′15′s.

In the inheader subgroup, the data format for 3270s is:

```
A   C   C
I   U   U (text)
D   R   R
```

The output 3270 data stream created by DEVCON:

```
E    C   W
S    M   C (text)
C    D   C
```

For output, the DEVCON macro leaves the escape character (ESC) which must be removed (using MSGEDIT) for output to 3270s.

Write at Line Address (WLA)

After successful execution of the DEVCON macro, register 15 is set to a return code of 0 and:

1. In the inheader subgroup:
   - The control characters are removed (5 leading characters and 3 leading characters for SNA 3270s)
   - User-specified displayable New Line symbols (NEWLINE=keyword) are replaced with the 2260 New Line symbol (X'15')
   - A 2260 SMI character is inserted as the first character of the newly converted data stream if GENSMI=YES
   - The data length is adjusted according to the input stream mode (formatted or unformatted). For formatted input, the length computation is based on the number of lines (user NL), on the buffer address of the first formatted field, and on the current cursor address. For unformatted input, the length remains as received.
2. In the outheader subgroup:
   - 2260 New Line symbols (X'15') are replaced with the user-specified displayable New Line symbol; SBA (set buffer address) orders are inserted to move the next set of data to the next 3270 row
   - After each New Line symbol appears, each row is padded with blanks using the Repeat-to-Address command
   - The buffer is scanned for an SMI character if SZE2260=NONE is not specified; if no SMI is found, an SMI is inserted at the end of the output data. The SMI character is then replaced by an IC command and an SF command and attribute-byte sequence.

The 3270 cursor is positioned after the last output character.

After an unsuccessful execution of the DEVCON macro, the scan pointer and all data in the buffer are unaltered and register 15 is set to a return code as shown in "Return Codes."

Supported Resources and General Requirements: SNA.

Valid subgroups: Inheader and outheader.

DEVCON

When Used in an Inheader Subgroup

NAME	OPERATION	OPERANDS

[symbol]	DEVCON	PROCESS=INPUT
		[,GENSMI={NO }
		{YES}
		[,NEWLINE={C'''' }]
		{C'char'}
		[,SZE3270={LARGE}]
		{SMALL}

When Used in an Outheader Subgroup

NAME	OPERATION	OPERANDS

[symbol]	DEVCON	PROCESS=OUTPUT
		[,ERASE={YES}
		{NO }
		[,LINSKIP={YES}]
		{NO }
		[,NEWLINE={C''''}]
		{char }
		[,SZE2260={LARGE}]
		{SMALL}
		{NONE }
		[,SZE3270={LARGE}]
		{SMALL}
		[,WCC={X'C3' }]
		{C'char'}
		[,WLA={NO }]
		{YES}

```
ERASE={YES}
      {NO }
```

Function: Specifies whether the screen should be erased prior to writing.

Format: YES or NO.

Default: ERASE=YES.

Note: YES specifies that the screen will be erased prior to writing. NO specifies that the screen will not be erased prior to writing.

```
GENSMI={NO }
       {YES}
```

Function: Specifies whether DEVCON is to provide a 2260 SMI character as the first character of the converted input stream.

Format: YES or NO.

Default: GENSMI = NO.

Note: NO specifies that DEVCON is not to generate the SMI. YES specifies that the first character in the converted data stream will be an SMI character.

```
LINSKIP={YES}
        {NO }
```

Function: Specifies whether the DEVCON macro should write only on alternate lines of the 3270 screen.

Format: NO or YES.

Default: LINSKIP = NO if SZE3270 = SMALL; otherwise LINSKIP = YES.

Note: NO specifies that the function is not to be performed and that all lines of the screen may be used for output data. YES specifies that:

- New Line symbols cause a line to be skipped after every line of data displayed on the 3270 screen.
- Write-at-line-address (WLA) characters are doubled to compute the correct starting address, and changed to the next odd-numbered line if the WLA specified is for an even-numbered line.

This function enables you to employ the full-screen size of the utilized 3270 even if the application programs were written only for a 12-row display screen.

This operand is valid only if SZE3270 = LARGE is specified and is ignored if it is coded in an inheader subgroup.

```
NEWLINE={C''''   }
        {C'char'}
```

Function: Specifies the displayable pseudo-New Line symbol; this permits simulation of the 2260 New Line function.

Format: C'''' or *C'char'*. *C'char'* is a single, nonblank character that must be framed with C'' characters. C'''' represents a single quote sign (the assembler requires that single quote signs be specified twice).

Default: C''''

Note: For PROCESS = OUTPUT, this character replaces the 2260 New Line symbols (X'15').

For PROCESS = INPUT, the incoming character string is scanned and, wherever detected, replaced with a 2260 New Line symbol (X'15').

```
PROCESS={INPUT }
        {OUTPUT}
```

Function: Specifies whether the macro is processing in the inheader or the outheader subgroup.

Format: INPUT or OUTPUT.

Default: None. Specification is required.

Note: INPUT specifies that the macro is being executed in an inheader subgroup and must convert a 3270 input stream to 2260 format. The NEWLINE, SZE3270, and GENSMI operands are valid when INPUT is specified; all other operands are ignored.

OUTPUT specifies that the macro is executing in an outheader subgroup and must convert a 2260 output stream to 3270 format. The NEWLINE, SZE2260, WCC, SZE3270, LINSKIP, WLA, and ERASE operands are valid if PROCESS=OUTPUT is coded; all other operands are ignored.

```
symbol
```

Function: Specifies the name of the macro.

Format: symbol must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

```
SZE2260={LARGE}
        {SMALL}
        {NONE }
```

Function: Specifies the size of the 2260 screen for which the application programs were written.

Format: SMALL, LARGE or NONE.

Default: SZE2260=LARGE.

Note: SMALL specifies a 12-row, 40-column screen size.

LARGE specifies a 12-row, 80-column screen size.

NONE specifies that the output has no special 2260 control characters. The user New Line characters are not displayed, and output is single-spaced from the top of the screen, with truncation, if it exceeds the size of a single screen. If SZE2260=NONE is specified, the LINSKIP and WLA operands will be ignored.

This operand is ignored if coded in an inheader subgroup.

```
SZE3270={SMALL}
        {LARGE}
```

Function: Specifies the size of the 3270 screen.

Format: SMALL or LARGE.

Default: SZE3270=LARGE.

Note: SMALL specifies a 12-row, 40-column screen size. LARGE specifies a 24-row, 80-column screen size.

```
WCC={X'C3'  }
    {X'char'}
```

Function: Specifies the 3270 Write Control Character (WCC) to be used for output (see the *3270 Component Description* manual for a discussion of the 3270 WCC).

Format: X′C3′ or X *′char′*. X *′char′* consists of two hexadecimal digits that must be framed with X″ characters.

Default: WCC=X′C3′.

Note: WCC X′C3′ indicates that MDT bits are to be reset and the keyboard restored to permit data entry when the message is displayed on the screen.

WCC=X′C7′ indicates that the same functions as X′C3′ are desired, and that an alarm is to be sounded.

```
WLA={NO }
    {YES}
```

Function: Specifies if this output stream contains a Write-at-Line Address (WLA) character as the first character of the output data stream. This operand is used when the application programmer wants to begin writing to the 2260 at a specific line address.

Format: NO or YES.

Default: WLA=NO.

Note: WLA=NO specifies that the first character is not a WLA character.

If WLA=YES is coded, DEVCON assumes that the character pointed to by the scan pointer (in the outheader subgroup) is the 2260 line address character and generates the appropriate 3270 SBA orders to cause the output to begin at the correct line address. If the character is not a valid line address, DEVCON constructs a character string that is written beginning on row 1 of the 3270.

Return Codes

One of the following return codes is set in register 15:

Code	Meaning
X'00000000'	Successful execution.
X'00000004'	Zero-length buffer.
X'00000008'	Scan pointer points to location beyond the first unit.
X'0000000C'	Either: ● Message is not complete in this buffer. ● Not a duplicate header buffer.
X'00000010'	Output data string is longer than the available buffer and there are no extra units available.
X'00000014'	Data in buffer exceeds 2300 characters.
X'00000018'	No data after control characters are removed from the buffer.
X'FFFFFFFC'	Scan pointer points to location beyond the data in the buffer.

DONTSAVE Macro

The DONTSAVE macro:

- Lists the names of external LUs and application programs having message queues on disk that are not to be saved when a Save Unsent Traffic command specifying ALL is executed by the save/restore message queues DKJSMQ
- Is coded only in the DKJNSQ module.

Supported Resources and General Requirements: Not applicable.

Valid subgroups: Not applicable.

NAME	OPERATION	OPERANDS
	DONTSAVE	(qname[,qname...,qname])

`(qname[,qname])`

Function: Specifies the names of all queues that should not be saved when the Save Unsent Traffic command specifying ALL is executed.

Format: *qname* must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. At least one name is required.

Note: Each *qname* is the name of an entry in the TCAM terminal table and must be specified in alphanumeric order. This macro can be issued only once in the NSQSRC module.

EDUNITS Macro

The EDUNITS macro:

- Reduces the possibility of MSGEDIT insert operation failure due to a shortage of available buffer units
- Analyzes utilization of system buffer units and updates peak utilization count for use by the Display Buffer Unit Status extended operator command
- Must be issued once in the initialization section of the MCP (after INTRO and before READY)
- May be issued more than once in an MH.

The MSGEDIT macro tries to obtain an available unit from the buffer-unit pool when you request an insert operation. If there are no units available, MSGEDIT does not process any data; it sets a return code and returns control to the MH. Therefore, unless the return code is checked and appropriate action taken, the unprocessed message is transmitted (on the outgoing side) or is placed on the queue (on the incoming side).

The EDUNITS macro is designed to reduce the chances of failure of an insert operation by maintaining a chain of emergency MSGEDIT units that are taken from the buffer by the MSGEDIT macro if there are insufficient buffer units to perform an insertion.

The EDUNITS macro with the RESERVE operand is issued once in the initialization section of the MCP. With this macro, you specify how many emergency MSGEDIT units must be reserved for the insert operation.

The EDUNITS macro may also be issued before any MSGEDIT macro requesting insert operations. In this case, you specify the NEED operand to indicate approximately how many units are required to perform the operation. If there are fewer buffer units available in the buffer-unit pool than are specified, the macro returns units to the buffer-unit pool to permit execution of the MSGEDIT insertion. If there are more buffer units available than are specified, the macro takes units from the buffer-unit pool and puts them on its chain of units (for possible future emergency use) until the RESERVE value is satisfied.

Supported Resources and General Requirements: ALL.

Valid subgroups: Inheader, inbuffer, outheader, and outbuffer.

NAME	OPERATION	OPERANDS

[symbol]	EDUNITS	RESERVE=integer
		[,NEED={1 }]
		{integer}

```
NEED={1       }
     {integer}
```

Function: Specifies the approximate number of units that are needed to perform the next MSGEDIT insert operation.

Format: *integer* is an unframed decimal integer from 0 through 10.

Default: NEED=1.

Note: If NEED=0 is coded, no additional units are required, but unit initialization analysis is performed for use by the Display Buffer Unit Status extended operator command. In an extended network, code at least one EDUNITS macro in the internodal message handler to ensure proper updating of the peak utilization count.

When the NEED operand is coded, that EDUNITS macro must be included only in an MH.

`RESERVE=integer`

Function: Specifies the number of emergency MSGEDIT units to be reserved for the insertion operation.

Format: *integer* is an unframed decimal integer from 1 through 30.

Default: None. Specification is required once in the MCP initialization section if EDUNITS macros are to be issued in any message handlers.

Note: When the RESERVE operand is coded, that EDUNITS macro must not be included in an MH.

`symbol`

Function: Specifies the name of the macro.

Default: None. Specification is optional.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	The number of units specified by the NEED operand are present
X'00000004'	There are fewer units available than were specified by the NEED operand.

ERRORMSG Macro

The ERRORMSG macro:

- Queues a message when either an error occurs or you want to be notified of a successful operation (non-error condition)
- May be used more than once in a subgroup
- Allows error message text with or without a header to be sent to a destination.

If the message being processed requires a fixed header prefix, the SENDMSG macros should be used to generate the message. For information on SENDMSG and NEWMSG macro processing, see the macro descriptions later in this chapter.

ERRORMSG sends a user-specified error message to a designated external LU or TPROCESS entry when errors indicated by the error-mask have occurred, provided you code CONNECT=OR or CONNECT=AND. The bits specified by the error mask operand are compared with the setting of the bits in the message error record for this message; if specified bits in the message error record are on, the error message is sent. If you code CONNECT=NAND, ERRORMSG sends messages to a designated station or TPROCESS entry if the bits specified by the error mask operand are off. The message may be sent unconditionally by specifying an all-zero mask or by omitting the *mask* operand.

The message sent may take one of three forms:

- It may include the text written by you preceded by the header of the message in error, which is recalled from the message queue. If the original message was queued in main storage, TCAM may not be able to recall it; therefore, the ERRORMSG macro may not execute.
- It may include text only (no header recall).
- It may include text plus RH and sense bytes.

The amount of user text returned depends on the location of the scan pointer at the time ERRORMSG is issued.

The error message, once formatted, is placed on the destination queue for the external LU or application selected to receive the message and is handled by the group of the MH for that queue. Therefore, unless a MSGTYPE or PATH macro distinguishes between different message types, the format of the header of the message in error must be compatible with the macros executed in the outgoing group handling messages routed to the destination selected to receive the error message. If the MSGTYPE macro is used for this purpose, the formats of the respective headers may differ after the message-type character.

The outgoing group that handles the error message for the destination may have an outmessage subgroup containing ERRORMSG and REDIRECT macros. TCAM does not generate subsequent error messages based on an error condition that may be related to the original error message; this prevents an accumulation of error messages.

You may prefer to use the MSGGEN function if the message header is not required as a part of the error message. MSGGEN uses less CPU processing since it does not access the message queue, but the user will not perceive a difference. However, ERRORMSG can return the header of the message in error, while MSGGEN does not. If it is necessary to send an error message for input errors when no data has been transferred (for example, due to an operator or hardware error), then either MSGGEN or ERRORMSG with HEADER = NO may be used.

If cancellation of an erroneous message is required, the CANCELMG macro must have been issued before the ERRORMSG macro. ERRORMSG may appear in inmessage and outmessage subgroups and can appear more than once in either subgroup.

Note: If HEADER = YES is coded then the message header must be recalled from a destination queue. This requires that the message have a valid destination prior to the execution of this macro. See the FORWARD macro for determining valid destinations.

The logical source of an error message is the same as the source of the message that caused the error message to be generated, except when HEADER = NO is coded in the outmessage subgroup.

When ERRORMSG is executed with HEADER = YES specified, only the first buffer of the message in error is retrieved from the destination queue. (If the header occupies more than one buffer, that portion of the header extending beyond the first buffer is not retrieved.) The actual error message is placed in that portion of the first header buffer that contains message text; the error message overlays the text. If HEADER = NO is specified, TCAM allocates a buffer from the available buffer pool to contain the message. If the first buffer is entirely filled with header information or does not contain enough space after the header to hold the entire error message, TCAM automatically assigns one extra unit to the buffer to hold as much as possible of the remainder of the message. If the entire message will not fit, the remainder is truncated on the right.

The message is inserted in the header beginning at the current location of the scan pointer. If an ERRORMSG macro is issued in the inmessage subgroup but there is additional header information that is recognized by the outheader subgroup, the message overlays this data and data is lost for outgoing processing. If data has been inserted or removed during inbuffer or outbuffer processing, the data in the buffer is moved either to the right or the left while the scan pointer remains fixed. Thus, when the error message is inserted at the scan pointer, data that is logically part of the header may be lost or may not be present and, data beyond the header or residual data may be included as part of the header information returned with the message.

Supported Resources and General Requirements: ALL (if APP, in inmessage subgroup only).

Valid subgroups: Inmessage and outmessage.

NAME	OPERATION	OPERANDS

[symbol]	ERRORMSG	[mask]
		,DATA={fieldname}
		{message }
		[,CONNECT={AND }]
		{OR }
		{NAND}
		[,DEST={destination name}]
		{opfield }
		{ORIGIN }
		{DESTIN }
		[,EXIT=name of routine]
		[,HEADER={YES}]
		{NO }

```
CONNECT={AND }
        {OR  }
        {NAND}
```

Function: Specifies the type of logical connection to be made between the mask and the message error record.

Format: AND, OR, or NAND.

Default: CONNECT=OR.

Note: AND specifies that the macro is to be executed only if *all* of the bits specified by *mask* are on in the message error record.

OR specifies that the macro is to be executed if *any* bit specified by *mask* is on in the message error record.

NAND specifies that the macro is to be executed only if *all* of the bits specified by *mask* are off in the message error record.

```
DATA={fieldname}
     {message  }
```

Function: Specifies the error message.

Format: *message* or *fieldname*. *message* must be specified within framing C'' or CLn'' *fieldname* must conform to the rules for assembler language symbols.

Default: None. This operand is required.

Maximum: The error message is a maximum length of 255 characters. This is exclusive of the binary count in the *fieldname* format.

Note: *message* is the actual error message to be sent.

fieldname is the name of a location containing in its first byte a binary count of the number of characters in the message, followed by the message itself.

If an error message is longer than a single buffer unit, one additional buffer unit is obtained and as much of the remainder of the message as will fit is placed in it. If the entire message will not fit into these two units, the remainder is truncated on the right.

```
DEST={destination name}
     {opfield         }
     {ORIGIN          }
     {DESTIN          }
```

Function: Specifies the destination for the error message.

Format: destination name, opfield, ORIGIN or DESTIN. *destination name* must be enclosed in framing C'' or CLn'' *opfield* must conform to the rules for assembler language symbols and must not be specified with framing characters.

Default: In an inmessage subgroup, DEST=ORIGIN. In an outmessage subgroup, DEST=DESTIN.

Note: destination name is the name of an external LU or a process entry in the terminal table.

opfield is the name of an option field defined by an OPTION macro that contains the name of an external LU or process entry in the terminal table.

ORIGIN specifies that the error message is to be sent to the origin of the message. If the origin is not known, the message is sent to the dead-letter queue, if one is specified by the DLQ= operand of the INTRO macro, or is lost otherwise. This operand may be specified in either an inmessage or outmessage subgroup when HEADER=YES is coded. If HEADER=NO is specified, ORIGIN may be coded only in an inmessage subgroup.

DESTIN specifies that the error message is to be sent to the destination specified in the header of the message in error. If an invalid destination is specified or if DESTIN is specified in an inmessage subgroup and no FORWARD macro has been issued for this message in the inheader subgroup, the message is sent to the dead-letter queue if one has been specified by the DLQ= operand of the INTRO macro. If no dead-letter queue is specified, the message is overlaid and lost. This operand may be specified in either an inmessage or outmessage subgroup if HEADER=YES is coded. If HEADER=NO is coded, DESTIN may be specified only in the outmessage subgroup.

A distribution list or a PUT process entry must not be specified as the destination of an error message.

```
EXIT=name of routine
```

Function: Specifies the name of a user-written routine that alters error message processing. This routine may alter the text of the error message before incorporating the text into a header buffer.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Note: The user exit is not entered if the destination specified by the DEST= operand is an invalid destination. If the user provides an exit routine for ERRORMSG, TCAM automatically saves and restores registers for this routine; the user need not save registers, and may change the contents of registers 2 through 12 as he likes. However, the contents of registers 13 and 14 must not be altered

When the routine receives control, register 1 contains the address of the header buffer, and register 5 contains the address of the error-message text.

The contents of registers 6 and 7 depend on whether ERRORMSG is issued in the inmessage or outmessage subgroup as illustrated in the following table. Within the inmessage subgroup, register contents are also affected by the coding order of the IEDRESP and ERRORMSG macros. In the following table, RRRRRR represents the RH of the exception response and SSSSSSSS represents the SNA sense bytes.

Subgroup	Register	Contents	Explanation
Inmessage	6	X'00000000'	ERRORMSG coded before IEDRESP.
	7	X'00000000'	When both registers contain zeros, no exception conditions have been detected up to the time of error message execution. A positive response either has already been sent or will be sent later if required.
	6	X'00RRRRRR'	IEDRESP coded before ERRORMSG.
	7	X'SSSSSSSS'	At the time of error message execution, either an exception response has already been sent (IEDRESP) or an exception response has been generated and will be sent later provided the request allows an exception response to be sent.
Outmessage	6	X'00000000'	A positive response was
	7	X'00000000'	received to this message (chain).
	6	X'00RRRRRR'	An exception response was received
	7	X'SSSSSSSS'	to this message (chain).

Register 14 contains the return address for the calling routine. Register 15 contains the address of the entry point for the user routine. TCAM expects no return code from the user routine. The routine should return control to TCAM by a BR 14 instruction.

The LOCOPT macro (STATION= operand) and the IEDSHOW macro (TSTATUS, STATION= and CURTERM operands) are valid in this user-written routine to obtain the contents of various TCAM control blocks. For detailed information on these macros, see the applicable macro description later in this chapter.

```
HEADER={YES}
       {NO }
```

Function: Specifies whether the error message text is to be inserted in a recalled copy of the message header or into a new buffer from the buffer pool.

Format: YES or NO.

Default: HEADER=YES.

Note: If HEADER=YES is coded, error message text is inserted in a recalled copy of the message header. If for any reason the recall fails, then you should not expect to receive the error message. This situation can arise any time a recall is attempted and there is no valid destination field in any of the messages on the destination queue. If HEADER=NO is coded, then the error message text is inserted into a new buffer from the buffer pool.

`mask`

Function: Specifies the 5-byte bit configuration used to test the message error record for the message. (The message error record is described in Appendix A.)

Format: Unframed decimal integer or hexadecimal field. If hexadecimal format is used, framing characters must be specified. If X'' is used, leading zeros must be coded. If XL5'' is used, leading zeros may be omitted.

Default: None. Specification is optional.

Note: Omitting this operand or specifying an all-zero mask causes unconditional execution of the macro.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Example

You can use the exit provided via the EXIT= operand of ERRORMSG to perform such functions as providing the station operator with the correct input sequence number if he enters an invalid number. The following example shows how you can code the routine:

```
GETSEQ   CSECT
         USING    GETSEQ,12
         USING    IEDQAVTD,11
         USING    IEDQPRF,3
         USING    IEDQTRM,1
         LR       12,15             SAVE ENTRY AND SET BASE
         LR       2,14              SAVE RETURN ADDRESS
         LR       3,1               SAVE BUFFER ADDRESS
         LR       4,0               SAVE REGISTER 0
         LH       1,PRFSRCE         GET SOURCE INDEX
         N        1,AVTCLRHI        CLEAR HIGH TWO BYTES
         BZ       NOGO              IF YES-CANNOT GET SEQUENCE
*
         L        15,AVTRNMPT       GET TCAM INTERNAL ROUTINE
         BALR     14,15             GIVE IT CONTROL
*
         LH       5,TRMINSEQ        GET INPUT SEQUENCE
*                                   IT IS IN BINARY FORMAT
*                                   PROCESS IT AS REQUIRED
         B        EXIT              BRANCH TO COMMON EXIT
*
NOGO     EQU      *                 DO WHATEVER PROCESSING IS
*                                   NEEDED IF NO SEQUENCE
*
EXIT     EQU      *
         LR       1,3               RESTORE BUFFER ADDRESS
         LR       0,4               RESTORE REGISTER 0
         LR       15,12             RESTORE ENTRY POINT
         LR       14,2              RESTORE RETURN ADDRESS
         BR       14                RETURN TO TCAM
*
         TAVTD                      AVT DSECT
         TPRFD                      PREFIX DSECT
         TTRMD                      TTE DSECT
*
         END
```

Return Codes

None.

ERRTEST Macro

The ERRTEST macro:

- Permits a test for zero-length buffers
- Permits a test of the flags in the message error record
- May be issued more than once.

The ERRTEST macro may be used in a message handler to find out if, for example, the header buffer of a message has been received with no transmission errors detected.

The IEDTCSD macro should be included in the MCP to provide dummy sectio.ı (DSECTS) of control blocks referenced by the ERRTEST macro.

Supported Resources and General Requirements: ALL.

Valid subgroups: Inblock, inheader, inbuffer, outheader, and outbuffer.

NAME	OPERATION	OPERANDS
[symbol]	ERRTEST	[BRANCH=label] [,MASK=m] [,ZEROBUF=label]

`BRANCH=label`

Function: Provides the label of the next instruction to be executed if any of the bits in the user-specified mask are on in the message error record.

Format: *label* must conform to the rules for assembler language symbols.

Default: None. Specification is required if the MASK= operand is coded.

Note: *label* is the symbolic name of the next instruction to be executed if the tested bits are on. This must be the name of an instruction in the current subgroup or the name of a delimiter macro.

`MASK=m`

Function: Provides the 5-byte mask to be tested against the message error record.

Format: *m* is a 5-byte hexadecimal value with framing characters XL5'' (a total of 15 coded characters) or X'' (a total of 13 coded characters).

Default: None. Specification is optional.

Note: The format and bit settings of the message error record are described in Appendix A.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

`ZEROBUF=label`

Function: Provides the label of the next instruction to be executed if the current buffer is a zero-length buffer.

Format: *label* must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Note: *label* is the symbolic name of the next instruction to be executed if the current buffer is a zero-length buffer. This must be the label of an instruction in the current subgroup or the name of a delimiter macro.

Return Codes

None.

EXMSG Macro

The EXMSG macro:

- Specifies the characteristics of one message to be generated
- Provides a user exit to allow modification of a message when it is generated
- May be coded in a non-executable section of an MCP.

The EXMSG macro specifies the destination, text, and other characteristics of a message to be generated by the NEWMSG and SENDMSG macros. All of the EXMSG macros defining generated messages must be coded contiguously and specified with a name. The entire series of EXMSG macros defines the exception request table, the end of which is delimited by an EXMSG macro coded with no name and TEXT=LAST. The name of the EXMSG macro is equated to the number of the entry being created the name of the first EXMSG macro is equated to 1, the second to 2, etc.

If EXMSG macros are being coded in an MCP that is not a model MCP (the model MCP generates the required labels) and does not have an IEDTCSD macro coded in it, one EXMSG macro specifying DSECT=YES must be included to generate certain labels required by the macro. Do not code a EXMSG macro specifying DSECT=YES in an MCP that contains an IEDTCSD macro because IEDTCSD also generates the required labels.

EXMSG is used with the NEWMSG and SENDMSG macros to generate messages in the following manner:

- NEWMSG: The name of the equated value of the EXMSG macro may be coded as the parameter of the MSG operand causing the NEWMSG option field to be set to the equated value associated with this message.
- SENDMSG: If the EXIT operand is omitted, SENDMSG processing automatically accesses the exception request table to get the appropriate message to be generated.

See the discussions of the NEWMSG and SENDMSG macros for more information on message generation.

Supported Resources and General Requirements: ALL.

Valid subgroups: Not applicable.

EXMSG

NAME	OPERATION	OPERANDS
[symbol]	EXMSG	[COUNT={NO }]
		{YES}
		[,CPYDEST={ORIG }]
		{DAF }
		{OPCTL }
		{KEY(name) }
		{TERMNAM(name)}
		[,CURRNDE=integer]
		[,DAF=integer]
		[,DATE={NO }]
		{YES}
		[,DEST={ORIG }]
		{DAF }
		{OPCTL }
		{KEY(name) }
		{TERMNAM(name)}
		[,DSECT={NO }]
		{YES}
		[,EXIT=exitname]
		[,EXITTYP={A}]
		{V}
		[,EXTEXT=integer]
		[,FHP={NO }]
		{YES}
		[,GRP=integer]
		[,MER=integer]
		[,NEWLINE={YES}]
		{NO }
		[,NOTFYOP={NO }]
		{YES}
		[,OAF=integer]
		[,OUTSEQ={NO }]
		{YES}
		[,PRI=integer]
		[,ROUTERR={NO }]
		{DLQ}
		[,TC=integer]
		[,TERMNAM=integer]
		[,TEXT={LAST }]
		{'data'}
		{addr }
		{USER }
		[,TIME={NO }]
		{YES}

```
COUNT={NO }
      {YES}
```

Function: Specifies whether to increment the logical error count for the external LU or application program associated with the message that caused the exception condition. The ERRCOUNT option field must have been defined for that external LU or application program. That option field may be displayed via a Display Resource Status (DATA) extended operator command.

Format: NO or YES.

Default: COUNT=NO.

Note: NO specifies that the count is not to be incremented. YES specifies that the count should be incremented.

```
CPYDEST={ORIG         }
        {DAF          }
        {OPCTL        }
        {KEY(name)    }
        {TERMNAM(name)}
```

Function: Specifies an additional destination that is to receive a copy of the message being generated.

Format: ORIG, DAF, OPCTL, KEY(name), or TERMNAM(name).

Default: None. Specification is optional.

Note: If EXIT is specified, the user exit will gain control when the message is being generated for its primary destination (as specified via the DEST operand) and when it is being generated for its copy destination (CPYDEST operand). If the user exit must determine which of these conditions it is being invoked for, it can test the NEMCOPY bit in the NEWMSG option field associated with source (for input processing) or destination (for output processing) of the message currently being processed. If this bit is on, the message is being generated for the destination specified.

```
CURRNDE=integer
```

Function: Specifies that the node identifier for the current host node is to be inserted in this exception request.

Format: *integer* is an unframed decimal integer from 1 to 255.

Default: None. Specification is optional.

Note: *integer* is the character position at which the node identifier is to be inserted. Allow 3 bytes for insertion of this node identifier. The node identifier is zero-suppressed and right-justified.

```
DAF=integer
```

Function: Specifies that the TDAF of the message being processed when this message is generated should be inserted in the generated message.

Format: *integer* is an unframed decimal integer from 1 to 255.

Default: None. Specification is optional.

Note: *Integer* is the character position at which the TDAF is to be inserted. Allow 9 bytes for insertion of the TDAF. The format is nnn-rrrrr; *nnn* represents the node, *rrrrr* represents the resource identifier. The node identifier is zero-suppressed and right-justified.

```
DATE={NO }
     {YES}
```

Function: Specifies whether to date stamp the generated message.

Format: NO or YES.

Default: DATE=NO.

Note: NO specifies that the message should not be date stamped. YES specifies that the message should be date stamped.

If requested, the output date is scheduled to be inserted in the message after the sequence number if OUTSEQ=YES is specified. Blanks need not be left in the text of the message for insertion of this field; but it is included in the total message length, which may not exceed 255 characters. The outgoing date field is 7 bytes long. For the date to be inserted, a TCSENDBL macro must be executed in the outgoing group processing the generated message.

```
DEST={ORIG         }
     {DAF          }
     {OPCTL        }
     {KEY(name)    }
     {TERMNAM(name)}
```

Function: Specifies the destination of this message.

Format: ORIG, DAF, OPCTL, KEY(name), or TERMNAM(name).

Default: DEST=ORIG.

Note: ORIG specifies that the destination is the originator of the message that caused this message to be generated.

DAF specifies that the destination is signified via a user-supplied TDAF found in a user-generated FHP.

OPCTL specifies that the destination is the extended operator control program. If the message being generated is an extended operator control command (DEST = OPCTL and/or CYPDEST = OPCTL), SENDMSG processing removes all blanks within the one or more operand fields of the message before routing it. The only generated reply to this extended operator control command is sent to the primary extended operator control station if NOTFYOP = YES is coded.

KEY(*name*) specifies that the message being generated is to be sent to the destination represented in the key table by the key specified by (*name*).

TERMNAM (*name*) specifies that the message being generated is to be sent to the destination specified by (*name*). *name* is the name of the TERMINAL, TPROCESS, or TLIST macro defining the destination.

If DAF is specified, a user-generated FHP must be provided; therefore FHP = YES and EXIT = *addr* must also be specified.

```
DSECT={NO }
      {YES}
```

Function: Generates a dummy section describing the format of an exception request table entry.

Format: NO or YES.

Default: DSECT = NO.

Note: If YES is specified, all other EXMSG operands are ignored and the dummy section describing an exception request table entry is generated.

```
EXIT=exitname
```

Function: Specifies the name of a user exit routine which may:

- Supply the text of the message
- Modify the text of the message
- Generate an FHP for the message.

Format: *exitname* must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary..")

Default: None. Specification is optional.

Note: On entry to the user exit routine, register 1 points to the SENDMSG parameter list.

The SENDMSG parameter list contains the length of the message text (the SNDUDALN parameter in the SENDMSG DSECT) and a pointer to the message text (the SNDUDATA parameter in the SENDMSG DSECT). If TEXT = USER is specified, both of these fields are zero and must be filled in by the user at this time (see the TEXT operand). If the length of the message text is changed by this exit, the corresponding value in the

SENDMSG parameter list must be changed. If FHP=YES is specified, this exit must supply an FHP and place its address in the SNDUFHP field.

```
EXITTYP={A}
        {V}
```

Function: Indicates the location of the user-written routine specified in the EXIT operand.

Format: A or V.

Default: EXITTYP=A.

Note: A specifies that the routine is assembled within the MCP (an A-type address constant is generated). V specifies that the user-written routine is a module in the load library (a V-type address constant is generated).

```
EXTEXT=integer
```

Function: Specifies that the text from the message being processed is to be inserted at the end of the generated message.

Format: *integer* is an unframed decimal integer from 1 to 255.

Default: None. Specification is optional.

Note: *integer* is the number of characters of the original message (starting after the FHP) to insert at the end of the generated message.

You need not leave blanks in the text of the message for insertion of this field. It is added on to the end of the generated message.

The resulting total length must not exceed 255 characters.

```
FHP={NO }
    {YES}
```

Function: Specifies whether a user-generated FHP is to be supplied for the generated message.

Format: NO or YES.

Default: FHP=NO.

Note: NO specifies that the user will not supply an FHP; in this case, SENDMSG processing creates an FHP for the message.

YES specifies that the user will supply an FHP. If YES is specified, EXIT=*addr* must also be specified. When the user exit gains control, it must build an FHP for the message and store its address (the SNDUFHP parameter in the SENDMSG DSECT) in the SENDMSG parameter list.

```
GRP=integer
```

Function: Specifies that the name of the group of the origin (for input processing) or destination (for output processing) of the message currently being processed is to be inserted in the generated message. If the relevant origin or destination is not a station, no action is taken.

Format: *integer* is a decimal integer from 1 to 255.

Default: None. Specification is optional.

Note: *integer* is the position within the generated message starting at which the group name is to be inserted. Allow 8 bytes for the insertion of the group names.

```
MER=integer
```

Function: Specifies that the TCAM message error record (MER) is to be inserted in the generated message.

Format: *integer* is an unframed decimal integer from 1 to 255.

Default: None. Specification is optional.

Note: *integer* is the position within the generated message starting at which the message error record is to be inserted. Allow 5 bytes for the insertion of the message error record. The format of the Message Error Record is a 10-digit hexadecimal constant.

```
NEWLINE={YES}
        {NO }
```

Function: Specifies whether any occurrence of // (double slash) characters in the message being generated should be replaced with the new-line symbol as specified by the NEWLINE operand on INTRO.

Format: YES or NO.

Default: If TEXT=USER is specified, then the default is NO; otherwise, the default is YES.

Note: YES specifies that any occurrence of // is to be replaced.

NO specifies that no replacement is to be made.

```
NOTFYOP={NO }
        {YES}
```

Function: When the message is an extended operator control command (indicated by either DEST=OPCTL or CPYDEST=OPCTL), this operand specifies whether to notify the primary extended operator control station of the disposition of that command.

Format: NO or YES.

Default: NOTFYOP=NO.

Note: NO specifies that the primary extended operator control station is not to be notified.

YES specifies that the primary-extended operator control station is to be notified.

```
OAF=integer
```

Function: Specifies that the TOAF of the message currently being processed when this message is generated should be inserted in the generated message.

Format: *integer* is an unframed decimal integer from 1 to 255.

Default: None. Specification is optional.

Note: *integer* is the character position at which the TOAF is to be inserted. Allow 9 bytes for insertion of the TOAF. The format is nnn-rrrrr; *nnn* represents the node identifier, *rrrrr* represents the resource identifier. The node identifier is zero-suppressed and right justified.

```
OUTSEQ={NO }
       {YES}
```

Function: Specifies whether to give the generated message an output sequence number.

Format: NO or YES.

Default: OUTSEQ=NO.

Note: NO specifies that the message is not to be sequenced. YES specifies that the message is to be sequenced.

If requested, the sequence number is scheduled to be inserted as the first data character of the message (after the FHP). Blanks need not be left in the text of the message for insertion of this field but it is included in the total message length, which may not exceed 255 characters. The output sequence number is 5 bytes long. For the output sequence number to be inserted, a TCSENDBL macro must be executed group processing the generated message.

```
PRI=integer
```

Function: Specifies the message priority of the generated message.

Format: *integer* is an unframed decimal integer from 1 to 255.

Default: None. Specification is optional.

Note: This message priority *overrides* the priority on the SENDMSG macro that is being processed. However, if FHP=YES is specified and the user-built FHP contains a valid message priority in the FHPICPRI, field, *that* message priority supersedes this operand.

```
ROUTERR={NO }
        {DLQ}
```

Function: Indicates the action to be taken if the generated message cannot be routed to its destination.

Format: NO or DLQ.

Default: ROUTERR=NO.

Note: NO specifies that the message is to be discarded entirely (that is, not sent). DLQ specifies that the message is to placed on the dead letter queue.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary")

Default: None. Specification is required unless TEXT=LAST is specified.

Note: The macro generates an equate of the *symbol* to the number of the entry being generated in the exception request table. The name of the first EXMSG macro coded is equated to 1, the second to 2, etc.

```
TC=integer
```

Function: Specifies which transmission category (and associated queue) to use when routing the exception request to another TCAM node.

Format: *integer* is an unframed decimal integer not greater than the number of transmission categories defined in this MCP; *integer* must not exceed 16, the maximum number of transmission categories in the MCP.

Default: None. Specification is optional.

Note: If this operand is omitted and the message destination is a key, the transmission category is taken from the key-table entry. If this operand is omitted and the message destination is not a key, or no transmission category is specified for a key, the transmission category is taken from the FHP or defaults to transmission category 1 if the transmission category in the FHP is invalid or missing.

```
TERMNAM=integer
```

Function: Specifies that the name of the origin (for input processing) or destination (for output processing) of the message currently being processed should be inserted in the generated message.

Format: integer is a decimal integer from 1 to 255.

Default: None. Specification is optional.

Note: integer is the position within the generated message starting at which the origin or destination name is to be inserted. Allow 8 bytes for the insertion of the name.

```
TEXT={LAST       }
     {'constant'}
     {addr       }
     {USER       }
```

Function: Specifies the text of the messages generated, or denotes the end of the EXMSG table.

Format: USER, *constant, addr,* or LAST. *constant* is a valid assembler language hexadecimal or character constant within appropriate framing characters. *addr* must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: TEXT=LAST.

Note: USER specifies that the text will be supplied at message generation through a user exit.

constant is the actual message text enclosed in quotes. It must be from 1 to 255 bytes in length and blanks must be left for the insertion of data when the message is generated if any of the following operands are specified: MER, TERMNAM, GRP, CURRNDE, OAF, or DAF.

addr is the address of an area containing the 1-byte binary length of the message text followed by the message text defined as a constant or list of constants. The text itself must conform to the length and format above for *constant* requirements given.

LAST denotes the end of the EXMSG table. If TEXT=LAST is specified, all other operands are ignored.

If TEXT=USER is specified, EXIT=name must also be specified. The user exit receives the address of the SENDMSG parameter list; it must build the message and store its length and address (in the SNDUDALN and the SNDUDATA fields in the SENDMSG DSECT) in the SENDMSG return parameter list. MER, TERMNAM, GRP, CURRNDE, OAF, DAF, and EXTEXT may not be specified; and NEWLINE=NO is assumed.

```
TIME={NO }
     {YES}
```

Function: Specifies whether to time-stamp the generated message.

Format: NO or YES.

Default: TIME = NO.

Note: NO specifies that the message is not to be time stamped. YES specifies that the message is to be stamped.

If TIME = YES is specified, the output time is scheduled to be inserted following the date and the sequence number of the message. Blanks need not be left in the text of the message for insertion of this field; but it is included in the total message length, which may not exceed 255 characters. The output time field is 5 bytes long. For the time to be inserted, a TCSENDBL macro must be executed in the outgoing group processing the generated message.

Examples

Suppose that under certain exception conditions detected during execution of the incoming group of an MH, the user wishes to generate an extended operator control command to stop certain traffic, and also to send a copy of that extended operator command to a external LU named MONITOR. The following in the inheader and inmessage subgroups causes the message to be generated:

```
INHDR
   .
   .
   .
NEWMSG MSG=STPLIN
   .
   .
   .
INMSG
   .
   .
   .
SENDMSG EXITCDE=NEWMSG.
```

Note that the EXIT operand is omitted from SENDMSG, causing the exception request table to be accessed to find the message. The EXMSG macro defining the message characteristics would be coded:

```
STPLIN  EXMSG    DEST=OPCTL,TEXT='OFFLN       ',PRI=1,GRP=7,
                 COUNT=YES,NOTFYOP=YES,CPYDEST=TERMNAM(MONITOR)
```

Note that the GRP operand causes the group name to be inserted starting at the 7th byte of the message text.

Suppose that you wished to define a message that would tell the originating station that a particular message which it had entered contained invalid data. The EXMSG macro would be coded in the following way:

```
name  EXMSG    DEST=ORIG,TEXT='INVALID DATA ENTERED AS FOLLOWS:',
               EXTEXT=24,OUTSEQ=YES,ROUTERR=DLQ
```

This would cause 24 characters from the error-causing message to be inserted (after the colon). The message would receive an output sequence number; if it could not be routed back to the origin, it would be sent to the dead letter queue.

The following message definition would cause a message containing the name of the origin of the message currently being processed, and the contents of the appropriate message error record (separated by one blank) to be transmitted to an error analysis application program represented by the routing key ERRPGM:

```
name   EXMSG     DEST=KEY(ERRPGM),TEXT=CL19' ',
                 TERMNAM=1,MER=10
```

Note: The model MCPs contain an exception table which illustrates the use of most EXMSG options.

Return Codes

None.

FHPBUILD Macro

The FHPBUILD macro:

- Constructs and initializes a fixed header prefix and inserts it after the buffer prefix in the first unit of the message header
- Is optional in the incoming group of the message handler
- Should not be executed more than once per message and is a prerequisite for some TCAM functions
- Must be issued when the scan pointer is pointing to the beginning of the first unit of the first buffer of the message
- Is not to be issued before the message is translated into EBCDIC (usually through the use of a CODE macro) unless no translation is to occur.

The FHPBUILD macro instruction constructs and initializes the fixed header prefix within the reserved section of the header buffer.

If the FHPBUILD macro is issued and no FHP is present in the message being processed, an FHP is inserted in the message. If BLDOAF=YES is specified, the TCAM origin address field (TOAF) in the FHP is initialized with the TCAM network address of the message originator. If BLDOAF=NO is specified or the BLDOAF operand is omitted, the TOAF in the FHP is initialized to all zeros.

If an FHP is already present in the message when the FHPBUILD macro is issued, the FHPBUILD macro reinitializes the input date and time stamps in the FHP. If BLDOAF=YES is specified, the TOAF in the FHP is reinitialized with the TCAM network address of the message originator. If BLDOAF=NO is specified or the BLDOAF operand is omitted, the TOAF in the FHP is not altered.

You must ensure that sufficient space is available for the FHP in the message header. For information on how to reserve space for the FHP, refer to the chapter of the *TCAM Installation Guide* titled, "Coding the Message Handler."

The FHPBUILD macro expects the scan pointer to point to the position before the first byte of data in the message; that is, to the start of the message header.

This macro is not issued in an internodal MH; in that case, an FHP will already be present. The INODEMH macro (GEN=INHDR), which must be issued in the inheader subgroup of an internodal MH, internally initializes the existing FHP from another network host node.

If an FHP already exists in a buffer and the message origin is an application program, the PRFFHP bit must be turned on prior to FHPBUILD execution. This may be done by coding the FHPTEST macro with ACTION=SETYES.

After successful completion of FHPBUILD processing:

1. The scan pointer is not moved.
2. Register 15 is set to a return code of 0.
3. An initialized FHP has been built, which includes the following data in the fields indicated:
   - A bit in the header buffer's prefix is turned on indicating an FHP has been built.
   - An offset value from the beginning of the FHP to the first message data beyond the FHP is placed in the FHPHEADP field.
   - An offset value from the beginning of the FHP to the last byte of data in the buffer unit is placed in the FHPTEXTP field.

- The input date and time are placed in coded form in the FHPIDATB and the FHPITIMB fields, respectively.
- If the user in an extended networking environment has coded BLDOAF=YES, the TCAM address of the origin has been placed in the FHPOAFLD field.
- Other FHP fields are set to hexadecimal zeros.

Note: FHP offset pointers are index values from the starting address of the FHP; that is, the first FHP field (FHPSTART) has an FHP offset value of 0. It is significant to note that offset pointer fields (such as the scan pointer 0 in a buffer prefix) are calculated as counts from the beginning of the buffer prefix as it appears in a main-storage buffer; that is, the first buffer prefix field in the buffer unit (PRFSUNIT) has a buffer prefix offset value of 1.

After unsuccessful completion of FHPBUILD processing:

1. The scan pointer's location is unpredictable.
2. Register 15 is set to a return code as shown in "Return Codes" at the end of this macro description.

Supported Resources and General Requirements: ALL.

Valid Subgroups: Inheader.

NAME	OPERATION	OPERANDS
[symbol]	FHPBUILD	[BLDOAF={NO }] {YES}

```
BLDOAF={NO }
       {YES}
```

Function: Indicates if the TCAM origin address field (TOAF) in the FHP should be initialized or re-initialized with the TCAM address of the message originator.

Format: NO or YES.

Default: BLDOAF=NO.

Note: If BLDOAF=YES is specified, the TCSEARCH macro with ARG=CURRTERM specified must be issued before this macro is issued.

If BLDOAF=YES is specified, the TCAM origin address field (TOAF) in the FHP will be set to binary zeros if one of the following conditions is encountered:

- The INTRO macro did not specify a node identifier for this host node.
- No valid resource identifier was assigned to the message origin via a RESOURCE option field.
- RETURNQ option field associated with the message origin does not contain a valid terminal-table entry name.

A return code of X'00000024' is set in register 15 if any of these conditions is encountered.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Return Codes

One of the following return codes is set in register 15:

Code	Meaning
X'00000000'	Successful execution.
X'00000004'	There is no data in the current message buffer.
X'00000008'	The scan pointer points to a location within the data in the message buffer but outside the first buffer unit.
X'0000000C'	The message buffer is not the first buffer of a message, or is a duplicate header.
X'00000010'	No message origin is indicated in the buffer prefix.
X'00000014'	In the GROUP or PCB macro, no reserved characters have been specified for the message buffer.
X'00000018'	In the GROUP or PCB macro, not enough reserved characters have been specified for the message buffer.
X'0000001C'	Using the macro to update an already-present FHP in an extended networking environment, the start-of-header pointer (the FHPHEADP field in the FHP) points to a location before the end of a minimum-size FHP.
X'00000020'	The message origin does not have a TCSOPTS option field initialized for it.
X'00000024'	BLDOAF=YES was specified, and one of the following conditions was encountered; the TOAF in the FHP was set to binary zeros as a result: • The INTRO macro did not specify a node identifier for this host node. • No valid resource identifier was assigned to the message origin via a RESOURCE option field. • The RETURNQ option field associated with this message origin does not contain a valid terminal-table entry name.
X'FFFFFFFC'	The scan pointer points to a location outside the buffer.

FHPTEST Macro

The FHPTEST macro:

- Tests for the presence of a fixed header prefix (FHP)
- Sets a bit to indicate the presence or absence of an FHP.

The FHPTEST macro generates in-line assembler language instructions to test for the presence of an FHP in the message being processed. It also is used to set or clear the flag PRFFHP to indicate the presence or absence of an FHP.

Supported Resources and General Requirements: ALL.

Valid Subgroups: Inheader, Inbuffer, Inblock, Inmessage, Outheader, Outbuffer, Outmessage.

NAME	OPERATION	OPERANDS

[symbol]	FHPTEST	ABSENT=name
		,PRESENT=name
		[,ACTION={TEST }]
		{SETYES}
		{SETNO }
		[,PRFSTAT={20(1)}]
		{name }

```
ABSENT=name
```

Function: Specifies the address of the next instruction to be executed.

Format: *name* must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. The ABSENT= operand is required if ACTION=TEST is specified or allowed to default.

Note: If ABSENT=name is not specified when ACTION=TEST is specified or allowed to default, an assembler error occurs. *name* is the address of the next instruction to be executed when ACTION=TEST is coded or allowed to default and the PRFFHP bit is off.

The ABSENT= operand is applicable only when used with the ACTION=TEST specification.

```
ACTION={TEST  }
       {SETYES}
       {SETNO }
```

Function: Specifies the function to be performed.

Format: TEST, SETYES, or SETNO.

Default: ACTION=TEST.

Note: TEST specifies that a test is to be made against the PRFFHP flag to determine the presence or absence of an FHP. See the PRESENT= and ABSENT= operand descriptions for more detail.

SETYES causes the PRFFHP flag to be set to indicate the presence of an FHP. SETNO causes the PRFFHP flag to be cleared to indicate the absence of an FHP.

The PRFSTAT= operand provides the address of the byte that contains the PRFFHP flag.

```
PRESENT=name
```

Function: Specifies the address of the next instruction to be executed.

Format: *name* must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. The PRESENT= operand is required if ACTION=TEST is specified or allowed to default.

Note: If PRESENT=name is not specified when ACTION=TEST is specified or allowed to default, an assembler error occurs. *name* is the address of the next instruction to be executed when ACTION=TEST is coded, or allowed to default, and the PRFFHP bit is on.

The PRESENT= operand is applicable only when used with the ACTION=TEST specification.

```
PRFSTAT={20(1)}
        {name }
```

Function: Specifies the address of the status byte that contains the PRFFHP flag.

Format: 20(1) or *name*. *name* must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: 20(1)

Note: If PRFSTAT= is allowed to default, you must ensure that register 1 points to the buffer unit.

FHPTEST

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Return Codes

None.

FORWARD Macro

The FORWARD macro:

- Queues messages for one or more specified destinations
- Sets bit 22 in the message error record when a specified number of queued messages is exceeded for a destination using main-storage-only queuing. Bit 22 is also set when the reusable disk queue is full for destinations using reusable disk queues or the core queue is full for destinations using main-storage-only queuing.

FORWARD allows scanning of the destination code field in the header of each incoming message and compares the field with the names of the terminal table entries. If this option is used, the message must be in EBCDIC code prior to the execution of FORWARD. If the destination code is valid (a matching entry is found in the terminal name table), FORWARD queues the message for the specific destination or destinations. If an invalid destination code (that is, one not appearing in the terminal-name table) is detected, control passes to the user routine specified by the EXIT= operand of FORWARD. If no user exit is specified, the message is queued for the external LU or application program specified by the DLQ= operand of the INTRO macro. If no external LU or application program is specified by DLQ= and no user exit is provided, messages with invalid destination codes are overlaid and lost.

Messages may be routed to one or more destinations in the following ways:

1. To the single destination specified in the message header or named by an operand of the FORWARD macro.
2. To the distribution list specified in the message header or named by an operand of the FORWARD macro.
3. To the cascade list specified in the message header or named by an operand of the FORWARD macro.
4. To the multiple destinations specified in the message header. The destination codes may be of equal length or of varying lengths. If there are any multiple destinations, an operand specifies the end-of-address character or characters included after the last destination code in the header of each incoming message.
5. To the group entry in the terminal table specified in the message header or in an operand of the FORWARD macro.
6. To a TPROCESS entry in the terminal table if that entry has been identified as an LU.

If multiple destinations are specified in the message header or if a distribution list is specified, the incoming group processes the message. Then copies are made and routed to the destination queue for each destination specified in the header or distribution list.

When multiple destinations are specified, only the first is checked for validity during execution of the FORWARD macro. Secondary destinations are checked for validity during execution of the inmessage subgroup, after the entire message has been received. Of the secondary destinations, a message for the first invalid destination is routed to the external LU or application program specified by the DLQ= operand of the INTRO macro. If a dead-letter queue is not specified, the message is lost. Any other invalid destinations are ignored. Care should be taken when specifying multiple destinations, whether in the message header or in a distribution list. If any one of the destinations has disk queuing, it should be specified ahead of all destinations with main-storage queuing. If the first (or primary) destination has main-storage queuing, the message may be serviced and freed from the queue before it can be copied to all secondary destinations. This could cause an ABEND (045-2). Having disk queuing for the primary destination ensures that the message will be available for a longer period of time, even after it is marked serviced. If message priorities are enforced by a

PRIORITY macro coded before FORWARD, you must code the PRISAVE= operand of PRIORITY to ensure that message priority is maintained for all destinations in a multiple-destination message and for all destinations that are part of a distribution or cascade list.

If no FORWARD is issued, if the TPROCESS entry has been identified as a host LU (that is, LU=YES has been coded on the TPROCESS macro), and if a session has been established using the TPROCESS name, then the message is routed to the destination queue associated with the TPROCESS. A FORWARD macro must be issued and executed if CANCELMG is included in the inmessage subgroup.

If DEST=*(count)* or DEST=** is specified, the CODE macro must be executed before FORWARD unless the line code is EBCDIC.

The message header of the message must contain the write characters X'F5C3' for FORWARD to send a message.

Note: Care must be taken in entering a character string in a destination field to ensure that it matches a terminal-table entry. A character string entered in lowercase characters from some stations, for example, will not match a terminal-table entry name that is in uppercase characters.

Supported Resources and General Requirements: ALL.

Valid subgroup: Inheader.

NAME	OPERATION	OPERANDS

[symbol]	FORWARD	[DEST={** }]
		{destname }
		{opfield }
		{(count) }
		{PUT }
		{ORIGIN }
		{REG(regname)}
		[,EOA=characters]
		[,EXIT=exitname]
		[,THRESH=qmaximum]
		[,TTCIN={reg) }]
		{fieldname}

```
DEST={**          }
     {destname    }
     {opfield     }
     {(count)     }
     {PUT         }
     {ORIGIN      }
     {REG(regname)}
```

Function: Specifies the destination for the message.

Format: destname, opfield, (*count*), PUT, **, ORIGIN, or REG(*number*). *destname* and *opfield* must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")
destname must be specified with framing C'', CLn'', X'', or XLn'' characters. *opfield* must not be specified with framing characters. *(count)* is a decimal integer from 1 to 8. REG*(regname)* may use any register (2-12).

Default: DEST=**.

Note: *destname* specifies the name of a single, distribution, cascade, or process entry in the terminal table.

opfield specifies the name of a 1- to 8- byte option field defined by an OPTION macro containing the name of an entry in the terminal table. If the destination name is shorter than the length of the option field, the name must be padded to the right with blanks to fill the field. *(count)* specifies the number of characters in each of a list of one or more destinations. If *(count)* is coded, the destination names in the message header must all be the same length. Delimiting and embedded blanks are ignored. If this operand is specified and there is more than one destination, the EOA operand must also be specified.

PUT specifies that the destinations of messages entered by an application program are placed by the user in an application-program work area.

** specifies that there are one or more destination names of variable length in the message header. If this parameter is specified and there is more than one destination, the EOA operand must be specified. If this parameter is specified, the scan pointer must be at the byte preceding the destination.

ORIGIN specifies that the message is to be returned to the origin of the message.

REG(*regname*) indicates the register containing the address of a work area that contains the name of the destination in the following format:

Byte 0 =	Length of the destination name in hexadecimal.
Byte 1-n =	Destination name specified with framing C″, CLn″, X″, or XLn″ characters.

If an invalid destination is specified, control passes to the user routine specified by the EXIT operand. If no user exit is specified, the message is queued for the destination specified by the DLQ operand of the INTRO macro. If no destination is specified by the DLQ operand and no user exit is provided, messages with invalid destination codes are overlaid and lost.

DEST=PUT should be specified in the inheader subgroups of the MH assigned to an application program when the MH is to handle messages coming from an application program that has OPTCD=W coded on its output DCB macro, if the user wishes the message to go to the destination specified in the work area. For more information on specifying the destination of a message in the application program, see the discussion of the OPTCD operand of the output DCB macro in *TCAM Application Programming.* Use of this operand is restricted to the case just described.

In the case of multiple destinations, TCAM checks the validity of the primary destination at execution of the FORWARD macro, replacing the last byte of the primary destination with a X′DF′. At inmessage processing, the scan to validate the secondary destinations is begun at this X′DF′, which is replaced with the original character upon completion. This is done to maintain a pointer regardless of any data movement that may occur between FORWARD and INMSG macro processing. Therefore, you should not insert a X'DF' preceding the EOA character in the message.

In case of multiple-buffer headers, a destination must be determined for the first header buffer. This can be ensured in one of two ways, depending upon how the first header and the subgroup are designed:

- If the destination is specified by the DEST operand, the FORWARD macro must occur sufficiently early in the subgroup so that it acts upon the first header buffer.
- If the destinations are specified in the header rather than by the DEST operand, the first destination must be completely contained within the first buffer.

If the second condition is not met, TCAM assumes an invalid destination has been specified and branches to the user exit, if provided. If no user exit is provided or if the first condition is not met, the message is routed to the dead-letter queue, or if no dead-letter queue is provided, the buffer is returned to the buffer pool.

`EOA=characters`

Function: Specifies the character or character string used to delimit the destination field of the header.

Format: One to eight nonblank characters specified in character or hexadecimal format. If character format is specified, the field must be unframed or framed with C'' or CLn'' characters. If hexadecimal format is specified, the field must be framed with X'' or XLn'' characters. *n* must be the actual number of the characters.

Default: None. With the DEST operand coded *destname, opfield,* or PUT, specification is optional. With the DEST operand coded *(count)* or ** and multiple destinations in the message, this operand is required.

Note: If this operand is specified and the DEST operand is coded *destname, opfield,* or PUT, the operand is ignored.

`EXIT=exitname`

Function: Specifies the name of a user-written exit routine that is given control when an invalid destination is detected.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Note: The routine may correct the destination, provide another destination, or indicate that the message is not to be processed for a destination. If an invalid destination is provided by the user-exit routine, the message is forwarded to the dead-letter queue if one is specified by the DLQ operand of the INTRO macro. Otherwise, it is overlaid and lost. In the case of multiple destinations, only the first one can be corrected by this exit routine.

If you provide an exit routine for FORWARD, TCAM automatically saves and restores registers for this routine; the user routine need not save registers and may change the contents of registers 2 through 12. However, the contents of registers 13 and 14 must not be altered. When the user routine receives control, register 1 contains the address of the header buffer. Register 14 contains the return address for the calling routine. Register 15 contains the address of the entry point for the user routine.

TCAM expects the user routine to place one of two items in register 15 before returning control:

- A return code of 0 in register 15 means that the user routine was unable to provide a satisfactory destination for this message. In this case, the message is forwarded to the dead-letter queue or is not processed for any destination if no dead-letter queue is provided.
- Register 15 may contain the main-storage address of a field that you set up, consisting of a length byte followed by the name of an external LU, distribution, cascade, or process entry in the terminal table. (The length byte must contain in binary form the number of bytes in the rest of the field.) TCAM assumes that the specified name is the destination of the message. The field must be padded to the right with blanks to the length of the longest terminal-table entry name (the value of the MAXLEN operand of the TTABLE macro).

The user routine should return control to TCAM by a BR 14 instruction.

This operand is ignored when DEST=PUT is specified.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

`THRESH=qmaximum`

Function: Sets bit 22 in the message error record when the user-specified number of messages has been queued for a destination that uses main-storage-only queuing.

Format: *qmaximum* is an unframed decimal integer or hexadecimal field, from 1 to 65,535, or a 2-byte hexadecimal field. If hexadecimal format is used, framing X'' or XL2'' characters must be specified.

Default: None. Specification is optional.

Note: *qmaximum* is a user-specified number that indicates the maximum number of messages that can be queued at any one time for a destination that uses main-storage-only queuing. This operand causes TCAM to set bit 22 in the message error record for a message forwarded to a message queue that contains the number of messages specified by *qmaximum.*

If the destination is an application program, this operand permits you to delay activation of the application program until a certain number of messages is queued for that destination. This operand can be used also as an aid in controlling the efficient use of system resources when one or more destinations use main-storage-only queuing. For instance, to prevent main storage being tied up with unsent messages, you can use the operand to warn of an excessive number of messages queued for a particular destination and could then issue a conditional macro in your inmessage subgroup such as CANCELMG, REDIRECT, ERRORMSG, or MSGGEN. If there are multiple destinations, the REDIRECT, ERRORMSG, and MSGGEN macros are effective only if the number specified by *qmaximum* is exceeded for the first destination.

After the threshold has been reached, subsequent messages are sent to the exit routine specified by the EXIT operand, if specified; the dead-letter queue, if specified; or returned to the buffer pool. This action prevents a main-storage-only queue from being filled with canceled header buffers. If the message is sent to the buffer pool, the CANCELMG, REDIRECT, and ERRORMSG (if HEADER=YES is coded) macros do not execute. If HEADER=NO is specified on the ERRORMSG macro, ERRORMSG executes.

In order for this function to be provided for a destination, the destination's entry in the terminal table must *not* be included in a cascade entry or a distribution entry (that is, do *not* code a TLIST macro in the terminal table that includes this destination).

```
TTCIN={(reg)     }
      {fieldname}
```

Function: Specifies the TNT index to the terminal entry to which the message is to be routed.

Format: A register number enclosed in parentheses that specifies a register from 2 to 12 containing the TTCIN in the two low-order bytes, or the label of a halfword containing the TTCIN.

Default: None. Specification is optional.

Note: The TTCIN operand is mutually exclusive with the DEST, EOA, and THRESH operands. Invalid indexes can be checked via the EXIT operand of the FORWARD macro. Otherwise, all messages specifying invalid destinations are routed to the dead letter queue with a return code of 0.

Return Codes

One of the following return codes is set in register 15:

Code	Meaning
X'00000000'	Successful execution.
X'00000004'	Invalid destination and no DLQ specified, specified number of queued messages is exceeded, main storage queue is full, or the reusable disk queue is in a FULL condition.
X'00000008'	EOA string is detected on entry from the multiple routing subtask.

GROUP Macro

The GROUP macro:

- Identifies the message handler for the external LUs in this group
- Specifies the number of buffers assigned to external LUs in this group for sending and receiving operations
- Specifies the buffer size for buffers servicing external LUs in this group
- Specifies the translation tables for translating incoming and outgoing messages.

The GROUP macro creates a logical grouping of external LUs so that specifications and manipulations may be made on the entire group at once rather than on each individual external LU. See the GROUP operand of the TERMINAL macro to see how external LUs are assigned to the group named by the GROUP macro.

Supported Resources and General Requirements: SNA.

Valid subgroups: Not applicable.

NAME	OPERATION	OPERANDS

grpname	GROUP	MH=mhname
		[,ALTCD=(table1,table2)]
		[,BUFOUT={integer}]
		{2 }
		[,BUFSIZE=integer]
		[,PACK={YES}]
		{NO }
		[,RESERVE={(integer1,integer2)}]
		{(0,0) }
		[,TRANS={tablename}]
		{EBCD }
		{EBCF }
		{ASC8 }
		{ASCI }

ALTCD=(table1,table2)

Function: Specifies the alternate translate tables to be used for translation performed by the CODE macro. If the BIND image specifies that alternate code set selection is allowed and the Alternate Code Processing Identifier in the BIND is B'0', then *table1* will be used when the alternate code is selected in the RH. If the BIND image specifies that alternate code set selection is allowed and the identifies in the BIND is B'1', then *table2* will be used.

Format: Valid translate table names for *table1* and *table2* are ASCI, ASC8, EBCD, EBCF and user-defined table names which must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: If ALTCD= is not specified, then translation is done by the table specified by TRANS= operand. If TRANS= is not specified, then the default is EBCD.

Note: To implement the SNA architecture, ALTCD=(ASCI,ASC8) should be coded.

```
BUFOUT={integer}
       {2      }
```

Function: Specifies the number of buffers assigned for sending data to a station in the group.

Format: An unframed decimal integer.

Default: BUFOUT=2.

Maximum: 15.

`BUFSIZE=integer`

Function: Specifies the output buffer size (in bytes) for all stations in this group.

Format: An unframed decimal integer greater than or equal to 76.

Default: Specification is optional. If it is not specified, the value assigned to the UNITSZ= operand of the INTRO macro is used.

Maximum: 65,535.

Note: The size specified can be overridden for outgoing messages on an external LU basis, by coding the BUFSIZE= operand on the TERMINAL macro.

`grpname`

Function: Specifies the name of the group.

Format: Must conform to the rules for assembler language symbols.

Default: None. A name must be specified.

`MH=mhname`

Function: Specifies the name of the STARTMH macro associated with this group.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. This operand is required.

```
PACK={YES}
     {NO }
```

Function: Specifies whether input data is to be accumulated prior to being sent to the MH for input processing.

Format: YES or NO.

Default: PACK=NO.

Note: If PACK=YES is coded, input data is accumulated until the value specified in the BUFSIZE operand is reached, an error is encountered, or normal end-of-message occurs.

```
RESERVE={(integer1,integer2)}
        {(0,0)              }
```

Function: integer1 specifies the number of bytes reserved in the first unit of a buffer receiving the first incoming segment of each message entered from the external LU. *integer2* specifies the number of bytes reserved in the first unit of all buffers except the first.

This keyword may also be used to indicate the number of reserved bytes for internally generated TCAM buffers.

Format: Unframed decimal integers that must conform to the rules for assembler language symbols. (See the *symbol* entry in the Glossary.)

Default: RESERVE=(0,0).

Maximum: 255 for each integer.

Note: integer1 must be less than the UNITSZ= value minus 37. *integer2* must be less than the UNITSZ= value minus 30. The UNITSZ= value is the value coded on the UNITSZ= operand (or KEYLEN=) of the INTRO macro.

```
TRANS={tablename}
      {EBCD     }
      {EBCF     }
      {ASC8     }
      {ASCI     }
```

Function: Specifies the translation table for this group.

Format: EBCD, EBCF, ASCI, ASC8 or *tablename*. *tablename* is the name of a user-defined table and must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: TRANS=EBCD.

Note: EBCD specifies translation to EBCDIC. EBCF specifies translation to folded EBCDIC, which translates lowercase input to uppercase EBCDIC. ASC8 specifies translation to ASCII-8. ASCI specifies translation to ASCII 7-bit code.

Return Codes

None.

HOLD Macro

The HOLD macro:

- Suspends transmission to a destination with either disk queuing or main-storage queuing with disk backup
- Allows an LU-LU session to be retained for an external LU in the case where TCAM would normally terminate the session because the LUs destination queue is held.
- Specifies whether the next message on the queue is to be held
- Allows recovery from permanent errors when coded in outmessage subgroups
- Specifies whether the queue is being held by the TCPRGHLD flag in the message handler for pacing purposes (by setting the TCPRGHLD flag in the TCSOPTS option field).

HOLD suspends transmission of output messages to a destination:

- For a time interval
- Until the messages are released by a Release Intercepted Resource basic operator command
- Until an MRELEASE macro is issued in an application program
- Until an IEDRELS macro is executed in the MH.

HOLD may be requested unconditionally by specifying an error mask of 0 or by omitting the mask or conditionally in which case the error mask specified in the first operand is compared to the message error record assigned to the message. If specified errors are detected, transmission is suspended. A destination that cannot accept messages because of the effect of a HOLD macro is said to be *intercepted*. For a discussion of holding, see the chapter of the *TCAM Installation Guide* titled "Coding the Message Handler."

If you want to hold a message using the Intercept a Resource basic operator command, you must code at least one HOLD macro in the message handlers.

An inquiry/reply function is provided by the HOLD and MRELEASE macro combination. (See the *TCAM Installation Guide*. The HOLD macro, when issued in the inheader or inmessage subgroup of a message handler, suspends transmission of outgoing messages to the destination entering the message until an MRELEASE macro in an application program releases the destination.

By using the macros in combination with TCAM's message-priority capability, you can ensure that the next message received by the intercepted station after it enters an inquiry is the reply from an application program to that inquiry.

Note: An external LU or application whose queue is located in main storage with no disk backup may not be intercepted; the HOLD macro is ignored in this case. If the HOLD is in the inheader or outheader subgroup, a return code of 4 will be set in register 15.

When HOLD is executed in an outmessage subgroup, a hold-current function is performed: the current message is held after transmission. When the queue is released, this message is retransmitted. When HOLD executes an inheader, inmessage, or outheader subgroup and the message causing execution is routed to another destination, a hold-next function is performed: the hold goes into effect with the message following the one causing suspension of transmission; therefore, when the queue is released, no retransmitting occurs.

If the HOLD macro is executed in the outmessage subgroup for a lock response, the lock is not broken, the destination is not intercepted, and the message is retransmitted immediately - that is, it is sent twice. This can result in an infinite loop if the condition for the HOLD is permanent and the destination is inoperative. If a destination is intercepted by an operator command while in lock mode, or if lock is initiated while the destination is intercepted, all lock responses will be sent as if the destination were not intercepted. No other messages are sent, however, until the destination is released.

Supported Resources and General Requirements: SNA.

Valid subgroups: Inheader, inmessage, outheader, and outmessage.

NAME	OPERATION	OPERANDS
[symbol]	HOLD	[mask] [,RELEASE] [,CONNECT={OR }] {AND} [,INTVL={seconds}] {opfield} [,NEXT={NO }] {YES} [,SESSION=KEEP] [,SYNCHLD={NO }] {YES}

```
CONNECT={OR }
        {AND}
```

Function: Specifies the type of logical connection to be be made between the mask and the message error record.

Format: OR and AND.

Default: CONNECT=OR.

Note: OR specifies that the macro is to be executed if *any* bit specified by *mask* is on in the message error record. AND specifies that the macro is to be executed only if *all* of the bits specified by *mask* are on in the message error record.

The CONNECT operand is only valid if specified in an inmessage or outmessage subgroup.

```
INTVL={seconds}
      {opfield}
```

Function: Specifies the number of seconds that transmission to the destination is to be suspended.

Format: seconds is an unframed positive integer, either decimal or hexadecimal. If hexadecimal format is used, framing X'' or XLn'' characters must be specified. *opfield,* if specified, must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary") and must be the name of a 2-byte option field defined by an OPTION macro for the destination to be intercepted. The maximum value that can be specified is 65,535 or a 2-byte hexadecimal field.

Default: None. Specification is optional.

Note: The TCAM checkpoint/restart function permits restart of a TCAM system after system closedown or failure. If the system fails or is closed down while the destination is intercepted, when the system is restarted by a point-of-last-environment (POLE) or point-of-failure (POF) restart, the interception will still be in effect, but the INTVL operand will no longer apply; transmission will be suspended until a Release Intercepted Resource basic operator command, an IEDRELS macro, or an MRELEASE application program macro causes transmission to be resumed.

This operand is valid only if specified in outheader or outmessage subgroup and may not be specified in an inheader or inmessage subgroup.

The *opfield* parameter is valid only in an outheader subgroup.

If INTVL=0 is coded or if the option field contains 0 when the HOLD macro is executed, TCAM defaults the INTVL operand to one second.

If both the RELEASE and INTVL operands are coded, RELEASE overrides the INTVL operand.

The INTVL operand must not be coded if the HOLD macro is to be executed in an inheader or inmessage subgroup.

```
mask
```

Function: Specifies the 5-byte bit configuration used to test the message error record for the message. (The message error record is described in Appendix A.)

Format: ***mask*** is an unframed decimal integer or hexadecimal field. If hexadecimal format is used, framing characters must be specified. If X'' is used, leading zeros must be coded. If XL5'' is used, leading zeros may be omitted.

Default: None. Specification is optional.

Note: Omission of the operand or an all-zero mask causes unconditional execution.

The ***mask*** operand is only valid if it is specified in an inmessage or outmessage subgroup.

```
NEXT={NO }
     {YES}
```

Function: Specifies whether the next message on the queue is to be held.

Format: NO or YES.

Default: NEXT=NO.

Note: YES specifies that the next message on the queue is to be held when it is processed by the outgoing group of this device message handler. NO specifies that the message currently being processed by the outgoing group is to be held and retransmitted when the HOLD condition is no longer in effect.

This operand has meaning in the outmessage subgroup only.

```
RELEASE
```

Function: Specifies that transmission to the destination is to be suspended until either a Release Intercepted Resource basic operator command or IEDRELS macro in the message handler is issued for the destination or until an MRELEASE macro is issued for the destination in an application program.

Format: RELEASE.

Default: None. Specification is optional.

```
SESSION=KEEP
```

Function: Specifies that TCAM should not request session termination for a TCAM-initiated LU-LU session in the case where the destination queue is held and TCMSESN = NORMAL was specified on the TERMINAL macro that defines the LU.

Format: KEEP

Default: None. Specification is optional.

Note: if SESSION = KEEP is not specified, the session will be terminated.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

```
SYNCHLD={NO }
        {YES}
```

Function: Specifies whether the TCPRGHLD flag bit in the current destination's TCSOPTS option field is to be turned on. YES specifies that the flag is to be turned on.

Format: YES or NO.

Default: SYNCHLD = NO.

Note: If SYNCHLD = YES is specified, the TCSOPTS option field should be defined for the destination.

TCPRGHLD can be turned off by issuing an IEDRELS macro with SYNCHLD = YES in an inheader, inbuffer, or inblock subgroup, by user code, or by the extended operator commands START and STOP.

Suggested use: To specify that a destination is being intercepted because the destination is temporarily busy and that the intercepted queue for the destination should be released when a positive signal is received from the destination to indicate that the intercepted messages can be received (for example, receipt of an LUSTAT DFC command from an LU).

Return Codes

If specified in an inheader or outheader subgroup, one of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	Successful execution.
X'00000001'	Option field specified by INTVL operand could not be found; interval is set to one second.
X'00000004'	Destination queue is located in main storage with no disk backup.
X'00000008'	Destination is already held.
X'0000000C'	Destination cannot be held because it is a process or list entry.
X'00000014'	SYNCHLD = YES is specified and the TCSOPTS option field is not defined for the destination, or destination is held, or TCSOPTS was found but TCPRGHLD is not turned on for the destination.
X'0000001C'	No buffers are available; auto release is not performed.

HOSTDEF Macro

The HOSTDEF macro:

- Is issued in defining an extended network.
- Generates a GROUP macro.

The HOSTDEF and TCPATH macros together define resources in other TCAM nodes that may participate in the extended networking environment. The extended networking system makes use of TCAM extended networking functions to ease MCP definition and increase data integrity and security. These functions are described in the *TCAM Installation Guide.*

Supported Resources and General Requirements: SNA.

Valid subgroups: Not applicable.

NAME	OPERATION	OPERANDS

[symbol]	HOSTDEF	NODEID=integer
		[,BUFOUT={2 }]
		{integer}
		[,BUFSIZE=integer]

```
BUFOUT={2       }
       {integer}
```

Function: Specifies the number of buffers assigned to sending data to the TCAM node associated with this HOSTDEF macro.

Format: *integer* is an unframed decimal integer from 2 to 15.

Default: BUFOUT=2.

```
BUFSIZE=integer
```

Function: Specifies the output buffer size (in bytes) for the TCAM host node associated with this HOSTDEF macro.

Format: *integer* is an unframed decimal integer from 52 to 65,535.

Default: None. Specification is optional.

Maximum: 65,535.

```
NODEID=integer
```

Function: Specifies the node identifier of another TCAM node associated with this HOSTDEF macro.

Format: *integer* must be an unframed decimal integer from 1 to 245.

Default: None. Specification is required.

HOSTDEF

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Return Codes

None.

IEDBUFTR Macro

The IEDBUFTR macro:

- Traces buffer contents into a trace table within TCAM's address space
- Specifies the first four characters of the trace entry
- Specifies the name of an option field, the contents of which may be used to execute the macro conditionally.

Supported Resources and General Requirements: ALL.

Valid Subgroups: Inbuffer, inheader, outbuffer, and outheader.

NAME	OPERATION	OPERANDS
[symbol]	IEDBUFTR	[ID={<u>USER</u> }] {chars} [,TRACESW=opfield]

```
ID={USER }
   {chars}
```

Function: Specifies the first four characters of the trace entry.

Format: USER or *chars*. *chars* is 4 bytes of unframed character data.

Default: ID=USER.

Note: The ID operand can specify any 4-character combination marking the start of the trace entry.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

`TRACESW=opfield`

Function: Specifies the name of a 1-byte option field that, if found and if set to a value of X'01', causes this buffer to be traced.

Format: *opfield* must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary") and must be the name of an option field defined by the OPTION macro.

Default: None. Specification is optional.

Note: If the TRACESW operand is not specified, a trace entry is made for each buffer for which this macro is executed.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	Full buffer trace occurred.
X'00000004'	TRACESW is off; no trace occurred.
X'00000008'	TRACESW not coded for terminal table entry; no trace occurred.
X'0000000C'	No TTCIN value in LCB; no trace occurred.
X'00000010'	Partial trace (one unit) occurred.
X'0000000F'	Trace table not specified on INTRO; no trace occurred.

IEDDFC Macro

The IEDDFC macro:

- Is used to send the SNA data flow control (DFC) commands BID, BIS, CANCEL, CHASE, LUSTAT, QC, QEC, RELQ, RSHUTD, RTR, SBI, SHUTC, SHUTD, or SIGNAL prior to or after processing current message
- May be executed more than once per subgroup

The IEDDFC macro allows the MH to receive verification that the destination LU is ready to receive a message before it is transmitted. This saves the time that otherwise would be used in attempting to send a message to an LU that cannot receive it. If the IEDDFC macro is executed more than once in a subgroup, subsequent executions repeat the same operation as if no prior execution had occurred.

Note: TCAM enforces DFC protocols for the particular session involved. If an IEDDFC request is coded for a session type that does not allow that session type, TCAM will generate a negative response for that request.

Supported Resources and General Requirements: SNA.

Valid subgroups: Outheader (BID), inmessage and outmessage.

NAME	OPERATION	OPERANDS
[symbol]	IEDDFC	{BID } {RELQ } {BIS } {RSHUTD} {CANCEL} {RTR } {CHASE } {SBI } {LUSTAT} {SHUTC } {QC } {SHUTD } {QEC } {SIGNAL} [,conchars] [,BLANK={YES } {NO } {chars}] [,REQCODE=XL4'chars']

```
{BID    }          {RELQ  }
{BIS    }          {RSHUTD}
{CANCEL}           {RTR   }
{CHASE }           {SBI   }
{LUSTAT}           {SHUTC }
{QC     }          {SHUTD }
{QEC    }          {SIGNAL}
```

Function: Specifies which DFC command to send before or after processing the current message in the MH; the current message causes the execution of IEDDFC.

Format: BID, BIS, CANCEL, CHASE, LUSTAT, QC, QEC, RELQ, RSHUTD, RTR, SBI, SHUTC, SHUTD, or SIGNAL.

Default: None. This operand is required.

Notes:

1. *The BID, BIS (Bracket Initiation Stopped), CANCEL, CHASE, LUSTAT (Logical Unit Status), QC (Quiesce Complete), QEC (Quiesce at End of Chain), RELQ (Release Quiesce), RSHUTD (Request Shutdown), RTR (Ready to Receive), SBI (Stop Bracket Initiation), SHUTC (Shutdown Complete), SHUTD (Shutdown), and SIGNAL commands are data flow control requests sent by TCAM to an LU. These commands are discussed in the chapter titled "System Network Architecture and TCAM" in the TCAM Installation Guide.*

2. *BID may be coded in the outheader subgroup only. BIS, CANCEL, CHASE, LUSTAT, QC, QEC, RELQ, RSHUTD, RTR, SBI, SHUTC, SHUTD, and SIGNAL may be coded in the inmessage or outmessage group; the conchars and BLANK operands may only be coded with the BID operand in the outheader subgroup.*

3. *After issuing the IEDDFC macro in the inmessage or outmessage subgroup, the message error record is modified according to the sense data returned on the response to the command. This is done only if the MER doesn't already contain a non-zero value.*

4. *If, after issuing SHUTD, a SHUTC is received, TCAM will terminate the session.*

```
BLANK={YES }
      {NO  }
      {char}
```

Function: Specifies whether EBCDIC blank characters are to be ignored when encountered in the character string in the message header being compared to the string specified by the *conchars* operand, or whether blanks are to be counted as part of the header string, this operand also specifies whether some other hexadecimal character is to be ignored when encountered in the header string.

Format: YES, NO, or *char*. *char* is a single character that may be specified in either character or hexadecimal format. If character format is specified, it may be unframed or framed with C'' or CL1'' characters. If hexadecimal format is specified, it must be framed with X'' or XL1'' characters.

Default: BLANK=YES.

Note: YES specifies that the EBCDIC blank character (X'40') is to be ignored by this macro whenever it is encountered in the header character string being checked against the control character string specified by the *conchars* operand. For example, if BLANK=YES and an 8-byte field in the header is being checked by this macro, a blank appearing in the fifth byte of the field is ignored, and the sixth through ninth bytes are considered to be the last four bytes of the field, assuming that no blanks are coded in the sixth through ninth bytes.

NO specifies that the EBCDIC blank character is to be treated in the same way as any other character when it is encountered by this macro as any other character when it is encountered by this macro in the header string being compared to the string specified by *conchars.*

char specifies that the single character replacing *char* is to be ignored by this macro whenever it is encountered in the header string being compared to the string specified by the *conchars* operand. (That is, the macro automatically skips over the character without performing a comparison and checks the next character in the header.) If BLANK=*char* is not specified, the EBCDIC blank is not ignored by this macro when it is encountered in the header string, but is compared to the character in the corresponding space in the *conchars* string, in the same was as any other character.

This operand is meaningless unless the *conchars* operand is also specified and may be used only with BID in the outheader subgroup.

`conchars`

Function: Specifies the character or character string that, if found in the header buffer as the next nonblank field, causes execution of the function.

Format: conchars is one to eight nonblank characters in character or hexadecimal format. If character format is used, the string may be unframed or framed with C'' or CLn'' characters. If hexadecimal format is used, the string must be framed with X'' or XLn'' characters.

Default: None. Specification is optional.

Note: If this operand is omitted, the IEDDFC function is performed unconditionally. If the next field in the header does not match this operand, the function is not performed.

conchars may be used only with the BID operand in the outheader subgroup.

`REQCODE=XL4'chars'`

Function: Specifies the LU status value and status extension field or the signal code and extension field.

Format: chars is 4 bytes of hexadecimal data framed with XL4'' characters.

Default: None. This operand must be specified with the LUSTAT and SIGNAL operands. It is invalid with all other operands.

Note: The REQCODE operand is used with the LUSTAT operand to give the status value and status extension field and with the SIGNAL operand to give the signal code and signal extension field.

The *SNA Reference Summary* gives the format and possible values for these fields.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	Successful execution.
X'00000004'	Not a logical unit.
X'00000008'	This session is not using brackets or bracket state manager is not loaded.
X'0000000C'	BID invalid with FM type.

IEDGTBND Macro

The IEDGTBND macro:

- Provides access to bind parameter data in the option table
- Specifies the register that contains the option-field address.

The IEDGTBND macro returns to a user-specified register the address of the desired saved session parameters in the IEDBNDPM option field, which is reserved if user-written code the MH needs access to the logon parameter data. If the option subfield IEDBNDSZ has been specified, the address of the proper subfield for a particular session is returned. If IEDBNDSZ is not specified or has a value of 0, the IEDBNDPM option field is not subdivided and is considered to hold only one set of session parameter data.

IEDBNDSZ and IEDBNDPM must be valid option field names, though not necessarily defined for the target terminal-table entry.

See the chapter of this manual titled "Option Fields Reserved for Use by TCAM Functions" for more information on the IEDBNDPM and IEDBNDSZ option fields.

See the chapter "Defining User-Replaceable Tables and Modules" of the *VTAM Planning and Installation Reference* for more information about the logon mode table and session parameters.

Supported Resources and General Requirements: SNA.

Valid subgroups: Inblock, inbuffer, inheader, outbuffer, and outheader.

NAME	OPERATION	OPERANDS

[symbol]	IEDGTBND	{(15)}
		{(reg)}

```
{(15) }
{(reg)}
```

Function: Specifies the register into which the address of the desired session parameters are to be placed.

Format: A decimal register number, 2 through 11 or 15, enclosed in parentheses.

Default: (15).

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
Option field address	If register 15 is specified, register 15 contains the address of IEDBNDPM or IEDBNDSZ (if available).
X'00000000'	1. If a register other than 15 is specified, register 15 contains X'00000000' and the specified register contains the option field address. 2. If register 15 is specified and an error occurs, register 15 contains X'00000000'.
X'00000004'	If register 15 is not specified and an error occurs, register 15 contains X'00000004' and the specified register contains X'000000FF'.

IEDHALT Macro

The IEDHALT macro:

- Causes termination of an LU-LU session
- May be conditionally executed.

When executed in an inheader subgroup, the IEDHALT macro causes an LU-LU session to be halted conditionally when a character string coded in the macro is found in a message or when a specified option field contains a character string that matches a character string in a message.

When coded in an inmessage or outmessage subgroup, the IEDHALT macro causes an LU-LU session to be conditionally halted when the mask specified in the macro is tested against the message error record.

Supported Resources and General Requirements: SNA.

Valid subgroups: Inheader, inmessage, and outmessage.

When Used in an Inheader Subgroup

NAME	OPERATION	OPERANDS

[symbol]	IEDHALT	`[CHARS={conchars}]`
		`       {opfield }`

When Used in an Inmessage or Outmessage Subgroup

NAME	OPERATION	OPERANDS
[symbol]	IEDHALT	[mask] [,CONNECT={AND}] {<u>OR</u> }

```
CHARS={conchars}
      {opfield }
```

Function: Specifies the character string or the option field containing the character string that, if found, causes execution of the IEDHALT function.

Format: Character or hexadecimal. If hexadecimal format is used, the string must be framed with X″ or XLn″ characters. If character format is used, the string must be framed with C″ or CLn″ characters. If framing characters are used, the field is treated as *conchars*. Without framing characters, it is treated as *opfield*. EBCDIC blank (X′40′) is skipped.

Default: None. Specification is optional. If this operand is omitted, the IEDHALT function is performed unconditionally.

Note: The *conchars* operand specifies the character or hexadecimal representation of the session termination sequence.

The *opfield* operand specifies the name of an option field containing the character string that, if found, causes execution of the IEDHALT function.

If the CHARS= operand is not coded, a return code is not set in register 15.

```
CONNECT={AND}
        {OR }
```

Function: Specifies the type of logical connection to be made between the mask and the message error record.

Format: AND or OR.

Default: CONNECT=OR.

Note: AND specifies that the macro is to be executed only if *all* bits specified by the mask are on in the message error record.

OR specifies that the macro is to be executed if *any* of the specified bits are on in the message error message.

```
mask
```

Function: Specifies the 1- to 40-bit configuration to test the message error record for the message (see "Appendix A. Message Error Record").

Format: Unframed decimal integer or hexadecimal field. If hexadecimal format is used, framing characters must be specified. If X'' is used, leading zeros must be coded. If XL5'' is used, leading zeros may be omitted.

Default: None. Specification is optional.

Note: Omitting the operand or specifying an all-zero mask causes unconditional execution.

Default: None. Specification is optional.

Return Codes

One of the following codes is returned in register 15:

Code	Meaning
X'00000000'	Successful execution.
X'00000004'	CHARS=characters not found, option field not found, or not a valid LU-LU session.

IEDLOGON Macro

The IEDLOGON macro:

- Tests for a logon data buffer.

This macro detects a message constructed by TCAM to serve as the first data message for a new LU-LU session to flow in the DMH associated with the host LU. This message is constructed only if logon data has been specified in the CINIT received by the LOGON exit, or if logon data has been specified by the user bind exit routine. The user bind exit routine may specify logon data for any session, independent of who initiated the session and independent of whether the host LU is primary or secondary. For more information, see the description of the user bind exit routine in the *TCAM Installation Guide.*

Supported Resources and General Requirements: SNA.

Valid subgroups: Inheader.

NAME	OPERATION	OPERANDS
[symbol]	IEDLOGON	(no operands)

symbol

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Return Codes

Code	**Meaning**
X'00000000'	This is a logon buffer.
X'00000004'	This is not a logon buffer.
X'00000008'	This is an invalid external LU type.

IEDLSCR Macro

The IEDLSCR macro:

- Provides compatibility between the VTAM logon mode table and the terminal-table entry for display screen size specification
- Is valid only in your bind exit routine for external LUs.

The IEDLSCR macro is used to provide compatibility between the logon mode specification and the TERMINAL macro definition for large-screen support. It specifies whether the large-screen information is to be moved from the terminal-table entry to the session parameters or from the session parameters to the terminal-table entry.

Supported Resources and General Requirements: SNA.

Valid subgroup: Not applicable.

NAME	OPERATION	OPERANDS
[symbol]	IEDLSCR	[TARGET={BI }] {TTE}

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

```
TARGET={BI }
       {TTE}
```

Function: Specifies whether the 5 bytes of large-screen information is to be moved from the terminal-table entry to the session parameters or from the session parameters to the terminal-table entry.

Format: BI or TTE.

Default: TARGET=BI.

Note: TARGET=TTE specifies that the large-screen information from the session parameters being sent is used to update the external LU's terminal-table entry large-screen information.

TARGET=BI specifies that the large-screen information currently in the external LU's terminal-table entry is to be used to define the large-screen portion of the session parameters used to establish the session.

Note: When TARGET=BI is specified, the input session parameters to the bind exit routine are modified. If the bind exit routine copies the session parameters into local storage to make modifications, the IEDLSCR macro should be issued prior to copying the parameters.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	Successful execution.
X'00000004'	TTE specified by destination TCAM address is not valid for large-screen support.
X'00000008'	Logon mode contains invalid specification.

IEDNCRPT Macro

The IEDNCRPT macro:

- Specifies selective encryption of output data being processed by the outgoing group of an MH
- Requires the presence of the IBM Programmed Cryptographic Facility program product.

Supported Resources and General Requirements: SNA.

Valid subgroups: Outheader and outbuffer.

When executed in the outheader subgroup, IEDNCRPT causes the entire message (SNA chain) to be encrypted; when executed in the outbuffer subgroup, IEDNCRPT causes the current buffer contents (SNA RU) to be encrypted.

When Used In An Outheader Subgroup

NAME	OPERATION	OPERANDS

[symbol]	IEDNCRPT	[conchars]
		[,BLANK={<u>YES</u> }]
		{NO }
		{chars}

When Used In An Outbuffer Subgroup

NAME	OPERATION	OPERANDS
[symbol]	IEDNCRPT	(no operands)

```
BLANK={YES  }
      {NO   }
      {chars}
```

Function: Specifies whether EBCDIC blank characters are to be ignored when encountered in the character string in the message header being compared to the string specified by the *conchars* operand or whether blanks are to be part of the header string when encountered in it. If EBCDIC blanks are to be counted as part of the header string, this operand also specifies whether some other hexadecimal character is to be ignored when encountered in the header string.

Format: YES, NO or *char*. *char* is a single character that may be specified in either character or hexadecimal format. If character format is specified, it may be unframed or framed with C'' or CL1'' characters. If hexadecimal format is specified, it must be framed with X'' or XL1'' characters.

Default: BLANK=YES.

Note: YES specifies that the EBCDIC blank character (X'40') is to be ignored by this macro whenever it is encountered in the header character

string being checked against the control character string specified by the *conchars* operand. For example, if BLANK=YES and an eight-byte field in the header is being checked by this macro, a blank appearing in the fifth byte of the field will be ignored and the sixth through ninth bytes will be considered to be the last four bytes of the field, assuming that no blanks are coded in the sixth through ninth bytes.

NO specifies that the EBCDIC blank character is to be treated in the same way as any other character when it is encountered by this macro in the header string being compared to the string specified by *conchars*.

char specifies that the single character replacing *char* is to be ignored by this macro whenever it is encountered in the header string being compared to the string specified by the *conchars* operand. That is, the macro automatically skips over the character without performing a comparison and goes on to check the next character in the header. If BLANK=char is specified and *char* is not the EBCDIC blank character (X'40'), the EBCDIC blank is not ignored by this macro when it is encountered in the header string, but is compared to the character in the corresponding space in the *conchars* string, in the same way as any other character.

This operand is meaningless unless *conchars* is also specified.

`conchars`

Function: Specifies the character or character string that, if found in the header as the next nonblank field, causes execution of the IEDNCRPT macro function.

Format: One to eight nonblank characters in character or hexadecimal format. If character format is used, the string may be unframed or framed with C'' or CLn'' characters. If hexadecimal format is used, the string must be framed with X'' or XLn'' characters.

Default: Specification is optional.

Note: If this operand is omitted and IEDNCRPT occurs in an outheader subgroup, the IEDNCRPT function is performed unconditionally. If this operand is included when the IEDNCRPT macro is coded in an outheader subgroup, an the next field in the header does not match this operand, the function is not performed.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

One of the following codes is set in register 15.

Code	Meaning
X'00000000'	Successful execution
X'00000004'	Destination not an SNA LU
X'00000008'	Session not encryption bound
X'0000000C'	RU type is not FM data
X'00000010'	Encrypted data indicator already set
X'FFFFFFFC'	Zero length buffer.

IEDOPCTL Macro

The IEDOPCTL macro:

- Tests for basic operator control commands
- Transfers control to the basic operator control subtask
- Allows you to bypass or continue inheader and inbuffer subgroup processing for operator control input
- Specifies the name of the external LU or application to receive the reply to each basic operator command detected
- Specifies the user correlation ID to be used on basic operator control commands.

The scan pointer must be pointing to the first byte of the operator control command character string in order for the command to be recognized.

If basic operator control is used, either this macro or the CODE macro must be issued in the inheader subgroup of the message handler. CODE may be used to translate the message to EBCDIC before testing for basic operator control; IEDOPCTL assumes that a basic operator command, if present, is already in EBCDIC.

Supported Resources and General Requirements: ALL.

Valid subgroups: Inheader.

NAME	OPERATION	OPERANDS
[symbol]	IEDOPCTL	`[MHPROC={YES}]`
		`        {NO }`
		`[,RSPDEST={**          }]`
		`          {'name'      }`
		`          {fieldname   }`
		`          {(register)  }`
		`[,USERID={X'id'      }]`
		`         {fieldname  }`
		`         {(register) }`
		`         {**         }`
		`         {NO         }`

```
MPPROC={YES}
       {NO }
```

Function: Allows continuation of message-handler processing with a return code from IEDOPCTL processing.

Format: YES or NO.

Default: MHPROC = NO.

Note: NO causes inheader and inbuffer processing to be bypassed. Inmessage processing is performed. YES specifies that MH processing continue with a return code of zero from IEDOPCTL.

```
RSPDEST={**          }
        {'name'      }
        {fieldname   }
        {(register)}
```

Function: Specifies the name of the external LU or application that is to receive the reply to a basic operator command.

Format: **, *'name', fieldname,* or *(register). 'name'* and *fieldname* must conform to the rules for assembler language symbols. *(register)* must be specified within framing parentheses as an explicit decimal integer from 2 through 12 or as a symbol that has been previously equated to a decimal value from 2 through 12.

Default: RSPDEST=**.

Note: ** specifies that the origin of the command is the destination of the reply.

'name' specifies the name of the external LU or application to receive the reply.

fieldname specifies the symbolic name of the field containing the name of the external LU or application to receive the reply.

(register) specifies the name or number of a register continuing the address of a field that contains the name of the external LU or application to receive the reply.

```
USERID={X'id'      }
       {fieldname  }
       {(register)}
       {**         }
       {NO         }
```

Function: Specifies whether a user correlation ID is to be accepted on basic operator commands.

Format: 'id', fieldname, (register), **, or NO. *'id'* is a 4-byte hexadecimal field. *fieldname* must conform to the rules for assembler language symbols.

Default: USERID=NO.

Note: NO specifies no correlation ID.

fieldname specifies the symbolic name of the field containing the four-byte user ID.

(register) specifies the name or number of a register containing the address of a field continuing the four-byte user ID.

** specifies that the user correlation ID is in the basic operator control command following the basic operator control command identifier (which is specified on the CONTROL= operand of the INTRO macro).

Return Codes

One of the following codes is returned in register 15:

Code	Meaning
X'00000000'	Successful execution. If MHPROC = YES, message is an operator control command.
X'00000004'	TCAM could not determine if this message is a basic operator control command for one of the following reasons: • The message source is not correct. • The external LU or application program is not a basic secondary operator control station. • The request was out of the buffer before the operator control string was found. • The next field in the buffer did not match the defined operator control string.
X'00000008'	An invalid response destination was specified.
X'0000000C'	Operator control string is not found; this message is not a basic operator command.

IEDRELS Macro

The IEDRELS macro:

- May be used in place of an application-program MRELEASE macro to release an intercepted destination within message handler
- Is executed unconditionally
- Is used to turn off the TCPRGHLD bit in the TCSOPTS option field for the destination.

One of the ways of releasing a destination from intercepted status is by execution of the IEDRELS macro in a message handler. For other methods of releasing an intercepted destination, see the HOLD macro in this chapter.

Unless SYNCHLD=YES is specified, IEDRELS does not allow conditional execution within the subgroup in which it is coded; however, the delimiter macro for the subgroup may be coded in such a way that the entire subgroup may be executed conditionally. When STATION= is not coded, the destination that is released when IEDRELS is executed is the external LU sending the message. When STATION= is coded, the destination released is specified by that operand.

Supported Resources and General Requirements: ALL.

Valid subgroups: Inblock, inbuffer, inheader, and outheader.

NAME	OPERATION	OPERANDS

[symbol]	IEDRELS	[BRANCH=name]
		[,STATION={'name' }]
		{fieldname }
		{(register)}
		{** }
		[,SYNCHLD={YES}]
		{NO }

```
BRANCH=name
```

Function: Specifies the name of the next MCP instruction to be executed when IEDRELS executes successfully (return code of zero).

Format: *name* must conform to the rules for assembler language symbols.

Default: Return control to the next sequential instruction following the IEDRELS macro.

Note: *name* is the label of a macro or a user label within the current subgroup or the name of a subgroup delimiter macro.

```
STATION={'name'     }
        {fieldname }
        {(register)}
        {**        }
```

Function: Identifies the destination to be released.

Format: **, *'name', fieldname,* or *(register).* Must conform to the rules for assembler language symbols.

Default: STATION=**, if IEDRELS is specified in the inblock, inbuffer, or inheader. This operand is required if IEDRELS is specified in outheader.

Note: ** specifies the message source.

'name', is the name of a destination defined by a TERMINAL macro.

fieldname is the symbolic name of the field containing the destination name. The length of the field must equal the value specified on the MAXLEN= operand of the TTABLE macro. The destination name in the field must be left-justified and padded with blanks.

(register) specifies a register containing the address of the field that contains the station name.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

```
SYNCHLD={YES}
        {NO }
```

Function: Specifies whether the TCPRGHLD flag bit in the TCSOPTS option field for the destination is to be checked prior to releasing the held message queue. (TCPRGHLD is turned on by issuing a HOLD macro with SYNCHLD=YES or by user code.)

Format: YES or NO.

Default: SYNCHLD=NO.

Note: SYNCHLD=YES specifies that TCPRGHLD is to be checked prior to releasing the queue. If TCPRGHLD is on, it is turned off and if the release of the destination is successful (return code of zero), control is returned to the instruction at the address specified by the BRANCH= operand, or to the next sequential instruction following the IEDRELS macro if no BRANCH= operand was specified. If the release is not successful, control is returned to the next sequential instruction. If TCPRGHLD is off the queue is not released and control is returned to the next sequential instruction.

SYNCHLD=NO specifies that TCPRGHLD is not to be examined when releasing the message queue.

If SYNCHLD=YES is specified, the TCSOPTS option field must be defined for the destination. If the TCSOPTS option field is not found, the queue is released (if the destination is held) and a return code of eight will be returned.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	Successful execution
X'00000004'	Destination not held; cannot be released
X'00000008'	SYNCHLD=YES specified, and no TCSOPTS option field could be found for the message origin. Destination is released if the destination is held. (Note that if no TCSOPTS is found and the origin destination is not held, a return code of 4 is issued).
X'00000010'	Destination not found, invalid name.
X'0000000C'	SYNCHLD=YES and TCPRGHLD not on for the message origin. Destination is not released.

IEDRESP Macro

The IEDRESP macro:

- Generates immediate and controlled responses to incoming messages
- Is executed either conditionally or unconditionally.

The IEDRESP macro is used to respond to incoming SNA chains (messages). When IEDRESP is executed, an immediate or controlled response (with specified sense) is sent to the external LU. Only one response per message is sent. TCAM transmits the required response.

There are two types of SNA responses: *positive* and *exception* (negative). IEDRESP may send exception responses out of inheader, inbuffer, inmessage, and outheader subgroups. Positive responses may only be sent out of the inmessage or outheader subgroup.

In the following cases, no response is sent to the external LU:

- The request chain requested no response.
- A response has already been sent (either by TCAM or a prior IEDRESP).

Note: If an error is detected by TCAM, an exception response is sent, if requested, reflecting the error. Any IEDRESP macro executed after TCAM detects an error does not send a response. When using the controlled response function, it is your responsibility to determine whether a response is necessary and what type of response you wish to send. You are also responsible for preventing the generation of a second response from the outheader subgroup if you have already generated a response from the execution of a previous IEDRESP macro (with HOLD=YES specified). (For more information on the controlled response function, see the *TCAM Installation Guide* and the description of the HOLD= operand in this section.)

- The Request is a Cancel command. TCAM sends all responses to Cancel.
- The request is Ready-to-Receive (RTR) and IEDRESP is trying to send a positive response. TCAM sends all positive responses to RTR. However, it is permissible to send an exception response to RTR using IEDRESP.

The allowable major sense codes (the major sense code is the first sense byte) are:

X'40'	RH usage error
X'20'	state error
X'10'	request error
X'08'	request reject
X'00'	user sense present.

No other major sense code is allowed.

When IEDRESP is executed in an inmessage subgroup, it is not possible to use a return code.

In order to send a response, IEDRESP must be coded before the first MSGGEN in the inmessage subgroup. Any IEDRESP executed after the first MSGGEN does not send a response. IEDRESP must also be executed before any ERRORMSG macro using the error exit because information concerning the error is passed to the error routine. (For more information on the error exits, see the ERRORMSG macro in this chapter.)

Supported Resources and General Requirements: SNA.

IEDRESP

Valid subgroups: Inbuffer, inheader, inmessage and outheader.

When Used in an Inheader or Inbuffer Subgroup

NAME	OPERATION	OPERANDS
[symbol]	IEDRESP	[conchars]
		[,BLANK={YES }]
		{NO }
		{char }
		[,CLEAR={YES }]
		{NO }
		[,COUNT=(opfield1,{operator},{opfield2})]
		{EQ } {integer }
		{0 }
		[,HOLD={YES}]
		{NO }
		[,OVERIDE={YES}]
		{NO }
		[,SENSE={XL4'chars'}]
		{(register)}
		{opfield }
		[,UPDATE={ADD}]
		{SUB}
		{NO }

When Used in an Inmessage Subgroup

NAME	OPERATION	OPERANDS
[symbol]	IEDRESP	[mask]
		[,CLEAR={YES}]
		{NO }
		[,CONNECT={NAND}]
		{AND }
		{OR }
		[,COUNT=(opfield1,{operator},{opfield2})]
		{EQ } {integer }
		{0 }
		[,OVERIDE={YES}
		{NO }
		[,SENSE={XL'chars'}]
		{opfield }
		[,UPDATE={ADD}]
		{SUB}
		{NO }

When Used in an Outheader Subgroup

NAME	OPERATION	OPERANDS

[symbol]	IEDRESP	`[conchars]`
		`[,BLANK={YES }]`
		`        {NO  }`
		`        {char}`
		`[,CLEAR={YES}]`
		`        {NO }`
		`[,COUNT=(opfield1,{operator},{opfield2})]`
		`                 {EQ      } {integer }`
		`                            {0       }`
		`[,SENSE={XL4'chars'}]`
		`        {(register)}`
		`        {opfield   }`
		`[,SNAVALS=opfield]`
		`[,UPDATE={ADD}]`
		`         {SUB}`
		`         {NO }`

```
BLANK={YES }
      {NO  }
      {char}
```

Function: Specifies whether EBCDIC blank characters are to be ignored when being counted or when encountered in the character string in the message header being compared to the string specified by the *conchars* operand, or whether blanks are to be counted as characters or as part of the header string when encountered in it. If EBCDIC blanks are to be counted as characters or as part of the header string, this operand also specifies whether another hexadecimal character is to be ignored when skipping or when encountered in the header string.

Format: YES, NO, or *char*. *char* is a single character that may be specified in either character or hexadecimal format. If character format is specified, it may be unframed or framed with C'' or CL1'' characters. If hexadecimal format is specified, it must be framed with X'' or XL1'' characters.

YES specifies that the EBCDIC blank character (X'40') is to be ignored by the macro whenever characters are being skipped or whenever it is encountered in the header character string being checked against the skip character string specified by the *conchars* operand. For example, if BLANK=YES is coded and an eight-byte field in the header is being checked by this macro, a blank appearing in the fifth byte of the field is ignored and the sixth through ninth bytes are considered to be the last four bytes of the field (assuming that no blanks are coded in the sixth through ninth bytes).

NO specifies that the EBCDIC blank character is to be treated in the same way as other character when characters are being skipped or whenever it is encountered in the header string being compared to the string specified by *conchars*.

char specifies that single character replacing *char* is to be ignored by this macro whenever characters are being skipped or whenever encountered in the header string being compared to the string specified by the *conchars* operand. This is, the macro automatically skips over the character without counting or performing a comparison and goes on to check the next character in the header. If BLANK=*character* is coded and *char* is not the EBCDIC blank character, the EBCDIC blank is not ignored by this macro when it is encountered in the header, but is counted or compared to the character in the corresponding space in the *conchars* string in the same way as any other character.

This operand is ignored if IEDRESP is coded in the inmessage subgroup.

```
CLEAR={YES}
      {NO }
```

Function: Specifies whether the option field specified as the comparator is to be cleared after successful execution of the COUNT= operand expression.

Format: YES or NO.

Default: CLEAR=YES.

```
conchars
```

Function: Specifies the character or character string that, if found in the header as the next nonblank field, causes the IEDRESP function to be execution.

Format: One to eight nonblank characters in character or hexadecimal format. If character format is used, the string may be unframed or framed with C'' or CLn'' characters. If hexadecimal format is used, the string must be framed with X'' or XLn'' characters.

Default: None. Specification is optional.

Note: If this operand is omitted, the response function is performed based on whether the COUNT= operand expression is satisfied. If the next field in the header does not match this operand, the function is not performed. This operand is invalid if IEDRESP is coded in an inmessage subgroup.

```
CONNECT={NAND}
        {AND }
        {OR  }
```

Function: Specifies the type of logical connection to be made between the mask and the message error record.

Format: NAND, AND, or OR.

Default: CONNECT=OR.

Note: NAND specifies that the macro is to be executed only if *all* of the bits specified by *mask* are off in the message error record. AND specifies that the macro is to be executed only if *all* of the bits specified by *mask* are on in the message error record. OR specifies that the macro is to be executed if *any* bit specified by *mask* is on in the message error record.

This operand is ignored if IEDRESP is coded in an inheader or inbuffer subgroup.

```
COUNT=(opfield1,{operator},{opfield2})
                {EQ      } {integer }
                           {0       }
```

Function: Specifies an expression to determine execution of the IEDRESP macro function.

Format: opfield1 and *opfield2,*if specified, must conform to the rules for assembler language symbols and must be the name of an option field defined by an OPTION macro. *operator* must be an acceptable algebraic operator *integer* if specified, must be a decimal or hexadecimal integer not to exceed 16,777,215.

Default: None. Specification is optional.

opfield1 specifies the name of a 1-, 2-, or 3-byte option field containing a binary count that is to be used as the comparator in the expression. This suboperand must be specified if COUNT= is used.

You may gain access to the field at any time to determine or reset the count (by operator commands or by user code, including the LOCOPT macro). The count is initially set using the OPDATA= operand of the TERMINAL or TPROCESS macro.

If the option field is not found, IEDRESP does not execute and control passes to the next MH instruction.

operator specifies the algebraic operator to be used in evaluating the expression. If *operator* is omitted, it defaults to Equal (EQ or =). One of the following may be coded: =, ≠, <, ≤, >, ≥, EQ, NE, LT, LE, GT, or GE.

COUNT=(,,*opfield2)* or COUNT=(,,*integer)* specifies the comparand to be used in evaluating the expression. If neither parameter is coded, the comparand defaults to 0. The comparand must be a decimal or hexadecimal integer not to exceed 16,777,215.

If COUNT= is not specified, execution is determined by the error mask (inmessage) or *conchars* (inheader/outheader) operands. If neither is specified, execution is unconditional.

```
HOLD={YES}
     {NO }
```

Function: Specifies whether the SNA response RU is to be held until the IEDRESP macro is executed in the outheader subgroup of the device message handler.

Format: YES or NO.

Default: HOLD = NO.

Note: HOLD = YES may be coded only in the inbuffer or inheader subgroup of the MH. It is valid only for the last segment of the message. This operand is part of the controlled response function of the IEDRESP macro. See the *TCAM Installation Guide* for more information on the controlled response function. A return code of X'00000014' will result when HOLD = YES is coded to hold a response.

```
mask
```

Function: Specifies a 1- to 5-byte field used to test the message error bits for conditional macro execution.

Format: Unframed decimal integer or hexadecimal field. If hexadecimal format is used, framing characters must be specified. If X'' is used, leading zeros must be coded. If XL5'' is used, leading zeros may be omitted.

Default: None. Specification is optional.

Note: Omitting the operand or specifying an all-zero mask causes execution based on whether the COUNT= operand is satisfied.

This operand is invalid if IEDRESP is coded in an inheader or inbuffer subgroup.

```
OVERIDE={YES}
        {NO }
```

Function: Specifies whether the current sense setting for a negative response can be changed.

Format: YES or NO.

Default: OVERIDE = NO.

```
SENSE={XL4'chars'}
      {(register)}
      {opfield   }
```

Function: Specifies the 4 sense bytes to be placed in the response.

Format: *chars, (register), or opfield. chars* must be specified in hexadecimal format with framing characters. *(register)* must specify a register (2-12) enclosed in parentheses. *opfield* must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary") and must be the name of an option field defined by an OPTION macro.

Default: None. Specification is required in the inheader or inbuffer subgroup, and is optional in the inmessage and outheader subgroups.

chars specifies the 4 sense bytes in hexadecimal format, i.e. XL4''. *(register)* specifies a register that contains the address of a 4-byte field containing the sense bytes. *opfield* represents the name of a 4-byte option field containing the sense bytes.

If SENSE= is coded, a negative response is generated; otherwise, a positive response is sent. See the *TCAM Installation Guide* for more information on positive and negative responses and the meaning and format of the sense data field.

When SENSE=*opfield* is specified, the first 4 bytes of the option field are used as the sense bytes. Therefore, the option field should be specified as being at least 4 bytes long. No check is made for an option field of less than 4 bytes.

SENSE=*(register)* can be coded in the inbuffer, inheader, or outheader subgroup only and is invalid in the inmessage subgroup.

If SENSE= is coded in the outheader subgroup, it is recognized only if the Sense Data Included bit in the RH is on.

`SNAVALS=opfield`

Function: Specifies the name of the 8-byte option field in which the RH and the sequence number are saved for the controlled response function of the IEDRESP macro. (For more information on the controlled response function, see the *TCAM Installation Guide*.)

Format: *opfield* must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary") and must not be specified with framing parentheses.

Default: None. Specification is required for the controlled response function of the IEDRESP macro.

Note: This operand is valid only in the outheader subgroup.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

```
UPDATE={ADD}
       {SUB}
       {NO }
```

Function: Specifies whether the option field specified as the comparator is to be incremented (ADD) by 1, decremented (SUB) by 1, or left unchanged (NO) before the expression is evaluated.

Format: ADD, SUB, or NO.

Default: UPDATE=ADD.

Return Codes

When IEDRESP is issued in an inheader or inbuffer subgroup, a return code indicates whether the response was successful. The return code values are defined as follows:

Code	**Meaning**
X'00000000'	Successful execution.
X'00000004'	• IEDRESP could not execute as a result of COUNT= operand comparison • Could not obtain a buffer
X'00000008'	Option field invalid; an option field specified by COUNT=, SENSE=, or SNAVALS= operand does not exist for this external LU.
X'0000000C'	Invalid response sense was specified on SENSE= operand. The first byte (major sense byte) of the 4 bytes of sense must be one of the following: X'40' - RH usage error X'20' - state error X'10' - request error X'08' - request reject X'00' - user sense See the *SNA Reference Summary* for SNA sense codes.
X'00000010'	A response has already been sent to this chain. Either TCAM or IEDRESP has caused a response to be generated.
X'00000014'	Not allowed to send a response to this request. Either the request asked for no response, the response has been held because HOLD=YES was coded, or this is an SNA Cancel command. If in OUTHDR, no buffers were available. See the *SNA Reference* for information on the Cancel command.
X'00000018'	The resource to which the response is to be sent is not an logical unit. (IEDRESP only sends responses to logical units.)
X'FFFFFFFC'	Zero length buffer.

IEDRH Macro

The IEDRH macro:

- Performs a checking function on indicators in an RH when:
  - It is coded in an input message handler, or
  - It is coded in an output message handler and CHKONLY is specified in the output MH.
- Performs a setting function on bits in an RH (unless CHKONLY is specified) in the output MH.

The IEDRH macro is used in a DMH serving as part of a TCAM host LU to gain information required for the enforcement of SNA protocols. See the *TCAM Installation Guide* for more information.

Supported Resources and General Requirements: SNA.

Valid subgroups: Inblock, inbuffer, inheader, outbuffer, and outheader.

NAME	OPERATION	OPERANDS

[symbol]	IEDRH	`[BSTATE={YES}]`
		`       {NO }`
		`[,RHIND=({+FMDATA},{+EXR},{+FMH},{*BB,}`
		`         {-FMDATA} {-EXR} {-FMH} {+BB }`
		`         {+DFC   }             {-BB }`
		`         {-DFC   }`
		`{*EB},{+SELCDE},{*CHNGDIR},{+RSP},CHKONLY)]`
		`{+EB} {-SELCDE} {+CHNGDIR} {-RSP})`
		`{-EB}           {-CHNGDIR} {+REQ}`
		`                           {-REQ}`

```
BSTATE={YES}
       {NO }
```

Function: Specifies whether the current bracket state is to be returned in register 15 after execution of the IEDRH macro.

Format: YES or NO.

Default: BSTATE=NO.

The current bracket state is returned in the second byte of register 15. It contains the following values for bracket states:

Between brackets (BETB); no RTR	X'00'
Between brackets (BETB); with RTR	X'10'
Pending BETB; no RTR	X'20'
Pending BETB; with RTR	X'30'
In bracket state (INB); no RTR	X'40'
In bracket state (INB); with RTR	X'50'
Pending INB; no RTR	X'60'
Pending INB; with RTR	X'70'
Begin bracket pending state (BBP)	X'80'

If BSTATE=YES is coded on the IEDRH macro on the input side of the MH, the following test for checking a return code is not adequate:

```
LTR         REG15,REG15
BNZ         ERROR.
```

Instead, to check for the current bracket state, use the following test:

```
CLM                 REG15,4,FIELD
BNE                 ERROR
```

where FIELD=BSTATE

The CLM instruction can also be used to check return code only or a combination of BSTATE with the return code. (See *IBM System/370 Principles of Operation* for more information on the CLM instruction.)

```
RHIND=( +FMDATA , +EXR , +FMH , *BB , *EB ,  +SELCDE , *CHNGDIR , +RSP ,CHKONLY)
       "-FMDATA" "-EXR  "-FMH  "+BB" "+EB"  "-SELCDE" "+CHNGDIR" "-RSP
       "+DFC    "              "-BB" "-EB"            "-CHNGDIR" "+REQ
       "-DFC    "                                                "-REQ
```

Function: Specifies the status flags in the RH to be checked on input or set on output or, if CHKONLY is coded, specifies that only checking of SNA indicator bits is to be done. The meanings of the various indicators and the RH bit settings corresponding to them are:

Indicator	**Meaning**	**RH Bit**	**Value**	**-Value**
FMDATA	Function management data	RU type	0	1
DFC	Data flow control	RU type	1	0
EXR	Exception request	Sense included	1	0
FMH	Formatted header	Format indicator	1	0
BB	Begin bracket	Begin bracket	1	0
EB	End bracket	End bracket	1	0
SELCDE	Select code	Code selection	1	0
CHNGDIR	Change direction	Change direction	1	0
RSP	Header is response header	Response	1	0
REQ	Header is request header	Response	0	1

*Format: indicator, -indicator, *indicator*. The indicators are not positional; they may be coded in any order.

Default: None. The indicators are optional.

When the IEDRH macro performs the checking function:

- All the indicators are valid with + or -
- Use of the * is valid for the input MH
- "+" means check the specified indicator to see if its value is set as defined in the preceding table
- "-" means check the specified indicator to see if its - value is set as defined in the preceding table

- The DFC and FMDATA indicators, and the RSP and REQ indicators are mutually exclusive (only one of each pair of these indicators may be coded per IEDRH macro)
- The low-order bytes of register 15 are set as follows:

Byte 0	zeros	
Byte 1	The value for the current bracket state if BSTATE=YES was specified; otherwise zeros.	
Byte 2	Error bits	
	Bits 0-5	Reserved
	Bit 6	The resource is not a logical unit. No RH bits were checked.
	Bit 7	Bracket state not available, that is, brackets are not specified on the SNA BIND command. Invalid to set bracket bits or specified BSTATE=YES.

When the IEDRH macro performs the setting function:

- In the outheader subgroup, only the BB, EB, FMH, SELCDE, and CHNGDIR indicators are invalid. (BB and EB are invalid only for the first buffer of each output message.)
- In the outbuffer subgroup, only the FMH, CHNGDIR, and SELCDE indicators are valid.
- "+" means set the specified indicator to its value, as defined in the preceding table.
- "-" means set the specified indicator to its -value, as defined in the preceding table.
- "*" (valid only for BB, EB, and CHNGDIR) means that TCAM will decide how to set the specified indicator. TCAM will set the indicator to either its value or its-value, depending on the current state of the LU session.
- If an indicator is not specified, no status flags are set by the IEDRH macro. (All of the RH bits are initialized to zeros by TCAM before the buffer starts through the output MH.)
- The IEDRH macro supplies the following information in register 15:

Byte 0	Reserved	
Byte 1	The value for the current bracket state if BSTATE=YES was specified.	
Byte 2	Error bits	
	Bits 0-2	Reserved
	Bit 3	An attempt has been made to send a change direction indicator. This is not allowed in FM profile type 2.
	Bit 4	FM head indicator is set in RH. This is not allowed in FM profile type 2.
	Bit 5	An attempt has been made by a secondary LU to send EB. This is not allowed in FM profile type 2.
	Bit 6	Resource is not a logical unit. RH bits are not set if the resource is not a logical unit.
	Bit 7	Bracket state not available. It is invalid to set bracket bits or specify BSTATE=YES.
Byte 3		
	Bit 0	Begin Bracket (BB) must be set between brackets.
	Bit 1	Invalid to send when between brackets and pending.
	Bit 2	Invalid to set begin bracket (BB) bit when in brackets.
	Bit 3	Invalid to set end bracket (EB) bit when BIND specifies that the primary LU cannot send EB.
	Bit 4	Invalid to set FM header bit when BIND specifies that FM headers cannot be used.
	Bit 5	Invalid to set alternate code bit when BIND specifies that alternate code cannot be used.

Bit 6	Invalid to set BB or EB on middle RU of chain or last RU of chain.
Bit 7	Reserved

If no error bits are on, the macro executed successfully. If one or more error bits are on, an error or multiple errors were detected.

symbol

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Return Codes

When IEDRH performs checking, one of the following return codes is set in the low-order byte of register 15:

Code	**Meaning**
X'00xxxx00'	All the checks specified result in 1's.
X'00xxxx04'	Some, but not all, of the checks result in 1's.
X'00xxxx08'	All checks result in 0's.

Note: x bit settings may vary; see the RHIND= operand.

IEDSENSE Macro

The IEDSENSE macro:

- Transfers SNA sense status bytes associated with an error buffer into a user-specified area.

The IEDSENSE macro does not affect and does not depend on the order or function of other MH macros. It moves 4 bytes of sense status information into a user-specified area. The specified area must be within the MH.

Supported Resources and General Requirements: SNA.

Valid subgroups: Inblock, inbuffer, inheader, outbuffer, and outheader.

NAME	OPERATION	OPERANDS

[symbol]	IEDSENSE	AREA={(register)}
		{name }

```
AREA={(register)}
     {name      }
```

Function: Provides the name or address of a user-specified area.

Format: *(register)* is the actual register number or a register name. Registers 2 through 12 may be used. If *(register)* is used, the number or name must be enclosed in parentheses. *name* is the name of a data area and must conform to rules for assembler language symbols.

Default: None. *(register)* or *name* is required.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	Move was successful.
X'00000004'	Not an error buffer; move was not performed.
X'00000008'	Resource is not an LU.

IEDSHOW Macro

The IEDSHOW macro:

- Provides the origin of the current message
- Provides the destination of the current message
- Provides the byte count in the current TCAM buffer (minus the prefix)
- Provides status information concerning the current operation
- Provides current SNA RH values and SNA sequence number
- Provides the input or output sequence number assigned by the MCP to the current message
- May be used in the ERRORMSG and REDIRECT exit routines, the STARTMH bind and unbind exit routines, the TYPETABL message-type exit routine, and in closed subroutines.

The IEDSHOW macro allows you to selectively extract information from TCAM control blocks during message processing. The specified data is moved to a field provided in the message handler or an option field. This MH field, whose name is specified by the RESULT= operand, or the option field whose name is specified by the TARGET= operand, can then be inspected by user code to determine a course of action.

Supported Resources and General Requirements: ALL.

Valid subgroups: Inblock, inbuffer, inheader, outbuffer, and outheader.

NAME	OPERATION	OPERANDS
[symbol]	IEDSHOW	{CURTERM }
		{DATASIZE}
		{DESTIN }
		{ORIGIN }
		{SEQIN }
		{SEQOUT }
		{SESSTAT }
		{SNAVALS }
		{TSTATUS }
		[,RESULT=fieldname]
		[,FORMAT={YES}]
		{NO }
		[,STATION={'name' }]
		{fieldname }
		{(register)}
		{** }
		[,TARGET=opfield]

```
{CURTERM }
{DATASIZE}
{DESTIN  }
{ORIGIN  }
{SEQIN   }
{SEQOUT  }
{SESSTAT }
{SNAVALS }
{TSTATUS }
```

Function: Specifies what information is desired from various TCAM control blocks.

Format: CURTERM, DATASIZE, DESTIN, ORIGIN, SEQIN, SEQOUT, SESSTAT, SNAVALS, or TSTATUS.

Default: None. This operand is required.

CURTERM provides the name of the currently connected external LU or an index to its location in the terminal name table. CURTERM cannot be coded outside of an MH.

DATASIZE provides the number of bytes of actual data in the current TCAM buffer in the leftmost 2 bytes of the field specified in the RESULT= operand.

DESTIN provides the destination name or an index to its location in the terminal name table of the current message. If DESTIN is coded in an MH incoming group, it will only act on the header buffer after the destination has been determined by a routing macro.

ORIGIN provides the source name or an index to its location in the terminal name table of the current message.

SEQIN provides the input sequence number of the current message (input sequence number of the next message if the SEQUENCE macro is issued prior to the IEDSHOW macro).

SEQOUT provides the output sequence number of the current message.

SESSTAT provides the identification of the LU that is responsible for error recovery.

SNAVALS provides the current values of the SNA response header and the SNA sequence number; the RH is located in the first three bytes of SNAVALS, and the sequence number is located in the fifth and sixth bytes of SNAVALS.

TSTATUS provides the type and status of the external session partner or of a external LU specified by the STATION= operand. When TSTATUS is coded outside the MH (in a bind or unbind exit), you must set register 13 to point to an 18-word save area before you issue IEDSHOW.

When SESNSTAT is coded, the information that is requested is formatted in the RESULT field as follows:

Byte	Value	Meaning
Byte 0	X'00'	The primary LU is responsible for recovery.
	X'20'	Sender of request is responsible for recovery.
Byte 1-7	Unused	

When TSTATUS is specified, the information supplied is formatted as follows:

Byte	Value	Meaning
0	X'00'	Reserved

The first three bits of the second byte, considered as an entity, indicate the type of resource entry.

Byte	Value	Meaning
1	X'00'	Single entry
	X'20'	Process entry
	X'40'	List entry

Byte	Value	Meaning
	X'60'	Logtype entry
	X'10'	TTE has a prefix
	X'08'	Process entry status
	X'04'	Destination in hold mode
	X'02'	Option fields used
	X'01'	Secondary operator control station
2	X'00'	Reserved
3	X'02'	Device type is 3270
	X'80'	Resource is started
4-5		If this is an LU that is currently in an LU-LU session, bytes 4 and 5 contain the relative entry (TTCIN) into the terminal name table of the entry for the session partner for the last session initiated. Otherwise, these bytes contain all zeros.
6	X'00'	Reserved. This byte only appears for single entries.

The following values in the seventh byte apply to stations defined as LUs:

Byte	Value	Meaning
6	X'00'	Station is an SNA LU

```
FORMAT={YES}
       {NO }
```

Function: Specifies whether the external LU name or the entry number in the terminal name table is provided.

Format: YES or NO.

Default: FORMAT=NO.

Note: If YES is coded, the station name is provided. If NO is coded, a 2-byte, left-justified index representing the number of the station in the terminal name table is provided in the RESULT=field.

FORMAT= is permitted only if DESTIN, CURTERM, or ORIGIN is coded as the first operand of this macro.

```
RESULT=fieldname
```

Function: Specifies the name of an 8-byte field to contain the result of your inquiry.

Format: *fieldname* must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary") and must not be specified with framing characters.

Default: None. This operand is optional.

Note: This field cannot be an option field. If you wish to have the result of your inquiry placed in an option field, specify TARGET=opfield (see the

TARGET= operand description in this section). RESULT=fieldname is invalid when coded with SNAVALS.

```
STATION={'name'     }
        {fieldname  }
        {(register) }
        {**         }
```

Function: Specifies the station whose status is desired.

Format: *'name'*, *fieldname* , (*register*), or **. *'name'* must conform to the rules for assembler language symbols and must be specified within single quotes. *fieldname* must conform to the rules for assembler language symbols and must not be specified with framing characters. *(register)* is the actual register number or the equated name of a register enclosed in parentheses. Registers 0 through 15 may be used.

Default: STATION=**.

Note: *'name'* is the name of an external LU specified by the TERMINAL macro. *fieldname* is the symbolic name of the field containing the external LU name, the length of which must correspond to the MAXLEN= operand of the TTABLE macro. *(register)* specifies a register containing the address of the field that contains the name. ** specifies that you want the status of the external session partner.

The STATION= operand can be used only when SESSTAT or TSTATUS is coded as the first operand. TSTATUS,STATION= is valid in all subgroups except inmessage and outmessage. It is also valid outside the message handler in closed subroutines entered from MH subgroups, the ERRORMSG exit, the FORWARD exit, and bind/unbind exits.

If TSTATUS,STATION=** is coded (or the default is taken) outside an MH, TCAM expects to be processing a buffer in buffer format so that it can find the LCB to locate the currently connected station.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

```
TARGET=opfield
```

Function: Specifies the name of an 8-byte option field to contain the result of your inquiry.

Format: *opfield* must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary") and must not be specified with framing parentheses.

Default: None. Specification is optional, unless SNAVALS is coded.

Note: This field must be an option field. The TARGET= operand is required when SNAVALS is coded.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	Successful execution.
X'00000004'	Function not supported.
X'00000008'	Resource not found; invalid name.
X'00000010'	Status information for this resource not found.

IEDSMODE Macro

The IEDSMODE macro:

- Allows the MH to request a set of LU-LU session parameters
- Specifies whether the brackets indicator is to be returned
- Specifies whether the session normal-flow send/receive mode is to be returned
- Specifies whether the primary/secondary indicator is to be returned
- Specifies whether the quiesce state indicator is to be returned
- Specifies whether the shutdown state indicator is to be returned.

The IEDSMODE macro returns a set of LU-LU session parameters in register 15. The operands specify whether you want the particular parameter to be returned; if you do, the indicators are returned in byte 0 of register 15 as specified under the return codes at the end of this macro description.

Supported Resources and General Requirements: SNA.

Valid subgroups: Inblock, inbuffer, inheader, outbuffer, and outheader.

NAME	OPERATION	OPERANDS
[symbol]	IEDSMODE	[BCKTSFS={YES}] {NO } [,MODE={YES}] {NO } [,PSIND={YES}] {NO } [,QSTATE={YES}] {NO } [,SSTATE={YES}] {NO }

```
BCKTSFS={YES}
        {NO }
```

Function: Specifies whether a bracket indicator for the session is to be returned in bit 5 of byte 0 of register 15.

Format: YES or NO.

Default: BCKTSFS=NO.

Note: YES specifies the bracket indicator is to be returned; NO specifies that it is not to be returned. This indicator shows whether the host is first speaker or bidder on the session.

```
MODE={YES}
     {NO }
```

Function: Specifies whether the session normal-flow send/receive mode is to be returned in bits 0-1 of byte 0 of register 15.

Format: YES or NO.

Default: MODE=NO.

Note: YES specifies that the normal-flow send/receive mode is to be returned; NO specifies that it is not to be returned.

The normal-flow send/receive mode controls the protocol regulating the flow between network addressable units. Possible modes of operation are full-duplex, half-duplex/contention, or half-duplex/flip-flop.

```
PSIND={YES}
      {NO }
```

Function: Specifies whether a primary LU and secondary LU indicator is to be returned in bit 4 of byte 0 of register 15.

Format: YES or NO.

Default: PSIND=NO.

Note: YES specifies that the primary LU and secondary LU indicator is to be returned; NO specifies that it is not to be returned. This indicator shows whether the host LU is primary or secondary on the session.

```
QSTATE={YES}
       {NO }
```

Function: Specifies whether the quiesce state is to be returned in bit 6 of byte 0 of register 15.

Format: YES or NO.

Default: QSTATE=NO.

Note: YES specifies that the quiesce state is to be returned; NO specifies that it is not. This indicator shows whether the Quiesce at End of Chain SNA command has been received and, if so, if the Quiesce Complete command has been sent.

```
SSTATE={YES}
       {NO }
```

Function: Specifies whether the shutdown state indicator is to be returned in bits 2-3 of byte 0 of register 15.

Format: YES or NO.

Default: SSTATE=NO.

Note: YES specifies that the shutdown state indicator is to be returned; NO specifies that it is not to be returned. This indicator reflects whether Shutdown (SHUTD) has been received or Shutdown Complete (SHUTC) has been sent (mutually exclusive indicators).

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Return Codes

The indicators are returned in byte 0 of register 15 as follows:

Bit	Value	Meaning
0-1	B'00'	Full-duplex
	B'01'	Half-duplex/contention
	B'10'	Half-duplex/flip-flop
2	B'1'	Shutdown received
3	B'1'	Shutdown Complete sent
4	B'0'	Host LU is primary
	B'1'	Host LU is secondary
5	B'0'	Host LU is bidder
	B'1'	Host LU is first speaker
6	B'0'	Quiesce at End of Chain (QEC) command not yet received
	B'1'	QEC received; Quiesce Complete (QC) not yet sent.

In register 15:

X'aa000000'	Successful exectuion (values for *aa* are defined above).
X'00000004'	Resource is not an LU.

IEDSTOP Macro

The IEDSTOP macro:

- Causes a MODIFY STOP basic operator control command to be executed for the external LU
- Causes, if an interval is specified, a MODIFY START basic operator control command to be executed for the LU once the interval has expired
- Is conditionally executed
- May be specified more than once within a subgroup
- May be specified in more than one subgroup of a message handler.

When executed in an MH handling messages for an external LU, IEDSTOP may be used to stop message flow to an external LU and terminate all LU-LU sessions that the LU has with TCAM host LUs. Message flow may be stopped when the content of an option field is tested against a specified value. (The option field and value are both specified by the COUNT= operand.) Message flow may also be stopped for a specified time interval specified by the INTVL= operand.

When coded in an inheader or outheader subgroup, execution may be made dependent on the occurrence of a specified character sequence in the header.

When coded in an inmessage or outmessage subgroup, IEDSTOP may be executed by testing a user-specified mask against the contents of the message error record.

Supported Resources and General Requirements: SNA.

Valid subgroups: Inheader, inmessage, outheader, and outmessage.

When Used in an Inheader or Outheader Subgroup

NAME	OPERATION	OPERANDS
[symbol]	IEDSTOP	[conchars]
		[,BLANK={YES }]
		{NO }
		{char}
		[,CLEAR={YES}]
		{NO }
		[,COUNT=(opfield1,{operator},{opfield2})]
		{EQ } {integer }
		{0 }
		[,INTVL=integer]
		[,UPDATE={ADD}]
		{SUB}
		{NO }

When Used in an Inmessage or Outmessage Subgroup

NAME	OPERATION	OPERANDS
[symbol]	IEDSTOP	[mask]
		[,CLEAR={YES}] {NO }
		[,CONNECT={AND}] {OR }

		[,COUNT=(opfield1,{operator},{opfield2})]
		{EQ } {integer }
		{0 }

		[,INTVL=integer]

		[,UPDATE={ADD}]
		{SUB}
		{NO }

```
BLANK={YES }
      {NO  }
      {char}
```

Function: Specifies whether EBCDIC blank characters are to be ignored when they are being counted or when they are encountered in the character string in the message header being compared to the string specified by the *conchars* operand or whether blanks are to be counted as characters or as part of the header string when encountered in it. EBCDIC blanks are to be counted as characters or as part of the header string, this operand also specifies whether another hexadecimal character is to be ignored when skipping or when encountered in the header string.

Format: YES, NO or *char*. *char* is a single character that must be specified in either character or hexadecimal format. If character format is specified, it may be unframed or framed with C″ or CL1″ characters. If hexadecimal format is specified, it must be framed with X″ or XL1″ characters.

Default: BLANK=YES

Note: YES specifies that the EBCDIC blank character (X′40′) is to be ignored by the macro whenever characters are being skipped or whenever it is encountered in the header character string being checked against the skip character string specified by the *conchars* operand. For example, if BLANK=YES is coded and an eight-byte field in the header is being checked by this macro, a blank appearing in the fifth byte of the field is ignored and the sixth through ninth bytes are considered to be the last four bytes of the field (assuming that no blanks are coded in the sixth through ninth bytes).

No specifies that the EBCDIC blank character is to be treated in the same way as any other character when characters are being skipped or whenever it is encountered in the header string being compared to the string specified by *conchars*.

char specifies that the single character replacing *char* is to be ignored by this macro whenever characters are being skipped or whenever encountered in the header string being compared to the string specified by the *conchars* operand. That is, the macro automatically skips over the character without counting or performing a comparison and checks the next character in the header. If BLANK=*char* is coded and *char* is not the EBCDIC blank character, the EBCDIC blank is not ignored by this macro when it is encountered in the header, but is counted or compared to the character in the corresponding space in the *conchars* string in the same way as any other character.

This operand is ignored if IEDSTOP is coded in the inmessage or outmessage subgroup.

```
CLEAR={YES}
      {NO }
```

Function: Specifies whether the option field specified as the comparator is to be cleared after successful execution of the COUNT= operand expression.

Format: YES or NO.

Default: CLEAR=YES.

```
conchars
```

Function: Specifies the character or character string that, if found in the header as the next nonblank field, causes the IEDSTOP macro function to be executed.

Format: One to eight nonblank characters in character or hexadecimal format. If character format is used, the string may be unframed or framed with C'' or CLn'' characters. If hexadecimal format is used, the string must be framed with X'' or XLn'' characters.

Default: None Specification is optional.

Note: If this operand is omitted, the IEDSTOP function is performed based on whether the COUNT= operand expression is satisfied. If the next field in the header does not match this operand, the function is not performed.

This operand is invalid if IEDSTOP is coded in the inmessage or outmessage subgroup.

```
CONNECT={AND}
        {OR }
```

Function: Specifies the type of logical connection to be made between the mask and the message error record.

Format: AND or OR.

Default: CONNECT=OR.

Note: AND specifies that the macro is to be executed only if *all* of the bits specified by the mask are set on in the message error record. OR specifies that the macro is to be executed if *any* bit specified by the mask is set on in the message error record.

This operand is invalid if IEDSTOP is coded in the inheader or outheader subgroup.

```
COUNT=(opfield1,{operator},{opfield2}
                {EQ      } {integer }
                           {    0   }
```

Function: Specifies an expression to determine execution of the IEDSTOP macro function.

Format: *opfield1* and *opfield,* if specified, must conform to the rules for assembler language symbols and must be the name of option field defined by an OPTION macro. *operator* must be an acceptable algebraic operator. *integer,* if specified, must be a decimal or hexadecimal integer not to exceed 16,777,215.

Default: None. Specification is optional.

Note: *opfield1* specifies the name of a 1-, 2-, or 3-byte option field containing a binary count that is to be used as the comparator in the expression. This suboperand must be specified if COUNT= is used.

You may gain access to the field at any time to determine or reset the count (by operator commands or user code, including the LOCOPT macro). The count is initially set using the OPDATA= operand of the TERMINAL macro. If the option field is not found, IEDSTOP does not execute, and control passes to the next MH instruction.

operator specifies the algebraic operator to be used in evaluating the expression. If *operator* is omitted, it will default to EQUAL (EQ or =). One of the following may be coded: =, ≠, <, ≤, >, ≥, EQ, NE, LT, LE, GT, or GE.

COUNT=(,,*opfield2*) or COUNT=(,,*integer*) specifies the comparand to be used in evaluating the expression. If neither parameter is coded, the comparand defaults to 0. The comparand must be a decimal or hexadecimal integer not to exceed 16,777,215.

If COUNT= is not specified, execution is determined by the *conchars* operand.

The IEDSTOP macro causes operator control code to be executed. Each time the macro is executed, an attempt is made to pass an operator control element to the operator control task, informing it of the status of the station for which an error has been detected and what operator control action is required. This function requires the allocation of buffers from the buffer unit pool. When the operator control task is unavailable, allocated buffers are accumulated and are therefore unavailable for use by other functions. You can minimize the accumulation of buffers by using the COUNT= operand to control the frequency of IEDSTOP execution.

```
INTVL=integer
```

Function: Specifies the number of seconds to elapse before the external LU is to be restarted.

Format: Positive decimal integer.

Default: None. Specification is optional.

Maximum: 43,199.

```
mask
```

Function: Specifies the 5-byte bit configuration used to test the message error record for the message. (The message error record is described in Appendix A.)

Format: Unframed decimal integer or hexadecimal field. If hexadecimal format is used, framing characters must be specified. If X″ is used leading zeros must be coded. If XL5″ is used, leading zeros may be omitted.

Default: None. Specification is optional.

Note: Omitting the operand or specifying an all-zero mask causes execution of IEDSTOP based on whether the COUNT= operand is satisfied.

This operand is invalid if IEDSTOP is coded in the inheader or outheader subgroup.

```
UPDATE={ADD}
       {SUB}
       {NO }
```

Function: Specifies whether the option field specified as the comparator is to be incremented (ADD) by 1, decremented (SUB) by 1, or left unchanged (NO) before the expression is evaluated.

Format: ADD, SUB, or NO.

Default: UPDATE=ADD.

Return Codes

None.

IEDTCSD Macro

The IEDTCSD macro:

- Generates DSECTs and labels required by the INODEMH macro when INODEMH is coded outside of a model MCP
- Is not needed if a model MCP is being used.

This macro has no operands and generates certain DSECTs and labels. These DSECTs and labels are supplied automatically with the model MCPs; if you are modifying a model MCP, you must not code this macro. If you are coding your own MCP, code this macro after all executable code in the MCP and no more than once in the MCP.

DSECTs for the following control blocks and data areas are generated by the IEDTCSD macro. For descriptions of these data and control blocks, see the *TCAM Program Reference Summary*

Address Vector Table
Binary Search Routine Answer Area
Binary Search Table Control Block
Bind Exit Parameter List
Buffer Prefix
Data Control Block
Dynamic Accounting Control Block
Dynamic Accounting Record
Exception Request Table Entry
Fixed Header Prefix
Line Control Block
NEWMSG Macro Parameter List
NEWMSG Option Field
Queue Control Block
Routing Key Table Entry
SENDMSG User Exit Parameter List
Station Control Block
TCSOPTS Option Field
TCSSORT Table Definition Control Block
Terminal Name Table
Terminal Table Entry
THRESH Option Field
Unbind Exit Parameter List

If this extended networking feature is present, the following DSECTs will also be generated:

DKJDR Routine Parameter List
INODFLAG Option Field
ISNS Message Prefix
Node Table Entry
Resource Identifier Table Entry
RESTABLE Macro Parameter List
Session Information Block

Supported Resources and General Requirements: Not applicable.

Valid Subgroups: Not applicable.

NAME	OPERATION	OPERANDS
	IEDTCSD	(no operands)

Return Codes

None.

INBLOCK Macro

The INBLOCK macro:

- Identifies the beginning of an inblock subgroup
- Tests a path switch to allow alternative courses of action
- Is required if logical messages are to be handled by this MH, or if a character string specified on MSGEDIT may cross buffer boundaries
- If present, must precede the inheader subgroup (INHDR macro).

If an incoming group is needed, the INBLOCK macro, when used, must be the first macro following STARTMH; otherwise, the INHDR delimiter macro must be the first instruction following STARTMH.

The inblock subgroup may contain only one SETEOM functional macro. Other functional macros that may be included in this subgroup are listed below. Their positions relative to SETEOM determine whether they operate on all data in an entire incoming transmission sequence or whether they operate on one or more blocked or deblocked logical messages. Any of the following functional macros that are issued before the SETEOM macro act on all the incoming data in a physical transmission sequence, and any issued after SETEOM act on one or more of the resulting logical messages:

- CODE
- COUNTER
- LOCOPT
- LOG
- MSGEDIT
- PATH
- SETEOM
- TERRSET.

Supported Resources and General Requirements: ALL.

Valid subgroup: Inblock.

NAME	OPERATION	OPERANDS
`[symbol]`	`INBLOCK`	`[PATH=(opfield,switch)]`

`PATH=(opfield,switch)`

Function: Specifies conditional execution of this macro and its subgroup.

Format: (opfield,switch) opfield must conform to the rules for assembler language symbols, and must be the name of a 1-byte option field defined by an OPTION macro. *switch* may be either decimal or hexadecimal. If hexadecimal format is used, framing X'' characters must be specified.

Default: None. Specification is optional.

Maximum: 255 or a 1-byte hexadecimal field for *switch*.

Note: If any bits that are specified in the *switch* parameter are on in the path switch in the option field, the macro and its subgroup are executed. If this operand is not specified, the subgroup is executed unconditionally.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Return Codes

None.

INBUF Macro

The INBUF macro:

- Identifies a subgroup that handles incoming message buffers
- Tests a path switch to allow alternative courses of action
- Is optional in the incoming group.

INBUF identifies the beginning of an inbuffer subgroup, which contains instructions concerned with both header and text portions of incoming messages.

If the PATH= operand of INBUF is coded, INBUF examines a path switch in a field of the option table. If any of the bits specified by INBUF are on in the path switch, this subgroup is executed. If none of the bits specified by INBUF are on in the path switch, processing goes to the next subgroup. If INBUF does not specify an operand, this subgroup is executed unconditionally.

Supported Resources and General Requirements: ALL.

Valid subgroup: Inbuffer.

NAME	OPERATION	OPERANDS
`[symbol]`	`INBUF`	`[PATH=(opfield,switch)]`

`PATH=(opfield,switch)`

Function: Specifies conditional execution of this macro and its subgroup.

Format: (opfield,switch) opfield must conform to the rules for assembler language symbols, and must be the name of a 1-byte option field defined by an OPTION macro. *switch* may be either decimal or hexadecimal. If hexadecimal format is used, framing X'' characters must be specified.

Maximum: 255 or a 1-byte hexadecimal field for *switch*.

Note: If any bits that are specified in the *switch* parameter are on in the path switch in the option field, the macro and its subgroup are executed. If this operand is not specified, the subgroup is executed unconditionally.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Return Codes

None.

INEND Macro

The INEND macro:

- Identifies the end of the incoming group to an MH
- Is required as the last macro of any incoming group.

INEND identifies the end of the instruction sequence that processes incoming messages. One and only one INEND macro is required for each MH with an incoming group, and it must be the last macro in the incoming group.

Supported Resources and General Requirements: ALL.

Valid subgroup: Incoming group.

NAME	OPERATION	OPERANDS
[symbol]	INEND	(no operands)

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Note: If an INMSG subgroup is included in the incoming group, no executable code is generated by the INEND macro; therefore, branching to location *symbol* is not permitted.

Return Codes

None.

INHDR Macro

The INHDR macro:

- Identifies the beginning of an inheader subgroup
- Tests a path switch to allow alternative courses of action
- Is required as the first macro of any incoming group if the INBLOCK macro is not included.

INHDR identifies the beginning of an inheader subgroup in which the functional macros are concerned only with incoming header segments. If the inblock subgroup is not required, an inheader subgroup must be the first subgroup in the incoming group. If MSGEDIT is used across buffer boundaries or if incoming logical messages are being handled by this MH, an inblock subgroup must be the first subgroup in the incoming group. Text segments are passed to the first inbuffer subgroup.

An incoming message segment is tested by INHDR to determine whether it is a header or text segment; the first segment of any message is always considered to be a header segment. If it is a text segment or a canceled message, the segment is passed to the next subgroup; if it is a header segment, the inheader subgroup is executed.

If the PATH= operand of INHDR is coded, INHDR examines a one-byte path switch in a field of the option table. If any of the bits specified by INHDR are on in the path switch, this subgroup is executed. If none of the bits are on, control is directed to the next subgroup. If INHDR does not specify an operand, this subgroup is executed unconditionally. For a more complete description of the path switch and its function, see the chapter of the *TCAM Installation Guide* titled "Coding the Message Handler."

Supported Resources and General Requirements: ALL.

Valid subgroup: Inheader.

NAME	OPERATION	OPERANDS
[symbol]	INHDR	[PATH=(opfield,switch)]

`PATH=(opfield,switch)`

Function: Specifies conditional execution of this macro.

Format: (opfield,switch) opfield must conform to the rules for assembler language symbols, and must be the name of a 1-byte option field defined by an OPTION macro. *switch* may be either decimal or hexadecimal. If hexadecimal format is used, framing X'' characters must be specified.

Default: None. Specification is optional.

Maximum: 255 or a 1-byte hexadecimal field for *switch.*

Note: If any bits that are specified in the *switch* parameter are on in the path switch in the option field, the macro and its subgroup are executed. If this operand is not specified, the subgroup is executed unconditionally.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Return Codes

None.

INMSG Macro

The INMSG macro:

- Identifies the beginning of an inmessage subgroup
- Tests a path switch to allow alternative courses of action
- Is required as the first macro in an inmessage subgroup
- Is optional in the incoming group.

INMSG identifies the beginning of an inmessage subgroup. The functional macros associated with this subgroup are executed after an entire message or block has entered the system. Inmessage subgroups are specified after other subgroups in the incoming group. No user-written code should be included in an inmessage subgroup or between such subgroups.

If the PATH= operand of INMSG is coded, INMSG examines a path switch in a field of the option table. If any of the bits specified by INMSG are on in the path switch, this subgroup is executed. If INMSG does not specify an operand, this subgroup is executed unconditionally. Only one inmessage subgroup per message can be executed.

INMSG causes empty buffer units at the end of a buffer handled by this message handler to be deallocated before the contents of the buffer are queued for a destination. Deallocated units are returned to the available-unit queue. When the inmessage subgroup is not included in a message handler, this deallocation function is performed by the INEND macro.

Supported Resources and General Requirements: ALL.

Valid subgroup: Inmessage.

NAME	OPERATION	OPERANDS
[symbol]	INMSG	[PATH=(opfield,switch)]

PATH=(opfield,switch)

Function: Specifies conditional execution of this macro and its subgroup.

Format: *(opfield,switch) opfield* must conform to the rules for assembler language and must be the name of a 1-byte option field defined by an OPTION macro. *switch* may be either decimal or hexadecimal. If hexadecimal format is used, framing X'' characters must be specified.

Default: None. Specification is optional.

Maximum: 255 or a 1-byte hexadecimal field for *switch*.

Note: If any bits that are specified in the *switch* parameter are on in the path switch in the option field, the macro and its subgroup are executed. If this operand is not specified, the subgroup is executed unconditionally.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is optional.

Return Codes

None.

INODEMH Macro

The INODEMH macro:

- Generates portions of four major subgroups of an internodal message handler in an extended network
- Generates portions of inheader and inmessage subgroups which process messages received from other TCAM systems and handles error conditions relating to lost, sequence-checked traffic
- Generates portions of outheader and outmessage subgroups which process messages being transmitted to other TCAM host nodes and handle alternate path selection in the event of errors
- Is used in defining an extended network.

The INODEMH macro has four distinct functions determined by the GEN= operand. In constructing your internodal message handler, you must issue the macro once in each of the four major subgroups of the MH: inheader (GEN=INHDR), inmessage (GEN=INMSG), outheader (GEN=OUTHDR), and outmessage (GEN=OUTMSG). In addition, the macro must be issued (GEN=SUPPORT) in a nonexecutable section of the MCP to generate several support subroutines used by the four MH subgroups.

INODEMH processing depends on user-specified data in the INODEID, INODFLAG, INODPACE, ALTDEST, TCSOPTS, and VIANODE option fields for TERMINAL macros defining, internodal destination queues. The option fields must be initialized (see chapter 3 for details).

If the INODEMH macro is coded outside of a model MCP, the IEDTCSD macro must be coded to generate certain DSECTs required by the INODEMH macro.

Note: The extended networking model MCP illustrates the use of the macro and may be consulted during the reading of the macro description.

The five forms of the macro are described separately in the following text.

INODEMH in the Inheader Subgroup

The main purpose of the code generated by the INODEMH macro is to process and perform further routing of messages received from other existing host nodes in the extended network. The messages processed by the code generated by the macro may be in either of two formats:

- Non-sequence-check messages, which consist of an FHP followed by data:

FHP	Data

- Sequence-checked messages which contain a 14-byte internodal sequence prefix (constructed by code generated by an INODEMH macro coded GEN=OUTHDR issued in the originating node), followed by an FHP and data:

Sequence Prefix	FHP	Data

The layout of the internodal sequence prefix is described by the NSPRIFXD DSECT.

INODEMH inheader subgroup functions include the following:

1. *Assuring no data loss for sequence-checked messages:* Messages with sequence prefixes are checked to assure that no messages are missing. If an incoming message's sequence number (from the prefix) matches the next-expected input sequence number for the appropriate internodal destination queue, the sequence prefix is removed from the message.

   If a higher number than expected is received, one or more messages have been lost, for which retransmission must be requested. Therefore, a retransmission request is scheduled (the code generated by the INODEMH GEN=INMSG form of the macro constructs and queues the retransmission request to the sending node's NODESYNC system service program).

2. *Guarding against looping and down-link failures:* The INODEMH macro checks for looping of messages being routed through the extended network.

   In addition, each time a message leaves a host node in the extended network, code generated by the GEN=OUTHDR form of the INODEMH macro places the node identifier of the current host node into the FHP prior to transmitting the message. This is done to allow INODEMH inheader code in any receiving host node to perform a test on its routing determination to guard against a situation called *down-link failure.*

3. *Performing normal routing based on the TCAM network address in the FHP:* This includes either placing the message on the destination queue for a host node, or placing it on the destination queue for an external LU or application program owned by the local host node.

INODEMH in the Inmessage Subgroup

The code generated by the INODEMH macro in the inmessage subgroup tests for the sequence-break error condition and, if necessary, generates a retransmission request.

The GEN=INMSG form of the macro should be issued directly after the CANCELMG macro.

INODEMH in the Outheader Subgroup

The code generated by the INODEMH macro in the outheader subgroup prepares an outgoing message for transmission to another extended network node. If the message is sequence-checked, an internodal sequence prefix is added and the TCSENDBL macro is issued to preserve the FHP, with the message data, when it is transmitted to the target node.

INODEMH in the Outmessage Subgroup

The INODEMH macro in the outmessage subgroup tests for and attempts to handle error conditions occurring in transmission to other host nodes. The type of error handling performed is based on specifications in the destination's INODFLAG and TCSOPTS options fields. Unless otherwise specified by you in one of these option fields, automatic alternate extended routing is performed when errors are detected. First, the internodal destination queue for the designated primary alternate destination (as specified by you in the ALTDEST option field) is examined. If it is not intercepted, transferred, or purged, if the queue threshold value has not been exceeded, the current internodal destination queue is placed in transfer mode and messages are routed to the queue named in the ALTDEST option field. If the primary alternate is not usable, the secondary internodal destination queue (as you have specified in the VIANODE option field) is examined; if it is usable, the current internodal destination queue is placed in transfer mode and messages are routed to the internodal destination queue named in the VIANODE option field.

Macro Formats

For ease of reference, the macro format is shown in five versions corresponding to the five forms of the GEN= operand with the additional operands that are applicable to each form.

Supported Resources and General Requirements: SNA, FHP.

Valid subgroups: Inheader, inmessage, outheader, and outmessage.

When used in INHEADER Subgroup

NAME	OPERATION	OPERANDS
[symbol]	INODEMH	GEN=INHDR ,DATABAD=name ,LOOP=name ,NODEBAD=name ,NODEDWN=name ,RESBAD=name ,RESDOWN=name ,SEQHI=name ,SEQLO=name [,DSECT={YES}] {NO }

When used in INMESSAGE Subgroup

NAME	OPERATION	OPERANDS

[symbol]	INODEMH	GEN=INMSG
		,SEQBAD=name
		[,DSECT={YES}]
		{NO }

When used in Outheader Subgroup

NAME	OPERATION	OPERANDS

[symbol]	INODEMH	GEN=OUTHDR
		,DATABAD=name
		[,DSECT={YES}]
		{NO }

When Used in Outmessage Subgroup

NAME	OPERATION	OPERANDS
[symbol]	INODEMH	GEN=OUTMSG ,AUTOTRN=name ,HOLD=name ,MASK=e ,NOERR=name ,OUTEND=name ,TRNFAIL=name ,USERERP=name [,DSECT={YES}] {NO }

When used in the MCP outside of an MH

NAME	OPERATION	OPERANDS

[symbol]	INODEMH	GEN=SUPPORT
		[,DSECT={YES}]
		{NO }

```
DSECT={NO }
      {YES}
```

Function: Generates the dummy section describing the format of the INODFLAG option field, and the internodal sequence prefix.

Format: YES or NO.

Default: DSECT=NO.

Note: If YES is specified, all other keywords are ignored and the dummy section describing the INODFLAG option field and the internodal sequence prefix is generated.

```
GEN={INHDR  }
    {OUTHDR }
    {INMSG  }
    {OUTMSG }
    {SUPPORT}
```

Function: Specifies the type of message-handler code to be generated by the macro.

Format: INHDR, OUTHDR, INMSG, OUTMSG, or SUPPORT.

Default: None. This operand is required unless DSECT=YES is coded.

Note: INHDR, OUTHDR, INMSG, and OUTMSG correspond to the MH subgroup in which the macro is issued. If SUPPORT is specified, the macro must be issued in a nonexecutable portion of the MCP.

GEN=INHDR operands:

DATABAD=name

Condition: There is insufficient data in the buffer to process.

User Action: Branch directly to an INMSG delimiter macro.

LOOP=name

Condition: Either the message was found to be looping, or a down-link failure was detected and the VIANODE option field either did not exist or the specified different extended route was not usable. The message has been scheduled for routing to the destination specified in the LOOPQ= operand of the INTRO macro. If LOOPQ= was not specified on INTRO, the message has not been scheduled for routing.

User Action: Refer to the model MCPs. Messages may be routed and held for later delivery, returned to the originator, or routed to LOOPQ depending on the strategy for the TC.

Use the NEWMSG macro to inform the originator that the message cannot be sent.

NODEBAD=name

Condition: The node identifier in the TDAF is invalid. (There is no entry in the node table corresponding to the node identifier.)

User Action: Use the NEWMSG macro to cancel the message and to inform the originator and the TCAM operator.

NODEDWN=name

Condition: The node table entry status information indicates that the next node on the extended route to the destination is inactive or that the internodal destination queue for this node is located in main storage without disk backup and the queue of unsent messages on this queue exceeds the value specified in the THRESH option field associated with it.

User Action: Refer to the model MCPs.

RESBAD=name

Condition: The resource identifier in the TDAF is invalid. (The node identifier in the TDAF indicates the resource is located on this local host node, but there is no entry in the resource table corresponding to the resource identifier.)

User Action: Use the NEWMSG macro to cancel the message and to inform the originator and the TCAM operator.

```
RESDOWN=name
```

Condition: The resource identifier in the TDAF indicates a resource in this TCAM system whose destination queue is located in main storage without disk backup. Either the number of unsent messages on the destination queue for this external LU or application is greater than the threshold value specified in its THRESH option field or the destination is an inactive application.

User Action: Use the NEWMSG macro to cancel the message and to inform the originator.

```
SEQHI=name
SEQLO=name
```

Conditions: A sequence-checked message was received. The sequence number was either higher or lower than expected. The INODEMH macro has internally noted the sequence-break condition (if the sequence number was high) and a retransmission request will be generated by the code associated with the GEN=INMSG form of the INODEMH macro. If the sequence number is low, this message is a duplicate and no action has been taken by INODEMH.

User Action: Branch directly to an INBUF or INMSG delimiter macro.

GEN=INMSG operands:

```
SEQBAD=name
```

Condition: A retransmission request has been generated and queued to the transmitting host node, based on an indication set previously during inheader processing.

User Action: Branch directly to an INEND delimiter macro.

GEN=OUTHDR operands:

```
DATABAD=name
```

Condition: There is insufficient data in the buffer to process.

User Action: Branch directly to an OUTMSG delimiter macro.

GEN=OUTMSG operands:

AUTOTRN=name

Condition: The MER as tested by the macro using MASK= operand indicates that error has occurred and automatic queue transfer has gone into effect. Code generated by the macro has turned on the transfer flag in the TCSOPTS option field for the internodal destination queue and redirected the message to either the primary alternate internodal destination queue (specified by the ALTDEST option field) or to the secondary alternate internodal destination queue (specified by the VIANODE option field).

User Action: Use SENDMSG to notify the network control operator that automatic transfer is in effect for the internodal destination queue.

HOLD=name

Condition: The MER as tested by the macro using the MASK= operand indicates that an error has occurred and the INODFLAG option field indicates that the internodal destination queue should be held.

User Action: Issue the HOLD macro for a given interval and use the SENDMSG macro to notify the network control operator of the event.

MASK=e

Function: Provides a 5-byte hexadecimal mask framed by X'' to be tested against the message error record (MER) (by using the MASK= operand) to determine if there were any errors in the transmission.

NOERR=name

Condition: The MER as tested by the macro using the MASK= operand indicates that no errors occurred in the transmission.

User Action: Clear the send-side error counter maintained in the internodal destination queue's THRESH option field and/or branch to an OUTEND delimiter macro.

OUTEND=name

Condition: The INODEMH macro has detected that the internodal destination queue is currently in transfer or purge mode. This operand provides the name of the next instruction to be executed if either condition exists. If transfer is in effect, the macro issues the REDIRECT macro prior to branching to the name.

User Action: None.

TRNFAIL=name

Condition: An error has occurred as tested by the macro using the MASK= operand and automatic queue transfer was attempted but neither the ALTDEST nor the VIANODE option fields contain the names of usable alternate internodal destination queues.

User Action: Use the SENDMSG macro to inform the originator that the message cannot be transmitted and to inform the network control operator of the error condition.

USERERP=name

Conditions: An error has been detected based on the MASK= operand error mask in the MER but the INODFLAG option field indicates that user ERP (error recovery procedures) will be provided.

User Action: Perform ERP as desired. If user ERP is not to be performed for any internodal destination queue, this keyword operand must specify the name of an OUTEND delimiter macro.

Return Codes

None.

INSRTFLD Macro

The INSRTFLD macro:

- Schedules the insertion of data into the outgoing message header
- May be issued more than once
- Requires the presence of an FHP in the header being processed
- Must be issued with the scan pointer located within the first unit of the first buffer containing the message.

The INSRTFLD macro is used within the inheader subgroup of an MH to *schedule* the insertion of specific data into the outgoing message header. Functionally, INSRTFLD converts the current scan pointer setting to a fixed header prefix (FHP) offset and saves this offset in the FHP field associated with the type of data requested. The actual insertion of the data into the message is performed when the TCSENDBL macro executes in the outheader subgroup.

The INSRTFLD macro requires that a fixed header prefix (FHP) be previously constructed for the message. If no FHP exists, INSRTFLD functions are terminated and a return is made to the user.

INSRTFLD schedules the insertion of data into the first unit only of the buffer containing the message.

If multiple data field requests are made for the identical scan pointer setting, the actual order of the inserted fields in the outgoing message as specified in the INSRTFLD macro has no effect.

If INSRTFLD is executed when the scan pointer is positioned at the first byte of message header data, the requested data appears as the first field or fields of the outgoing message. In all other cases, the requested data field or fields are inserted one position after the scan pointer setting when INSRTFLD is executed.

The scan pointer is never moved by this macro instruction.

Note: An overlay of message data can occur when the MSGEDIT macro is used to insert or remove message data *and* the INSRTFLD macro is also used to insert message data. The overlay condition occurs when MSGEDIT inserts or removes message data preceding the area used to contain the INSRTFLD data. This insertion or removal of message data causes the INSRTFLD pointers in the FHP to point to the wrong area of the message.

Supported Resources and General Requirements: ALL, FHP.

Valid subgroups: Inheader.

NAME	OPERATION	OPERANDS
symbol	INSRTFLD	DATA=(fieldname,...)

`DATA=(fieldname,..)`

Function: Specifies the data field or fields to be scheduled for insertion.

Format: fieldname must conform to the rules for assembler language symbols. (If more than one field name is specified, each name but the last must be followed by a comma, and the entire suboperand must be framed by parentheses.)

Default: None. Specification is required.

The data fields must be named as follows:

Data scheduled	**Fieldname**
Output sequence number	SEQOUT
Input date stamp	DATEIN
Input time stamp	TIMEIN
Output data stamp	DATEOUT
Output time stamp	TIMEOUT

The insertions take place in the predetermined order (as shown above) regardless of the sequence in which the words are entered in this operand, if multiple data field requests are made for the same scan pointer setting.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Examples

Example 1 below shows the normal scheduling of fields to be inserted; example 2 shows scheduling to force the date and time expression to appear preceding the output sequence number in the outgoing message.

Example 1:

```
Inheader subgroup code:
.
.
.
DESTFLD
INSRTFLD     DATA=(TIMEIN,DATEIN,SEQOUT)
.
.
.
```

Incoming message:

:
LA 10 SF
LA - Destination

Outgoing message:

LA 22 101584/1258 10 SF
LA - Destination
22 - Output sequence number
101584/1258 - Input date and time expression

Example 2:

```
Inheader subgroup code:
.
.
.
DESTFLD
INSRTFLD    DATA=(TIMEIN,DATEIN)
SETSCAN     1,BLANK=NO
INSRTFLD    DATA=(SEQOUT)
.
.
.
```

Incoming message:

LA 10 SF
LA - Destination
Outgoing message:
LA 101584/1258 22 10 SF
LA - Destination
101584/1258 - Input date and time expression
22 - Output sequence number

Return Codes

One of the following return codes is set in register 15:

Code	Meaning
X'00000000'	Successful execution.
X'00000010'	No FHP in the message buffer.

INTRO Macro

The INTRO macro:

- Is the first TCAM macro in an MCP
- Creates the address vector table (the primary control block in the TCAM system)
- Performs the bulk of TCAM system initialization
- Creates the exit list that contains the addresses of TCAM's asynchronous exit routines
- Establishes addressability and entry linkages for the message control program
- Specifies the name of the message control program
- Specifies the number of channel program blocks to be provided for transferring data between buffer units and queues maintained on disk
- Specifies the maximum number of command input blocks that may be used at any one time to contain operator commands entered at the system console
- Identifies the basic and extended primary operator control stations
- Specifies a character string to identify operator commands
- Specifies the size of buffer units
- Specifies the maximum number of units that may be assigned to a main-storage message queue
- Provides you with a means of determining when your main-storage message queue is nearly full, and when this condition has abated
- Identifies the external LU or application program to which messages having an invalid destination are to be forwarded
- Specifies which user registers are to be saved when in-line user code is located in an inheader or outheader subgroup that may handle multiple-buffer headers
- Specifies the interval between environment checkpoints
- Specifies the number of environment records to be retained at any one time
- Provides system optimization by specifying that unnecessary options are to be omitted
- Specifies the type of restart to be performed following system closedown or failure
- Specifies a mask to indicate protected terminal-table fields
- Specifies that a copy of responses to certain extended operator commands that alter system status should be sent to a specified destination
- Creates the TCAM vector table (TVT)
- Builds a table definition control block defining the TCAM terminal name table and places its address into the TVT; the TCSEARCH macro can then be used to search the terminal name table
- Specifies the node identifier of this MCP if it is part of an extended network
- Specifies the new-line symbol to be used by internal routines that generate messages exceeding a single line of output
- Specifies parameters governing detection of message loops in an extended network, and names a destination for messages that are looping among host nodes
- Specifies whether a TCAM is ASCB-based or CVT-based
- Specifies whether message IED159 is to be retained on the operators console
- Specifies when or if the Access Method Interface is to be started automatically (at each startup, on warm start only, on cold start only or not on any startup)
- Specifies the number of entries in the Access Method Interface trace table
- Specifies the password that will be used on the Access Method Control Blocks that are opened for the host LUs
- Specifies the number of external LUs that can be in session concurrently with TCAM host LUs.

At the time the INTRO macro is executed, it may cause the following WTOR message to appear on the system console:

```
nn   IED002A SPECIFY TCAM PARAMETERS.
```

This WTOR message is issued if at least one of the following operands is omitted from the INTRO macro: STARTUP=, UNITSZ=, LNUNITS=, or (if DISK=YES is coded on INTRO) CPB=. If the WTOR message IED002A is displayed, its first appearance is preceded by another message.

```
IED001I    TCAM JOB jobname, stepname, procstepname ADDRESS
           OF AVT xxxxxxxx
```

This AVT address may be used by the operator to display (or to modify, according to your instructions) areas of the message control program during the wait for the WTOR reply. These displays and modifications cannot be done via TCAM operator control commands. They must be done via commands to the operating system (MVS) or to the debugging monitor under which TCAM is running. After the TCAM system issues the WTOR message, it waits for a user response to be entered at the system console. The user has two options in responding: enter either response keywords (as shown in the *response keyword* line in the list of INTRO operands) or INTRO operands, together with appropriate values. Several keywords or operands may be coded in one response. Keywords or operands coded in a response are separated by commas or vertical bars (|). Responses may be entered in upper or lowercase letters. They are automatically translated into uppercase. Each response is limited to 41 characters.

After a response has been entered, TCAM reissues the WTOR message and continues to issue it after each response is entered until the operator indicates that he has nothing more to specify; he does this by entering U at the end of a response. If he enters U and has not yet specified values for STARTUP=, UNITSZ=, LNUNITS=, or (if DISK=YES is specified on the INTRO macro) CPB=, either on the INTRO macro or in a response to the WTOR message, TCAM prompts him with the following message:

```
nn    IED004A REQUIRED PARAMETER MISSING.  SPECIFY operand
```

where *operand* is the name of the missing INTRO operand.

An error in specifying a response keyword or operand (such as an invalid response keyword or operand or an invalid value for either) causes an error message to be printed at the console; the operator may respecify the response keyword or operand when he receives such a message. An error in one response keyword or operand prevents interpretation of any keywords in the same response to the right of the keyword in error. A response keyword or operand may be coded more than once in a sequence of WTOR responses; the latest value specified applies.

Example:

The following WTOR messages and responses occur at INTRO execution if you omit the STARTUP= and LNUNITS= operands from the INTRO macro. The operator specifies LNUNITS=, MSMIN=, MSMAX=, CPRCDS=, and CONTROL= but forgets to specify STARTUP= (a required operand) and is prompted for this operand:

message:	00 IED002A SPECIFY TCAM PARAMETERS
response:	r 00,'B=2,MSMIN=80,X=95,E=5'
message:	01 IED002A SPECIFY TCAM PARAMETERS
response:	r 01,'CONTROL=OPCONT,U'
message:	02 IED004A REQUIRED PARAMETER MISSING. SPECIFY STARTUP=
response:	r 02,'s=c,u'.

If the NEWMSG, DESTFLD, or SENDMSG macro is issued in a message handler, the DLQ operand must be specified on the INTRO macro or must be entered as a parameter at TCAM initialization. If the DLQ operand is not specified, the following WTO message is written to the system console:

```
IED659I VALID DLQ NOT SPECIFIED - RESTART MCP AND USE Q= RESPONSE
TO IED002A TO SPECIFY
```

and a return code of X'2C' is returned to the MCP to indicate that the NEWMSG, DESTFLD, or SENDMSG macro was issued without a dead letter queue being specified.

Note: If no response keyword is shown for a particular operand, the value for that operand may not be specified when the INTRO macro is executed.

If you wish to execute user-written code before the INTRO macro is executed, this code must not contain any TCAM macros. In the case where you must execute some routine before the INTRO macro is executed, the message control program (with the INTRO macro as the first TCAM macro) should be called as a subroutine of the inserted user code with register 15 containing the address of the INTRO macro, register 14 containing the address to which the MCP returns upon termination of TCAM, register 13 containing the address of a standard 18-word save area, and register 1 containing the parameter-list pointer as originally passed in register 1 from the TCAM initiator.

Register 1 contains the address of a word which points to an area containing the 'parm' string specified on the PARM operand of TASKDEF macro which defined the MCP. The first two bytes of the area pointed to by this word contain the length of the string (in bytes). The 'parm' string text follows these two bytes.

If no PARM operand was specified on the MCP TASKDEF macro, the word pointed to by register 1 is zero.

After INTRO execution, the address passed in register 1 is stored in AVTSPLPT.

Supported Resources and General Requirements: Not applicable.

Valid subgroup: Not applicable.

INTRO

NAME	OPERATION	OPERANDS
[symbol]	INTRO	LNUNITS=integer
		,STARTUP={C}[{Y}][I]
		{W} {E}
		{N}
		{,UNITSZ=integer}
		{,KEYLEN=integer}
		[,ACCLIM=n]
		[,AMIPASS=password]
		[,AMISTRT={AUTO }]
		{NOAUTO }
		{COLDAUTO}
		{WARMAUTO}
		[,AMITRCE={integer}]
		{500 }
		[,APDUMP={YES}]
		{NO }
		[,APWAS={184 }]
		{440 }
		{952 }
		{1976 }
		{4024 }
		{integer}
		[,AUTHA={YES}]
		{NO }
		[,BASED={ASCB}]
		{CVT }
		[,BFRRTN={HI}]
		{LO}
		[,BUFFTR=({integer},{FULL},{U})]
		{0 } {PAR } {C}
		[,CIB={integer}]
		{2 }
		[,CKREQS={integer}]
		{0 }
		[,CONTROL={characters}]
		{0 }
		[,CPB={integer}]
		{0 }
		[,CPINTVL={integer}]
		{1800 }

NAME	OPERATION	OPERANDS

	INTRO (Cont'd)	[,CPRCDS={integer}]
		{2 }
		[,DISK={YES}
		{NO }
		[,DLQ={entry}]
		{0 }
		[,DTRACE={integer }]
		{(integer,option)}
		{500,POST }
		[,LOOPCNT={n }]
		{254}
		[,LOOPQ=destination]
		[,MANUAL={ON }]
		{OFF}
		[,MSMAX={integer}]
		{70 }
		[,MSMIN={integer}]
		{50 }
		[,MSUNITS={integer}]
		{0 }
		[,NETMON={termname }]
		{KEY(keyname)}
		{SYSCON }
		[,NEWLINE={X'2615'}]
		{X'15' }
		[,NODEID={n}]
		{0}
		[,OPMASK={(hexadecimal integer,...)}]
		{(FCFC,FCFC,C000,FC0C) }
		[,PASSWRD=characters]
		[,PRIMARY={termname}]
		{SYSCON }
		[,PROGID={characters}]
		{IEDIMCP }
		[,RESTART={integer}]
		{0 }
		[,REUSMSG={RETAIN }]
		{NORETAIN}
		[,USEREG={integer}]
		{0 }

Note: The operands may be specified in any order according to assembler language conventions.

The operands for the INTRO macro are described in the following list. The list of operands for the INTRO macro also shows the response keywords that may be substituted for the operand names in responses to the WTOR message SPECIFY TCAM PARAMETERS sent to the system master console at INTRO execution.

ACCLIM=n

Response Keyword: AL=.

Function: Specifies the *access limit* which is the number of external LUs that can be is session concurrently with TCAM host LUs.

Format: *n* where *n* is an unframed integer decimal format.

Default: The number of external LUs defined in the MCP. If ACCLIM=0 is coded, then default is taken.

Maximum: 32,767.

Note: This operand *may* be specified at TCAM startup.

```
AMIPASS=password
```

Response Keyword: None.

Function: Specifies the password that will be used on ACBs that are opened for the host LUs.

Format: 1 to 8 EBCDIC characters.

Default: None. Specification is optional.

Notes:

1. *This operand may not be specified at TCAM startup.*
2. *If password protection is desired, the PRTCT parameter on the VTAM APPL definition statement must specify the same value as this operand.*

```
AMISTRT={AUTO     }
        {NOAUTO   }
        {COLDAUTO}
        {WARMAUTO}
```

Response Keyword: AS=.

Function: Specifies when or if the Access Method Interface (AMI) is to be started automatically.

Format: AUTO, NOAUTO, COLDAUTO, or WARMAUTO.

Default: AMISTRT=AUTO.

Note: AUTO or NOAUTO may be specified at TCAM startup.

AUTO specifies that the AMI will be started automatically on any startup of TCAM.

NOAUTO specifies that the AMI will *not* be started automatically an any startup of TCAM.

COLDAUTO specifies that the AMI will be started automatically on an initial start or cold restart of TCAM but not on a warm or continuation restart.

WARMAUTO specifies that the AMI will be started automatically on a warm or continuation restart of TCAM but not on an initial start or cold restart.

```
AMITRCE={integer}
        {500    }
```

Response Keyword: AT=.

Function: Specifies the number of entries in the Access Method Interface trace table.

Format: *n* where *n* is an unframed integer in decimal format.

Default: AMITRCE=500.

Minimum: AMITRCE=100.

Maximum: AMITRCE=32,767.

Note: This operand may be specified at TCAM startup.

```
APDUMP={YES}
       {NO }
```

Response Keyword: None.

Function: Specifies whether the 046 dump for an application program executing as a separate job is to be printed when the MCP terminates processing abnormally.

Format: YES or NO.

Default: APDUMP=YES.

Note: This operand is not applicable for attached application programs.

```
APWAS={ 184   }
      { 440   }
      { 952   }
      {1976   }
      {4024   }
      {integer}
```

Response Keyword: WA=.

Function: Specifies the size of the interface work area used to transfer data between the application program and the MCP. The size must be equal to or greater than the largest work area for any TCAM application-program function or macro system. See the discussion of the BLKSIZE= operand on the "Input DCB Macro" and "Transferring Data between TCAM and an Application Program" in the *TCAM Application Programming* publication for more information regarding work areas.

Format: 184, 440, 952, 1976, 4024, or any integer between 4025 and 32,695 in unframed decimal format.

Default: APWAS=4024.

Maximum: 32,695.

Notes:

See the particular TCAM application-program function or macro in TCAM Application Programming for the return code if APWAS= is specified too small. Specifying 184, 440, 592, 1976, or 4024 gives optimum performance and utilization of system resources. Do not specify an integer between 4025 and 32,695 unless absolutely necessary.

This work area is dynamically acquired and released in the common storage area as needed. The number of work areas acquired depends upon the number of concurrent TCAM application functions or macros coded and will not be freed until TCAM termination.

```
AUTHA={YES}
      {NO }
```

Response Keyword: AU=.

Function: Specifies whether all TCAM application programs must be authorized; that is, link-edited in an authorized library.

Format: YES or NO.

Default: AUTHA=YES.

Note: If AUTHA=YES is specified or this operand is omitted, and the application program is *not* authorized, the application program is abnormally ended with a system 043-7 ABEND code at open.

```
BASED={ASCB}
      {CVT }
```

Response Keyword: BA=.

Function: Determines which MVS control block is to be used to establish TCAM addressability.

Format: ASCB or CVT.

Default: BASED=CVT.

Note: CVT specifies that this TCAM will use the field in the CVT. An attempt to start a second TCAM with BASED=CVT coded will result in an error.

ASCB specifies that this TCAM will not use the field in the CVT. Multiple ASCB-based TCAMs can be started concurrently.

```
BFRRTN={HI}
       {LO}
```

Response Keyword: BB=.

Function: Specifies whether TCAM buffers being returned to the buffer pool are placed on the top or the bottom of the queue of free buffers.

Format: HI or LO.

Default: BFRRTN = LO.

Note: HI specifies the top of the queue, while LO specifies the bottom of the queue. If LO is specified, then the peak utilization count of buffers, MSUNITS, and LNUNITS is not accurate when a display is requested (Display Pool Statistics basic operator command).

```
BUFFTR=({integer},{FULL},{U})
        {0      } {PAR } {C}
```

Response Keyword: BF = .

Function: Specifies a buffer trace table within TCAM's storage region, provides the option of tracing the entire buffer contents or one buffer unit only, and provides the option of tracing from STARTMH for all buffers or for only buffers associated with a station for which the AMI trace is active.

Format: *integer* is an integer of 0 to 200 that specifies the number of trace-table entries for which TCAM should reserve space. Each entry must be large enough to contain status information and the contents of one TCAM buffer unit. FULL causes the trace routine to trace the entire contents of the buffer, while PAR causes the trace routine to trace the contents of the first unit only. FULL is effective only if a positive integer is coded in the *integer*. U causes the trace entry taken at STARTMH to occur for every buffer if IEDQFE30 is loaded, while C causes STARTMH to trace only if the station associated with the buffer has the AMI trace active.

Default: BUFFTR = (0,PAR,C).

Maximum: 200.

integer is an integer from 0 to 200 that specifies the number of entries in the trace table. The size of each entry depends upon the size of the buffer units being used by the MCP, as specified via the UNITSZ = (or KEYLEN =) operand of the INTRO macro. Each entry is large enough to contain the contents of one buffer unit plus associated status information. This entry size is then rounded up to the nearest multiple of 32.

If no positive value for *integer* is specified via the BUFFTR = operand, then the buffer trace table is located in the IEDQFE30 service aid routine in the storage region occupied by the COMWRITE Service Aid Writer Task, rather than in the MCP itself. The IEDQFE30 routine is loaded via the Start TCAM Service Aid Routine operator command. The table in the COMWRITE storage region consists of twenty entries, each of which is 96 bytes long.

FULL causes the buffer trace routine to copy the entire contents of a buffer being traced into as many trace table entries as are required to contain it. PAR causes the buffer trace routine to copy into a trace table entry only the first unit of each buffer being traced. If no positive integer is coded for the *integer* suboperand, then the PAR option is in effect even if FULL is coded; in this case, if the entire unit plus status information will not fit into one of the 96-byte entries in the trace table located in the COMWRITE storage region, then the data in the last part of the buffer unit is not recorded in the trace.

U causes buffer contents to be traced at the initiation of MH processing for all buffers being processed within the MCP, while C causes buffer contents to be traced at the initiation of MH processing only if the origin (for an incoming message) or the destination (for an outgoing message) has an AMI trace currently active. To trace buffer contents at the initiation of MH processing, the COMWRITE subtask must be active and the IEDQFE30 service aid routine must be loaded into the COMWRITE storage region via the Start TCAM Service Aid Routine operator command.

The FULL, PAR, U, and C suboperands may be modified dynamically by means of the Change Buffer Trace Parameters operator command.

In order for buffer contents to be recorded at the initiation of MH processing, three conditions must be met:

1. The COMWRITE subtask must be defined in the TCAM subtask table (TST) and must be attached by the initiator.
2. The IEDQFE30 service aid routine must be loaded via the Start TCAM Service Aid Routine operator command.
3. If U is not specified as a suboperand of the BUFFTR= operand of INTRO, buffer contents are recorded only for buffers going to or coming from an external LU for which an AMI trace is currently active.

In order for buffer contents to be recorded at some particular point during processing by an inheader, outheader, inbuffer, or outbuffer subgroup of an MH, the IEDBUFTR macro must be executed in the MH at that point. Execution of the IEDBUFTR macro is not effective unless a nonzero integer is specified for the *integer* suboperand of the BUFFTR= operand of INTRO (causing the trace table to be located in the MCP's storage region).

```
CIB={integer}
    {2      }
```

Response Keyword: C=.

Function: Specifies the maximum number of command input blocks (CIBs) that can be used at any one time in the TCAM system.

Format: Unframed decimal integer greater than zero.

Default: CIB=2.

Maximum: 255.

Note: CIBs are buffer-like areas used to contain operator commands entered at the system console. Space for them is allocated dynamically when needed, and the main storage assigned to a CIB is freed once the operator command contained within the CIB has been processed. Only one CIB need be specified for operator commands entered from the system console. However, more than one CIB should be specified if you anticipate attempting to simultaneously process more than one operator command from the console. If an attempt is made to enter an operator command from the system console when the number of CIBs specified by the CIB= operand is already present in the system (because that many operator commands from the system console are now being processed), the message being entered is rejected by TCAM.

```
CKREQS={integer}
       {0      }
```

Response Keyword: R=.

Function: Specifies the maximum number of destination queues that may be in use at any time for application programs that include a CKREQ macro.

Format: An unframed decimal integer.

Default: CKREQS=0.

Maximum: 255.

Note: This operand specifies the number of checkpoint request records to be set up in a checkpoint data set.

If an attempt is made to increase or to decrease the *integer* during a point-of-last-environment (POLE) or point-of-failure (POF) restart, the smaller value prevails.

```
CONTROL={characters}
        {0         }
```

Response Keyword: L=.

Function: Specifies the character string used to identify each operator command as such to TCAM.

Format: One to eight unframed characters with no embedded commas or blanks.

Default: CONTROL=0.

Note: CONTROL=0 indicates that no character string is being specified and is valid only if all operator commands are to be entered at the system console.

```
CPB={integer}
    {0      }
```

Response Keyword: D=.

Function: Specifies the number of channel program blocks to be provided for transferring data between buffer units and message queues maintained on disk.

Format: Unframed decimal integer.

Default: CPB=0.

Maximum: 65,535.

Minimum: 5.

Note: One CPB is involved in transferring the data in one unit to disk, or in filling one unit with data from disk. See the *TCAM Installation Guide* for more details.

CPB= must be specified if DISK=YES is coded. This operand is ignored if DISK=NO is coded and may be omitted in this case. If DISK=NO is coded, this operand is ignored by INTRO and both CPB= and D= are invalid responses at INTRO execution.

```
CPINTVL={integer}
        {1800   }
```

Response Keyword: V=.

Function: Specifies the maximum number of seconds between environment checkpoints when the TCAM checkpoint/restart facility is used.

Format: An unframed decimal integer greater than 29.

Default: CPINTVL=1800.

Maximum: 43,199.

Note: If the checkpoint/restart facility is specified, environment records are taken automatically after the time interval specified in CPINTVL= has elapsed. If you are synchronizing TCAM checkpoints with OS/VS checkpoints of the application program, specify the same time interval in this operand as in the application program.

```
CPRCDS={integer}
       {2      }
```

Response Keyword: E=.

Function: Specifies the number of environment records to be retained the checkpoint data set at any time.

Format: An unframed decimal integer greater than 1.

Default: CPRCDS=2.

Maximum: 75.

Note: The most recent records are the ones retained. For example, if CPRCDS=2 is specified, the most recent two environment checkpoints are kept in the checkpoint data set. When a new environment checkpoint is taken, its record overlays the oldest environment record then being held in the data set. If an attempt is made to increase or decrease *integer* during a POLE or POF restart, the smaller value prevails. Guidelines for coding this operand are included in the chapter of the *TCAM Utilities* manual titled "Checkpoint/Restart Service Facility."

```
DISK={YES}
     {NO }
```

Response Keyword: None.

Function: Specifies whether any of the message queue data sets defined for this MCP are located on a direct-access secondary-storage device.

Format: YES or NO.

Default: DISK=YES.

Note: DISK=YES is coded if any of the message queue data sets are located on disk. DISK=NO is coded if no message queue data sets are located on disk. If DISK=NO is coded, then MSUNITS= must specify a non-zero integer.

For further information, see the chapter "Defining Data Sets" in the *TCAM Installation Guide.*

```
DLQ={entry}
    {0    }
```

Response Keyword: Q=.

Function: Specifies the name of the dead-letter queue to which messages with invalid destinations are sent.

Format: *entry* is the name of a external LU or application as defined by a TERMINAL or TPROCESS macro. DLQ=0 specifies that no dead-letter queue is to be used.

Default: DLQ=0.

Note: Dead-letter messages are messages having invalid destination as determined by a FORWARD macro. If a user-specified routine is coded for the EXIT= operand of the FORWARD macro, messages with invalid destinations may have the destination corrected. If both the DLQ= operand of INTRO and the EXIT= operand of FORWARD are omitted, dead-letter messages are overlaid and lost.

This operand is required if the DESTFLD, NEWMSG, or SENDMSG macro is coded in the MCP.

```
DTRACE={integer         }
       {(integer,option)}
       {500,POST        }
```

Response Keyword: A=.

Function: Specifies the number of entries in the TCAM dispatcher trace table and specifies whether the subtask trace facility is used.

Format: integer is an unframed decimal integer. *option* represents the subtask trace facility and should be replaced by ALL, OFF, ON, POST, or STCBEX.

Default: DTRACE=(500,POST).

Maximum: 65,535 for *integer*.

Note: The dispatcher trace table is a debugging aid that keeps a sequential record of subtasks activated by the TCAM dispatcher. This table is discussed in the chapter of *TCAM Diagnosis Guide* titled "TCAM Diagnostic Aids". One entry is created for each activated subtask; when the end of the table is reached, the table is wrapped and new entries overlay the oldest entries.

The *option* suboperand can be ALL, OFF, ON, POST, or STCBEX. ALL specifies that all traces are to be activated. OFF specifies that traces activated by ON, POST, or STCBEX are to be terminated. ON and POST specifies normal subtask trace plus a trace of the elements as they are placed on the ready queue. STCBEX specifies extended STCB trace with the normal subtask trace and tracing of elements posted on the ready queue.

If *integer* is zero, the *option* default is OFF. However, if *integer* is not equal to zero, the default is POST. ON or POST cannot be specified if *integer* is zero.

The *integer* operand can be changed at startup by using the response keyword A= format. The *option* suboperand cannot be changed, but is reset if A=0 is specified.

```
KEYLEN=integer
```

See the UNITSZ= operand for the description of this operand.

```
LNUNITS=integer
```

Response Keyword: B=.

Function: Specifies the number of buffer units that may be used to build buffers to contain incoming and outgoing message segments.

Format: Unframed decimal integer greater than zero.

Default: None. This operand is required.

Maximum: 65,535 minus the value on the MSUNITS= operand.

Note: This value is added to that for MSUNITS= to determine the total number of units in the buffer unit pool.

```
LOOPCNT={n  }
        {254}
```

Response Keyword: None.

Function: Specifies the maximum number of times a message can be routed on utility sessions between host nodes in an extended network before internodal message handler code presumes that an infinite loop condition exists, and branches to the location specified by the LOOP= operand of the INODEMH macro.

Format: An explicit decimal integer from 1 through 254.

Default: LOOPCNT=254.

Note: The LOOPCNT= operand also pertains to internal looping due to transfer of a message to an alternate destination; when such looping is detected, the message is routed to the destination specified by the LOOPQ= operand.

```
LOOPQ=destination
```

Response Keyword: None.

Function: Specifies the destination name for messages found by the Internodal MH to be in a loop between destination queues as a result of being processed multiple times by REDIRECT macros specifying EXIT=DKJAZX. When the number of times the message has been redirected exceeds the value specified on the LOOPCNT= operand of the INTRO macro, the message is routed to the destination specified by LOOPQ=.

This operand is valid only if the extended networking facility is being used.

Format: destination is the name of a single or cascade-type TLIST entry in the terminal table.

Default: None. Specification is optional. If not specified, messages found to be in a loop condition continue to loop.

```
MANUAL={ON }
       {OFF}
```

Response Keyword: None.

Function: Specifies whether the automatic extended network control facility be activated after a cold restart.

Format: OFF or ON.

Default: MANUAL=OFF.

Note: OFF specifies that any valid extended operator control commands generated as a result of SENDMSG macro execution is to be honored. ON disables the execution of all extended operator control commands generated as the result of the execution of SENDMSG macros.

The Start/Stop Automatic Network Control extended operator control commands MANUAL ON and MANUAL OFF permit you to dynamically disable and activate the automatic network control function after a cold start.

```
MSMAX={integer}
      {70     }
```

Response Keyword: X=.

Function: Specifies the percentage of the number of units specified by the MSUNITS= operand to be queued on a main-storage message queue data set before a warning is provided that the data set is nearly full. When this percentage of units is queued, bit 9 is set in every message error record in the system.

Format: An unframed decimal integer greater than zero.

Default: MSMAX=70.

Maximum: 100.

Note: This operand is discussed in greater detail in the section on main-storage message queue data sets in the chapter "Defining Data Sets" in the *TCAM Installation Guide*. See also, the note under the MSMIN operand of this macro.

```
MSMIN={integer}
      {50     }
```

Response Keyword: Y=.

Function: Specifies the percentage of the number of units queued on a message queue data set (specified by the MSUNITS= operand) below which bit 8 is set in every message error record in the system.

Format: An unframed decimal integer.

Default: MSMIN=50.

Maximum: 99.

Note: The operand can inform you that your message queue data set is no longer crowded. The value specified for MSMIN= must be less than that specified for MSMAX=; otherwise, the INTRO macro does not execute. The value specified for MSMIN= (or MSMAX=) at INTRO execution by means of a response to a WTOR macro is checked against the current value of MSMAX= (or MSMIN=) if specified, to ensure that this rule is not broken. If the rule is broken, the value specified in the WTOR response is rejected and an error message is sent to the system master console informing the operator of this fact. The operator may then respecify the value. For example, if MSMIN=95 and MSMAX=99 are coded in the INTRO macro, at INTRO execution the user should not reply:

```
r 00,   'MSMAX=90,MSMIN=85'
```

as a WTOR response, because the WTOR response is read from left to right and the new MSMAX value will be compared with the old MSMIN value and be rejected. If however, the user replies:

```
r 00,   'MSMIN=85,MSMAX=90'
```

these values will be accepted since the new MSMIN value is less than the old MSMAX value, and the new MSMAX value is greater than the new MSMIN value, with which it is compared.

```
MSUNITS={integer}
        {0      }
```

Response Keyword: M=.

Function: Specifies the maximum number of buffer units that may be assigned to the main-storage message queue data set at any one time.

Format: Unframed decimal integer.

Default: MSUNITS=0.

Maximum: 65,535 minus the value on the LNUNITS= operand.

Note: Guidelines for coding this operand are given in the discussion of main-storage message queue data sets in "Defining Data Sets", in the *TCAM Installation Guide.*

This value is added to that specified for LNUNITS= to determine the total number of units in the buffer-unit pool.

If specified as part of the WTOR response at INTRO execution, M=0 is considered an invalid response. At execution, a value greater than zero must be specified. If MSUNITS=0 is specified or assumed, the expression

M= may not be entered in the WTOR response at INTRO execution. Therefore, if a main-storage message queue data set is desired, MSUNITS= must be coded with a nonzero integer, even if the value specified is to be overridden at INTRO execution by an M= expression.

Either MSUNITS= must be nonzero or DISK=YES must be coded.

```
NETMON={termname     }
       {KEY(keyname)}
       {SYSCON      }
```

Response Keyword: None.

Function: Specifies that a copy of all responses to certain extended operator commands that alter the system status is to be sent to the specified extended primary operator control station.

Format: *termname,* KEY(*keyname*), or SYSCON. *termname* and *keyname* must conform to the rules for assembler language symbols.

Default: NETMON=SYSCON.

Note: *termname* specifies the name of the external LU or application that is to receive the extended operator control system-status messages. KEY(*keyname*) specifies the routing key for the destination that is to receive the extended operator control system-status messages. *keyname* is assumed to be up to 8 characters unless hexadecimal data is indicated by XL8''. SYSCON is the name of the system console.

NETMON=OPRA is not valid. If OPRA is specified, NETMON=SYSCON is assumed.

```
NEWLINE={X'2615'}
        {X'15'  }
```

Response Keyword: None.

Function: Specifies the new-line symbol that is used by internal routines that generate messages exceeding a single line of output.

Format: X'15' or X'2615'.

Default: NEWLINE= X'15', the EBCDIC new-line symbol recognized by most IBM terminal devices.

Note: Messages generated by extended operator control, as well as the SENDMSG and TCSUP (GEN=STANDARD) macros, depend on the specification of this operand.

X'2615' is the new-line symbol, often referred to as Carriage Return/Line Feed (CR/LF), for teletypewriter devices. In a mixed environment that has both teletypewriter and other devices, code a MSGEDIT macro to remove the X'26' during processing of an incoming MH group of messages from the teletypewriter devices, in order to provide an internally compatible format. The outgoing MH group for the teletypewriters can then reinsert the X'26' prior to output message transmission. If, however, the system handles only teletypewriter devices, specify NEWLINE=X'2615'.

```
NODEID={n}
       {0}
```

Response Keyword: None.

Function: Specifies the node identifier assigned to this MCP by the designer of an extended network. This identification number is stored in the TCAM vector table (TVT) and is used internally by TCAM macros.

Format: *n* must be either a decimal integer from 1-245 or a symbol previously equated to a decimal value from 1-245. Each node identifier must be unique in its network.

Default: NODEID=0. If this MCP is part of an interconnected multicomputer network using extended networking capabilities, a value other than 0 must be specified.

Note: The node identifiers 246-255 are reserved for system use. If this MCP is to make use of TCAM extended networking capability, a positive integer must be specified for NODEID=.

```
OPMASK={(hex integer,..      )}
       {(FCFC,FCFC,C000,FC0C)}
```

Response Keyword: OP=.

Function: Specifies a list of security masks that allow only select fields of terminal-table entries to be modified by the TCHNG application program macro function.

Format: Four 2-byte hexadecimal fields without framing characters. The four entries must be separated by commas and enclosed with parentheses.

Default: The following list gives the default for each terminal-entry type supported by OPMASK=. These masks must be coded in the exact order shown.

Terminal-Table Entry Type	**Default Mask**
Log	FCFC
LU	FCFC
LIST	C000
TPROCESS	FC0C.

Note: Each bit in the 2-byte *opmask* field corresponds to a a byte in the terminal-table entry. If the bit equals 1, its corresponding field can be changed; if the bit equals 0, then its corresponding field in the terminal-table entry cannot be changed.

For all terminal-table entry types except LIST entries:

Bit in Mask	Corresponding Byte in Terminal-Table Entry
0	Byte 4
1	Byte 5
2	Byte 6
3	Byte 7
4	Byte 8
5	Byte 9
6	Byte 10
7	Byte 11
8	Byte 12
9	Byte 13
10	Reserved
11	Reserved
12	All option fields as a group
13	All device-dependent fields as a group
14-15	Reserved

Note: For TPROCESS entries, bytes 12 and 13 cannot be changed regardless of the mask.

For LIST entries:

	Corresponding Byte in Bit Terminal-Table Entry
0	Byte 4-5 of the entry, which is the field that contains the number of entries in the list.
1	Byte 6-n of the entry, which is the TTCIN of the names in the list.
2-15	Reserved

Example:

```
OPMASK=(0008,0008,0000,0008)
```

This specification allows the TCHNG macro to change only option fields in all entries except LIST entries; it does not allow any changes to LIST entries.

For more information on modifying terminal-table entries with the TCHNG macro, see "Optional TCAM Facilities for the Application Programmer" in *TCAM Application Programming*.

`PASSWRD=characters`

Response Keyword: W=.

Function: Specifies a character string that must be entered in an MRELEASE, MCPCLOSE, TCHNG macro issued in an application program. This password is used to verify this MCP's authority to accept or provide data by the name specified in the PROGID= operand.

Format: One to eight unframed characters with no embedded blanks or commas.

Default: None. This operand is optional.

Note: If this operand is coded, none of the macros previously listed under *Function* is executed unless they have the correct password. A macro with an incorrect password or no password is ignored. PASSWRD=0 indicates that no password is being specified. An internal TCAM routine scrambles the password at INTRO execution.

```
PRIMARY={termname}
        {SYSCON  }
```

Response Keyword: P=.

Function: Specifies the name of the external LU or application program to be used as the basic primary operator control station.

Format: *termname* or SYSCON. *termname* must conform to the rules for assembler language symbols.

Default: PRIMARY=SYSCON.

Note: *termname* is the name of a external LU or application program (defined by a TERMINAL or PUT/WRITE TPROCESS macro). It must be able to enter and to accept messages.

SYSCON is the name of the system console. The functions of the basic primary operator control station are given in *TCAM Operation* publication.

If *termname* is changed by a Change Basic Primary Operator Control Station basic operator command, execution of a POLE or POF restart causes the value specified in the macro to be overridden by the value specified by the last Change Primary Operator Control Station command executed before closedown or failure.

```
PROGID={characters}
       {IEDIMCP   }
```

Response Keyword: None.

Function: Specifies the name of the message control program.

Format: One to 230 unframed characters with no embedded blanks or commas.

Default: PROGID=IEDIMCP.

Note: This operand must be the same as the name used to identify the "TCAM" logical unit. TCAM inserts this name in a DC C'characters' field located in the MCP. In a dump, this name appears in the EBCDIC field at the right of each page of the listing and identifies the beginning of executable code for the MCP.

```
RESTART={integer}
        {0      }
```

Response Keyword: N=.

Function: Specifies which environment record the TCAM restart is to use in attempting to reconstruct the MCP environment as it existed at the time of closedown or failure.

Format: An unframed decimal integer.

Default: RESTART=0.

Maximum: 255.

Note: For more information on the use of this operand, see the chapter "Checkpoint/Restart Service Facility" in *TCAM Utilities*.

If 0 is specified, the latest environment record is used; if 1 is specified, the next to the latest record is used, and so forth.

Although the maximum that may be specified is 255, the value entered must be at least two (2) less than the number of environment records kept, as specified by the CPRCDS= operand. This value should not be changed during a warm restart. A scan is performed at restart if scanning is specified in the STARTUP= operand.

If a message queue data set is on reusable disk and the integer specified causes TCAM to attempt to restructure the environment from a checkpoint record that was taken before serviced messages in a certain queue were overlaid, it is unlikely that a POLE or POF restart will be successful.

```
REUSMSG={RETAIN  }
        {NORETAIN}
```

Response Keyword: RM=.

Function: Specifies whether message IED159 is to be retained on the operators console until either TCAM or the operator deletes the message.

Format: RETAIN or NORETAIN.

Default: REUSMSG=NORETAIN.

Note: If REUSMSG=RETAIN is coded, the "STARTS" and "FULL" forms of message IED159 are retained on the operators console as critical messages (the message then becomes IED159). Before issuing the "QUITS" form of the message, TCAM deletes the "STARTS" form. If a message queue full condition is reached after the "STARTS" form of IED159 is issued, TCAM deletes the "STARTS" form and issues the "FULL" form.

The "FULL" form of the message is not deleted until the message queue reorganization is complete. Before issuing the "ENDS" form, TCAM deletes the "FULL" form of the message.

If REUSMSG=NORETAIN is coded, message IED159 is deleted whenever the operators console is cleared. See the *TCAM Messages* publication for more information concerning message IED159.

```
STARTUP={C}[{Y}][I]
        {W} {E}
            {N}
```

Response Keyword: S=.

Function: Specifies the type of startup to be performed following closedown of the message control program or system failure.

Format: C, CE, CEI, CI, CN, CNI, CY, CYI, W, WE, WEI, WI, WN, WNI, WY or WYI.

Default: None. This operand is required.

The types of restart are defined in the chapter of the *TCAM Utilities* manual titled "Checkpoint/Restart Service Facility."

The values may be specified in any order. For instance, IC is equally valid and produces the same result as CI.

C specifies that a cold restart is to be performed following a normal quick closedown or flush closedown and that a point-of-failure (POF) restart (including scanning of the message queues) is to be performed following system failure.

CE specifies that a cold restart is to be performed following a normal quick closedown or flush closedown and that a POF restart is to be performed following system failure. The POF restart includes an additional scan of the message queues to reconstruct queue chaining fields. The reconstruction allows complete retrievability by input sequence number of the messages placed on the queue between the last checkpoint and system failure.

CN specifies that a cold restart is to be performed following a normal quick closedown or flush closedown and that a POF restart including scanning message queues (no queue back scan) is to be performed following system failure. N only has significance in the POF restart.

CY specifies that a cold restart is to be performed following a quick closedown, a flush closedown, or a system failure.

W specifies that a POF restart is to be performed following a normal quick closedown, a flush closedown, or system failure.

WE specifies that a POF restart is to be performed following a normal quick closedown, a flush closedown, or system failure. The POF restart will include an additional scan of the message queues to reconstruct queue chaining fields. The reconstruction allows complete retrievability by input sequence number of the messages placed on the queue between the last checkpoint and system failure.

WN specifies that a POF restart is to be performed following a normal quick closedown, a flush closedown, or system failure. A queue back scan is not performed. This is to be used only if retrievability by sequence number is not desired.

WY specifies that a point-of-last-environment (POLE) restart is to be performed following a quick closedown, a flush closedown, or system failure.

I specifies the status of each external LU is to be included in the checkpoint record.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the symbol entry in the "Glossary").

Default Value: None. Specification is optional.

```
{UNITSZ=integer}
{KEYLEN=integer}
```

Response Keyword: K=.

Function: Specifies the size (in bytes) of a buffer unit.

Format: An unframed decimal integer greater than or equal to 76.

Default: None. One of these two operands is required.

Maximum: 255.

Note: Guidelines for coding these operands are given in the *TCAM Installation Guide*. If disk queuing is used, *integer* must be identical with the unit size specified in the DCB= (KEYLEN=) parameter of the IEDQDATA DD statement for the IEDQXA utility program used to preformat the disk queues.

```
USEREG={integer}
       {0      }
```

Response Keyword: None.

Function: Specifies the number of registers to be saved when in-line user code is located in an inheader or outheader subgroup that may handle multiple-buffer headers.

Format: An unframed decimal integer.

Default: USEREG=0.

Maximum: 10.

Note: For guidelines on specifying this operand, see the chapter of the *TCAM Installation Guide* titled "Coding the Message Handler." USEREG= specifies sequential registers, beginning with register 2. For instance, if USEREG=4 is coded, registers 2, 3, 4, and 5 are saved.

Return Codes

Following the INTRO macro, the user must include a section of code that tests the return code in register 15 to determine whether INTRO has executed correctly. If the MCP is not halted when the INTRO's return code is nonzero, results are unpredictable.

If a nonzero code is to be returned by the INTRO routine, TCAM displays the message:

```
IED065I    TCAM INITIALIZATION ERROR xxxx
```

where *xxxx* is the decimal equivalent of the value returned in register 15; the values that may appear in this variable field follow:

Code	**Meaning**
X'00000000'	Successful execution.
X'00000004'	TCAM is already in this address space or an attempt was made to start a CVT-based TCAM and a CVT-based TCAM is already started.
X'00000008'	There is insufficient main storage for generating one of the following: 1. Buffer-unit pool - refer to the LNUNITS= and MSUNITS= operands. 2. CPB free pool - refer to the CPB= operand. 3. Subtask trace table - refer to the DTRACE= operand.
X'00000010'	An invalid value was specified on the ALTDEST= operand of either a TERMINAL or a TPROCESS macro.
X'00000014'	The basic primary operator control station is invalid. Either no such terminal entry exists, or SECTERM=YES is not specified for the station named on the PRIMARY= operand of the INTRO macro.
X'0000001C'	The MCP is not executing in authorized protection key 6.
X'00000024'	The job step task is not the initiator.
X'00000028'	TCAM basic operator control was not attached or did not initialize successfully. Verify that a TASKDEF macro for SYSOPCTL is in the subtask table.
X'0000002C'	A NEWMSG, DESTFLD, or SENDMSG macro was issued in an MH and the DLQ operand on the INTRO macro was not specified, or DLQ was not specified as a parameter at initialization.
X'00000030'	An attempt was made to start an ASCB-based TCAM but the operating system does not support ASCB-based TCAMs (not at MVS SP.1.3.1 or above).

KEYDEF Macro

The KEYDEF macro:

- Creates an entry in the key table
- May be coded in a nonexecutable section of an MCP or in an externally assembled module
- Must be coded contiguously with all other KEYDEF macros defining other routing keys in the table
- Requires the presence of a KEYTABLE macro in the MCP.

Supported Resources and General Requirements: ALL.

Valid subgroups: Not applicable.

The KEYDEF macro creates an entry in the key table, which is used by the KEYPROC macro in routing the key.

If KEYDEF macros are being coded in an externally assembled module or in an MCP that is not a model MCP and does not have an IEDTCSD macro coded in it, one KEYDEF macro specifying DSECT=YES must be included to generate certain labels required by the macro. Do *not* code a KEYDEF macro specifying DSECT=YES in an MCP that contains an IEDTCSD macro because IEDTCSD also generates the required labels. The model MCP generates the required labels.

NAME	OPERATION	OPERANDS
[symbol]	KEYDEF	[RESOURC={name } {OPCTL}]
		[,CLEARSC={NO } {YES}]
		[,CONVERT={NO } {YES}]
		[,DSECT={NO } {YES}]
		[,FEATURE={BASE} {NET }]
		[,HNLOGOF={0 } {logoff}]
		[,INITSES={NO } {YES}]
		[,KEY={LAST } {chars}]
		[,KEYMODE={SESSION} {TRANSAC}]
		[,KEYTYPE={USERPROG} {TERMINAL} {SPECIAL } {CICS }]
		[,LOCAL={NO } {YES}]

NAME	OPERATION	OPERANDS
[symbol]	KEYDEF	[,LOGAUTO={0 }]
	(cont'd)	{logauto}
		[,MONITOR={YES}]
		{NO }
		[,NODE={0}]
		{n}
		[,PGMCOPY={NO }]
		{YES}
		[,PRIORTY={0 }]
		{pri}
		[,SECURTY={0 }]
		{secmask}
		[,SENDERR={NO }]
		{YES}
		[,SNDLOGO={NO }]
		{YES}
		[,STATUS={NOTUPYET}]
		{UP }
		[,TC={n}]
		{1}
		[,TERMNAM={NO }]
		{YES}

```
CLEARSC={NO }
        {YES}
```

Function: Specifies whether the application represented by the key should be informed when the Clear key is depressed on a 3270 display station with which it is in session.

Format: NO or YES.

Default: CLEARSC = NO.

Note: If CLEARSC = YES is specified, the KEYCLERS flag is set to 1 in the device-dependent status byte of the key table entry.

Processing related to the KEYCLERS flag of the key table entry must be user-provided. See the model MCP for an example.

This operand is ignored if KEYTYPE = TERMINAL, KEYTYPE = SPECIAL, or RESOURC = OPCTL is specified, and it is valid only if KEYMODE = SESSION is specified.

```
CONVERT={NO }
        {YES}
```

Function: Specifies whether all messages using this key-table entry must be uniformly formatted regardless of origin.

Format: NO or YES.

Default: CONVERT = NO.

Note: If YES is specified, the KEYCONTO flag is set to 1 in the device-dependent status byte (KEYSTDEV) of the key table entry.

Processing related to the KEYCONTO flag of the key table entry must be user-provided.

You can refer to the setting of this key-table entry flag in an MH that supports several types of LUs. If this key table entry represents an application which handles only TWX LU type message formats, you can test this flag for messages from 3270 LU types to determine if user-provided edit processing is required.

If different edit procedures are required for each originating station type, this flag may indicate that more specific requirements relating to message consistency can be found in the user-extended portion of the key-table entry.

```
DSECT={YES}
      {NO }
```

Function: Generates a dummy section describing the format of a key table entry.

Format: YES or NO.

Default: DSECT=NO.

Note: If DSECT=YES is specified, all other KEYDEF operands are ignored and the dummy section describing the format of the key-table entry is generated.

```
FEATURE={BASE}
        {NET }
```

Function: Permits assembly of the KEYDEF macro outside of the MCP.

Format: BASE or NET.

Default: FEATURE=BASE.

Note: This operand is required only when KEYDEF macros are issued outside of an MCP and only on the first KEYDEF macro of a series. This operand is not required and is ignored when KEYDEF macros are issued within an MCP.

```
HNLOGOF={0      }
        {logoff}
```

Function: Specifies if end-to-end sessions may be ended via a logoff message, and, if so, provides the text of the logoff message.

Format: *logoff* is a variable-length character string up to 8 bytes in length in either CLn'' or XLn'' format, where *n* is a decimal integer from 1 to 8.

Default: HNLOGOF=0.

Note: The *logoff* message text is placed in the KEYLOGOF field of the key table entry, and its length is placed in the KEYLOLEN field of the key table entry.

0 specifies that session logoff is not supported for the key.

This operand is ignored if KEYTYPE=TERMINAL is specified and is invalid unless KEYMODE=SESSION or RESOURC=OPCTL is specified.

```
INITSES={NO }
        {YES}
```

Function: Specifies whether the logon message establishing an end-to-end session is to be sent to the destination application.

Format: NO or YES.

Default: INITSES=NO.

Note: No specifies that the logon message is not to be sent to the application.

YES specifies that the logon message is to be sent to the application. The KEYSESIN flag is set to 1 in the session related status byte (KEYSTSES).

This operand is ignored if KEYTYPE=TERMINAL, KEYTYPE=SPECIAL, or RESOURC=OPCTL is specified and is invalid unless KEYMODE=SESSION is specified.

```
KEY={LAST }
    {chars}
```

Function: Specifies the character string to be used as the key for the entry being defined or else signals the end of the table.

Format: *chars* or LAST. *chars* is an 8-byte character string in either CL8'' or XL8'' format.

Default: KEY=LAST.

Note: LAST signals the end of the table. If KEY=LAST is specified, all other operands are ignored.

Keys of less than 8 characters must be left-justified and padded out to eight bytes with blanks.

```
KEYMODE={SESSION}
        {TRANSAC}
```

Function: Specifies the type of routing to be used for messages specifying this key.

Format: SESSION or TRANSAC.

Default: KEYMODE=TRANSAC.

Note: SESSION specifies that end-to-end sessions are to be used as a routing mechanism for messages specifying this key. The KEYSESSN flag is set to 1 in the session-related status byte (KEYSTSES).

TRANSAC specifies that transaction-based routing is to be used for messages specifying this key. The KEYTRANS flag is set to 1 in the session-related status byte (KEYSTSES).

This operand is ignored if KEYTYPE=TERMINAL or RESOURC=OPCTL is specified.

```
KEYTYPE={USERPROG}
        {TERMINAL}
        {SPECIAL }
        {CICS    }
```

Function: Specifies the type of key entry. It is ignored if RESOURC=OPCTL is coded.

Format: USERPROG, TERMINAL, SPECIAL, or CICS.

Default: KEYTYPE=USERPROG.

Note: CICS specifies that the key represents a CICS/VS subsystem.

USERPROG specifies that the key represents a TCAM application program.

TERMINAL specifies that the key represents a external LU or a distribution list, or cascade list.

SPECIAL specifies that the key represents a user-defined function. For example, in the model MCP the “special” key is used to represent broadcast and multiple-destination messages.

If KEYTYPE=TERMINAL, the KEYTERML flag is set to 1 in the key status byte (KEYSTAT) of the key entry.

If KEYTYPE=SPECIAL, the KEYSPECL flag is set to 1 in the key status byte (KEYSTAT) of the key entry.

If the KEYTYPE= operand specifies a valid operand other than TERMINAL or SPECIAL, the KEYPROGM flag is set to 1 in the key status byte (KEYSTAT) of the key entry. Additionally, for the KEYTYPE=CICS, the flag KEYCICS will be set to 1 in the program related status byte field (KEYSTPGM). The KEYCICS flag is not used by TCAM but is reserved for future use.

```
LOCAL={NO }
      {YES}
```

Function: Specifies if the local internodal awareness system service program is to inform other host nodes of the status of the application specified via the RESOURC= operand.

Format: NO or YES.

Default: LOCAL=NO.

Note: No specifies that the application key is known globally in the system. If the MONITOR= operand specifies YES and KEYTYPE= specifies an application in this TCAM system, the local internodal awareness system service program monitors and broadcasts its availability to other nodes.

YES specifies that this key is to be locally known only. If the MONITOR= operand specifies YES and the KEYTYPE= operand specifies an application in this TCAM system, the internodal awareness system service program monitors its availability but does not broadcast its availability status to other host nodes. The KEYLOCAL flag is set to 1 in the routing-related status byte (KEYSTRTE) of the key-table entry.

```
LOGAUTO={0      }
        {logauto}
```

Function: Specifies whether an end-to-end session is to be ended automatically whenever a permanent I/O error occurs, and if so, provides the text of the logoff message to be sent to the application in this case.

Format: *logauto* is a variable length character string up to 8 bytes in length in either CLn" or XLn" format, where n is a decimal integer from 1 to 8.

Default: LOGAUTO=0.

Note: The *logauto* character string is placed in the KEYALOGO field of the key table entry, and its associated character string length in the KEYALOGL field of the entry.

0 specifies that the automatic logoff facility is not supported.

Processing related to the KEYALOGO field must be provided by the user.

This operand is ignored if KEYTYPE=TERMINAL or KEYTYPE=SPECIAL is specified, or if RESOURC=OPCTL is specified. It is invalid if the KEYMODE= operand does not specify KEYMODE=SESSION.

```
MONITOR={YES}
        {NO }
```

Function: Specifies whether or not to monitor the availability of the application program specified by the RESOURC= operand. If the extended networking facility is being used, the internodal awareness system service program does the monitoring; otherwise, the MCP does the monitoring.

Format: YES or NO.

Default: MONITOR=YES.

Note: NO specifies that monitoring of the application program availability is not to be performed. The KEYNOMON flag is set to 1 in the key status byte field (KEYSTAT).

YES specifies that the MCP or internodal awareness system service program is to monitor the availability of the associated application program and update its key-table entry or entries, as required to reflect its current availability.

This operand is ignored if KEYTYPE=TERMINAL, KEYTYPE=SPECIAL, or RESOURC=OPCTL is specified.

```
NODE={0}
     {n}
```

Function: Provides the host node identifier on which the destination associated with this key is located. This operand is valid only if a non-zero value was coded for the NODEID operand on the INTRO macro.

Format: *n* should either be a decimal integer from 1 through 245 or a symbol that has previously been equated to a decimal value from 1 through 245.

Default: NODE=0.

Note: The value specified by NODE= is placed in the KEYNODE field of the key-table entry. (Numbers 246 through 255 are reserved for TCAM system use.)

NODE=0 assumes that the entry represents an application program that is never executed on this node and that this key table entry is updated at a later time by the internodal awareness system service program.

```
PGMCOPY={NO }
        {YES}
```

Function: Specifies if the application program handles copy request processing from a 3270 station.

Format: NO or YES.

Default: PGMCOPY = NO.

Note: If YES is specified, the KEYCOPY flag is set to 1 in the device-dependent status byte (KEYSTDEV) of the key-table entry.

Processing related to the PGMCOPY flag in the key-table entry must be user-provided.

An example of how this flag setting might be used appears in the model MCP 3270 MH, where the AID byte of a message indicates a copy request.

If the external LU is currently in session with a program whose key-table entry has the KEYCOPY flag on, you could merely forward the message to the application program for processing. If you choose to support all copy requests in the MH, this flag could be tested to determine when MH processing of the message is necessary.

This operand is ignored if KEYTYPE = TERMINAL, KEYTYPE = SPECIAL, or RESOURC = OPCTL is specified. It is valid only if KEYMODE = SESSION is specified.

```
PRIORTY={0  }
        {pri}
```

Function: Specifies the message priority level to be used for messages specifying this key.

Format: *pri* is a decimal integer from 1 to 255.

Default: PRIORTY = 0.

Note: The *pri* value is placed in the KEYPRIOR field of the key table entry.

```
RESOURC={name }
        {OPCTL}
```

Function: Provides the name of the destination associated with this key entry.

Format: *name* or OPCTL. *name* must conform to rules for assembler language symbols.

Default: None. Specification is required.

name is the name of a TERMINAL or TPROCESS macro representing an external LU or application to which messages specifying the key as a destination are to be routed.

If an application program is specified as the destination, the TPROCESS entry named by this operand should be the one that is capable of originating messages (the application GET/READ TPROCESS entry).

If OPCTL is specified, the resource identification reserved for the extended operator control system service program is assigned automatically.

The destination specified by RESOURC= is translated by TCAM into a TNT index which is placed in the KEYRESOR field of the key entry.

If RESOURC=OPCTL is coded, the key entry is additionally initialized with the following flag settings:

- KEYPROGM and KEYNOMON flag is set to 1 in the key status byte field (KEYSTAT)
- KEYOPCTL flag is set to 1 in the program-related status byte (KEYSTPGM)
- KEYSESSN flag is set to 1 in the session-related status byte field (KEYSTES)
- KEYCONTO flag is set to 1 in the device-dependent status byte field (KEYSTDEV).

```
SECURTY={0       }
        {secmask}
```

Function: Specifies whether security checking is to be performed by the MCP before a message is routed to the destination and if so, provides the 3-byte security mask to be used for the authorization check.

Format: *secmask* is a 3-byte character string in either CL3'' or XL3'' format.

Default: SECURTY=0.

Note: The 3-byte mask is placed in the KEYSECMK field of the key table entry. This mask must match the contents of the SECURITY option field associated with the source in order for the message to be routed.

0 specifies that no security checking is to be performed.

```
SENDERR={NO }
        {YES}
```

Function: Specifies whether error status messages should be sent to the application program associated with this key-table entry.

Format: NO or YES.

Default: SENDERR=NO.

Note: If YES is specified, the KEYERPRG flag is set to 1 in the device dependent status byte (KEYSTDEV) of the key-table entry.

Processing relating to KEYRPRG flag of the key-table entry must be user-provided.

This operand is ignored if KEYTYPE = TERMINAL, KEYTYPE = SPECIAL, or RESOURC = OPCTL is specified and is invalid unless KEYMODE = SESSION is specified for this key.

```
SNDLOGO={NO }
        {YES}
```

Function: Specifies whether the application program associated with this MCP is to receive the logoff used to terminate end-to-end sessions.

Format: YES or NO.

Default: SNDLOGO = NO.

Note: If SNDLOGO = YES is specified, the KEYSNDLO flag is set to 1 in the logoff-related status byte (KEYSTLOF) of this key-table entry, and the KEYPROC macro routes the logoff message to the application when it is encountered.

This operand is valid only if KEYMODE = SESSION is specified and if the HNLOGOF = operand specifies the text of a logoff message.

```
STATUS={NOTUPYET}
       {UP      }
```

Function: Specifies the initial status of a program.

Format: NOTUPYET or UP.

Default: STATUS = NOTUPYET.

Note: NOTUPYET specifies that the program has not been initially activated and needs to be specified if program status is to be monitored by the MCP, by the internodal awareness system service program, or by a user monitoring the program. The KEYPSDAY flag is set at 1 in the key status byte (KEYSTAT).

UP specifies that the program is initially available for processing. UP may be specified only when the MONITOR = operand specifies NO, indicating user monitoring of program availability.

This operand is ignored if KEYTYPE = TERMINAL, KEYTYPE = SPECIAL, or RESOURC = OPCTL is specified.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

```
TC={n}
   {1}
```

Function: Specifies the transmission category to be used when routing messages from another TCAM node to the destination associated with this key entry. This operand is valid only if a non-zero value was coded for the NODEID operand on the INTRO macro.

Format: An unframed decimal integer from 1 through 16.

Default: TC=1.

Note: When a message is routed by the KEYPROC macro and the key indicates that the destination is located in another TCAM node, the node table entry for the other TCAM node is accessed to determine the type of routing. Depending upon the value of this operand, the message is placed on the queue named by either the TC1 (transmission category) or TC2 (transmission category 2) operand of the NODEDEF entry for that node.

```
TERMNAM={NO }
        {YES}
```

Function: Specifies whether the name of the originating external LU or application is to be inserted by MCP code in the MH at the beginning of message text.

Format: NO or YES.

Default: TERMNAM=NO.

Note: YES specifies that the name of the originating external LU or application is to be inserted as the first 8 positions (padded with blanks) beyond the FHP in each message header. The KEYTRMNM flag is set to 1 in the routing related byte (KEYSTRTE) of the key-table entry.

Coding TERMNAM=YES only sets the KEYTRMNM flag in the key-table entry; actual insertion of the origin name into the message must be done by user-supplied code in the inheader subgroup of the message handler. For an example of such code, see the model MCPs.

This operand is ignored if RESOURC=OPCTL is specified.

Example

By defining constants following each entry, you may incorporate any additional data you might wish in a KEYDEF entry. For example, if you want to append an 8-character field to each entry you could code:

```
KEYDEF KEY=CL8'PAYR'
       DC     CL8' '
KEYDEF KEY=CL8'C'
       DC     CL8' '
KEYDEF KEY=LAST
```

Return Codes

None.

KEYPROC Macro

The KEYPROC macro:

- Examines an incoming message and conditionally queues it on the appropriate destination queue based on the current status of the message origin, the message destination, and the system
- May be issued only after the FHPBUILD and TCSEARCH ARG=CURRTERM macros have been issued
- May be issued more than once in a message handler
- Requires the presence of an FHP in the message being processed, unless a special key is being processed.

As each message is entered from an external LU or application, the message handler must cope with device dependencies and route the message. The KEYPROC macro can perform most of the work associated with routing messages.

KEYPROC macro processing depends on whether or not the origin is a external LU that is currently in an established end-to-end session when the message is entered. (If the SESSION option field associated with the originating external LU contains an asterisk, the external LU is not in an end-to-end session.)

If the originating external LU is currently in an end-to-end session, KEYPROC gets the routing key for the destination application from the SESSION option field. The message is then processed based on specifications in the routing key entry. If session logoff is specified, the data (addressed by the FRSTCHR= operand) is compared to the session logoff sequence in the routing key entry (KEYLOGOF). If they match and if SNDLOGO=NO was specified, the message is not routed. Otherwise, the message is routed to the destination queue associated with the key entry.

If the originating external LU is not currently in an end-to-end session, KEYPROC checks the location pointed to by the FRSTCHR= operand to gain access to a key, and attempts to either establish an end-to-end session or perform transaction-based routing of the message (depending on the specifications in the key's entry in the key table).

After finding the key for the message, KEYPROC matches the key with the appropriate key-table entry and the code that executes is based on the specification in the entry as follows:

- If the destination associated with the key is a external LU, security checking is performed (if required), the message is scheduled for queuing on the destination queue, and the KEYPROC macro returns control to the user KEYTERM= branch, where the user can apply appropriate formatting and insertion scheduling using the INSRTFLD macro.
- If the destination associated with the key is an application using end-to-end sessions, security checking is performed (if required), and the message is scheduled for queuing on the appropriate destination queue. The SESSION option field is initialized with the destination's key, and the macro returns control to the user at the NEWSESS= label if the key entry specifies that the application is to receive logon requests, or at the SETSESS= label if the key entry specifies that the application is not to receive logon requests; in this case, the message is not scheduled for routing.
- If the key describes an application using transaction-based routing, a security check is performed (if required), the message is scheduled for queuing on the appropriate destination queue, and control is returned to the TRANSAC= branch label.

- If the key describes a special user processing key, a security check is performed (if required), the message is *not* scheduled for queuing, and control is immediately returned to the user at the SPECIAL= branch label.
- All but special keys must have an FHP or the DATABAD= branch will be taken.

The KEYPROC macro never alters the message it processes. It is device independent. For that reason, you must handle device peculiarities before the issuance of the macro and after the macro has executed.

You must do the following before issuing KEYPROC:

1. Screen out error messages via the ERRTEST macro.
2. Issue the TCSEARCH and FHPBUILD macros.
3. Identify the first valid data character (or the first character of the key) and pass its address to the KEYPROC macro using the FRSTCHAR= operand.

You must do the following after issuing KEYPROC:

1. Using the returned key entry address, test the entry for device-dependency requirements. Examples of this type of processing are:
   - Issuing the DEVCON macro to convert 3270 input to 2260/1050-like data streams
   - Formatting the message, perhaps by adding the 8-byte terminal name in front of the message, if required by the program.
2. Issue NEWMSG macros to schedule the generation of advisory or error messages. For example, you can issue a NEWMSG macro at the LOGOFF= label to advise the originator that he is now logged off as he requested.
3. If the destination associated with the key is a external LU, you can format the ultimate output appearance of the message (using the INSRTFLD macro, for example).

The model MCPs illustrate the use of the KEYPROC macro for all device message handlers. Study the models to gain a more comprehensive understanding of the macro.

The KEYPROC macro uses the SESSION, SECURITY, TCSOPTS, and THRESH option fields.

Supported Resources and General Requirements: ALL, FHP.

Valid subgroups: Inheader.

KEYPROC

NAME	OPERATION	OPERANDS
[symbol]	KEYPROC	DATABAD=name ,FRSTCHR=(register) ,INSESS=name ,KEYADDR=(register) ,KEYBAD=name ,KEYDOWN=name ,KEYTERM=name ,LOGOFF=name ,NEWSESS=name ,NODEBAD=name ,NODEDWN=name ,RESBAD=name ,RESDOWN=name ,SECURTY=name ,SETSESS=name ,SPECIAL=name ,TRANSAC=name [,SCAN={<u>YES</u>}] {NO }

DATABAD=name

Condition: Either a zero-length buffer was being processed, an invalid FRSTCHAR= address was given, the keyname being scanned in the message contained a character that was not uppercase EBCDIC alphanumeric, or no FHP was in the message and a special key was not being processed.

User Action: Use the NEWMSG macro to trigger the generation of a message to inform the origin of this condition.

FRSTCHR=(register)

Function: Provides the macro with the address of the first valid data character in the message (or the first character of the key, if it is not the first field in the message). This address will be used to locate the key if the origin is not currently in an end-to-end session, or to test for a logoff message if the origin is currently in an end-to-end session.

Format: *(register)* is a general register, 2 through 12, that can be specified as a decimal integer from 2 through 12 between framing parentheses, or as a symbol that has previously been equated to a value from 2 through 12. It is your responsibility to assign this operand a value that is within the prescribed limits.

Default: None. Specification is required.

Note: You must have previously loaded the register with the address of the first data character or the first character of the key in the message.

`INSESS=name`

Condition: The originating external LU is currently in end-to-end session and the message has been scheduled to be queued on the destination queue for the application which is the session partner.

User Action: Perform device-dependent processing and insert the external LU name, if required.

`KEYADDR=(register)`

Function: Specifies the general register in which KEYPROC returns the address of the key entry. KEYPROC returns this address to the user unless either the DATABAD= or KEYBAD= branch point is taken.

Format: *(register)* is a general register, 2 through 12, which can be specified as an explicit decimal integer from 2 through 12 in framing parentheses, or a symbol, in framing parentheses, that has previously been equated to a value from 2 through 12. It is the user's responsibility to assign this operand a value that falls within the prescribed limits.

Default: None. Specification is required.

`KEYBAD=name`

Condition: The origin was not in an end-to-end session and the key (addressed by the FRSTCHR= operand) could not be matched in the key table.

User Action: Use the NEWMSG macro to trigger the generation of a message to inform the origin of this condition.

`KEYDOWN=name`

Condition: One of the following conditions has been found:

- The KEYDEF macro defining the key table entry associated with the destination application specified MONITOR=YES and the application is not currently active.
- The KEYDEF macro defining the key table entry associated with the destination application specified MONITOR=YES, and the destination has a main-storage queue of unsent messages greater than the threshold value specified in its THRESH option field.
- The destination application associated with the key has a main storage queue and is not currently active.
- The KEYDEF macro defining the key-table entry associated with the destination application specified MONITOR=YES, the application is located in another TCAM system and either the host node for the other TCAM system or the application itself is unavailable.

KEYTERM=name

Condition: The origin was not in an end-to-end session. The key was accessed and validated. The destination associated with the key entry is a station to which the message has been scheduled for routing.

User Action: Perform device-dependent processing and message formatting, if required.

LOGOFF=name

Condition: The origin was in an end-to-end session when that message was received. The key entry included a logoff sequence specified via the HNLOGOF= operand of the KEYDEF macro, which matched the message data pointed to by FRSTCHR=. The end-to-end session has been terminated. The origin SESSION option field has been cleared by KEYPROC (that is, an asterisk has been placed in the high-order byte of the SESSION option field). If the application that was the session partner expects to receive logoff messages (specified by SNDLOGO=YES in the KEYDEF entry), the message has been scheduled for queuing on the destination queue for the application; otherwise, it has not been scheduled.

User Action: Use the NEWMSG macro to trigger the generation of a message to inform the station operator that his end-to-end session is terminated and, if desired, to present the operator with a *menu* message outlining possible subsequent actions.

NEWSESS=name

Condition: The originating external LU was not in session with the destination. The key was accessed and validated. The key entry specifies that the destination is an application to which messages are routed via end-to-end sessions, and therefore the external LU was placed into session. The message was scheduled to be queued on the destination queue for the application which is the session partner because the key entry specified that the destination application receives logon requests.

User Action: Use the NEWMSG macro to trigger the generation of a message to inform the originator that an end-to-end session has been established.

NODEBAD=name

Condition: This operand is valid only if a non-zero value was coded for the NODEID operand or the INTRO macro. Either the node identifier of the destination network address is invalid (that is, the passed node identifier) or the terminal table entry named in the operand of the NODEDEF macro for the destination is invalid. This error is generally indicative of faulty user definition of the network routing table.

```
NODEDWN=name
```

Condition: This operand is valid only if a non-zero value was coded for the NODEID operand on the INTRO macro. One of the following conditions has been found:

- The node-table entry status information indicates that the target TCAM host node is inactive.
- The internodal destination queue for the other TCAM host node is located in main storage without disk backup, and the queue of unsent messages exceeds the threshold value specified in the THRESH option field associated with the internodal destination queue.

Note: This operand applies only to internodal routing in which the destination is a key representing a external LU or an application not monitored by the internodal awareness system service program.

```
RESBAD=name
```

Condition: This operand is valid only if a non-zero value was coded for the NODEID operand on the INTRO macro. The resource identifier of the destination in the key entry is invalid. This error is generally indicative of faulty user definition of the network routing tables or improper specification of the RESOURCE option field associated with the TERMINAL or TPROCESS entry.

```
RESDOWN=name
```

Condition: This operand is valid only if a non-zero value was coded for the NODEID= operand on the INTRO macro. One of the following conditions has been found:

- The resource identifier in the key entry reflects a destination external LU or application in this TCAM system with a main-storage queue of unsent messages greater than the threshold value specified in its THRESH option field.
- The resource identifier in the key entry reflects a destination application that uses main-storage queuing without disk backup which is not currently active.

Note: This operand applies only to keys representing external LUs or applications not monitored by the internodal awareness system service program.

This exit is driven internally by the DAFROUTE macro.

```
SCAN={YES}
     {NO }
```

Function: Indicates if the key pointed to by FRSTCHR= is already set up as an 8-byte field, padded to the right with blanks.

Format: YES or NO.

Default: SCAN=YES.

Note: YES indicates that the FRSTCHR= operand points to the beginning of the key field and that the macro must scan for the end of the key; delimited by a non-alphanumeric or blank character, 8 characters, or the end of the message, whichever is encountered first.

NO indicates that the FRSTCHR= operand points to the beginning of an 8-byte key field padded to the right with blanks. The field pointed to by FRSTCHR=, if SCAN=NO, need not be within the message buffer, but may be located elsewhere in the MCP (in an option field, for example).

`SECURTY=name`

Condition: The origin attempted to access a destination via this key which the user has not authorized it to communicate with.

For example, the origin attempted to enter into an end-to-end session with extended operator control but is not, however, an authorized extended operator control station. Or, the origin may have been in end-to-end session with the extended operator control program and entered a Send command but may not have been authorized to do retrievals. (Authorization for extended operator control stations and authorization for retrievals is specified in the origin's TCSOPTS option field.)

User Action: Trigger the generation of a message to notify a central security station of the attempted security branch. You may inform the originator that the key "does not exist."

Note: Except for the RESBAD= operand, the following KEYPROC conditional branch points are applicable only in an extended networking environment, and all are executed only when serious network malfunction has been detected. User action in all such cases should be to inform the origin that the function is temporarily suspended. The network control operator will be informed of the situations through the monitoring of other facilities.

`SETSESS=name`

Condition: Same as NEWSESS= except that the key entry specified that the destination application *does not* receive logon requests. KEYPROC has initialized the SESSION option field to place the external LU into end-to-end session, but has *not* caused the message to be queued on the destination queue for the application which is the session partner.

User Action: Use the NEWMSG macro to trigger generation of a message to inform the originator that an end-to-end session has been established.

SPECIAL=name

Condition: The origin was not in session. The key was accessed and validated. The key entry is defined as a special key. The message has *not* been scheduled for routing to a destination.

User Action: User processing as desired. User data in the key table entry may be used to indicate the type of special handling appropriate for the key.

symbol

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Note: All of the remaining operands are required; they provide the name of the next instruction to be executed if the condition indicated by each of the particular operands is detected.

TRANSAC=name

Condition: The origin was not in an end-to-end session. The destination associated with the key entry is an application program utilizing transaction-based routing. The message has been scheduled to be queued on the destination queue for the application.

User Action: Same as INSESS= with the following consideration:

The key is in the message, and some applications may not expect to receive a key in the text. For those applications, the key must be deleted at some time before the message is transferred to the application's work area. This deletion can be done either immediately in the TRANSAC= routine, or in the outheader subgroup of the AMH.

Another option you may exercise is the replacement of the key with user data in the key entry, for example, a four character CICS transaction code could replace the key.

Return Codes

None.

KEYTABLE Macro

The KEYTABLE macro:

- Causes a key table to be built at MCP execution
- Is optional and may be issued once in the initialization section of the MCP following the OPEN macros and before the READY macro.

When it is executed during initialization of the MCP, KEYTABLE dynamically constructs the key table using information provided by KEYDEF macros. The KEYDEF macros may be located in the resource definition section of the MCP, or they may be in an externally assembled module. By locating your KEYDEF macros in an externally assembled module, you can make changes to keys without reassembling the MCP.

Supported Resources and General Requirements: ALL.

Valid subgroups: Not applicable.

NAME	OPERATION	OPERANDS
[symbol]	KEYTABLE	ADDRESS=name ,ENTRY1=name ,TABNAME=name [,ENTRYLN={0}] {n}

ADDRESS=name

Function: Allows you to assemble your KEYDEF macros externally and provide the KEYTABLE macro with the address of the first entry.

Format: *name* must conform to the rules for assembler language symbols.

Default: None. Either ADDRESS= or ENTRY1= must be specified.

Note: *name* is the name of a fullword-aligned field that contains the address of the first KEYDEF macro coded for the key table that is to be built.

If the ADDRESS= operand is specified, TABNAME= must also be coded.

If the external module containing your KEYDEF macros were named KYTABLE, you could use the following code to initialize the address field:

```
          LOAD     EP=KYTABLE           Load the table
          ST       0,KYADDR             Store address
          KEYTABLE ADDRESS=KYADDR,      Issue the macro
                   TABNAME=name
          .
          .
KYADDR    DS   F
```

```
ENTRYLN={n}
        {0}
```

Function: Indicates the length of the user-data field in each key table entry.

Format: *n* is a decimal value from 1 to 100.

Default: ENTRYLN=0

Note: This operand allows you to append data to the basic KEYDEF entries. If 10 characters are added to each entry, ENTRYLN= would be 10. See the description of the KEYDEF macro for instructions on how to append user data to key-table entries.

`ENTRY1=name`

Function: Specifies the name of the first KEYDEF macro in the series defining the table.

Format: name must conform to the rules for assembler language symbols.

Default: None. Either ADDRESS= or ENTRY1= must be specified.

Note: *name* is the name of the first KEYDEF macro coded after this KEYTABLE macro in the assembly.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

`TABNAME=name`

Function: Provides a name for the key table being created.

Format: *name* must conform to the rules for assembler language symbols.

Default: None. Specification is required.

Return Codes

None.

LOCK Macro

The LOCK macro:

- Connects an external LU to an application program to await the reply to an inquiry message entered by the external LU
- Holds the connection for a single message or for an extended period.

LOCK keeps the connection between an external LU and an application program for a period of time not less than the duration of a message and its reply. An external LU connected in this manner is said to be in *terminal lock mode.* The application to which an external LU is locked depends upon the destination of the message. If the destination is not an application program, the station is not placed in lock mode.

For more information on how the lock function works, see the *TCAM Installation Guide.*

Supported Resources and General Requirements: SNA.

Valid Subgroup: Inheader.

NAME	OPERATION	OPERANDS

[symbol]	LOCK	{EXTEND }
		{MESSAGE}
		[,conchars]
		[,BLANK={YES }]
		{NO }
		{char}

```
BLANK={YES }
      {NO  }
      {char}
```

Function: Specifies whether EBCDIC blank characters are to be ignored when encountered in the character string in the message header that is being compared to the string specified by the *conchars* operand, or whether blanks are to be part of the header string when encountered in it. If EBCDIC blanks are to be counted as part of the header string, this operand also specifies whether some other hexadecimal character is to be ignored when encountered in the header string.

Format: YES, NO, or *char*. *char* is a single character that may be specified in either character or hexadecimal format. If character format is specified, it may be unframed or framed with C'' or CL1''characters. If hexadecimal format is specified, it must be framed with X'' or XL1'' characters.

Default: BLANK=YES.

Note: This operand is ignored unless the *conchars* operand is also specified.

YES specifies that the EBCDIC blank character (X'40') is to be ignored by this macro whenever it is encountered in the header character string being checked against the control character string specified by the *conchars* operand. For example, if BLANK=YES is coded and an eight-byte field in the header is being checked by this macro, a blank appearing in the fifth byte of the field is ignored, and the sixth through ninth bytes are considered to be the last four bytes of the field (assuming that no blanks are coded in the sixth through ninth bytes).

NO specifies that the EBCDIC blank character is to be treated in the same way as any other character when it is encountered by this macro in the header string being compared to the string specified by *conchars*.

char specifies that the single character replacing *char* ignored by this macro whenever it is encountered in the header string being compared to the string specified by the *conchars* operand. That is, the macro automatically skips over the character without performing a comparison and goes on to check the next character in the header. If BLANK=*char* is coded and *char* is not the EBCDIC blank character, the EBCDIC blank is not ignored by this macro when it is encountered in the header string but is compared to the character in the corresponding space in the *conchars* string, in the same way as any other character.

```
conchars
```

Function: Specifies the character or character string that, if found in the header as the next nonblank field, causes execution of the lock function.

Format: One to eight nonblank characters in character or hexadecimal format. If character format is used, the string may be unframed or framed with C'' or CLn'' characters. If hexadecimal format is used, the string must be framed with X'' or XLn'' characters.

Default: None. Specification is optional.

Note: If this operand is omitted, the lock function is performed unconditionally. If the next field in the header does not match this operand, the function is not performed.

For an external LU in extended lock mode, control characters are meaningful only for the header of the message being processed at the time the external LU is placed in lock mode. The LOCK macro does not examine headers or messages entered by an external LU already in extended lock mode for control characters.

```
{EXTEND }
{MESSAGE}
```

Function: Specifies the type of lock mode required.

Format: EXTEND or MESSAGE.

Default: MESSAGE

Note: EXTEND specifies that the external LU transmitting the message to be placed in lock mode until an UNLOCK macro is executed.

MESSAGE specifies that the external LU transmitting the message is to be placed in lock mode for the duration of the message and its reply.

For logical messages, the LOCK macro is restricted to an incoming message formed by blocking two or more incoming physical transmissions. In this case, PROCESS=NO must be specified on the SETEOM macro.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	Character string not found.
X'00000004'	Destination not specified.
X'00000008'	Destination not an open process entry.
X'0000000C'	Source is already locked.
X'FFFFFFFC'	Zero length buffer.

LOCOPT Macro

The LOCOPT macro:

- Provides access to fields in the option table
- Specifies the destination whose option field is requested
- May be used outside an MH, for example, an ERRORMSG exit routine
- Must be coded within an MCP assembly but does not have to execute within a message handler.

LOCOPT enables you to obtain the address of any option field for the appropriate terminal-table entry. The address of the desired field or a not-found indicator is placed in a user-specified register. A user-written routine may then examine and modify the contents of the option field.

If you specify this macro outside of an MH (but not outside the MCP), prior to issuing the macro, you must set up register 13 to point to an 18-word save area that you have provided. You must perform standard save-area linkage conventions prior to issuing LOCOPT and again prior to returning to TCAM.

Supported Resources and General Requirements: ALL.

Valid Subgroup: Inblock, inbuffer, inheader, outbuffer, and outheader. It is also valid outside message handlers and in all TCAM exit routines in the MCP.

NAME	OPERATION	OPERANDS

[symbol]	LOCOPT	opfield
		,(register)
		[,STATION={'name' }]
		{fieldname }
		{(register) }
		{<u>**</u> }

```
opfield
```

Function: Specifies the name of the option field whose address is desired.

Format: Must be the name of an option field as defined by an OPTION macro, and must conform to the rules for assembler language symbols.

Default: None. Specification is required.

Note: If the option field is not found, LOCOPT does not execute, a X'00000004' code is set in register 15, and the specified register will contain a X'000000FF'. If the default register 15 is used and the option field address cannot be located, register 15 will contain a fullword of zeros on return.

```
(register)
```

Function: Specifies the register into which the address of the desired option field is to be placed.

Format: A register number or the equated name of a register enclosed in parentheses. Registers 2 through 11 or 15 may be used.

Default: (15) Within a MH subgroup
(1) Outside a MH subgroup (but within the MCP).

Note: Do not specify register 15 outside an MH.

```
STATION={'name'     }
        {fieldname  }
        {(register) }
        {**         }
```

Function: Specifies the resource whose option field address is desired.

Format: 'name', fieldname, (register), or **. *'name'* must be specified in character format. *fieldname* must conform to the rules for assembler language symbols and must not be specified with framing characters. *(register)* is the actual register number or the equated name of a register enclosed in parentheses. Registers 0 through 15 may be used.

Default: STATION=** (** is the default if the macro is specified within an MH. If specified outside an MH, a resource name must be coded.)

Note: name is the name of a resource defined by a TERMINAL macro. *fieldname* is the symbolic name of the field containing the resource name, the length of which must correspond to the MAXLEN= operand of the TTABLE macro. *(register)* specifies a register containing the address of the field that contains the resource name.

** specifies that you want the option field address of the external LU. If specified in the incoming group, LOCOPT addresses option fields for the originating resource; if specified in the outgoing group, LOCOPT addresses option fields for the destination.

If ** is specified in an AMH handling messages to or from an application program, LOCOPT locates the option fields in the process entry for the queue to which the GET or READ is directed (if LOCOPT is issued in the outgoing group) or the fields in the process entry for the queue to which the PUT or WRITE is directed (if LOCOPT is issued in the incoming group).

symbol

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

One of the following return codes is set in register 15:

Within a MH Subgroup

Code	Meaning
X'00000000'	The option field address cannot be located.
Non-zero	Address of the option field.

Outside MH Subgroup (when register 15 is not used to return the option field address)

Code	Meaning
X'00000000'	LOCOPT function successful.
X'00000004'	Option field not found; LOCOPT did not execute.
X'00000008'	Resource not found; invalid name.

Note: When outside an MH subgroup and register 15 is used to return the option field address, the contents are unpredictable if the option field cannot be found.

LOG Macro

The LOG macro:

- Enables you to log complete messages or message segments.

LOG enables you to maintain a record of incoming or outgoing message traffic on a sequential medium. Message segments or full messages, as determined by the placement of LOG macros in an MH, are placed on an output device. If logging is for both message segments and complete messages in the same MCP, a data control block must be provided for each function. The various types of logs, and the corresponding MH subgroups in which LOG appears are:

- Either entire incoming message, individual logical messages, or portions of individual logical messages (inblock)
- Incoming header segments only (inheader)
- All incoming segments (inbuffer)
- Complete incoming messages (inmessage)
- All outgoing segments (outbuffer)
- Complete outgoing messages (outmessage).

When LOG is specified in an inblock subgroup, its position relative to the SETEOM macro determines what is logged. If LOG appears *before* SETEOM, TCAM logs the entire incoming message by segment. If LOG appears *after* SETEOM, either individual logical messages or portions of individual logical messages are logged, depending on whether deblocking or blocking operations are being performed. For deblocking operations, individual logical messages are logged. This requires that PROCESS=YES be specified on SETEOM which would cause any data following an EOM indicator to be processed in a new buffer. For blocking operations, portions of individual logical messages are logged. This requires that PROCESS=NO be specified on SETEOM which would cause any data following an EOM indicator to be discarded. When LOG is specified in an inbuffer or outbuffer subgroup, segments are logged in the sequence of different multiple-segment messages handled by the message handlers. In this case, segments of different multiple-segment messages handled at about the same time are likely to be intermixed on the logging medium. When segments are logged, their buffer prefixes are logged with them. The 20-byte control area connected with each buffer unit is not logged.

LOG may appear at any point in an MH subgroup in which it is used. However, the results of any alteration of segments or messages by macros preceding LOG in the subgroup will appear in the log. For example, if LOG is preceded by DATETIME, a logged header segment will contain the date or time, as specified in DATETIME, depending on the location of the date and time in a multiple-segment message.

LOG may be specified in any subgroup of an MH and may be used more than once in a subgroup if desired. The message log may be maintained on any available output medium. You must supply, define, and open the message log data sets. For each log data set used to log complete messages, a logtype entry in the terminal table must be defined by a LOGTYPE macro (this is not necessary if only segments are logged). The buffer prefix is logged with the message if message logging is used, but the size, status, and source fields are the only fields that are meaningful to the user. For a description of TCAM's message logging facility, see the chapter on the *TCAM Utilities* manual titled "Message Logging Service Facility." For information on specifying the message log data set, see "Defining Data Sets" in the *TCAM Installation Guide*.

Note: When logging segments after a FORWARD or DESTFLD macro with multiple destinations, the last character of the first destination is overlaid with an unprintable character. This byte is restored at the inmessage subgroup and thus appears if messages are logged.

Supported Resources and General Requirements: ALL.

Valid Subgroups: All subgroups.

NAME	OPERATION	OPERANDS

`[symbol]`	`LOG`	`{dcbname }`
		`{typename}`

```
{dcbname }
{typename}
```

Function: Specifies the name of the data control block or the logtype entry used for logging.

Format: Must conform to rules for assembler language symbols.

Default: None. This operand is required.

Note: *dcbname* is the name of the data control block for the message log data set and is used if the macro is specified in the inblock, inheader, inbuffer, outheader, or outbuffer subgroup.

typename is the name of a logtype entry in the terminal table and is used if the macro is specified in the inmessage or outmessage subgroup.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	Good return.
X'00000004'	*dcbname* does not match the name of a valid data control block or *typename* does not match the name of a logtype entry in the terminal table. LOG does not execute.
X'00000008'	Zero length buffer.

LOGTYPE Macro

The LOGTYPE macro:

- Initializes TCAM's logging facility
- May not be omitted if TCAM's logging facility is to be used for logging complete messages, and is unnecessary if only segments are logged
- Must have a corresponding LOG macro specified in an inmessage or outmessage subgroup of the MH to log complete messages
- If coded, must be specified among the macros defining the terminal table and must not be the last such macro.

Supported Resources and General Requirements: ALL.

Valid Subgroup: Not applicable.

NAME	OPERATION	OPERANDS
typename	LOGTYPE	dcbname ,BUFSIZE=size ,QUEUES=form

BUFSIZE=size

Function: Specifies the size of the buffers to handle messages destined for the logging medium.

Format: Unframed decimal integer greater than or equal to 76.

Default: None. This operand is required.

Maximum: 65,535.

dcbname

Function: Specifies the name of the data control block for the log data set.

Format: Must conform to the rules for assembler language symbols.

Default: None. This operand is required.

Note: This name must be the same as the name of the DCB macro specifying the log data set.

QUEUES=form

Function: Specifies where the messages are to be queued while awaiting transfer to the logging medium.

Format: DR, DN, MO, MR, or MN.

Default: None. This operand is required.

Note: DR specifies reusable disk queues. DN specifies nonreusable disk queues. MO specifies main-storage-only queues. MR specifies main-storage queues with reusable disk backup. MN specifies main-storage queues with nonreusable disk backup.

If MR or DR is specified, the original destination is automatically designated as the alternate destination for zone reorganization. (See "Defining Data Sets" in the *TCAM Installation Guide.)* Unlike the TERMINAL and TPROCESS macros, there is no ALTDEST= operand for the LOGTYPE macro.

`typename`

Function: Specifies the name of the LOGTYPE macro and is the same as the *typename* operand of a LOG macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Return Codes

None.

MCPGEN Macro

The MCPGEN macro:

- Generates message handler sections and MCP definition statements required to support certain TCAM system service programs
- Must be issued in the correct section of the MCP, depending on the function requested
- May be issued more than once.

The MCPGEN macro may be used to generate any or all of the following TCAM facilities:

- The AMH used by the extended operator control and online retrieval system service programs
- The AMH used by the save/restore message queues (SMQ) system service program
- The AMH used by the internodal awareness system service program for extended networking
- The AMH used by the internodal sequence number synchronization (ISNS) system service program for extended networking
- The TPROCESS and PCB macros required for the extended operator control, online retrieval, SMQ, internodal awareness, and ISNS system service program.

The MCPGEN macros consolidate elements of the MCP required for the TCAM system service programs. Naming conventions required by the online retrieval, extended operator control, SMQ, internodal awareness, and ISNS system service programs are followed; correct cross references between TPROCESS macros, PCBs, and AMHs are assured.

The location of MCPGEN macros in the MCP depends on the function requested; that is:

- If the macro is generating TPROCESS macros, MCPGEN must be issued after the OPTION macros and before the last entry in the terminal table (the name of which is specified in the LAST= operand of the TTABLE macro).
- If the macro is generating an MH, MCPGEN must be issued in an MCP section where a STARTMH macro could be issued.
- If the macro is generating PCB macros, MCPGEN must be issued in the MCP data set definition section.

The MCPGEN macro does not create option fields.

If you are coding MCPGEN macros outside of the model MCP, you will need to invoke two DSECTs containing labels required by MCPGEN. The IEDTCSD macro generates these DSECTs, along with many others. If you do not code the IEDTCSD macro in your MCP, you may generate only the two DSECTs required by the MCPGEN macro by coding the following three statements in a non-executable part of your MCP:

```
TVT     DSECT
        TCSTVT
        DKJFHD
```

Supported Resources and General Requirements: ALL, FHP.

Valid Subgroups: Not applicable.

NAME	OPERATION	OPERANDS

[symbol]	MCPGEN	GEN={MH }
		{TPROC}
		{PCB }
		,PROG={OPCTL }
		{SMQ }
		{NODEPATH}
		{NODESYNC}
		[,ALTDEST={entry}]
		{OPRB }
		[,BUFSIZE={252 }]
		{number}
		[,QUEUES={DN}]
		{DR}
		{MN}
		{MO}
		{MR}

```
ALTDEST={entry}
        {OPRB }
```

Function: Specifies the destination of responses to basic operator control commands issued by extended operator control (as a result of an operator issuing an extended operator command).

Format: *entry* or OPRB. *entry* is the name of a single process entry in the terminal table.

OPRB is the extended operator control command response queue.

Default: ALTDEST=OPRB.

Note: See the TPROCESS ALTDEST= operand for more information concerning the ALTDEST= operand.

Coding the ALTDEST= operand has no effect on the operation of extended operator control.

```
BUFSIZE={252    }
        {number}
```

Function: Specifies the BUFSIZE= operand on the PCB generated for the message retrieval system service program when PROG=OPCTL,GEN=PCB is specified. When PROG=SMQ,GEN=PCB is specified, the BUFSIZE= operand applies to the PCB for the save/restore message queues system service program.

Format: See the PCB macro.

Default: BUFSIZE=252.

Maximum: 65,535.

Note: This value determines the buffer size for messages queued by the online retrieval SSP or the restore function of the save/restore message queues SSP and may be chosen to correspond to buffer sizes normally used. Efficiency of multiple retrieval operations could be improved if a value for BUFSIZE= is chosen that is the smallest of:

- the APWAS= value of the INTRO macro
- 3072
- the typical message size of messages on disk queues.

For the most efficient save/restore operations, choose a value for BUFSIZE= that is equal to the largest application work area.

```
GEN={MH   }
    {TPROC}
    {PCB  }
```

Function: Specifies what kind of MCP data is to be generated for the related PROG= keyword.

Format: MH, TPROC, or PCB.

Default: None. This operand is required.

MH specifies that a message handler is to be generated. The internodal awareness and internodal sequence number synchronization system service programs use the same message handler; to generate their message handler, code either:

```
MCPGEN GEN=MH,PROG=NODEPATH
```

or

```
MCPGEN GEN=MH,PROG=NODESYNC
```

But not both.

This form of the MCPGEN macro is issued in an MH section of the MCP.

TPROC specifies that the required TPROCESS macros are to be generated. This form of the MCPGEN macro is issued in the terminal table section of the MCP. Message queues can be specified by coding the QUEUES= operand to use any valid queuing form for a TPROCESS entry.

PCB specifies that one or more required PCBs are to be generated. This form of the MCPGEN macro is issued in the PCB section of the MCP.

```
PROG={OPCTL   }
     {SMQ     }
     {NODEPATH}
     {NODESYNC}
```

Function: Specifies the program for which the MCP macros are to be generated.

Format: OPCTL, SMQ, NODEPATH or NODESYNC.

Default: None. Specification is required.

Note: OPCTL specifies the extended operator control and online retrieval system service programs.

SMQ specifies the save/restore message queues system service program.

NODEPATH specifies the internodal awareness system service program.

NODESYNC specifies the internodal sequence number synchronization system service program.

```
QUEUES={DN}
       {DR}
       {MN}
       {MO}
       {MR}
```

Function: Specifies how and where the message queues for the system service program TPROCESS queues are to be maintained.

Format: DN, DR, MN, MO, or MR.

Default: QUEUES=DR.

See the QUEUES= operand of the TPROCESS macro for an explanation of the queuing options and considerations when GEN=TPROC is coded.

The QUEUES= operand can be coded only when GEN=TPROC is coded in the MCPGEN macro. If the QUEUES= operand is coded without GEN=TPROC, the QUEUES= operand will be ignored.

The affected system service program TPROCESS entries for the QUEUES= operand are:

MCPGEN Operands	**Queues**
GEN=TPROC,PROG=OPCTL	OPRA, OPRB, RARA, and RARB
GEN=TPROC,PROG=SMQ	SMQR
GEN=TPROC,PROG=NODEPATH	NODEPATH
GEN=TPROC,PROG=NODESYNC	NODESYNC, NODESYNR

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Return Codes

None.

MHGET Macro

The MHGET macro:

- Transfers the contents of the current buffer into a user-specified work area, or
- Gives you the address of the data in the current buffer
- Is optional in serially reusable, user-written, open or closed subroutines.

The work area is a part of the MH or the called subroutine and should not be confused with an application-program work area. MHGET does not affect or depend on any functional macros used in the MH or subroutine. Therefore, any order of functional macros and MHGET will work. MHGET may be specified either to return the address of the first data character in the buffer or to move the entire data contents of the buffer into a user-specified work area. The entire message, or as much of the message as will fit into the user-specified work area, is moved. The MHGET macro moves data from multiple buffers into the work area consecutively, if the length of the data previously moved in is left in the DATLEN field of the work area prefix. If the work area is to contain only the data from a single buffer and you want the data to start at the beginning of the work area, the DATLEN field must be zeroed before issuing the MHGET macro.

The format of the work area is:

WKLEN	DATLEN	PRFSTAT	UNRES	Data . . .
0----1----	2----3----	4----	5----	6-n ----

Bytes	Field Name	Description
0-1	WKLEN	User-supplied data length of the work area. (Does not include the 6-byte work-area prefix length.)
2-3	DATLEN	Length of data moved, supplied by MHGET on return to the user. You must zero this field before issuing a single-buffer MHGET or the first MHGET of a series of multiple-buffer MHGETs. For the second and subsequent MHGETs, when moving the data from multiple buffers consecutively into a single work area, this field must contain the data length put there by the previous MHGET.
4	PRFSTAT	The status byte from the buffer prefix indicating: X′00′ Header last X′01′ Text last X′02′ Header not last X′03′ Text (not header) not last X′08′ Duplicate Header
5	UNRES	The number of unused reserve characters
6-n	Data	The first to *n-5* data characters in the buffer.

Supported Resources and General Requirements: ALL.

Valid subgroups: Inbuffer, inheader, outbuffer, and outheader.

NAME	OPERATION	OPERANDS

[symbol]	MHGET	REG=(register)
		[WORK={(register)}]
		{name }
		[,RESERVE={YES}]
		{<u>NO</u> }

```
REG=(register)
```

Function: Returns the address of the first data character in the buffer in the specified register. The specified register is the lower of a pair of consecutive registers. The next higher register contains the following:

Byte 0	Zero
Byte 1	Buffer status (PRFSTAT)
Bytes 2-3	Length of the data in the buffer

Format: (register) may be the actual register number or the equated name of a register enclosed in parentheses. Registers 2 through 11 may be used.

Default: None. Must be specified if WORK= operand is not specified.

```
RESERVE={YES}
        {NO }
```

Function: Moves unused reserve spaces into the work area in front of the data. The unused spaces are put into the UNRES field of the work area header.

Format: YES or NO.

Default: RESERVE=NO.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

```
WORK={(register)}
     {name      }
```

Function: Provides the name or address of the user-supplied work area.

Format: *(register)* is the actual register number or the equated name of a register enclosed in parentheses. Registers 2 through 12 may be used. *name* is the name of a work area that must be aligned on a halfword boundary.

Default: None. WORK=*(register)* or *name* is required if REG= is not specified.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	MHGET was successful
X'00000004'	Work area too small
X'0000000C'	Empty buffer processing permitted.

MHPUT Macro

The MHPUT macro:

- Transfers the contents of a user-specified work area into the current buffer
- Resets the scan pointer to the beginning of data in the buffer
- Is optional in serially reusable, user-written open or closed subroutines.

MHPUT will transfer whatever is in the user-specified work area into the current buffer, leaving room for the GROUP specified reserve characters. Up to 65,535 (X'FFFF') characters may be written, and the routine will set up the maximum number of units per buffer to contain the data. Care should be taken that data (in the header), which subsequent functional macros will need, is not destroyed or overlaid by execution of the MHPUT macro. Since MHPUT sets the scan pointer to the beginning of data in the buffer, before issuing a macro that uses the scan pointer, you should be certain that the scan pointer is properly set for that macro. If these precautions are observed, MHPUT does not affect or depend on any functional macros.

MHGET and MHPUT are to be used in a message handler as opposed to GET and PUT macros for an application program. The function of MHGET is to transfer the contents of the current buffer into a user-specified work area. The function of MHPUT is to transfer the contents of a user-specified work area into the current buffer. The implication is that these macros may be used to perform intermediate operations on messages before they are passed on to an application program or to an output destination buffer. The flow might be:

MHGET	MHGET
Process or	process
MHPUT	FORWARD

The format of the work area is:

Unused	DATLEN	Reserved	Data . . .
0----1----	2----3----	4----5----	6-n ----

Bytes	Field Name	Description
0-1		Unused and unchanged by MHPUT.
2-3	DATLEN	Length of the user-supplied data starting at the first position of the data field of the work area. Maximum data length can be 65,535 (X'FFFF').
4-5		Reserved.
6-n	Data	User data to be written (MHPUT).

Note: If the multiple destination facility of the FORWARD macro is being used, care should be taken that the data up to the EOA character(s) is not destroyed by the MHPUT macro. If MHPUT is coded in the inheader subgroup when there is multiple destination data in the header, the multiple destination data must be saved, by using MHGET, and restored as part of the data area used by MHPUT. It is necessary to save the multiple destination data in the header because of the X'DF' that is inserted in the data as a place-holder by the FORWARD macro. The place-holder identifies the starting point of the scan to verify the secondary destinations which is performed by the INMSG subgroup. For more information on multiple destinations, refer to the FORWARD macro.

MHPUT

Supported Resources and General Requirements: ALL.

Valid subgroups: Inbuffer, inheader, outbuffer, and outheader.

NAME	OPERATION	OPERANDS

[symbol]	MHPUT	WORK={(register)}
		{name }
		[,RESERVE=integer]

```
RESERVE=integer
```

Function: Allows the user to specify the reserve character count.

Format: Unframed decimal integer.

Default: None. Specification is optional.

Maximum: 221.

Note: When this operand is not specified, MHPUT begins transferring the data into the header buffer starting at the first position following the unused GROUP reserves. This operand allows the user to override that specification.

If RESERVE=0 is specified, MHPUT begins moving data into the buffer beginning at the first byte after the buffer prefix. If RESERVE=1 is specified, MHPUT begins moving data into the second byte after the prefix, etc. If the reserve character count you specify is greater than the maximum allowable in the GROUP, the maximum allowable is used instead of what was specified by this operand, and the return code X'0000000C' is placed in register 15.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

```
WORK={(register)}
     {name      }
```

Function: Provides the name or address of the user's work area.

Format: *(register)* is the actual register number or the equated name of a register enclosed in parentheses. Registers 2 through 12 may be used. *name* is the name of a work area that must be aligned on a halfword boundary.

Default: None.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	MHPUT was successful.
X'00000004'	MHPUT could not allocate enough units; data is truncated. This return code is also set when the number of units per buffer has reached the maximum.
X'0000000C'	Reserves specified are too large; data was moved.
X'00000010'	Length of work area not initialized.
X'00000014'	Zero length buffer.

MRCHECK Macro

The MRCHECK macro:

- Is an expansion of the functions of the CHECKPT macro and is intended as a means of notifying you, through an option field, of one of the following:
  - An environment checkpoint has occurred during the processing of a multiple routing function if S=Y is coded at startup.
  - TCAM abnormally terminated during a multiple routing function.
- Places the input sequence number of the current message and a Receive Status into the first option field of a terminal entry or the MRCHECK option field indicated by USE=MRCHECK.
- Causes an incident checkpoint record to be taken of the option fields for the originating external LU or application program, after the MHCHECK macro initializes the first option field, and before the message is routed to multiple destinations or the destinations in a distribution list (TYPE=D).
- Causes an incident checkpoint record to be taken of the option fields for the originating external LU or application program after the message is routed to multiple destinations or destinations in a distribution list, and the first option field is cleared.
- Requires that the SEQUENCE macro be coded in the inheader subgroup to provide an input sequence number.
- Executes unconditionally.

You must supply an option field initially defined as three bytes of zeros (not an address-type constant). This option field must be specified by the USE=MRCHECK operand of the option macro. If omitted, the first option field will be used.

The MRCHECK macro function is performed only if the FORWARD macro was coded with an EOA= operand or if a distribution list was specified as the destination of this message. If no multiple routing function has been requested or if the option field specified is not defined for this station, the MRCHECK function is not performed.

If the MRCHECK function is to be completed, the option field is set to the input sequence number of this message and to X'02' before the processing of the multiple routing function. Then an incident record checkpoint is taken for the sequence numbers and the option fields. Upon completion of the multiple routing function, the option field is set to zero and another checkpoint is taken.

If this macro is coded, you should interrogate this option field during the execution of your startup message routine. After taking whatever action you choose, you must reset the field to zeros in order to ensure the reliability of your first checkpoint.

Supported Resources and General Requirements: ALL.

Valid subgroup: Inmessage.

Name	Operation	Operands
[symbol]	MRCHECK	(no operands)

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

None.

MSGEDIT Macro

The MSGEDIT macro:

- Inserts specified characters at specified locations in a message
- Deletes specified characters from a message and contracts remaining data
- Replaces deleted characters with other specified characters
- Dynamically allocates buffer units to contain data inserted in message segments
- Should not be used to alter the data in the first buffer unit of a message containing a fixed header prefix.

The MSGEDIT macro allows you to edit incoming and outgoing messages from within a message handler. Each editing operation performed by MSGEDIT falls into one of three categories: they are insertion, deletion, or replacement.

An insert operation is one in which specified characters are inserted at a specified point in a message, with no characters being deleted in the operation. The operands of MSGEDIT allow characters to be inserted:

- At a single point in a message
- At a specified offset from the beginning of each message segment
- Whenever a certain character string appears in a message
- After every n bytes of message data, where n is a number specified by you
- At the current position of the scan pointer.

The inserted data may consist of a single character, an ordinary character string, or identical characters. If the MSGEDIT macro is issued in an inheader or outheader subgroup, the insert operation is performed only on a single segment of a message. This is usually the first segment but may be a subsequent segment if the message has a multiple-buffer header and the MSGEDIT macro is issued in a portion of the subgroup that is processing header fields in the second or subsequent segments. (The manner in which inheader and outheader subgroups are executed for multiple-buffer headers is described in the chapter titled "Coding the Message Handler" in *TCAM Installation Guide*). If the MSGEDIT macro is issued in an inbuffer or out buffer subgroup, the insert function might add a new destination name to the destination field in a message header or insert idle characters into an outgoing message going to a station with a printer requiring such characters prevent "printing on the fly" during carriage- return operation. For other uses, see the examples below. If the insert function is for a character string that may cross buffer boundaries, MSGEDIT should be coded in the inblock subgroup.

A remove operation is one in which a specified character string is removed from a message. The remaining data in the buffer is contracted to fill in the space left after the deletion. However, the MSGEDIT will not remove a character string if that removal deletes all data in the buffer. If there is a possibility of this occurring (such as the removal of an EOT at the end of a message), an idle or another suitable character replaces the deleted string. You may remove:

- A single character string
- A specified character string whenever it appears
- A specified number of bytes of data whenever a certain character string appears
- The data located in a specified section of a buffer
- The data at the current position of the scan pointer.

If MSGEDIT is code in an inheader or outheader subgroup, data is removed from only a single header segment of a message. If MSGEDIT is coded in an inbuffer or outbuffer subgroup, data is removed from all message segments. Also if the remove function is for a character string that may cross buffer boundaries, MSGEDIT should be coded in the inblock subgroup.

The replace operation is one in which a specified character string is removed and replaced with another. You may replace:

- A single character string
- A specified character string whenever it appears
- A specified number of bytes of data whenever a certain character string appears
- The data located in a specified buffer
- Substitute one destination for another

If a substituted character string is longer or shorter then the deleted string, TCAM automatically spreads or contracts the data remaining in the buffer to fit the new string; buffer units are allocated as needed to accommodate the new data. The replacement function might substitute one destination name for another in the header, or replace a specified character with a logical record delimiter that is recognized by application-program GET macros.

If the buffer containing a message segment is not long enough to accommodate additional data inserted by a MSGEDIT macro, additional buffer units are automatically added to the buffer as needed. Because of internal requirements, for insert operations MSGEDIT automatically allocates a minimum of one buffer unit before execution. This unit, whether it eventually contains data or not, remains with the buffer. This fact should be considered when determining the unit count for the MCP. The effect of this automatic unit allocation can be diminished by the multiple-operand feature of the MSGEDIT macro. Empty units at the end of a buffer are automatically deallocated when the buffer is passed to an INMSG or OUTMSG macro; deallocated units are returned to the available-unit queue.

If no buffers are available when you request an insert operation, MSGEDIT will not process any data ane will set a return code and return control to the MH. An EDUNITS macro is issued in the initialization section of the MCP provides a specified number of emergency MSGEDIT units to be used by the MSGEDIT macro if there are insufficient buffer units to perform insertion.

Up to 31 separate insert and remove operations may be specified by issuing a single MSGEDIT macro having up to 31 groups of four-position positional operands. Since any operation that can be performed with multiple identical groups of operands also can be performed with a single group, duplicate groups of operands are not allowed in the same MSGEDIT macro. Assembler language restrictions on the length of a macro operand also apply.

The MSGEDIT macro operand field consists of from one to 31 groups of four operands each and a single keyword operand that is coded as the last operand of the macro. Each group of positional operands is enclosed in parentheses, and each specifies a single insert, remove, or replace operation. If you wish to perform many insert, remove, or replace operations on messages, you may code either a single MSGEDIT macro having many groups, or you may code several MSGEDIT macros with one group each.

If end-of-block checking is specified for the message handler, the MSGEDIT macro may not be used in an incoming group to expand or contract the amount of data in the buffer. If the MSGEDIT macro is used in this manner and an error occurs in transmission, the retransmission of the block results in duplicated data if the buffer is contracted and lost data if it is expanded. This restriction does not apply if static allocation and deallocation of buffers is specified for receive operations (by coding N as the first suboperand of the PCI= operand of the line group DCB macro).

When multiple groups of positional operands are coded for a MSGEDIT macro, rather than multiple MSGEDIT macros each with a single group, data inserted by one operation is not considered to be part of the message segment when another operation is being performed. For example, if one group caused a B character to be inserted after every A character in the message and another group of the same MSGEDIT macro specified that a C character be inserted after every B character in the message, no C character would be inserted after a B character, which was itself inserted as a result of an A character being encountered in the message segment by the MSGEDIT macro.

Insertion, removal or replacement of data using MSGEDIT macro always results in a movement of data in the buffer, even when a MSGEDIT macro specified only a single remove operation and the replacement string is equal in length to character string being replaced. As a rule, when a MSGEDIT macro operates on any data in a buffer, all of the data between characters affected by the first insert or remove operation and end of the buffer are shifted once by means of MVC instructions issued internally by TCAM.

When coded in an inblock subgroup, the MSGEDIT macro can delete or replace a character string that extends across the message segments and buffer boundaries. The MSGEDIT macro has certain limitations:

- When issued in an inheader or outheader subgroup, MSGEDIT acts only upon on header segment of messages having multiple-buffer headers. The segment acted upon is the one being processed by the inheader or outheader subgroup at the time MSGEDIT is executed. Moreover, a MSGEDIT macro issued in an inheader subgroup specifies that NYC is to replace BOS whenever the latter character string occurs in the header, and if the header ends midway through the first message segment, even though it is outside of the header.

- Any character string in an operand specified in character format rather than as hexadecimal data cannot include a comma or a right parenthesis. If the character field requires the use of these characters, the field must be specified in hexadecimal format.

- Avoid performing message-editing functions that either add or delete data to the left of the scan pointer while you are performing sequential processing of header fields. Because the scan pointer points to a particular physical location in the buffer, rather than to a particular character, addition of data to the left of the scan pointer results in the shifting of the original scan pointer to the left. The following example illustrates the possible problem resulting from the improper placement of a MSGEDIT macro in the message handler:

```
SETSCAN   C'X'
ORIGIN    5
MSGEDIT   ((I,C'INSERT',1))
FORWARD   DEST=5,EOA=*
```

After the SETSCAN and the ORIGIN macros are executed, the buffer might look like this:

```
prefix   X   TERMA   TERMB   TERMC  *  message data
                 |
                 |scan pointer
```

After the MSGEDIT macro executes, the buffer looks like this:

```
prefix   INSERT   X   TERMA   TERMB   TERMC  *  message data
                 |
                 |scan pointer
```

When the FORWARD macro executes, the origin (TERMA) is considered to be the first destination (TERMB).

To avoid such problems, you may follow these guidelines:

1. Perform as many of the functions of MSGEDIT as possible in an inbuffer or outbuffer subgroup rather than in an inheader or outheader subgroup.
2. Perform all functions of MSGEDIT that affect header fields either before all sequential processing of header fields begins, or after all sequential processing of header fields has been completed.

Examples are

```
a.      MSGEDIT     ((I,C'INSERT',1))
        SETSCAN     C'X'
        ORIGIN      5
        FORWARD     DEST=5,EOA=*

b.      SETSCAN     C'X'
        ORIGIN      5
        FORWARD     DEST=5,EOA=*
        MSGEDIT     ((I,C'INSERT',25,SCAN))
```

MSGEDIT adjusts the scan pointer backwards for you for one special case. This is a delete (or replace) function specifying the scan pointer itself to the TO operand. Examples of this are:

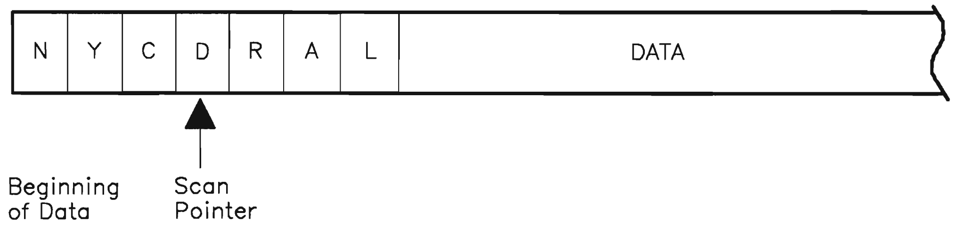
```
MSGEDIT     ((R,,25,SCAN))
MSGEDIT     ((R,C'INSERT',25,SCAN))
```

In these examples, if the delete or replace function deletes more bytes than exist between the scan pointer and the end of data in the buffer after the macro executed, the scan pointer would, if not adjusted, erroneously point beyond the end of the data in the buffer and prevent any subsequent sequential processing. Therefore, in these cases, the scan pointer is moved backward a distance equal to:

- The length of the data deleted, or
- The length of data removed less the length of data inserted

With one exception (when both the AT and the TO operands are coded as or with SCAN), the first byte of a string of data to be removed (as determined by the AT operand) must be to the left of, or in the same position as, the last byte of the string of data to be removed (as determined by the TO operand). See "Examples" later in this topic.

The first character in a character string to be deleted (as specified by the AT operand) must not be to the right of the last character of the character string (as specified by the TO operand). If both operands specify the same byte, that byte only is removed. As an example, consider the following initial portion of a buffer with the scan pointer located at D:



MSGEDIT

A MSGEDIT macro coded:

```
MSGEDIT ((R,CL3'BOS',SCAN,4))
```

would result in the character D being replaced with the string BOS in the buffer.

A MSGEDIT macro coded:

```
MSGEDIT ((R,CL3'BOS',CL1'D',CL3'RAL'))
```

would result in BOS inserted after D; this macro is coded to remove the character between D and R and replace it with BOS. Since there is no character between D and R, none is removed, but BOS is still inserted.

MSGEDIT Summary:

1. The MSGEDIT macro allow a maximum of 31 groups of functions.
2. Multiple MSGEDIT macros versus multiple function groups with one MSGEDIT macro is trade off between speed and flexibility.
3. Group execution is independent of coding position of the MSGEDIT macro but is dependent on buffer contents.
4. All groups work on the original message contents; inserted characters from one function cannot be the search argument for another group.
5. Inserted characters from a MSGEDIT macro can be the search argument for a subsequent MSGEDIT macro.
6. The data moved is from the first AT= position to the end of the buffer.
7. MSGEDIT does not require reserved space but allocates units automatically if additional space is needed.
8. MSGEDIT reallocates units automatically if remove functions result in empty units.
9. Execution of MSGEDIT in the inheader/outheader subgroups acts on only one header buffer.
10. Execution of MSGEDIT across buffers is possible only in an inblock subgroup.
11. The maximum length of a contiguous character string is 8 bytes, which is the size of the AVT work area.

General Information

Supported Resources and General Requirements: ALL.

Valid subgroups: Inblock, inbuffer, inheader, outbuffer, and outheader.

NAME	OPERATION	OPERANDS
[symbol]	MSGEDIT	((group1),(group2),...)
		[,BLANK={char}]
		{NO }
		{YES }
		[,LAST={ALL}]
		{YES}
		{NO }

```
BLANK={char}
      {NO  }
      {YES }
```

Function: This operand specifies whether EBCDIC blank characters are to be ignored when encountered in searching the message for a field, or whether blanks are to be considered part of the field, or whether blanks are to be considered part of the field when encountered. If the EBCDIC blanks are to be counted when found, this operand also specifies whether another hexadecimal character is to be ignored when encountered in searching the message fur a field.

Format: char, NO, or YES. *Char* is a single character that may be specified in either character or hexadecimal format. If character format is specified, it may be unframed or framed with C″ or CL1″ characters. If hexadecimal format is specified, it must be framed with X″ or XL1″ characters.

Default: BLANK = YES

Note: char specifies that the single character replacing *char* is to be ignored by this macro whenever it is encountered in the header. That is, the macro automatically skips over the character without checking it. If BLANK = *char* is coded and *char* is not the EBCDIC blank, the EBCDIC blank is treated in the sane way as any other character.

NO specifies that the EBCDIC blank character is to be treated in the same way as any other character when it is encountered by this macro in the message.

YES specifies that the EBCDIC blank character (X'40') is to be ignored by this macro whenever it is encountered in a message. For example, if BLANK = YES is coded and an eight-byte field is being acted upon by this macro, a blank appearing in the fifth byte is ignored and the sixth through ninth bytes are considered to be the last four bytes of the field (assuming that no blanks are coded in the sixth through ninth bytes.) If a delete or replace operation is being performed, any embedded blanks are automatically deleted or replaced and are not included in the count.

```
((group1),(group2),...)
```

Function: Each group specifies a single insert, delete, or remove function.

Format: Each group contains (function, data, AT, TO) operands. They must be provided in the order shown, enclosed in parentheses, and separated from each other by a comma. If more than one group of operands is included, the AT operand for each group must be specified as a character string.

Default: None. At least one group is required. Multiple groups may be coded only if the AT suboperand for each group is specified as characters and each character string specified as AT suboperand begins with a different character.

Maximum: 31 groups.

Notes: Due to the complexity of the macro, the operands are explained individually below.

```
LAST={ALL}
     {YES}
     {NO }
```

Function: Specifies whether to insert characters at the end of the existing message and at the end of each buffer in which the message is contained.

Format: ALL, YES, or NO.

Default: LAST=NO

ALL specifies that characters are to be inserted at the end of each buffer containing the existing message and also at the end of the message. It can be used only with the insert-character function.

YES specifies that characters are to be inserted at the end of the existing message and can be used only with the insert characters function.

NO specifies that characters are not to be inserted at the end of the message.

The following coding example shows the only way in which LAST=YES/ALL can be coded:

```
MSGEDIT ((I,characters,offset,)),LAST={YES}
                                      {ALL}
```

An offset value must be coded; this value is recalculated for each message.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Required when *data* is otherwise, it is optional.

The structure of each group of positional operands is:

```
function
operand      data operand       AT operand                TO operand

{I      }    ,[{characters }]   ,[{characters      }]     ,[{characters,...}]
{R[A][T]}      {(hexform,n)}      {offset          }        {offset        }
               {(register)  }     {(integer,opfield)}       {SCAN          }
```

function operand	data operand	AT operand	TO operand
	{DELIMIT }	{SCAN }	{(count) }
	{CONTACT }		{*0* }

Insertion

```
function operands
  {I}
```

Function: Specifies that this group will perform an insert function.

Format: I.

I specifies that an insert function is to be performed. The data specified in the *data* operand is the data inserted in the message.

Note: An overlay of message data can occur when the MSGEDIT macro is used to insert message data and the INSRTFLD macro is also used to insert message data. The overlay condition occurs when MSGEDIT inserts message data preceding the area used to contain the INSRTFLD data. This insertion of message data causes the INSRTFLD pointers in the FHP to point to the wrong area of the message.

```
data operands
  {characters }
  {(hexform,n)}
  {(register )}
```

Function: Specifies the data to be inserted in the message.

Format: characters, (hexform,n), (register). characters may be one to eight nonblank characters in character or hexadecimal format. If character format is used, framing C″ or CLn″ characters can be used. If hexadecimal format is used, framing X″ or XLn″ characters must be specified, *(hexform,n)* must be coded within parentheses. *hexform* is a single character in hexadecimal or character format surrounded by framing X″ or C″ characters. *n* is a decimal integer and must not be framed. *(register)* must specify a register (2-12) enclosed in parentheses.

Default: None.

Maximum: *n* may have a maximum length of the length of one buffer unit.

Note: If messages are to be translated, inserted characters should be in EBCDIC; if they are not to be translated, inserted characters should be in transmission code used by the station.

(hexform,n) specifies that the single character represented by *hexform* is to be inserted the number of times indicated by *n*. The inserted characters will be contiguous. This operand may be used to insert idle characters in outgoing message.

(register) is the number of a register that has been loaded with the address of a field containing the data. The field must have as its first byte the hexadecimal length (maximum of 255 bytes) of the data following, not including the length byte.

```
AT operands
  {characters       }
  {offset           }
  {(integer,opfield)}
  {SCAN             }
```

Function: Specifies the location at which the insertion is to be made.

Format: characters, offset, (integer,opfield), or SCAN. *characters* may be one to eight nonblank characters in character or hexadecimal format. If character format is used, framing C″ or CLn″ characters can be used. If hexadecimal format is used, framing X″ or XLn″ characters must be specified, *offset* is a decimal integer specified without framing characters.

(integer,opfield must be coded within framing parentheses. *integer* may be specified either in decimal or hexadecimal format. If hexadecimal format is used, the value must be coded within framing X″ or XL″ characters. *opfield* is the name of a halfword option field defined by an OPTION macro.

Default: SCAN.

Maximum: For *offset*, 32,767. For *integer*, 32,767, or X′7FFF′.

Note: If the *data* operand is *(register)* the AT operand must be *offset* or SCAN.

If more than one group of operands is included in this macro, the AT operand for each group must be specified as characters, and each character string specified as an AT operand must begin with a different character.

characters specifies a string immediately following which the data specified in the *data* operand is to be inserted. The *characters* string must be located within a single buffer unless this MSGEDIT macro is coded within an inblock subgroup. If the MSGEDIT macro is included in an inheader or outheader subgroup, the specified data is inserted each time this string is encountered in the message header. If the MSGEDIT macro is issued in an inbuffer or outbuffer subgroup, the specified data is inserted each time this string is encountered anywhere in the message.

If *characters* is coded, either *characters* or *(count)* should be specified as a the TO operand. If SCAN is specified as the TO operand, TCAM assumes a count of zero has been specified for TO. If an offset is specified for the TO operand, TCAM assumes that the offset is count.

offset specifies the number of bytes beyond the buffer prefix immediately following which the first character specified in the data operand is to be inserted.

If the MSGEDIT macro is specified in an inheader or outheader subgroup, *offset* applies to a single header segment only, and insertion of data occurs only once. If the macro is coded in an inbuffer or outbuffer subgroup, data is inserted at the specified offset in every segment of the message. For example, if an offset of two is specified, the first character inserted immediately follows the contents of the second byte beyond the the buffer prefix.

(integer,opfield) specifies that the data coded for the *data* operand is to be inserted after every number of bytes specified by *integer* If *integer* is 20, for instance, the data specified in the *data* operand is inserted after every 20 bytes of message. An *integer* greater than 1 may *not* be specified if the originating station (when coded in an inbuffer subgroup) or the destination station (when coded in an outbuffer subgroup) uses the block-checking feature for this message. Insertion occurs in both the header and text. *opfield* is the name of an option field assigned to the origin (if MSGEDIT is coded in the incoming group) or to the destination (if MSGEDIT is coded in the outgoing group).

The option field must be initialized by the OPDATA= operand of the TERMINAL or TPROCESS macro (it may be set to a halfword of zero).

Coding considerations for *(integer,opfield)*

- This MSGEDIT macro suboperand may be coded in an inbuffer or outbuffer subgroup only.
- Only one group of positional operands may be specified.
- *characters* or *(hexform,n)* must be specified for the *data* operand.

SCAN specifies that insertion is to begin with the character at which the scan pointer is currently pointing. (See the description of the scan pointer in "Designing the Message Handler" in the *TCAM Installation Guide.*) This operand may be specified only when the MSGEDIT is issued in an inheader or outheader subgroup.

`TO operand`

TO operand is invalid with the insert function.

Insertion examples

Example 1: Insertion of a single character string after specified data in header buffer. The following MSGEDIT macro can be coded in an inheader subgroup to add destination RAL to the list of destinations specified in the message header whenever the destination NYC is specified in the header.

```
EDIT1 MSGEDIT ((I,CL3'RAL',CL3'NYC'))
```

Note that only the function, data and AT operands are coded for this macro; the TO operand must not be coded for an insert operation.

Example 2: Insertion of a character string after every 50 bytes of message data. The following MSGEDIT macro might be coded in the outbuffer subgroup of a message handler assigned to an application program to cause the EBCDIC Z character (specified as a record delimiter by the RECDEL= operand of the TPROCESS macro creating the process entry specified as the destination of the message data.

```
EDIT1 MSGEDIT ((I,C'Z',(50,EDITOPT)))
```

EDITOPT is the name of a halfword option field created by an OPTION macro and initialized with zeros by the OPDATA= operand of the TPROCESS macro creating the process entry specified as the destination of this message.

Example 3: Insertion of idle characters. The following macro, when coded in an inbuffer or outbuffer subgroup, causes 13 EBCDIC idle characters (X'17') to be inserted whenever a period is encountered in a message.

```
EDIT1 MSGEDIT ((I,(X'17',13),CL1'.'))
```

Example 4: The following MSGEDIT macro, coded in an outbuffer subgroup causes three EBCDIC SYN control symbols (X'32') to be inserted in each segment, beginning at the thirty-first byte.

```
EDIT1 MSGEDIT ((I,(X'32',3),31))
```

Deletion

```
function operands
  {R(A)(T)}
```

Function: Specifies that a remove function is to be performed and also whether the characters delimiting the beginning and the end of removal are themselves to be removed.

Format: R,RA,RT,RAT,RTA.

R specifies that a remove function is to be performed; any data delimited by the AT operand and the TO operand is to be removed from the message. If no data is specified by the AT operand or by the TO operand, MSGEDIT removes one byte of data beginning at the location currently designated by the scan pointer. Data remaining in a buffer after a deletion is contracted to fill the space left by the deleted data.

A specifies that removal is to begin with the first character of the character string specified by the AT operand. If A is omitted, removal begins with the character immediately following the last character in the string specified in the AT operand. If A is coded in a group, a character string must be coded at the AT operand.

T specifies that removal is to end with the last character of the string specified in the TO operand; if T is not coded, the character immediately preceding the first character removed. If T is coded in a group, a character string must be specified as the TO operand.

```
data operands
  {CONTRACT}
```

Function: Specifies that the data remaining in a buffer after deletion is to be contracted to fill the space originally occupied by the deleted data.

Default: CONTRACT

CONTRACT specifies that after the appropriate data has been deleted from a message segment, succeeding characters in the buffer are to be moved to overlay deleted characters. If contraction results in one or more empty units at the end of the buffer, these are released when the segment leaves the incoming or outgoing group of the MH.

```
AT operands
  {characters  }
  {offset      }
  {SCAN        }
```

Function: Specifies the location of the beginning of the string to be deleted.

Format: *characters, offset,* or SCAN. *characters* may be one to eight nonblank characters in character or hexadecimal format. If character format is used, framing C″ or CLn″ characters can be used. If hexadecimal format is used, framing X″ or XLn″ characters must be specified, *offset* is a decimal integer specified without framing characters.

Default: SCAN.

Maximum: For *offset* 32,767. For *integer*, 32,767, or x′7FFF′.

Note: If the *data* operand is *(register)* the AT operand must be *offset* or SCAN.

If more than one group of operands is included in this macro, the AT operand for each group must be specified as characters, and each character string specified as an AT operand must begin with a different character.

characters specifies a string that delimits the beginning of the data to be removed. The *characters* string must be located within a single buffer unless this MSGEDIT macro is coded within an inblock subgroup. If the A suboperand of the *function* operand is included, deletion begins with the first character of this string; if A is not included, deletion begins with the character immediately following the last character of this string. If A is coded in the *function* operand and the TO operand is coded (0) or is omitted, only the string specified in the AT operand is deleted. If the MSGEDIT macro is included in an inheader or outheader subgroup, deletion occurs each time the character string is encountered in the message header. If the macro is issued in an inbuffer or outbuffer subgroup, deletion occurs each time the character string is encountered in the message.

If *characters* is coded, either *characters* or *(count)* should be specified as a the TO operand. If SCAN is specified as the TO operand, TCAM assumes a count of zero has been specified for TO. If an offset is specified for the TO operand, TCAM assumes that the offset is count.

offset specifies the number of bytes beyond the buffer prefix immediately following which deletion of data is to begin.

If the MSGEDIT macro is specified in an inheader or outheader subgroup, *offset* applies to a single header segment only, and deletion of data occurs only once. If the macro is coded in an inbuffer or outbuffer subgroup, data is deleted at the specified offset in every segment of the message. For example, if an offset of two is specified, the first byte whose contents are deleted from a segment is the third byte beyond the buffer prefix.

SCAN specifies that deletion is to begin with the character at which the scan pointer is currently pointing. (See the description of the scan pointer in "Designing the Message Handler" in the *TCAM Installation Guide.*) This operand may be specified only when the MSGEDIT is issued in an inheader or outheader subgroup.

```
TO operands
  {characters}
  {offset    }
  {SCAN      }
  {(count)   }
  {(0)       }
```

Function: Specifies the end of the character string to be deleted.

Format: characters, offset, SCAN, (count), or (0). characters may be one to eight nonblank characters in character or hexadecimal format. If character format is used, framing C" or CLn" characters can be used. If hexadecimal format is used, framing X" or XLn" characters must be specified. *offset* is a decimal integer specified without framing characters. *(count)* must be coded within its framing parentheses, and is a decimal integer specified without framing characters.

Default: (0).

Maximum: Both *offset* and *(count)* have a maximum value of 65,535.

Note: characters indicates the location of the last character to be deleted. If the T suboperand of the *function* operand is coded, deletion ends with the last character of the string specified; otherwise, deletion ends with the character immediately preceding the first character string. The entire string must be located in the buffer that contains the delimiter specified by the AT operand, since deletion must begin and end in the same buffer. If both the AT operand and the TO operands specify character strings, TCAM assumes that the the first byte of the TO string is to the right of the last byte of the AT string.

offset specifies an offset from the beginning of the data in a message segment; this offset defines the end of the string to be deleted in this operation. If the offset is 20, for instance, the character occupying the twentieth byte from the beginning of the data in the buffer is deleted. The offset must specify a byte that is in the same buffer as, and either in the same position as or to the right of, the first byte of data removed (as specified by the AT operand); each deletion must begin and end in the same buffer. If the offset specified by the TO operand is identical with the offset specified byte the AT operand, the single character located at this offset is removed If the offset is beyond the end of the buffer, data will be deleted to the end of the buffer.

If this MSGEDIT macro is specified in an inheader or outheader subgroup, *offset* applies to a single header segment only and deletion occurs only once. If the macro is coded in an inbuffer or outbuffer subgroup, data is deleted from each segment.

SCAN specifies that the character preceding the character indicated by the current position of the scan pointer is to be the last character deleted in this remove operation. This operand may be coded only in a MSGEDIT macro issued in an inheader or outheader subgroup. If SCAN is coded for both the AT and the TO operands, and R is specified in the *function* operand, the single character located at the current position of the scan pointer is deleted.

(count) and its default value (0) specify the number of bytes of data to be deleted, starting with the byte immediately following the AT operand. If the AT delimiter is a character string and if A is coded in the *function* operand, the amount of data removed is equal to the sum of the number of characters in the AT delimiter string plus the number of bytes specified by *(count)*. If the integer specified by *(count)* is greater than the number of bytes between the AT delimiter and the end of the buffer are deleted. A count of zero indicates that no data is to be deleted (except for the characters in the AT delimiter if A is coded in the *function* operand); if the TO operand is omitted, a count of 0 is assumed. If A is coded in the *function* operand and a string is coded in the AT operand, the string is removed each time it is encountered if (0) is coded or if no TO operand is specified.

Deletion examples

Example 1: The following MSGEDIT macro can be issued in the inheader or outheader subgroup. It causes the 10 bytes immediately following the current position of the scan pointer to be deleted; all data following the deleted 10 bytes in the first message segment is shifted to the left 10 spaces to fill in the space occupied by the deleted data. The shift may result in an empty unit at the end of this buffer; empty units are dynamically deallocated and returned to the available unit queue when the buffer leaves the MH group.

```
EDIT1 MSGEDIT ((R,,,(10)))
```

Note that the data and AT operands were not coded because their default values are CONTRACT and SCAN, respectively. The figure illustrates how a single buffer containing an entire message might look after this macro executed. Assume that the units are 60 bytes long, that the buffer consists of 2 units, and that the second unit contains only 6 bytes of data before the MSGEDIT macro is executed. Assume also that all of the 10 bytes immediately following the position of the scan pointer contain the meaningful data (none of the bytes contain blanks).

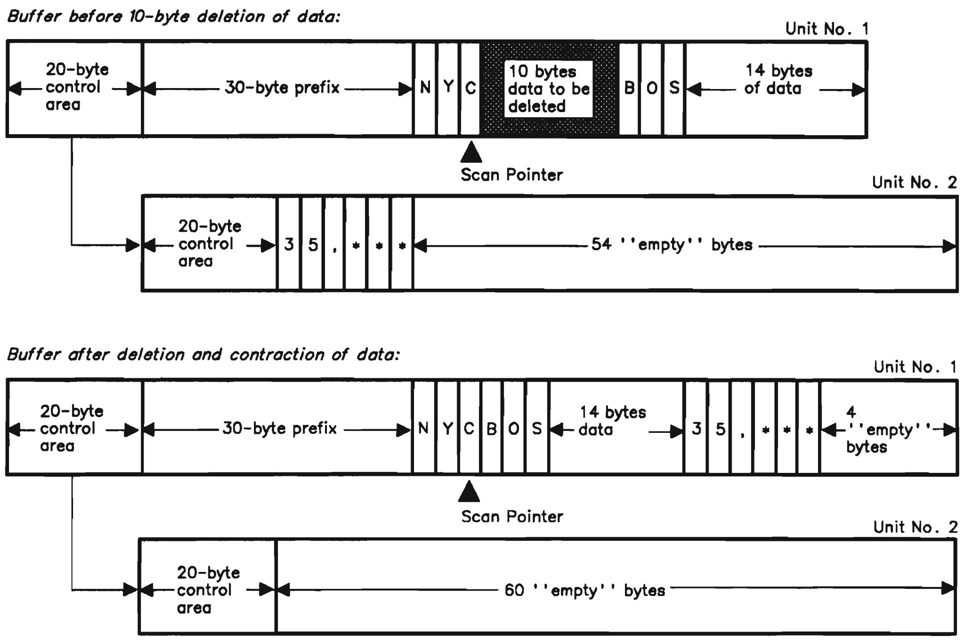


Figure 2-1. Deletion of Data from a Message Segment followed by Contraction of the Segment (UNITSZ=60 and BUFSIZE=120)

Notice that after the deletion was made, all data following the deleted characters was moved 10 bytes to the left; as a result the second unit contains no meaningful data after the remove operation.

Example 2: The following MSGEDIT macro, coded in an inbuffer or outbuffer subgroup, causes the character string OUT and the 10 characters immediately following OUT to be deleted from a message segment wherever OUT appears in a segment. Data following the 13 deleted characters is moved to the left to fill the gap caused by the deletion. If EBCDIC blanks are counted as characters in this example.

```
EDIT1 MSGEDIT ((R,CONTRACT,CL3'OUT',(10))),BLANK=NO
```

Example 3: The following macro, coded in an inbuffer or outbuffer subgroup causes the characters occupying the tenth through twentieth bytes of each buffer to be deleted and the remaining data to be shifted left to fill the gap caused by deletion.

```
EDIT1 MSGEDIT ((R,,9,20))
```

Example 4: The following MSGEDIT macro, coded in an inheader or outheader subgroup causes the character occupying the byte to which the scan pointer is currently pointing to be removed, subsequent data in the segment is shifted 1 byte left to fill the gap. Note the defaults.

```
EDIT1 MSGEDIT ((R,,,SCAN))
```

Replacement

```
function operands
  {R(A)(T)}
```

Function: Specifies that a remove function is to be performed and also whether the characters delimiting the beginning and the end of removal are themselves to be replaced.

Format: R,RA,RT,RAT,RTA.

R specifies that a remove function is to be performed; any data delimited by the AT operand and the TO operand is to be removed from the message and replaced with the data specified by the *data* operand.

A specifies that replacement is to begin with the first character of the character string specified by the AT operand. If A is omitted, replacement begins with the character immediately following the last character in the string specified in the AT operand. If A is coded in a group, a character string must be coded at the AT operand.

T specifies that replacement is to end with the last character of the string specified in the TO operand; if T is not coded, the character immediately preceding the first character replaced. If T is coded in a group, a character string must be specified as the TO operand.

```
data operands
  {characters }
  {(hexform,n)}
  {(register )}
  {DELIMIT    }
```

Function: Specifies the data to replace the characters removed from the message.

Format: characters, (hexform,n), (register), DELIMIT characters may be one to eight nonblank characters in character or hexadecimal format. If character format is used, framing C″ or CLn″ characters can be used. If hexadecimal format is used, framing X″ or XLn″ characters must be specified and *(hexform,n)* must be coded within parentheses. *hexform* is a single character in hexadecimal or character format surrounded by framing X″ or C″ characters. *n* is a decimal integer and must not be framed. *(register)* must specify a register (2-12) enclosed in parentheses.

Default: None.

Maximum: *n* may have a maximum length of the length of one buffer unit.

Note: If messages are to be translated, inserted characters should be in EBCDIC; if they are not to be translated, inserted characters should be in transmission code used by the station.

(hexform,n) specifies that the single character represented by *hexform* is to be inserted the number of times indicated by *n*. The inserted characters will be contiguous. This operand may be used to insert idle characters in outgoing message.

(register) is the number of a register that has been loaded with the address of a field containing the data. The field must have as its first byte the hexadecimal length (maximum of 255 bytes) of the data following, not including the length byte.

DELIMIT is valid only if the *function* operand specifies a replace function. DELIMIT specifies that the character in the RECDEL= operand of the TPROCESS macro whose name is entered as this message's destination is to replace the character string delimited by the AT and TO operands. This character is recognized by the AT and TO operands. This character is recognized by the application program's GET macro as the delimiter of a variable-length record. The MSGEDIT macro in which this operand is coded is normally located in the outbuffer subgroup of the MH for the application program or inbuffer subgroup for the LU from which the message is received. If MSGEDIT is located in an inheader subgroup, only a single header segment is scanned for the character to be replaced. The destination queue must be identified by means of a FORWARD macro before the MSGEDIT macro is issued. If the destination of this message is not an application program, the MSGEDIT group containing DELIMIT does not execute at the end of the buffer, these are released when the segment leaves the incoming or outgoing group of the MH.

```
AT operands
  {characters  }
  {offset      }
  {SCAN        }
```

Function: Specifies the location of the beginning of the string to be replaced.

Format: *characters, offset,* or SCAN. *characters* may be one to eight nonblank characters in character or hexadecimal format. If character format is used, framing C'' or CLn'' characters can be used. If hexadecimal format is used, framing X'' or XLn'' characters must be specified, *offset* is a decimal integer specified without framing characters.

Default: SCAN.

Maximum: For *offset*, 32,767. For *integer*, 32,767, or X'7FFF'.

Note: If the *data* operand is *(register)* the AT operand must be *offset* or SCAN.

If more than one group of operands is included in this macro, the AT operand for each group must be specified as characters, and each character string specified as an AT operand must begin with a different character.

characters specifies a string that delimits the beginning of the data to be replaced. The *characters* string must be located within a single buffer unless this MSGEDIT macro is coded within an inblock subgroup. If the A suboperand of the *function* operand is included, replacement begins with the first character of this string; if A is not included, replacement begins with the character immediately following the last character of this string. If A is coded in the *function* operand and the TO operand is coded (0) or is omitted, only the string specified in the AT operand is replaced. If the MSGEDIT macro is included in an inheader or outheader subgroup, replacement occurs each time the character string is encountered in the message header. If the macro is issued in an inbuffer or outbuffer subgroup, replacement occurs each time the character string is encountered in the message.

If *characters* is coded, either *characters* or *(count)* should be specified as a the TO operand. If SCAN is specified as the TO operand, TCAM assumes a count of zero has been specified for TO. If an offset is specified for the TO operand, TCAM assumes that the offset is count.

offset specifies the number of bytes beyond the buffer prefix immediately following which replacement of data is to begin.

If the MSGEDIT macro is specified in an inheader or outheader subgroup, *offset* applies to a single header segment only, and replacement of data occurs only once. If the macro is coded in an inbuffer or outbuffer subgroup, data is replaced at the specified offset in every segment of the message. For example, if an offset of two is specified, the first byte whose contents are replaced from a segment is the third byte beyond the buffer prefix.

SCAN specifies that replacement is to begin with the character at which the scan pointer is currently pointing. (See the description of the scan pointer in "Coding the Message Handler" in the *TCAM Installation Guide.*) This operand may be specified only when the MSGEDIT is issued in an inheader or outheader subgroup.

```
TO operands
  {characters}
  {offset    }
  {SCAN      }
  {(count)   }
  {(0)       }
```

Function: Specifies the end of the character string to be replaced.

Format: characters, offset, SCAN, (count), or (0)characters may be one to eight nonblank characters in character or hexadecimal format. If character format is used, framing C" or CLn" characters can be used. If hexadecimal format is used, framing X" or XLn" characters must be specified, *offset* is a decimal integer specified without framing characters. *(count)* must be coded within its framing parentheses, and is a decimal integer specified without framing characters.

Default: (0).

Maximum: Both *offset* and *(count)*, have a maximum value of 65,535.

Note: *characters* indicates the location of the last character to be replaced. If the T suboperand of the *function* operand is coded, substitution ends with the last character of the string specified; otherwise, substitution ends with the character immediately preceding the first character string. The entire string must be located in the buffer that contains the delimiter specified by the AT operand, since replacement must begin and end in the same buffer. If both the AT operand and the TO operands specify character strings, TCAM assumes that the the first byte of the TO string is to the right of the last byte of the AT string.

offset specifies an offset from the beginning of the data in a message segment; this offset defines the end of the string to be replaced in this operation. If the offset is 20, for instance, the character occupying the twentieth byte from the beginning of the data in the buffer is replaced. The offset must specify a byte that is in the same buffer as, and either in the same position as or to the right of, the first byte of data replaced (as specified by the AT operand); each substitution must begin and end in the same buffer. If the offset specified by the TO operand is identical with the offset specified by the AT operand, the single character located at this offset is replaced. If the offset is beyond the end of the buffer, data will be replaced to the end of the buffer.

If this MSGEDIT macro is specified in an inheader or outheader subgroup, *offset* applies to a single header segment only and substitution occurs only once. If the macro is coded in an inbuffer or outbuffer subgroup, data is replaced from each segment.

SCAN specifies that the character preceding the character indicated by the current position of the scan pointer is to be the last character replaced in this operation. This operand may be coded only in a MSGEDIT macro issued in an inheader or outheader subgroup. If SCAN is coded for both the AT and the TO operands, and R is specified in the *function* operand, the single character located at the current position of the scan pointer is replaced.

(count) and its default value (0) specify the number of bytes of data to be replaced, starting with the byte immediately following the AT operand. If the AT delimiter is a character string and if A is coded in the *function* operand, the amount of data replaced is equal to the sum of the number of characters in the AT delimiter string plus the number of bytes specified by *(count)*. If the integer specified by *(count)* is greater than the number of bytes between the AT delimiter and the end of the buffer are replaced. A count of zero indicates that no data is to be replaced (expect for the characters in the AT delimiter if A is coded in the *function* operand); if the TO operand is omitted, a count of 0 is assumed. If A is coded in the *function* operand and a string is coded in the AT operand, the string is replaced each time it is encountered if (0) is coded or if no TO operand is specified.

Replacement examples

Example 1: Replacement of one character string in a message with another character string. The following MSGEDIT macro is coded in the inheader subgroup; it causes the character string BOS to be replaced with the character string OMAHA wherever the former string appears in the first segment of the message. (Remember, however, that the entire character string BOS must occur in the segment in order for MSGEDIT to operate on it.) If a buffer is not long enough to accommodate the longer character string, TCAM will automatically allocate extra units as the buffer as needed.

```
EDIT1 MSGEDIT ((RA,CL5'OMAHA',CL3'BOS'))
```

Note that no TO operand is coded. The A in the function operand specifies that the AT character string is to be deleted and that the O in OMAHA is to be positioned at the location occupied by the B in BOS. If the TO operand had been coded BOS, all data in the segment between the first BOS and a second BOS would be deleted. If the segment contained no second BOS, the remove operation would not take place; the macro would not execute, and control would pass to the next macro.

Example 2: Replacement Insertion of a record delimiter. The following macro, when coded in an inbuffer or outbuffer subgroup, causes the logical record delimiter X to be substituted for the character D wherever the latter character appears in a message. The X delimiting character, which would be coded in the RECDEL= operand of a TPROCESS macro, is considered by a GET command issued by an application program to be the delimiter of a logical record.

```
EDIT1 MSGEDIT ((RA,DELIMIT,CL1'D'))
```

Example 3: The following macro, coded in an inbuffer or outbuffer subgroup causes the data between every R character and E character to be replaced with the character string EPLAC. If the data being deleted occupies less space than the replacement string, the data in the buffer is automatically spread out to make room for the insertion, and another buffer unit is added to the buffer if necessary. If the data being deleted occupies more space than the replacement string, data to the right of the replacement string is automatically moved to the left to fill the gap.

```
EDIT1 MSGEDIT ((R,CL5'EPLAC',CL'R',CL1'E'))
```

Multiple Group Examples:

Example 1: A single MSGEDIT macro can be issued in an inheader subgroup to accomplish the editing functions described earlier. This macro causes RAL to be inserted after each NYC in the first segment and also causes BOS to be replaced with OMAHA each time BOS appeared in the first segment.

```
EDIT1 MSGEDIT ((I,CL3'RAL,Cl3'NYC'),(RA,CL5'OMAHA',CL3'BOS'))
```

Example 2: The following MSGEDIT macro, coded in an inblock, inbuffer, or outbuffer subgroup, causes the EBCDIC idle characters (X'17') wherever a blank occurs (BLANK=NO must be specified for this operation). In addition, the character string DOLLARS is replaced with the character $ wherever it appears, and two blanks are inserted after each period in the message.

```
EDIT1 MSGEDIT ((RA,(XL1'17'13),Cl1' '),(RA,CL1'$',CL'DOLLARS'),
(I,CL2' ',CL1'.')),BLANK=NO
```

When multiple operations are performed by a single MSGEDIT macro, the data inserted by the operations is not considered when remove operations are performed. The macro should be in an inblock subgroup if the character string 'DOLLARS' can extend across buffer boundaries. Thus in the above example, the two blanks inserted after each period would not be replaced by 13 idle characters each.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	Successful execution
X'00000004'	Either no units available or trying to remove from or insert into buffer prefix or reserved area.

MSGFORM Macro

The MSGFORM macro specifies the type of chain in which the FM header will be sent.

Supported Resources and General Requirements: SNA.

Valid subgroups: Outheader.

NAME	OPERATION	OPERANDS

[symbol.]	MSGFORM	FMHCHN={FOC}]
		{OC }
		[,conchars]
		[,BLANK={char}]
		{NO }
		{<u>YES</u> }

```
BLANK={char}
      {NO  }
      {YES }
```

Function: This operand specifies whether EBCDIC blank characters are to be ignored when encountered in searching the message for a field, or whether blanks are to be considered part of the field when encountered. If EBCDIC blanks are to be counted when found, this operand also specifies whether another hexadecimal character is to be ignored when encountered in searching the message for a field.

Format: YES, NO, or *char*. *char* is a single character that may be specified in either character or hexadecimal format. If character format is specified, it may be unframed or framed with C'' or CL1'' characters. If hexadecimal format is specified, it must be framed with X'' or XL1'' characters.

Default: BLANK=YES.

Note: *char* specifies that the single character *char* is to be ignored by this macro whenever it is encountered in the header. That is, the macro automatically skips over the character without checking it. If BLANK=*char* is coded and *char* is not the EBCDIC blank, the EBCDIC blank is treated in the same was as any other character.

NO specifies that the EBCDIC blank character is to be treated in the same way as any other character when it is encountered by this macro in the message.

YES specifies that the EBCDIC blank character (X'40') is to be ignored by this macro whenever it is encountered in a message. For example, if BLANK=YES is coded and an eight-byte field is being acted upon by this macro, a blank appearing in the fifth byte is ignored, and the sixth through ninth bytes are considered to be the last four bytes of the field (assuming that no blanks are coded in the sixth through ninth bytes.) If a remove or replace operation is being performed, any embedded blanks are automatically removed or replaced and are not included in the count.

char specifies that the single character replacing *char* is to be ignored by this macro whenever it is encountered in the header string being compared to the string specified by the *conchars* operand.

This operand is meaningless unless the *conchars* operand is also specified.

```
conchars
```

Function: Specifies the character or character string that, if found in the header buffer as the next nonblank field, causes execution of the function.

Format: One to eight nonblank characters in character or hexadecimal format. If character format is used, the string may be unframed or framed with C'' or CLn'' characters. If hexadecimal format is used, the string must be framed with X'' or XLn'' characters.

Default: None. Specification is optional.

Note: If this operand is omitted, this MSGFORM function is performed unconditionally. If the operand is coded and the next field in the message header matches this operand, the function is performed.

```
FMHCHN={FOC}
       {OC }
```

Function: Specifies the type of chain in which the FM header will be sent.

Format: FOC or OC.

Default: None. This operand is required.

Note: FOC indicates that the FM header is to be sent in an RU that is a first RU of chain element (FOC). OC indicates that the FM header is to be sent as an only RU of chain element (OC).

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

Code	Meaning
X'00000000'	Successful execution. Indicator has been set to send an FM header as OC or FOC. *Note:* No verification is made to see that the FM header indicator in the RH of the buffer is turned on or that the FM header is contained within the header buffer. This is the user's responsibility.
X'00000004'	Destination not an SNA LU.
X'00000008'	Indicator already set to send the FM header as OC or FOC.
X'0000000C'	RU type not FMDATA.

MSGGEN Macro

The MSGGEN macro:

- Generates an unqueued message
- May be used to send a null message (RH only) to an external LU
- May be issued more than once in a subgroup.

MSGGEN generates a message if the bits contained in the error mask operand match the bits set in the message error record. (See "Appendix A. Message Error Record" for a description of the message error record.) If a zero mask is specified, the message is generated unconditionally. The generated message bypasses all normal functions, such as MH processing, queuing, logging, and buffer requesting. No error recovery is attempted for messages generated by MSGGEN. Errors encountered while attempting to send such a message are ignored. The MSGGEN macro informs the user of an error more rapidly than does the ERRORMSG macro, but it does not return the header of the message in error as the latter macro does.

If MSGGEN is specified in an inmessage subgroup, the generated message, as specified by an operand, is sent to the origin; if specified in an outmessage subgroup, the message is sent to the destination. MSGGEN may be specified more than once within a subgroup.

The MSGGEN macro causes TCAM default responses to be generated also; therefore, all user responses must be generated before MSGGENs are executed.

Supported Resources and General Requirements: SNA

Valid subgroups: Inmessage and outmessage.

NAME	OPERATION	OPERANDS

[symbol]	MSGGEN	`[mask]`
		`[,{'message'}]`
		`  {fieldname}`
		`[,CONNECT={AND }]`
		`          {NAND}`
		`          {OR  }`
		`{,RH={X 'rh' }]`
		`     {'label'}`
		`     {CL3' ' }`

```
CONNECT={AND }
        {NAND}
        {OR  }
```

Function: Specifies the type of logical connection to be made between the mask and the message error record.

Format: AND, NAND, or OR.

Default: CONNECT = OR.

Note: AND specifies that the macro is to be executed only if *all* of the bits specified by the *mask* operand are on in the message error record. OR specifies that the macro is to be executed if *any* bit specified by the *mask* operand is on in the message error record. NAND specifies that the macro is to be executed only if *all* of the bits specified by the *mask* operand are off in the message error record.

A message generated by MSGGEN may not be directed to an application program when specified in an inmessage subgroup.

No error recovery is attempted for a message generated by MSGGEN. Errors encountered while attempting to send such a message are ignored.

```
mask
```

Function: Specifies the 5-byte bit configuration used to test the message error record for the message. (See the description of the message error record in Appendix A.)

Format: Decimal or hexadecimal. If hexadecimal format is used, framing characters must be specified. If X'' is used, leading zeros must be coded. If XL5'' is used, leading zeros may be omitted.

Default: None. Specification is optional.

Note: Specifying an all-zero mask causes unconditional execution.

```
{'message'}
{fieldname}
```

Function: Specifies the message or the location of the message to be sent to the origin or destination, depending on whether MSGGEN is issued in an inmessage or outmessage subgroup, respectively.

Format: *'message'* or *fieldname*. *'message'* is the actual message to be sent; it must be framed, either by C'', CLn'', X'', or XLn'' framing characters. *fieldname* is the symbolic name of the field containing the message. It must not be specified with framing characters.

Default: None. This operand is optional if RH= is coded; otherwise it is required.

Note: The field referred to by *fieldname* must have as its first byte the hexadecimal count equal to the number of bytes in the rest of the field.

```
RH={X'rh'   }
   {'label'}
   {CL3''   }
```

Function: Specifies a 3-byte request header (no message—a null RU) or the location of the RH to be sent to the origin or destination, depending on whether MSGGEN is issued in an inmessage or outmessage subgroup, respectively.

Format: X'*rh*', '*label*' or CL3''. X'*rh*' is the actual 3-byte RH to be sent. It must be framed by X'' framing characters. '*label*' is the symbolic name of the field containing the RH. It must be specified in hexadecimal or character format within single quotes. CL3'' is the actual 3-byte RH to be sent. It must be 3 bytes in length and framed by CL3'' framing characters.

Default: None. This operand is optional.

Note: TCAM expects certain RH values. If you specify an invalid RH, TCAM corrects the specified bit setting in error. See the *TCAM Installation Guide* for more information on a skeleton RH. See also the *Systems Network Architecture Reference Summary* for the complete RH format.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

None.

MSGTYPE Macro

The MSGTYPE macro:

- Controls the path of a header buffer through an MH
- May be used more than once in a subgroup.

MSGTYPE enables you to categorize incoming or outgoing messages into two or more message types, each of which you process in a different manner.

Use of MSGTYPE is optional. Any number of MSGTYPE macros may be issued within a subgroup, provided that they all examine the same position in the buffer for the message-type characters. Only one field in a header per inheader or outheader subgroup may contain message-type characters, and only one sequence of code beginning with a MSGTYPE macro is executed in an inheader or outheader subgroup for any one incoming or outgoing message. MSGTYPE may be used only within inheader and outheader subgroups.

The use of MSGTYPE is discussed in the chapter of the *TCAM Installation Guide* titled "Coding the Message Handler."

Supported Resources and General Requirements: ALL.

Valid subgroups: Inheader and outheader.

NAME	OPERATION	OPERANDS

[symbol]	MSGTYPE	`[{conchars    }]`
		` {,TABLE=name}`
		` {,EXIT=name }`
		`[,BLANK={YES  }]`
		`        {NO   }`
		`        {char}`

```
BLANK={YES }
      {NO  }
      {char}
```

Function: Specifies whether EBCDIC blank characters are to be ignored when encountered in the character string in the message header being compared to the string specified by the *conchars* operand, or whether blanks are to be part of the header string when encountered in it. If EBCDIC blanks are to be counted as part of the header string, this operand also specifies whether another hexadecimal character is to be ignored when encountered in the header string.

Format: YES, NO, or *char*. *char* is a single character that may be specified in either character or hexadecimal format. If character format is specified, it may be unframed or framed with C'' or CL1'' characters. If hexadecimal format is specified, it must be framed with X'' or XL1'' characters.

Default: BLANK=YES.

Note: YES specifies that the EBCDIC blank character (X'40') is to be ignored by this macro whenever it is encountered in the header character

string being checked against the control character string specified by the *conchars* operand. For example, if BLANK=YES is coded and an eight-byte field in the header is being checked by this macro, a blank appearing in the fifth byte of the field is ignored, and the sixth through ninth bytes are considered to be the last four bytes of the field (assuming that no blanks are coded in the sixth through ninth bytes).

NO specifies that the EBCDIC blank character is to be treated in the same way as any other character when it is encountered by this macro in the header string being compared to the string specified by *conchars*.

char specifies that the single character replacing *char* is to be ignored by this macro whenever it is encountered in the header string being compared to the string specified by the *conchars* operand. That is, the macro automatically skips over the character without performing a comparison and checks the next character in the header. If BLANK=*char* is coded and *char* is not the EBCDIC blank character, the EBCDIC blank is not ignored by this macro when it is encountered in the header string, but is compared to the character in the corresponding space in the *conchars* string, in the same way as any other characters.

This operand is meaningless unless the *conchars* operand is also specified.

`conchars`

Function: Specifies the character or character string to be compared with the message-type field in the message handler.

Format: One to eight nonblank characters in character or hexadecimal format. If character format is used, the string may be unframed or framed with C'' or CLn'' characters. If hexadecimal format is used, the string must be framed with X'' or XLn'' characters.

Default: None. Specification is optional.

Note: When using the *conchars* operand, the next nonblank character or character string in the header buffer (after the current setting of the scan pointer) is compared with a character or character string specified by *conchars*. If the *conchars* field matches the field in the header buffer, all instructions between this MSGTYPE macro and the next MSGTYPE macro (or the next delimiter if there is not another MSGTYPE macro) are executed. The next MSGTYPE macro is not executed, and a branch is made to the next subgroup. If the *conchars* field does not match the field in the header buffer, the scan pointer is reset to its position before the comparison, and control passes to the next MSGTYPE macro in the current subgroup, or, if this is the last MSGTYPE macro in the subgroup, to the next MH subgroup.

If the MSGTYPE macro has no operands, the group of instructions that immediately follows this MSGTYPE macro will process the message header not processed by a preceding MSGTYPE macro with a *conchars* operand. A MSGTYPE macro with no operands may be used only at the last of a series of MSGTYPE macros with *conchars* operands.

If MSGTYPE macros are used with and without operands, either some message-type field should always be specified, or care should be taken, if the field is omitted, that the next field cannot match any of the strings specified by the *conchars* or TABLE=operands in the series of MSGTYPE macros.

The *conchars* operand cannot be coded when the TABLE= and EXIT= operands are coded.

`EXIT=name`

Function: Provides the address to which this MSGTYPE macro will be if the message-type character in the header buffer does not match any entry in the table pointed to by the TABLE= operand of this MSGTYPE macro.

Format: Must conform to the rules for assembler language. (See the "Glossary.")

Default: None. Specification is required when the TABLE= operand is specified.

Note: *name* is the symbolic address to which this MSGTYPE macro will branch on an unequal comparison.

This operand is invalid when the *conchars* operand is specified.

The MSGTYPE macro branches to the user exit specified on the EXIT= operand, via BR14 with register 2 containing the address of the next subgroup to be executed. The MSGTYPE macro does not expect to regain control from the user exit.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

`TABLE=name`

Function: Provides the name of the first of a series of one or more TYPETABL macros. (See the description of the TYPETABL macro later in this chapter.)

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is required when EXIT= operand is specified.

Note: When using the TABLE= operand, the next nonblank character in a header buffer (after the current setting of the scan pointer) is compared with a table of message-type characters generated by one or more TYPETABL macros, and the address of the next subgroup is placed in register 2. If a match is found, the scan pointer is moved one position to the right and a branch is taken to the address in the table for that message type. When user processing of the message is completed, you can branch to the next subgroup or to another address within the current subgroup. If no match is found in the table, the scan pointer is not moved and a branch is taken to the address specified by the EXIT= operand.

Since one execution of a MSGTYPE macro (using the TABLE= operand) examines one character in the message-type field of a header buffer, multiple characters can be included in the message-type field and multiple MSGTYPE macros can be issued to examine these characters.

The TABLE= and EXIT= operands cannot be used if the *conchars* operand is coded.

Examples

Identifying Different Types of Messages within the Same MH

This example shows how you may identify different types of messages within the same MH. After the various types are identified, they may be processed and forwarded by different codes.

```
MHA     STARTMH
        INHDR                   DELIMITER
        SEQUENCE                MACRO INSTRUCTIONS
        ORIGIN                  EXECUTED FOR ALL
        DATETIME                HEADER SEGMENTS
        COUNTER     FIELD       COUNT INCOMING SEGMENTS
        MSGTYPE     C'A'        TEST FOR TYPE A MESSAGES; IF NOT
*                               A TYPE A MESSAGE, CONTINUE EXECUTION
*                               AT THE NEXT MSGTYPE MACRO.
```

Macro instructions executed for all type A messages:

```
       FORWARD      DEST=CL3'NYC'
       MSGTYPE      C'B'        TEST FOR TYPE B MESSAGES:  IF NOT
                                A TYPE B MESSAGE, CONTINUE EXECUTION
                                AT THE NEXT MSGTYPE MACRO.
```

Macro instructions executed for all type B messages:

```
    FORWARD      DEST=CL3'BIX'
    MSGTYPE
    FORWARD      DEST=CL8'PROCESSQ'    MACRO INSTRUCTION EXECUTED
*                                      FOR ALL OTHER MESSAGES TYPES
    INMSG                              DELIMITER.
```

Coding a Series of MSGTYPE Macros

This is an example of a series of MSGTYPE macros using the *conchars* operand followed by the TABLE= operand.

```
MTA   MSGTYPE     C'ATYPE'    IF THE MESSAGE HEADER IS AN ATYPE, THE
*                             USER CODE FOLLOWING THE MSGTYPE MACRO
*                             NAMES MTA IS EXECUTED; THEN, A BRANCH
*                             IS TAKEN TO THE NEXT SUBGROUP.
```

user code

```
MTB   MSGTYPE     C'BTYPE'    IF THE MESSAGE HEADER IS A BTYPE, THE
*                             USER CODE FOLLOWING THE MSGTYPE MACRO
*                             NAMED MTB WILL BE EXECUTED;  THEN, A
*                             BRANCH WILL BE TAKEN TO THE NEXT SUBGROUP.
```

user code

```
MT1    MSGTYPE     TABLE=TABL1,*   IF THE MESSAGE TYPE IS NEITHER ATYPE
*                  EXIT=EXIT1      NOR BTYPE, THE SCAN POINTER IS RESET
*                                  TO ITS POSITION BEFORE THE COMPARISON
*                                  AND THIS MSGTYPE MACRO IS EXECUTED.
*                                  IF THE MESSAGE TYPE IS C, THE USER
*                                  CODE AT LABEL C WILL BE EXECUTED.
*                                  IF THE MESSAGE TYPE IS D, THE USER
*                                  CODE AT LABEL D IS EXECUTED.
*                                  OTHERWISE, THE ROUTINE AT EXIT1 IS
*                                  EXECUTED.
C      EQU *
```

user code

```
        BR   2
D       EQU *
```

user code

```
        BR          2
EXIT1   BR          2
TABLE1  TYPETABL    C,ROUTINE=C
        TYPETABL    D,ROUTINE=D
```

Return Codes

None.

NEWMSG Macro:

The NEWMSG macro:

- Specifies the name of an EXMSG macro defining a message to be subsequently generated by a SENDMSG macro
- Places in the exception request table the offset of the message in the next available slot of the appropriate NEWMSG option field
- Requires the presence of an FHP in the message being processed
- May be issued multiple times within one message handler group to schedule the generation of up to three messages
- May be used to clear the NEWMSG option field to zero
- May be used to signal cancellation in the current message (only when issued in an incoming subgroup)
- Requires that DLQ= operand on the INTRO macro be specified on the INTRO macro in the MCP or specified at TCAM startup time.

To use the NEWMSG macro, you must define a 4-byte option field, named NEWMSG, and initialize it to zero for each external LU or application that uses the macro. When, during MH processing, a message must be generated (either an error message or an advisory message), you may issue the NEWMSG macro to insert the offset of one or more messages in the exception request table into the next available message number slot in the NEWMSG option field. The current message may be marked for cancellation if so specified by the user. If all the message number slots in the NEWMSG option field are full, the specified message number overlays the number in the last message slot of the NEWMSG option field.

During inmessage or outmessage subgroup processing, the actual message will be generated by a SENDMSG macro specifying EXITCDE=NEWMSG using the exception request table, or by a user exit routine called by SENDMSG. Refer to the EXMSG macro and the SENDMSG macro for a description of message generation and transmittal processing when used in conjunction with the NEWMSG macro.

Supported Resources and General Requirements: ALL, FHP.

Valid subgroups: Inheader, inbuffer, outheader, and outbuffer.

NAME	OPERATION	OPERANDS
[symbol]	NEWMSG	[BRANCH={name }]
		{(register)}
		[,DSECT={YES}]
		{NO }
		[,MSG={number }]
		{(register)}
		{CLEAR }
		{0 }
		[,OLDMSG=CANCEL]

```
BRANCH={name      }
       {(register)}
```

Function: Provides the name or address of an instruction in the current MH subgroup or the name of a delimiter macro to be executed after NEWMSG processing.

Format: name or *(register)* must conform to the rules for assembler language symbols. *(register)* can be specified as a decimal integer from 2 through 12, or a symbol that has previously been equated to a decimal value from 2 through 12.

Default: None. Specification is optional.

Note: *name* is the assembler language name of the next instruction to be executed after NEWMSG processing.

(register) is a general register, 2 through 12, containing the address of the instruction to be executed after NEWMSG processing.

If this keyword is omitted, control passes to the next sequential instruction following the NEWMSG macro.

```
DSECT={YES}
      {NO }
```

Function: Generates the dummy section describing the format of the NEWMSG option field.

Format: YES or NO.

Default: DSECT=NO.

Note: If YES is specified, all other keywords are ignored and the dummy section describing the NEWMSG option field is generated.

```
MSG={number    }
    {(register)}
    {CLEAR     }
    {0         }
```

Function: Specifies either the number to be inserted into the NEWMSG option field or that the NEWMSG option field is to be cleared to zero for the current station.

Format: *number, (register),* CLEAR or 0. *number* is an unframed decimal integer with a value from 1 through 255. *(register)* is a general register, 2 through 12, containing a binary value 1 through 255.

Default: MSG=CLEAR.

Note: *number* represents the relative position in the exception request table of the message being scheduled for generation. A symbolic name, equated to such a number elsewhere in the assembly, is also acceptable. The symbolic name could be the label of an EXMSG macro. In this case, the NEWMSG macro would be scheduling the generation of the message defined by the EXMSG macro with that label.

(register) represents a message number that can be specified as a decimal integer from 2 through 12, or a symbol that has previously been equated to

a decimal value from 2 through 12. It is your responsibility to assign this operand a value that falls within the prescribed limits.

CLEAR specifies that all three message number slots in the NEWMSG option field are to be cleared to binary zeros.

Issuing a NEWMSG macro specifying MSG=CLEAR at the beginning of the inheader subgroup is the recommended method for initializing the NEWMSG option field.

If MSG=0 is specified, then OLDMSG=CANCEL must be coded. If MSG=0 is specified, the current message is cancelled; no new message is generated, and the new message option field is not cleared.

`OLDMSG=CANCEL`

Function: Specifies that the message currently being processed is to be scheduled for cancellation by setting on bit 20 in the TCAM message error record. A conditional CANCELMG macro (testing bit 20) must be the first functional macro executed in the inmessage subgroup to cause the actual cancellation.

Format: CANCEL.

Default: None. Specification is optional.

Note: This entry may be specified only in the incoming section of a message handler. It should be specified if the message currently being processed cannot, or should not, be routed.

If you code both the MSG=CLEAR and the OLDMSG=CANCEL operands, the message currently being processed is scheduled for cancellation, but no new message generation is scheduled. Bit 20 of the message error record (user error) is set. This is equivalent to the operation of the TERRSET macro.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols. (See the "Glossary.")

Default: None. Specification is optional.

Return Codes

None.

NODEDEF Macro

The NODEDEF macro:

- When the extended networking facility is present, creates an entry in the node table which is used for routing
- May be coded in a nonexecutable section of an MCP or in an externally assembled module
- Is used with the NODETABL macro to create a table permitting routing on the node identifier
- Must be coded contiguously with all NODEDEF macros defining other TCAM nodes in the network
- Requires the presence of a NODETABL macro in the MCP.

A group of NODEDEF macros together define the node table, which is created during MCP initialization via a NODETABL macro and is used for routing messages to TCAM nodes in other TCAM systems based upon node identifiers when using the extended networking facilities. One NODEDEF macro should be coded to represent each TCAM node in the extended network, including the local TCAM node. Each macro specifies the node identifier for the node it represents, and (other TCAM nodes) specifies the internodal destination queues associated with the node it represents. One node table should be created in each TCAM node that is part of the extended networking environment.

NODEDEF macros are coded as a group; they may either be coded in the resource definition section of the MCP or be a member of a partitioned data set known to the MCP.

Supported Resources and General Requirements: SNA, FHP.

Valid subgroups: Not applicable.

NAME	OPERATION	OPERANDS

[symbol]	NODEDEF	TC1=qname,TC2=qname[,...,TC16=qname]
		[,DSECT={YES}]
		{NO }
		[,MONITOR={YES }]
		{NO }
		{tclist}
		[,NODEID={n }]
		{LAST}

```
DSECT={YES}
      {NO }
```

Function: Generates a dummy section describing the format of a NODEDEF table entry.

Format: YES or NO.

Default: DSECT=NO.

Note: If YES is specified, all other operands are ignored and the section describing the NODEDEF entry format is generated.

By coding constants following each entry, you may insert data you wish to include in a NODEDEF entry. For example, if you wish to append an eight character field to each entry, you can code:

```
NODEDEF  NODEID=10,....
DC       CL8''
NODEDEF  NODEID=20,....
DC       CL8''
NODEDEF  NODEID=LAST
```

```
MONITOR={YES      }
        {NO       }
        {(tclist)}
```

Function: Specifies whether the TCAM node being defined is to be monitored for availability via the local internodal awareness system service program.

Format: YES or NO or *(tclist)* where *(tclist)* is a list of decimal numbers in the range of 1-16 and are separated by commas. Parentheses are not required if there is only one number in the list.

Default: MONITOR = YES.

Note: NO specifies that the local node is not to send messages relating to node availability to this node and is not to monitor this node for availability.

YES specifies that normal processing by the internodal awareness system service program is to occur. The local node sends node and program availability messages to this node, and monitors it for availability.

If MONITOR = NO is specified, the status field of the NODEDEF entry is flagged available (NODESTA1 is set to X'80').

(tclist) specifies that monitoring is requested for the transmission categories in the list. This list may be all categories or a subset. It is recommended that at least the first three categories should be in the list.

```
NODEID={n   }
       {LAST}
```

Function: Specifies the node identifier for this entry or signals the end of the node table.

Format: *n* or LAST. *n* should either be a decimal integer from 1 to 245 or a symbol that has previously been equated to a decimal value from 1 to 245. (Node identifiers 246 through 255 are reserved for TCAM system use.) It is the user's responsibility to assign this operand a value that is within the prescribed limits.

Default: NODEID = LAST.

Note: LAST signals the end of the table. If NODEID=LAST is specified, all other operands are ignored.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

```
TC1=qname
TC2=qname
    .
    .
    .
TCn=qname
```

Function: These operands may be coded for overriding default names of the internodal destination queues used to route messages other TCAM nodes which is represented by this macro. Each TC*n* operand specifies the terminal table entry name of an internodal destination queue to be used when messages are routed to this node using transmission category *n*.

Format: TC*n*=*qname*. *n* is a decimal integer not greater than the maximum number of transmission categories defined in this MCP; *n* may not exceed 16, the maximum number of transmission categories allowed in any MCP. *qname* must conform to the rules for assembler language symbols.

Default: $H*id*C*n*, where:

id = explicit decimal value specified for the NODEID= operand
n = transmission category.

The macro always generates 16 TC= suboperands. The default internodal destination queue names are used for any of these suboperands which are not coded by the user.

Note: *qname* may be any single or cascade-list entry representing an internodal destination queue in the terminal table.

Transmission categories 1 and 2 must be defined as another TCAM node; transmission category 1 should name a main storage queue; TCAM control facilities use transmission category 1 for their messages and may not function effectively if this category does not specify a main storage queue. Transmission category 2 should name a disk queue.

This operand is not applicable if the NODEID= operand specifies the local TCAM node.

Return Codes

None.

NODETABL Macro

The NODETABL macro:

- Creates a node table for routing using the extended networking facility
- Is optional and may be issued once in the initialization section of the MCP following an OPEN macro and before the READY macro.

When it is executed during initialization of the MCP, NODETABL dynamically constructs the node table using information provided by the NODEDEF macro. The NODEDEF macros may be located in the resource definition section of the MCP, or they may be a member of another assembly module. By locating your NODEDEF macros in another assembly module, you can change node routing information without MCP reassembly.

Supported Resources and General Requirements: SNA, FHP.

Valid subgroup: Not applicable.

NAME	OPERATION	OPERANDS

[symbol]	NODETABL	{ENTRY1=name }
		{ADDRESS=name}
		,TABNAME=name
		[,ENTRYLN={n}]
		{0}

ADDRESS=name

Function: Allows you to assemble your NODEDEF macros externally and provides the NODETABL macro with the address of the first entry.

Format: *name* must conform to the rules for assembler language symbols.

Default: None. Either ADDRESS= or ENTRY1= must be specified.

Note: *name* is the name of a fullword-aligned field that contains the address of the first NODEDEF macro coded for the node table that is to be built. If the ADDRESS= operand is specified, the TABNAME= operand must also be coded.

```
ENTRYLN={n}
        {0}
```

Function: Indicates the length of the user-data field in each node-table entry.

Format: *n* should be either a decimal integer from 1 to 100 or a symbol that has previously been equated to a decimal value from 1 to 100. It is your responsibility to assign this operand a valid value.

Default: ENTRYLN=0.

Note: This operand allows you to append data to the basic NODEDEF entries. If 10 characters are added to each entry, ENTRYLN=10. See the description of the NODEDEF macro for instruction on how to append user data in the node-table entries.

`ENTRY1=name`

Function: Specifies the name of the first NODEDEF macro in the series defining the node table.

Format: *name* must conform to the rules for assembler language symbols.

Default: None. Either ENTRY1= or ADDRESS= must be specified.

Note: *name* is the name of the first NODEDEF macro coded after this NODETABL macro in the assembly.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

`TABNAME=name`

Function: Provides a name for the node table being created.

Format: *name* must conform to the rules for assembler language symbols.

Default: None. Specification is required.

Example

If the external module containing your NODEDEF macros were called NDTABLE, you can use the following code to initialize the address table:

```
         LOAD        EP=NDTABLE       LOAD THE TABLE
         ST          0,NDADDR         PUT ITS ADDRESS
                                      IN THE FIELD
         NODETABL    ADDRESS=NADDR    ISSUE THE MACRO
         .
         .
         .
NDADDR   DS    F
```

Return Codes

None.

OPEN Macro

The OPEN macro:

- Completes initialization and activation of data sets belonging to the message control program.

The OPEN macros for data sets must be coded in the following order:

1. Message queue data sets
2. MCP checkpoint data set
3. Log data set.

Execution of the OPEN macro completes initialization and activation of MCP data sets and provides an interface with the BSAM routines handling the logging function for TCAM. Each data set required for execution (with the exception of a message queue data set residing in main storage) must be activated in the message control program by an OPEN macro. Each MCP data set may be activated by a separate OPEN macro, or all data sets of the same type (for example, all message queue data sets) may be activated as a group by a single OPEN.

Instead of a standard OPEN macro, you may code a list and an execute form of the macro, which are used in conjunction with each other. (For general information on the list and the execute form of macro, including a discussion of the advantages of using these forms, see the *Supervisor Services and Macros* manual.)

If an OPEN macro fails to properly open a TCAM data set, the TESTOPEN macro may be used to send an error message to the system console. This error message specifies the data set that could not be satisfactorily opened and gives the operator the option of either terminating the MCP abnormally or continuing the operation without the data set. This is discussed more fully in the chapter titled "Initiating and Terminating TCAM" in the *TCAM Installation Guide*. For some data sets, TCAM also provides you with the capability of specifying (via a DCB macro) a user-written subroutine that receives control when an OPEN macro is not executed properly. See the chapter of the *TCAM Installation Guide* titled "Defining Data Sets" for details. If you do not provide this subroutine or the TESTOPEN macro, TCAM issues an ABEND macro for the MCP program when an OPEN is not executed properly.

The operand fields of the OPEN macro consist of one or more groups of positional operands, followed by a single keyword operand. Each group of positional operands consists of the name of the data control block for the data set being opened and some optional information about that data set. The name of the block is the name of the DCB macro that created it; a comma is coded between groups. The optional keyword operand at the end permits the list or the execute form of the macro to be specified.

Supported Resources and General Requirements: Not applicable.

Valid Subgroup: Not applicable.

NAME	OPERATION	OPERANDS

[symbol]	OPEN	dcbname
		[,{OUTPUT}]
		  { INOUT }
		[,MF={L }
		    {(E,listname)}

```
dcbname
```

Function: Specifies the name of a data control block identical with the name specified in the *symbol* field of the DCB macro for the data set being opened.

Format: Must conform to the rules for assembler language symbols.

Default: None. This operand is required.

Note: Register notation may also be used, in which case the specified register enclosed in parentheses (2 through 12) should contain the address of the data control block for the data set being opened.

```
MF={L             }
   {(E,listname)}
```

Function: Specifies whether the OPEN macro is to generate a parameter list only or is to generate executable code.

Format: *listname* must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Note: *listname* specifies the name of an OPEN macro specifying MF=L. MF=L creates a parameter list based on the OPEN operands. No executable code is generated. You must specify this form of the OPEN among your program constants. The parameters in the list are not used until the program issues an OPEN or CLOSE macro with an MF=(E,*listname)* operand that refers to the list. The name specified in the name field of the OPEN macro becomes the name assigned to the parameter list.

MF=(E,*listname)* causes execution of the OPEN routine, using the macro having the MF=L operand specified. Parameters specified through a macro having the MF=(E,*listname)* operand override corresponding parameters in the list.

```
{OUTPUT}
{INOUT }
```

Function: Specifies the type of data set with respect to the direction in which message traffic may flow.

Format: OUTPUT or INOUT.

Default: None. Specification is required.

Note: OUTPUT specifies an output data set and must be specified for a log data set.

INOUT specifies a data set that can be used for both input and output. INOUT must be specified for a DASD message-queue data set or a checkpoint data set.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional if MF=L is coded.

Note: *symbol* becomes the name of the parameter list generated by the macro.

Return Codes

The DCB macros for the message-queue data sets permit specification of a user-written routine to be given control if an OPEN macro fails to open the data set for which the DCB macro is coded. The user routine is specified by coding a special entry in the problem-program exit list named in the EXLST= operand of the appropriate DCB macro. (The format and contents of the problem-program exit list are shown in the publication *Data Management Services.)* The special entry, called the user ABEND entry, consists of a 1-byte code of X'0E' followed by the 3-byte address of the user routine.

If the OPEN macro for a particular data set fails to execute properly and if a user ABEND entry is included in the EXLST= operand of the DCB macro for the data set, the user routine is given control. The user routine must save and restore registers. When control is passed to the user routine, the general registers contain the following information:

Register	**Contents**
0	Error code
1	Options available to the user ABEND routine
2-13	Contents before execution of the OPEN macro
14	Return address (must *not* be altered by the exit routine)
15	Address of the user-routine entry point.

The error code in register 0 tells you why the OPEN failed. The action taken depends on the return code provided in the user's asynchronous error routine. If no error routine is provided, the value in register 0 following the abnormal end is meaningless.

Hex Code Reg 0	Meaning	Programmer Action
X'00000001'	There is not enough main storage to build a data extent block (DEB).	Specify a larger region or partition size on the JOB statement for the MCP.
X'0000000A'	Disk queuing is specified in a DCB macro defining a message queue data set, but the INTRO macro that generates the AVT specifies DISK=NO.	Specify DISK=YES in the INTRO macro, reassemble, and execute again.
X'0000000B'	The key length specified by the UNITSZ= or KEYLEN= operand of the INTRO macro does not agree with the key length specified in the IEDQDATA DD statement of the IEDQXA utility used to format the message queue data set.	Reassemble the job with the proper length specified in the UNITSZ= or KEYLEN= operand (whichever was specified) of the INTRO macro and rerun the MCP, or restart the TCAM job and override the UNITSZ= or KEYLEN= value by entering REPLY XX,'K=nn,U' to message IED002A, or reformat the disk to the proper key length using the IEDQXA utility and rerun the MCP.
X'0000000C'	Dissimilar disk types are defined for message queuing.	Ensure that the disk types specified for message queuing are similar.
X'0000000D'	The OPTCD= operand for this DCB specifies something other than reusable or nonreusable queuing.	Check and correct the contents of the DCB field.
X'0000000E'	A GETMAIN macro was issued by TCAM to obtain main storage to build a data extent block for a message queue data set, but there was not enough main storage to satisfy the request.	Specify a larger region or partition size on the JOB statement for the MCP.
X'0000000F'	A GETMAIN macro was issued by TCAM to obtain main storage to build input/output supervisor blocks (IOSBs) for a message queue data set, but there was not enough main storage to satisfy the request.	Specify a larger region size on the JOB statement for the MCP.
X'00000010'	The message queue data set was allocated but not formatted correctly; the last record number written on a track is zero.	Reformat the data set using the IEDQXA utility and rerun the MCP job.
X'00000017'	There was not enough main storage to satisfy a GETMAIN request to build a TCAM control area in the subpool.	Specify a larger region or partition on the JOB statement for the MCP.
X'00000020'	An OPEN was issued for a TCAM DCB by a program that was not executing with a protection key of 6. Protection key 6 is reserved for TCAM.	Specify IEDIMCP as the name of the MCP that is executing or add the name of this MCP to the program properties table as a program designated to run in protection key 6. Be sure that the MCP is link-edited as an authorized program (AC=1 coded on the link-edit execute card).

Hex Code Reg 0	Meaning	Programmer Action
X'00000025'	For reusable queuing, an attempt has been made to define a greater number of records then allowed by the reusable queuing device.	Specify a smaller number in the JCL and rerun the job.

Before control is passed to the user ABEND routine, TCAM sets the bits in the rightmost byte of register 1 to indicate to the ABEND routine what courses of action it may take. The code in register 1 indicates possible user options; the TCAM open routines are set up to work properly if any of the courses of action indicated by the code in register 1 are taken. The user ABEND routine should restrict its activities to the options indicated in register 1. The following are possible user ABEND alternatives and the codes associated with them:

Code in Reg. 1	Permissible User Options	
X'03'	1.	You can abnormally terminate the MCP job either by issuing an ABEND macro in your subroutine or by placing a return code of X'02' or higher in the rightmost byte of register 15 and returning control to TCAM.
	2.	You can tell the TCAM open routine to stop attempting to open this data set and to pass control to the next instruction in the MCP by placing a return code X'00' in the rightmost byte of register 15 and returning control to TCAM. In this case, your MCP will run with restricted capabilities, since it will not be able to use this data set.
X'07'	1.	Same as Option 1 for X'03' code.
	2.	Same as Option 2 for X'03' code.

Note: If you specify a return code of X'01' in register 15 and the option code passed to you in register 1 was X'03', TCAM immediately takes the ABEND exit again; unless the user routine has code providing for this possibility, a loop will result.

OPTION Macro

The OPTION macro:

- Permits space to be reserved for an option field related to an external LU or application program
- Provides an efficient method of accessing option field data, if numerous option fields are accessed in an MH for any terminal-table-entry
- Generates two main-storage areas (CSECT) which may be used as an answer area for the TCSEARCH macro instruction and to contain terminal entry information, including the addresses of all of an external LU's associated option fields
- Must be specified before any TERMINAL, TLIST, or TPROCESS macro
- Is optional among the macros defining the terminal table.

OPTION macros are issued as a group; in conjunction with the OPDATA= operands of the TERMINAL and TPROCESS macros, they define the *option table,* a storage area containing option fields related to individual external LUs or application programs. The option fields are accessed by certain message-handler routines that need origin- or destination-related storage in order to perform their functions. Among the MH macros that invoke routines that gain access to the option fields are: STARTMH, INHDR, INBUF, INMSG, OUTHDR, OUTBUF, OUTMSG, COUNTER, ERRORMSG, FORWARD, IEDHALT, LOCOPT, PATH, and REDIRECT. To gain some insight into the function of option fields, turn to the individual discussions of these macros in this manual. User-written routines can also access information in an option field.

Taken together, the OPTION macros issued define a complete set of option fields; all or part of this set may be assigned to a particular external LU or application program by means of the OPDATA= operand of the TERMINAL or TPROCESS macro. An OPTION macro merely gives an option field a name and describes the type and length of the field in assembler language format. Up to 254 option fields, each of which may be up to 255 bytes, may be defined in an MCP by OPTION macros. All or any part of the set of option fields may be allocated to each external LU, or application program represented by a terminal-table entry. For the set of option fields for a particular entry in the terminal table, the last option field must be within 254 bytes of the first.

A new area of storage, having the name and attributes specified by the OPTION macro defining an option field, is assigned to each external LU or application program whose TERMINAL or TPROCESS macro initializes that field. Each TERMINAL or TPROCESS macro may initialize a field differently; hence different external LUs or application programs may be assigned option fields having identical names and attributes but different contents. This feature allows you to tailor the functions of a macro accessing an option field to meet the needs of a particular external LU or application program. For example, the COUNTER macro maintains a count of messages or message segments received from or sent to a destination. This counter is located in an option field for that destination. If the OPTION macro for this field is named COUNT and if the COUNTER macro names COUNT as the field in which the counter should be maintained, then a separate counter will be maintained for each destination that uses the OPDATA= operand of the TERMINAL macro to initialize COUNT.

A macro coded in an inheader, inbuffer, or inmessage subgroup handling messages entered by external LUs accesses the specified option field for the external LU that entered the message being processed. A macro coded in an outheader, outbuffer, or outmessage subgroup handling messages destined for external LUs gains access to the specified option field for the external LU that is to accept the message being processed. A macro coded in an inheader, inbuffer, or inmessage subgroup handling messages being received from an application program gains access to the specified option field for the process entry associated with the PUT or WRITE macro. A macro coded in an outheader, outbuffer, or outmessage subgroup handling messages destined for an application program accesses the specified option field associated with the process queue to which the GET or READ macro that is moving this message to the application program is directed.

At least one OPTION macro must be issued if a TCSEARCH macro coded with ARG=CURRTERM is issued in the MCP.

OPTION generates two main-storage areas (CSECTs) that can be utilized by the TCSEARCH macro to gather together terminal-table-entry data, including the addresses of all terminal table option field data existing for any particular terminal-table-entry. The TCSEARCH method of accessing terminal entry option field data (using the answer areas generated by the OPTION macros) is generally more efficient and less time-consuming than issuing numerous LOCOPT macros to accomplish the same data access.

The answer areas generated by the OPTION macros conform to the extended answer area data format required for TCSEARCH use, as described in the *TCAM Program Reference Summary*. The answer areas are generated by OPTION as two separate control sections (one called IEDXOAF, the other IEDXDAF) both of which (CSECTs) are directly addressable from the user's MHs. The symbolic labels for the fields within these two answer area CSECTs vary, depending on the number and names of the OPTION-defined option fields in the MCP. The following points apply to the IEDXOAF and IEDXDAF CSECTs:

- 232 bytes of storage plus 8 bytes per OPTION macro issued is reserved for IEDXOAF and IEDXDAF areas.
- User code may symbolically access fields in these CSECTs only if an OPTION macro specifying DSECT=YES is coded as the first OPTION macro.
- All symbolic field names in the IEDXOAF CSECT begin with the letter "O"; all IEDXDAF field names start with a "D." Except for these initial letters, each field name in IEDXOAF is identical to the name of a corresponding IEDXDAF field.
- The data fields in IEDXOAF and IEDXDAF are appropriate for holding the data that can be filled in by the TCSEARCH macro with the SRCHTYP=TNT, EXDATA=YES, and OPTION=ALL operands specified. In addition to the option fields, the most frequently referenced data fields in the answer area format are also assigned symbolic names in the IEDXOAF and IEDXDAF CSECTs. The names of these fields, and their offsets with the answer area are:

IEDXOAF Name	**IEDXDAF Name**	**Field Def.**	**Contents**	**Offset**
IEDXOAF	IEDXDAF	CSECT	CSECT name	0
OTNTNMLH	DTNTNMLH	AL1	terminal name length	8
OTNTENTR	DTNTENTR	AL4	TNT entry address	8
OTRMTNT	DTRMTNT	AL	TTCIN	12
OTRMENTR	DTRMENTR	AL4	TTE address	16
OTRMNAME	DTRMNAME	CL8	terminal name	20.

The TNT entry address and the TTE address each occupy the low-order 3 bytes of a 4-byte field. The high-order byte of these fields is non-zero.

After the main portion of the answer area (116 bytes), IEDXOAF and IEDXDAF contain a variable number of fullword fields, based on the OPTION-defined option fields in the MCP. As required by TCSEARCH, there is one field in each of the two answer areas for each option field defined. The name of each field consists of the first seven characters of the associated OPTION macro name appended to the leading "O" (if there is a field in IEDXOAF), or leading "D" (in the IEDXDAF CSECT). These fields are suitable to contain the option field addresses that TCSEARCH will return if the user specified an OPTION= operand value greater than zero.

As an example of the method used to generate and name these fields, assume that the user has coded three OPTION macros, defining the TCSOPTS, THRESH, and REALNAME option fields; the generated IEDXOAF and IEDXDAF areas would contain the following fields:

IEDXOAF Name	IEDXDAF Name	Definition	Data Inserted
OTCSOPTS	DTCSOPTS	DS F	Address of TCSOPTS option
OTHRESH	DTHRESH	DS F	Address of THRESH option
OREALNAM	DREALNAM	DS F	Address of REALNAME option.

Note that the last character of any 8-position option field names will be truncated in the corresponding IEDXOAF and IEDXDAF address slot. For this reason, the first seven characters of each option field name must form a unique character string, in order to prevent duplicate labels during assembly.

The final element generated by the user-coded OPTION macros is a statement defining a symbolic label, MAXOPTS, which is equated to the total number of option fields that have been defined.

Using the OPTION answer areas:

The user-coded OPTION macros only construct the IEDXOAF and IEDXDAF answer areas; they do not place any terminal-table entry data into the areas. The TCSEARCH macro is intended to collect the terminal-table entry data and place it into one of these work areas. In its simplest form, TCSEARCH with ARG=CURRTERM will fill in all the fields of the IEDXOAF area with information about the originating terminal-table entry (if it is issued during inheader or inbuffer processing), or about the destination terminal-table entry if it is issued during outheader or outbuffer processing). Note that the IEDXDAF area is not used by TCSEARCH unless it is specifically assigned as the TCSEARCH macro ANSWER= operand. IEDXDAF is used internally by the INODEMH macro; the user might want to specify IEDXDAF to contain data about a terminal table entry other than the origin in the incoming section or the destination in the outgoing section of the MH.

In addition to placing terminal-table entry data into one of the answer areas, the user must also establish addressability for the answer area CSECT used, if any fields within the area are to be accessed by user MH code. The following code illustrates the method used:

```
     USING IEDXOAF,R5         SET UP ANSWER AREA ADDRESSABILITY
     TCSEARCH ARG=CURRTERM
     L R5,TVTOAFAN            GET ANSWER AREA ADDRESS FROM TVT
```

Now, only one assembler instruction is required to get the address of an option field (for example, TCSOPTS or THRESH):

```
L R9,OTCSOPTS GET TCSOPTS ADDRESS
L R10,OTHRESH GET THRESH ADDRESS
```

Note: Refer to the description of the TCSEARCH answer area description for the name of all fields in the IEDXOAF answer area.

Supported Resources and General Requirements: ALL.

Valid subgroup: Not applicable.

NAME	OPERATION	OPERANDS
opfldname	OPTION	typelength [,DSECT={YES}] {NO } [,USE=MRCHECK]

`opfldname`

Function: Specifies the name of the option field.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. This name is required.

Note that certain option field names have special uses in TCAM; see chapter 3, "Option Fields Reserved for Use by TCAM Functions", for details. Do not use these names for option fields unless the fields are to be used for purposes described in that chapter. Also refer to Chapter 3 for a description of additional option field names that are used by certain TCAM functions.

If the reserved name is specified with the USE= operand, you will not be able to issue the LOCOPT macro specifying the reserved name.

```
DSECT={YES}
      {NO }
```

Function: If this operand is coded on the first OPTION macro in a series, user code may symbolically access fields in the IEDXOAF and IEDXDAF answer areas.

Format: YES or NO.

Default: DSECT=NO

Note: If DSECT=YES is coded, the USE= operand is ignored. If DSECT=YES is coded, the first seven characters of the name of each OPTION macro coded must form a unique character string, in order to prevent duplicate labels during assembly.

`typelength`

Function: Specifies the type and length of the option field.

Format: Standard assembler-language format (for example, H, CL8, AL3). All assembler-language type codes may be used. However, B, C, P, X, and Z must be coded with a length attribute (for example, CL5, BL4). Duplication factors are not allowed; that is, ABC OPTION 3DL5 is an invalid macro.

Default: None. This operand is required.

Note: When the option field is used in conjunction with the FORWARD, ERRORMSG, REDIRECT, or SETEOM macro, a character string of length *n* must be specified, where *n* is the length in bytes of the data in the OPDATA= operand of the TERMINAL or TPROCESS macro that initializes the fields.

If used with COUNTER, *typelength* should be specified as H, since this macro requires a halfword field on a halfword boundary.

If used with INBLOCK, INBUF, INHDR, INMSG, OUTBUF, OUTHDR, OUTMSG, or PATH macros, *typelength* must specify a 1-byte field (for example, FL1, AL1). No boundary alignment is required.

If used with STARTMH, *typelength* specifies a 1- or 4-byte field, depending upon which STARTMH operand names the option field.

`USE=MRCHECK`

Function: USE=MRCHECK allows the MRCHECK macro to use an option field other than the first option field. This operand should be coded USE=MRCHECK if this option field is not the first option field defined and is to be used by the MRCHECK macro.

Format: MRCHECK.

Default: None. Specification is optional.

Note: If USE= is not coded and the MRCHECK macro is coded, MRCHECK uses the first option field defined.

Other Examples of Coding Option Macros

OPTION macros, if used, must be issued as a group and must immediately follow the TTABLE macro. The order in which OPTION macros are arranged determines the order in which initialization data must be specified in the OPDATA= operand of the TERMINAL or TPROCESS macros. If a field specified by an OPTION macro is not to be defined for a particular external LU or application program, then a comma must be coded in place of the data for this field in the OPDATA= operand (but trailing commas must not be coded). Option fields defined as address-type constants are not checkpointed by TCAM.

OPTION macros should be arranged so as to prevent waste of storage space in the option table. For example, if four OPTION macros are coded:

```
BA  OPTION F
BB  OPTION CL1
BC  OPTION H
BD  OPTION CL1
```

as much as 4 bytes of storage in the option table are wasted for each resource after the first for which these option fields are defined. To conserve storage, code the above macros:

```
BA  OPTION F
BC  OPTION H
BB  OPTION CL1
BD  OPTION CL1
```

In coding an OPTION macro, you must specify the type and length of the option field to be generated. Exact specification of the option field size and its assembler attributes is critical to TCAM's operation, particularly in checkpoint/restart example, if an option field is to contain an address, the operand of the related OPTION macro must be coded *A* or some representation of an address constant. This information is contained in the discussion of the individual macro that accesses the option field.

Example: In the following example, the TTABLE macro defines the beginning and end of the terminal table section of the message control program. The OPTION macros, which are a part of this section of code, define fields in the option table that are accessed by the COUNTER, REDIRECT, ERRORMSG, and PATH macros.

```
TTABLE   LAST=LASTTERM
COUNT    OPTION   H
REDRECT  OPTION   CL3
ERRMSG   OPTION   CL4
PATHSW   OPTION   FL1
```

TTABLE defines LASTTERM as the name of the last entry in the terminal table. The OPTION macros define an 10-byte optional area for entries in the terminal table. The optional area consists of four fields:

- COUNT defines a halfword for decimal data to be used by the COUNTER macro.
- REDRECT defines a character string consisting of 3 bytes naming the resource; this data is used by the REDIRECT macro.
- ERRMSG defines a character string consisting of a 4-byte station name; this data is used by the ERRORMSG macro.
- PATHSW defines 1 byte for eight binary path switches to be tested by various delimiter macros.

If the OPDATA= operand of a TERMINAL macro is coded:

```
OPDATA=(0,NYC,PITT,3)
```

an 10-byte storage area is set aside in the option table for use by MH macros in handling messages to and from that station. The COUNT fields initially contain 0; the REDRECT field contains NYC; the ERRMSG field contains PITT; and the PATHSW field contains 3.

If the OPDATA= operand of another TERMINAL macro is coded:

```
OPDATA=(,NYC,PITT)
```

a 7-byte storage area is set aside in the option table for use by MH macros in handling messages to and from that resource. Only the REDRECT and ERRMSG fields are created.

Note that for an option field to be created for any particular external LU, two conditions must be satisfied:

1. An OPTION macro defining the field must be issued.
2. The field must be initialized in the OPDATA= operand of the TERMINAL macro for that external LU. If a comma is coded in place of a field in the OPDATA= operand, no space is set aside for that field. If the OPDATA= operand of a TERMINAL macro is omitted, no option fields are set aside for that external LU.

Return Codes

None.

ORIGIN Macro

The ORIGIN macro:

- Checks the validity of the origin field in a message header
- Sets a bit in the message error for the message if the origin field is invalid
- Allows you to change the indicated origin of a message in the header buffer prefix.

The ORIGIN macro specifying or defaulting to TYPE=VERIFY verifies that the origin field in the header contains the symbolic name of the origin of the message; that is, the origin field is compared with the name in the terminal-name-table entry for the external LU. If the names are not the same, an error flag is set in bit 1 of the message error record for the message.

A CODE macro must be issued before ORIGIN (unless the code is already EBCDIC).

Care must be taken in entering a character string in an origin field in the message header to ensure that it matches a terminal-table entry. A character string entered in lowercase characters does not match a terminal-table entry name that is in uppercase characters.

Supported Resources and General Requirements: ALL.

Valid subgroup: Inheader if the TYPE= operand specifies TYPE=VERIFY or is omitted; otherwise, inheader or outheader.

NAME	OPERATION	OPERANDS
[symbol]	ORIGIN	[{integer}]
		{X'FF' }
		[,STATION={'name' }]
		{fieldname }
		{(register)}
		[,TTCIN={** }]
		{(register)}
		{address }
		[,TYPE={SET }]
		{VERIFY}

```
{integer}
{X'FF'   }
```

Function: Specifies the number of characters in the origin field of a message header. This operand is meaningful only if TYPE=VERIFY is specified.

Format: *integer* or X'FF'. *integer* can be decimal or hexadecimal. If hexadecimal format is specified, framing X'' or XL1'' characters must be used.

Default: X'FF'.

Maximum: 8.

Note: If *integer* is specified, that many characters are accessed and considered to be the origin field. Embedded blanks are ignored.

X'FF' indicates that the origin field is of variable length. The origin field is considered to end at the next blank.

```
STATION={'name'     }
        {fieldname }
        {(register)}
```

Function: Specifies the name of the resource to be used in processing when the TYPE= operand specifies SET.

Format: *'name,' fieldname* or (*register*). *'name'* must conform to the rules for assembler language symbol and must be within single quotes. *fieldname* must conform to the rules for assembler language a must not be specified with framing characters. *(register)* is the actual register number or the equated name of a register enclosed in parentheses. Registers 0 thru 15 may be used.

Default: None. Specification is optional.

Note: *'name'* is the name of an entry in the terminal name table; the length of the name must correspond with the value of the MAXLEN operand.

fieldname is the symbolic name of the field which contains the resource name, the length of which must correspond to the MAXLEN= operand of the TTABLE macro. *fieldname* must be padded with blanks in order to correspond with the value of the MAXLEN operand.

(register) specifies a register containing the address of the field that contains the name of the desired resource.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

```
TTCIN={**         }
      {(register)}
      {address    }
```

Function: Specifies the resource to be used in processing when the TYPE= operand specifies SET. This operand is meaningful only if TYPE=SET is specified.

Format: **, *(register),* or *address*. *(register)* is the actual register number or the equated name of a register enclosed in parentheses. *address* must conform to the rules for assembler language symbols.

Default: TTCIN=**.

Note: ** causes the external LU session partner to be used for processing when the TYPE= operand specifies SET. *(register)* specifies that the terminal name table index of the desired resource is contained in the register whose name or number is specified.

address specifies that the terminal name table index of the desired resource is contained in the storage location whose address is specified.

The terminal name table index of the message origin can be determined from the contents of a field in the IEDXOAF answer area following execution of a TCSEARCH macro specifying ARG=CURRTERM. See the OPTION macro description for a discussion of IEDXOAF.

This operand is invalid if the STATION= operand is coded.

```
TYPE={SET   }
     {VERIFY}
```

Function: Specifies whether the ORIGIN macro is to check the validity of the origin field in the message header, or is to change the source of the message as indicated in a field in the header buffer prefix.

Format: SET, VERIFY.

Default: TYPE=VERIFY.

Note: SET specifies that a field in the header-buffer prefix containing the terminal name table index for the message origin and named PRFSRCE is to be changed to indicate that the message origin is the resource specified in the STATION= operand.

VERIFY specifies that TCAM is to validity-check the origin field in the message header.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	The ORIGIN macro executed successfully.
X'00000004'	The resource name was invalid.
X'00000008'	The TTCIN or STATION operand specified an invalid resource.
X'00000010'	Resource not found in terminal table; new source not set.
X'00000014'	Although PRFSRCE was set to the new source there is not an active LU-LU session between the destination and the new source.
X'00000018'	Buffer is not a header buffer when TYPE=SET is coded.
X'0000001C'	This is a zero length buffer, or the scan pointer out of buffer.
X'FFFFFFFC'	Delimiting blank not found or scan pointer beyond or is equal to the end of buffer; or the integer specified was greater than the buffer space.

OUTBUF Macro

The OUTBUF macro:

- Identifies a subgroup that handles outgoing message buffers
- Tests a switch to allow alternative courses of action
- Is optional in an outgoing group.

OUTBUF identifies the beginning of an outbuffer subgroup that contains instructions concerned with both header and text portions of outgoing messages. If included, an outbuffer subgroup may be located either before or after an outheader subgroup in the outgoing group.

If the PATH= operand of OUTBUF is coded, OUTBUF examines a path switch in a field of the option table. If any of the bits specified by OUTBUF are on in the path switch, this subgroup is executed. If none of the bits specified by OUTBUF are on, control passes to the next subgroup. If OUTBUF does not specify an operand, this subgroup is executed unconditionally.

Supported Resources and General Requirements: ALL.

Valid subgroup: Outbuffer.

NAME	OPERATION	OPERAND
[symbol]	OUTBUF	[PATH=(opfield,switch)]

`PATH=(opfield,switch)`

Function: Specifies conditional execution of this macro and its subgroup.

Format: (opfield,switch). opfield must conform to the rules for assembler language symbols, and must be the name of a 1-byte option field defined by an OPTION macro. *switch* may be either decimal or hexadecimal. If hexadecimal format is used, framing X'' characters must be specified.

Default: None. Specification is optional.

Maximum: 255 or X'FF' for *switch.*

Note: If any bits that are on in the *switch* parameter are on in the path switch in the option field, the macro and its subgroup are executed.

If this operand is not specified, the subgroup is executed unconditionally.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the “Glossary”).

Default: None. Specification is optional.

Return Codes

None.

OUTEND Macro

The OUTEND macro:

- Identifies the end of any outgoing group
- Is required as the last macro in any MH
- Identifies the end of an MH
- Cannot be branched to.

Supported Resources and General Requirements: ALL.

Valid subgroup: Outgoing group.

Name	Operation	Operands
[symbol]	OUTEND	(no operands)

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Note: If an OUTMSG subgroup is included in the outgoing group, no executable code is generated by the OUTEND macro; therefore, branching to location *symbol* is not permitted.

Return Codes

None.

OUTHDR Macro

The OUTHDR macro:

- Identifies the beginning of an outheader subgroup
- Tests a path switch to allow alternative courses of action
- Is optional in an outgoing group.

OUTHDR identifies the beginning of an outheader subgroup, which is concerned only with the header portions of outgoing messages and, if included, may be either before or after an outbuffer subgroup in the outgoing group.

If the PATH= operand of OUTHDR is coded, OUTHDR examines a path switch in a field of the option table. If any of the bits specified by OUTHDR are on in the path switch, this subgroup is executed. If none of the bits are on, control passes to the next subgroup. If OUTHDR does not specify an operand, this subgroup is executed unconditionally.

Supported Resources and General Requirements: ALL.

Valid subgroup: Outheader.

NAME	OPERATION	OPERAND
`[symbol]`	`OUTHDR`	`[PATH=(opfield,switch)]`

`PATH=(opfield,switch)`

Function: Specifies conditional execution of this macro and its subgroup.

Format: (opfield,switch). opfield must conform to the rules for assembler language symbols and must be the name of a 1-byte option field defined by an OPTION macro. *switch* may be either decimal or hexadecimal. If hexadecimal format is used, framing X'' characters must be specified.

Default: None. Specification is optional.

Maximum: 255 or a X'FF' for *switch*.

Note: If any bits that are on in the *switch* parameter on in the path switch in the option field, the macro and its subgroup are executed.

If this operand is not specified, the subgroup is executed unconditionally.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the “Glossary”).

Default: None. Specification is optional.

Return Codes

None.

OUTMSG Macro

The OUTMSG macro:

- Identifies the beginning of an outmessage subgroup of a DMH
- Tests a path switch to allow alternative courses of action
- Is required as the first macro in an outmessage subgroup of a DMH
- Is optional in an outgoing group.

OUTMSG identifies the beginning of an outmessage subgroup, which is executed only after an entire message has been sent. Outmessage subgroups are specified after other subgroups in the outgoing group.

If the PATH= operand of OUTMSG is coded, OUTMSG examines a path switch in a field of the option table. If any of the bits specified by OUTMSG are on in the path switch, this subgroup is executed. If none of the bits specified by OUTMSG are on, control passes to the next subgroup. If OUTMSG does not specify an operand, this subgroup is executed unconditionally. Only one outmessage subgroup per message can be executed.

OUTMSG causes empty units at the end of buffers handled by this outgoing group to be deallocated and returned to the available-unit queue. If an outmessage subgroup is not coded, this deallocation function is performed by OUTEND.

Supported Resources and General Requirements: SNA.

Valid subgroup: Outmessage of a DMH.

NAME	OPERATION	OPERAND
[symbol]	OUTMSG	[PATH=(opfield,switch)]

`PATH=(opfield,switch)`

Function: Specifies conditional execution of this macro and its subgroup.

Format: (opfield,switch). opfield must conform to the rules for assembler language symbols, and must be the name of a 1-byte option field defined by an OPTION macro. *switch* may be either decimal or hexadecimal. If hexadecimal format is used, framing X'' characters must be specified.

Default: None. Specification is optional.

Maximum: 255 or X'FF' for *switch.*

Note: If any bits that are on in the *switch* parameter in the path switch in the option field, the macro and its subgroup are executed. Macros in an outmessage subgroup of an AMH do not execute.

If this operand is not specified, the subgroup is executed unconditionally.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

None.

PATH Macro

The PATH macro:

- Alters a path-switch byte, thereby permitting dynamic variation of the path of a message through a message handler

One-byte option fields maintain switches known as *path switches*. These switches are located in option fields defined by OPTION macros. The switches must be set initially by the OPDATA= operand of the TERMINAL or TPROCESS macro. (If the option fields are not initialized, the PATH macro provides a return code of X'00'.) The setting of path switches is examined by each delimiter macro as the message reaches the subgroup it controls; you specify (by the PATH= operand of each delimiter) which path-switch bits are to be examined. More than one option field may be specified for each resource; each path-switch byte so defined consists of eight binary switches.

If any of the binary switches tested by a delimiter is on, the subgroup controlled by that delimiter is executed; if none of the binary switches tested is on, control passes to the next delimiter.

You may specify a character string (consisting of one to eight nonblank characters). If this character string appears in the header of a message, the PATH macro with that character string sets one or more specified path switches. If no character string is specified, the switches are set unconditionally.

The PATH macro may specify any number between 0 and 255 inclusive. The switches remain set until reset by a PATH macro specifying the same option field, until modified by user code and LOCOPT, or until modified by an Insert Option Field Data operator command. (The Insert Option Field Data operator command generates an MVI instruction.)

This use of the PATH macro is discussed in the chapter titled "Coding the Message Handler" in the *TCAM Installation Guide.*

Supported Resources and General Requirements: SNA.

Valid Subgroups: Inblock, inbuffer, inheader, outbuffer, and outheader.

NAME	OPERATION	OPERANDS
[symbol]	PATH	switch ,opfield [,conchars] [,BLANK={YES }] {NO } {char}

```
BLANK={YES }
      {NO  }
      {char}
```

Function: Specifies whether EBCDIC blank characters are to be ignored when encountered in the character string in the message header being compared to the string specified by the *conchars* operand, or whether blanks are to be part of the header string when encountered in it. If EBCDIC blanks are to be counted as part of the header string, this operand also specifies whether some other hexadecimal character is to be ignored when encountered in the header string.

Format: YES, NO, or *char*. *char* is a single character that may be specified in either character or hexadecimal format. If character format is specified, it may be unframed or framed with C'' or CL1'' characters. If hexadecimal format is specified, it must be framed with X'' or XL1'' characters.

Default: BLANK=YES.

Note: YES specifies that the EBCDIC blank character (X'40') is to be ignored by this macro whenever it is encountered in the header character string being checked against the control character string specified by the *conchars* operand. For example, if BLANK=YES is coded and an eight-byte field in the header is being checked by this macro, a blank appearing in the fifth byte of the field is ignored and the sixth through ninth bytes are considered to be the last four bytes of the field (assuming that no blanks are coded in the sixth through ninth bytes).

NO specifies that the EBCDIC blank character is to be treated in the same way as any other character when it is encountered by this macro in the header string being compared to the string specified by *conchars*.

char specifies that the single character replacing *char* be ignored by this macro whenever it is encountered in the header string being compared to the string specified by the *conchars* operand. That is, the macro automatically skips over the character without performing a comparison and goes on to check the next character in the header. If BLANK=*char* is coded and *char* is not the EBCDIC blank (X'40'), the EBCDIC blank is not ignored by this macro when it is encountered in the header string, but is compared to the character in the corresponding space in the *conchars* string, in the same way as any other character.

This operand is meaningless unless the *conchars* operand is also specified.

`conchars`

Function: Specifies the character or character string that, if found in the header as the next nonblank field, executes the function.

Format: One to eight nonblank characters in character or hexadecimal format. If character format is used, the string may be unframed or framed with C'' or CLn'' characters. If hexadecimal format is used, the string must be framed with X'' or XLn'' characters.

Default: None. Specification is optional.

Note: If this operand is omitted, the PATH function is performed unconditionally. If the next field in the header does not match this operand, the function is not performed.

This operand should be coded only in the PATH macro issued in an inheader or outheader subgroup.

`opfield`

Function: Specifies the path-switch byte to be operated upon.

Format: The name of a 1-byte field in the option table as defined by an OPTION macro.

Default: None. This operand is required.

Note: If the option field cannot be found, the path-switch byte is not operated upon and a return code of X'00' is set in the low-order byte of register 15.

If PATH is coded in the incoming group of a DMH, the specified option field for the external LU entering the message is operated upon. If PATH is coded in the outgoing group of a DMH, the specified option field for the destination is operated upon. If PATH is coded in the outgoing group of an AMH assigned to an application program, the option field associated with the process queue to which the GET/READ macro is directed is operated upon. If the macro is coded in the incoming group of an AMH assigned to an application program, the option field for the process entry associated with the DCB named in the PUT/WRITE macro is operated upon.

`switch`

Function: Specifies the path-switch setting that replaces the byte residing in the option field named by the *opfield* operand.

Format: *switch* may be decimal or hexadecimal. If hexadecimal format is specified, framing X'' or XL1'' characters must be used.

Default: None. This operand is required.

Maximum: 255 or X'FF'.

Note: An MVI instruction is generated. If 0 is specified, all eight path switches are reset. If 255 (or X'FF') is specified, all switches are set.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Example

The following code shows the outline of an inmessage subgroup of an MH. Messages with A, B, or C in an appropriate field are routed through the incoming group by PATH macro instructions. The switch settings enable you to select appropriate inbuffer and inmessage subgroups. Message type A passes through the first inbuffer subgroup and the second inmessage subgroup, etc.

Notes: In the case of multiple-buffer headers, the entire control-character field must appear in the first header segment.

```
VARYPATH  STARTMH INHDR                  INHEADER SUBGROUP EXECUTED FOR ALL
          .                              MESSAGES.
          .
          .
*         PATH      4,SWITCH,C'A'        SETS SWITCH FOR TYPE A MESSAGES
          .                              (NOT EXECUTED FOR OTHERS).
          .
          .
          PATH      2,SWITCH,C'B'        SETS SWITCH FOR TYPE B MESSAGES.
          PATH      1,SWITCH,C'C'        SETS SWITCH FOR TYPE C MESSAGES.
          .
          .
          .
*         INBUF     PATH=(SWITCH,4)      THIS INBUFFER SUBGROUP IS EXECUTED
          .                              IF SWITCH IS 4 (TYPE A MESSAGES).
          .
          .
*         INBUF     PATH=(SWITCH,2)      THIS INBUFFER SUBGROUP IS EXECUTED
          .                              IF SWITCH IS 2 (TYPE B MESSAGES).
          .
          .
*         INBUF     PATH=(SWITCH,1)      THIS INBUFFER SUBGROUP IS EXECUTED
          .                              IF SWITCH IS 1 (TYPE C MESSAGES).
          .
          .
*         INMSG     PATH=(SWITCH,3)      THIS INMESSAGE SUBGROUP IS EXECUTED
          .                              IF SWITCH IS 1 OR 2 (TYPE B AND C MESSAGES).
          .
          .
*         INMSG     PATH=(SWITCH,4)      THIS INMESSAGE SUBGROUP IS EXECUTED
          .                              IF SWITCH IS 4 (TYPE A MESSAGES).
          .
          .
          INEND
```

Return Codes

When the PATH macro is coded in an MH, one of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	Successful execution.
Option field address	Option field was found.

PCB Macro

The PCB macro performs for the application program many of the functions performed for external LUs by the GROUP macro. The PCB macro:

- Provides a control block in the MCP to interface with an application program
- Is required for each application program running with the MCP
- Is coded in the MCP, not the application program
- Identifies the application message handler to be used to process messages flowing to and from the application program.

The PCB macro generates a named control block known as a *process control block* (PCB). A process control block provides information needed to communicate between the MCP and an application program.

One and only one PCB macro is required for each active application program although you may assign more than one PCB to a single application program. A PCB may *not* be shared by two active application programs since a PCB is an intertask control block. A maximum of 255 TPROCESS entries may use the same PCB.

Supported Resources and General Requirements: APP.

Valid subgroup: Not applicable.

NAME	OPERATION	OPERANDS
pcbname	PCB	BUFSIZE=integer ,MH=mhname [,BUFIN={number}] {2 } [,BUFOUT={number}] {2 } [,DATE={YES}] {NO } [,PUTCNT={number}] {0 } [,RESERVE=(integer1,integer2)] [,SFLAG={YES}] {NO } [,TIMEDLY={number}] {0 }

```
BUFIN={number}
      {2     }
```

Function: Specifies the initial number of buffers requested into which the data in the user's PUT/WRITE work area will be emptied.

Format: Unframed decimal integer greater than 1.

Default: BUFIN=2.

Maximum: 15.

Note: The optimum number specifies enough buffers to contain the entire work area.

```
BUFOUT={number}
       {2     }
```

Function: Specifies the initial number of buffers that may be filled in anticipation of a GET or READ.

Format: Unframed decimal integer greater than 1.

Default: BUFOUT=2.

Maximum: 15.

Note: These buffers are used as a read-ahead queue for a process entry.

```
BUFSIZE=integer
```

Function: Specifies the size of the buffers to be assigned to handle messages for the associated application program.

Format: Unframed decimal integer greater than or equal to 76.

Default: None. This operand is required.

Maximum: 65,535.

Note: This value may be overridden by specifying in the application program the BUFL= operand of the input or output DCB for the application program.

The number of units allocated for each PUT TPROCESS entry will be determined by BUFSIZE= times BUFIN=. These units will remain allocated until filled. Minimum recommended size is half of average message size or UNITSZ, whichever is greater.

```
DATE={YES}
     {NO }
```

Function: Specifies whether the date and time of each message received for the process entry are to be recorded.

Format: YES or NO.

Default: DATE=NO.

Note: When a message is received for the application program, TCAM records the date and time. When the application program issues a GET or a READ macro, TCAM places the recorded date/time and the source of the message in the area specified by the DTSAREA= operand of the TPDATE application program macro.

This operand requires that the DATE= operand also be specified on the TPROCESS macro for this process entry.

```
MH=mhname
```

Function: Specifies the message handler to be used for an application program.

Format: *mhname* must conform to the rules for assembler language symbol (see the "Glossary").

Default: None. Specification is required.

Note: *mhname* must be the name specified in the name field of a STARTMH macro.

```
pcbname
```

Function: Specifies the name of the macro and the name of the process control block generated by the macro referred to in the TPROCESS macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. This name is required.

```
PUTCNT={number}
       {0     }
```

Function: Specifies the number of PUTs or WRITEs to be issued before TCAM forces a time delay.

Format: Unframed decimal integer.

Default: PUTCNT=0.

Maximum: PUTCNT=1000.

Note: TCAM processes the number of PUTs, then waits the number of seconds specified in the TIMEDLY= operand. This process is repeated until all PUTs or WRITEs have been processed. Both operands must be specified in order to activate the procedure.

```
RESERVE=(integer1,integer2)
```

Function: Specifies the number of bytes to be reserved in buffers.

Format: Unframed decimal integer.

Default: None. Specification is optional.

Maximum: 255 for each.

Note: *integer1* specifies the number of bytes to be reserved in the buffer receiving the first incoming segment of each message entered by an application program; the space is reserved for insertion of data by DATETIME and SEQUENCE functional MH macros.

integer2 specifies the number of bytes to be reserved in all buffers except the first for insertion of characters by the DATETIME macro. *integer2* is relevant only in a multiple-buffer header situation when the DATETIME macro is to insert data in a portion of the header that is not in the first buffer.

Data may be inserted in either an incoming or an outgoing message header, but space must be reserved in the incoming header. On the outgoing side, reserved space is retained for the first buffer only; thus, DATETIME and SEQUENCE macros, if specified in an outheader subgroup, operate on the first segment of the message.

No space need be reserved for data inserted by a MSGEDIT functional MH macro.

The *TCAM Installation Guide* describes how TCAM handles reserve bytes. Each buffer containing header data should be large enough to accommodate the segment itself plus any data that may be inserted by DATETIME and SEQUENCE macros. If a buffer containing header data does not have a sufficient number of bytes reserved in it to accommodate data inserted by a DATETIME or SEQUENCE macro, the macro is not executed and control passes to the next instruction in the MH. Unused reserve bytes are not sent out with an outgoing message segment when it is sent to its destination.

```
SFLAG={YES}
      {NO }
```

Function: Specifies whether or not the last message obtained by a GET or READ macro is to be marked serviced if the application program terminated abnormally.

Format: YES or NO.

Default: SFLAG=YES.

```
TIMEDLY={number}
        {0     }
```

Function: Specifies the time delay (in seconds) to follow the number of PUTs or WRITEs specified on the PUTCNT= operand.

Format: Unframed decimal integer.

Default: TIMEDLY=0.

Maximum: TIMEDLY=3600.

Note: This operand may be specified to create a time delay in the processing of PUTs or WRITEs to pace buffer utilization and to keep from flooding the queue.

This operand must be used in conjunction with PUTCNT= to activate the procedure.

Return Codes

None.

PRIORITY Macro

The PRIORITY macro:

- Specifies message priority handling
- May be used more than once in the inheader subgroup.

PRIORITY provides handling of messages according to priority levels. You must enter the message priority level in the message header, or it may be specified by an operand of the PRIORITY macro. The permissible message priority levels for each station or application program are specified in the TERMINAL or TPROCESS macro for that destination. If a message priority is requested that is not permitted, the message is assumed to have the next lower permissible priority. The PRIORITY macro must be specified within the subgroup in the same relative order as the header field on which it acts.

Absence of the PRIORITY macro causes a priority level of zero to be assigned to the messages.

For more information on message priority, see the *TCAM Installation Guide*.

TCAM converts the decimal message priority levels specified by the LEVEL= operand of the TERMINAL or TPROCESS macro to their 1-byte hexadecimal equivalents. If the message priority is specified in a message header, it may occupy a 1-byte field and should provide the hexadecimal equivalent of a decimal message priority level specified by the LEVEL= operand of the TERMINAL or TPROCESS macro.

In the case of multiple-buffer headers, the message priority, if desired, must be determined for the first header segment to pass through the inheader subgroup. This can be ensured in one of two ways:

1. The message priority field in the header, if used, must be in the first header segment.
2. The *integer* operand must be specified to provide the priority. Any control characters used to execute the PRIORITY macro must be in the first buffer.

Storing and Retrieving the Message Priority Level

The PRIEXIT= operand of the READY macro specifies the name of a user-coded subroutine that is given control whenever a message is to be queued to any but the first of multiple destinations. On return to TCAM, register 15 must contain a priority level, which is then assigned to the message. User code appearing in the inheader section of the DMH after the PRIORITY macro could be used to retrieve the message priority from the SCBPRI field of the SCB associated with the destination station store it in an option field. (The SCB address may be obtained from the LCB, the address of which may be obtained from the buffer prefix, whose address may be obtained from the AVT; consult the *TCAM Program Reference Summary* details.) A LOCOPT macro may be coded in the exit routine to locate the option field for each destination after the first, and the message priority may be returned in register 15.

If this facility is not used, messages sent to each destination after the first coded in a distribution or cascade list, and message copies sent to multiple destinations after the first, are assigned zero message priority when they are queued, unless you specify PRIEXIT=DKJHMX on the READY macro and specify PRISAVE=YES on PRIORITY.

Supported Resources and General Requirements: ALL.

Valid subgroup: Inheader.

PRIORITY

NAME	OPERATION	OPERANDS
[symbol]	PRIORITY	[integer] [,conchars] [,BLANK={<u>YES</u> }] {NO } {char} [,PRISAVE={YES}] {<u>NO</u> }

```
BLANK={YES }
      {NO  }
      {char}
```

Function: Specifies whether EBCDIC blank characters are to be ignored when encountered in the character string in the message header being compared to the string specified by the *conchars* operand, or whether blanks to be part of the header string when encountered in it. If EBCDIC blanks are to be counted as part of the header string, this operand also specifies whether another hexadecimal character is to be ignored when encountered in the header string.

Format: YES, NO, or *char*. *char* is a single character that may be specified in either character or hexadecimal format. If character format is specified, it may be unframed or framed with C'' or CL1'' characters. If hexadecimal format is specified, it must be framed with X'' or XL1'' characters.

Default: BLANK=YES.

This operand is meaningless unless the *conchars* operand is also specified.

YES specifies that the EBCDIC blank character (X'40') is to be ignored by this macro whenever it is encountered in the header character string being checked against the control character string specified by the *conchars* operand. For example, if BLANK=YES is coded and an eight-byte field in the header is being checked by this macro, a blank appearing in the fifth byte of the field is ignored and the sixth through ninth bytes are considered to be the last four bytes of the field (assuming that no blanks are coded in the sixth through ninth bytes).

NO specifies that the EBCDIC blank character is to be treated in the same way as any other character when it is encountered by this macro in the header string being compared to the string specified by *conchars*.

char specifies that the single character replacing .*char* be ignored by this macro whenever it is encountered in the header string being compared to the string specified by the *conchars* operand. That is, the macro automatically skips over the character without performing a comparison and goes on to check the next character in the header. If BLANK=*char* is coded and *char* is not the EBCDIC blank character (X'40'), the EBCDIC blank is not ignored by this macro when it is encountered in the header string, but is compared to the character in the corresponding space in the *conchars* string, in the same way as any other character.

`conchars`

Function: Specifies the character or character string that, if included in the message header, executes the PRIORITY macro specifying that string.

Format: One to eight nonblank characters in character or hexadecimal format. If character format is used, the string may be unframed or framed with C'' or CLn'' characters. If hexadecimal format is used, the string must be framed with X'' or XLn'' characters.

Default: None. Specification is optional.

If this operand is omitted, PRIORITY is executed unconditionally. If the control characters do not match, the PRIORITY macro is not executed and control passes to the next instruction.

If this operand is specified, but the *integer* operand is omitted:

- The message priority is assumed to be contained in the message header as the next nonblank character following the control characters.
- A comma must precede the *conchars* operand.

`integer`

Function: Specifies the assigned message priority level.

Format: Unframed decimal integer.

Default: None. Specification is optional.

Maximum: 255.

If this operand is omitted, TCAM assumes that the message priority level is contained in the next nonblank byte (as a 1-byte binary value) following the current setting of the scan pointer. This is explained in the *conchars* operand. If the message priority level is not one that the TERMINAL or TPROCESS macro specifies as permissible, the next lower permissible priority is assumed.

If the *integer* and *conchars* operands are omitted, the message priority is assumed to be in the message header, in the next nonblank character following the current setting on the scan pointer.

In the case of multiple-buffer headers, the message priority, if desired, must be determined for the first header segment to pass through the inheader subgroup. This can be ensured in one of two ways:

1. The message priority field in the header, if used, must be in the first header segment.
2. The *integer* operand must be specified to provide the message priority, and any control characters used to execute the PRIORITY macro must be in the first buffer.

```
PRISAVE={YES}
        {NO }
```

Function: PRISAVE=YES allows message priority level queuing to be maintained for all destinations in a multi-destination message, for all destinations that are part of a distribution or cascade list and for redirected messages.

Format: YES or NO.

Default: PRISAVE=NO.

Note: For maintaining message priority for multi-destination messages and TLISTs, a fixed header prefix (FHP) must be present in the message header and PRIEXIT=DKJHMX must be coded on the READY macro.

To maintain message priority for redirected messages, code REDIRECT with EXIT=DKJAZX.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	Successful execution.
X'00000004'	Not a header buffer or no *conchars* match.
X'00000008'	No fixed header prefix when PRISAVE=YES is specified.
X'FFFFFFFC'	Zero length buffer.

READY Macro

The READY macro:

- Permits "good morning" and "restart in progress" messages to be specified
- Permits specification of a user exit routine that facilitates the transfer of messages from one queue to another and the purging of messages from queues
- Permits specification of an exit to a user-written routine that allows you to set a priority for messages directed to second or subsequent entries on cascade or distribution lists and to other than the first of multiple destinations
- Must be located between the OPEN and CLOSE macros.

The READY macro completes initialization and activation of the message control program; once the READY macro has executed, the TCAM MCP is ready for message traffic. One READY macro is specified for each MCP. This macro must be located between the OPEN macros and the CLOSE macros in the activation and deactivation section of the MCP.

Two optional operands of the READY macro, GMMSG= and RSMSG=, allow you to provide the addresses of user-written routines that build "good morning" or "restart in progress" messages. These routines can also be coded to alter option fields and other control areas so that they indicate that a restart has occurred. The exit to the "good morning" message is taken for the initial startup each cold restart; the exit to the "restart in progress" message is taken. (For a discussion of the various types of TCAM startup, refer to the chapter titled "Checkpoint/Restart Service Facility" in *TCAM Utilities*.)

When initial startup or a restart occurs, the appropriate routine is given control for each external LU defined by a TERMINAL macro. The user routine should save and restore registers. When control passes to the user routine, register 1 contains the address of a two-word parameter list. The first word in the list contains the address of the terminal-table entry for the external LU to which the message generated by the user is to be sent, while the second word contains the address of the option fields for the destination. If the destination has no option fields specified, the second word The user routine may use this information to build a message tailored to this particular external LU if a message is desired and may also alter fields in the terminal-table entry and the option fields for the external LU to reflect the fact that a restart has occurred. (For a warm restart, the data in the terminal-table entries and the contents of the option fields before closedown or failure are preserved by the checkpoint facility.)

The user routine returns the address of a message to be sent to the external LU. An all-zero address indicates that no message is to be sent to this external LU. At the specified address is a one-byte field indicating, in binary form, one more than the number of bytes of data in the message, followed by the text of the message. The maximum length of the message is 255 bytes.

TCAM places the message at the head of the queue for the destination so that it is the first message sent to that destination following startup or restart. The message is handled by the outgoing group of the message handler for the destination and is transmitted in the same way as any other message. Since the message is handled by an MH group, it must have a header similar in format to the headers of messages normally handled by the group. You must construct this header in your exit routine and include it as the first part of the message.

The PRIEXIT= operand provides an exit to a user-written routine that receives a message is queued to any but the first entry in a cascade or distribution list or other than the first of multiple destinations. Unless changed by the user routine, all messages to destinations in cascade or distribution lists or multiple destinations except the first have a message priority of zero. This exit can set any priority to the message as it is queued to each destination. The same message can have a different message priority on each destination's queue.

The PURGEXT= operand provides the address of a user-written routine that will gain control prior to TCAM's establishing a session with the external LU for outbound messages. This user-written routine permits you to transfer messages from one queue to another or to purge messages from a queue in conjunction with the REDIRECT macro.

Immediately following the specification of the READY macro, you should begin coding the termination section of your MCP.

Supported Resources and General Requirements: Not applicable.

Valid subgroup: Not applicable.

NAME	OPERATION	OPERANDS
[symbol]	READY	[GMMSG=routine] [,PRIEXIT=routine] [,PURGEXT=routine] [,RSMSG=routine]

GMMSG=routine

Function: Specifies the name of a user-written, closed subroutine that builds "good morning" messages on each line.

When the IBM-supplied startup/restart message generation service facility is used, the name of the TCSUP macro should be specified here.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Note: If this operand is coded, the routine is given control, once for each external LU, for the initial startup, and for each cold restart.

TCAM does not automatically save and restore the registers for this routine. The user must restore any registers used.

The routine is coded and assembled as part of the MCP in the same manner as the message handlers. Its exact location is not important since it is called as a closed subroutine. See "Using Exits" in the *TCAM Installation Guide* for the correct linkages.

PRIEXIT=routine

Function: Specifies the name of a user-written exit routine that receives control whenever a message is to be queued to any but the first entry in a cascade or distribution list or other than the first of multiple destinations. (The TCAM PRIORITY macro determines the message priority of the first entry of multiple destinations.)

If PRISAVE=YES is coded on the PRIORITY macro, the IBM-supplied routine DKJHMX may be specified here.

Format: Must conform to the rules for a V-type address constant in assembler language.

Default: None. Specification is optional.

Note: You are passed the current message buffer address in register 1 and must return the desired message priority in the low-order byte of register 15 when returning to TCAM. If an invalid message priority is specified (that is, the priority level is not one that the destination's TERMINAL macro specifies as permissible), the message is assumed to have the next-lower defined message priority for that destination.

TCAM automatically saves and restores registers for this routine, enabling the user to change the contents of registers 2 through 12. The contents of registers 13 and 14, however, must not be changed. When this routine receives control, register 1 contains the address of the message header buffer, register 14 contains the return address of the calling TCAM routine, and register 15 contains the address of the entry point of this routine. On return to TCAM, register 15 must contain a message priority as specified above.

This routine should not use any system macro or other function that would result in a wait state since this would also place TCAM in a wait state.

`PURGEXT=routine`

Function: Allows the specification of the name of a user-written exit routine that receives control before establishing a session with an external LU for the purpose of deciding whether to purge the messages or send them.

To support transfer or purge functions of the extended operator control system service program , the IBM-supplied routine DKJKAX may be specified here.

Format: Must conform to the rules for a V-type address constant in assembler language.

Default: None. Specification is optional.

Note: TCAM automatically saves and restores registers for the routine, and the user may change the contents of registers 2 through 12 and 15. The contents of registers 13 and 14, however, must not be altered.

When the purge exit receives control, register 1 contains the address of the TTE control block for the destination. If SPURGE=YES is specified on the TERMINAL macro for the destination, register 2 contains the address of a parameter list (see the SPURGE parameter list below). If SPURGE=YES is not specified on the TERMINAL macro for the destination, register 2 contains all zeroes. Register 14 contains the return address to TCAM, and register 15 contains the address of the entry point of the purge exit. TCAM expects a return code of either 0 or 4 in register 15 from the routine. The routine must return control to TCAM by a BR 14 instruction.

This routine should not use any system macro or other function that results in a wait state since this places TCAM into a wait state.

When a return code of X'00' is passed to TCAM, normal processing continues. When a return code of X'04' is passed to TCAM, a session is not established with the external LU and bits 24 and 27 in the message error record are set and a zero-length (error) buffer is passed to the outheader subgroup.

The SPURGE Parameter List

Following is a Dsect that maps the SPURGE Parameter List:

```
************************************************************************
THIS DSECT MAPS THE SPURGE PARAMETER LIST.

THE USE OF EACH FIELD IS DOCUMENTED IN THE COMMENTS.

* ON ENTRY TO THE PURGE EXIT, REGISTER 2 CONTAINS THE ADDRESS          *
* OF THIS PARAMETER LIST IF THE ENTRY OT THE EXIT IS FOR A             *
* DESTINATION FOR WHICH SPURGE=YES IS SPECIFIED.  IF THE ENTRY         *
* IS FOR A DESTINATION THAT DOES NOT HAVE SPURGE=YES SPECIFIED,        *
* THEN REGISTER 2 CONTAINS ZEROES.                                     *
*                                                                      *
************************************************************************
IEDPRG     DSECT                PURGE EXIT PARAMETER LIST
           DS    OF             START ON WORD BOUNDARY
************************************************************************
PRGDEST    DS    0CL12          DESTINATION INFO
PRGDSTA1   DS    XL1            DESTINATION STATE 1
PRGDLU     EQU   X'80'          DEST IS AN SNA LU
*          EQU   X'40'-X'01'    RESERVED
PRGDSTA2   DS    CL1            DESTINATION STATE 2
PRDGINTL   EQU   X'80'          ONLY VALID IF PRGDLU IS ON.
*                               IF ON, THE DLU IS NOT CURRENTLY
*                               IN AN ACTIVE LU-LU SESSION BUT
*                               AN ATTEMPT TO INITIATE A SESSION
*                               WILL BE MADE UNLESS THIS MESSAGE
*                               IS PURGED.
PRGDACTV   EQU   X'40'          ONLY VALID IF PRDGLU IS ON.
*                               IF ON, THE DLU IS CURRENTLY IN
*                               AN ACTIVE LU-LU SESSION AND THE
*                               NAME OF THE DLU IS IN PRGONAME.
PRGDFAIL   EQU   X'20'          ONLY VALID IF PRGDLU IS ON.
*                               IF ON, THEN A PREVIOUS ATTEMPT TO
*                               SET UP AN LU-LU SESSION HAS FAILED.
*                               IF THIS MESSAGE IS NOT PURGED BY
*                               THE PURGE EXIT, THEN TCAM WILL
*                               PURGE IT WITH BITS 24 AND 26 SET
*                               IN THE MESSAGE ERROR RECORD.
*          EQU   X'10'-X'01'    RESERVED
           DS    XL2            RESERVED
PRGDNAME   DS    CL8            DESTINATION NAME -
*                               PADDED ON RIGHT WITH BLANKS CAN
*                               BE USED AS INPUT TO LOCOPT MACRO.
***********************************************************************
PRGSOURC   DS    0CL12          SOURCE INFO
PRGSSTA1   DS    XL1            SOURCE STATE 1
PRGSALTD   EQU   X'80'          SOURCE IS A PUT/WRITE TPROCESS
*                               ENTRY WITH AN ALTDEST SPECIFIED, AND
*                               THE ALTDEST NAME IS IN PRGANAME.
*                               THIS ALTDEST NAME MAY BE THE NAME
*                                OF THE GET/READ TPROCESS ENTRY WITH
*                                LU=YES SPECIFIED WHICH IS INTENDED
*                                TO BE THE DLU FOR THE MESSAGE.
*
*          EQU    X'40'-X'01'    RESERVED
           DS     XL3            RESERVED
PRGSNAME   DS     CL8            SOURCE NAME -
*                                PADDED ON RIGHT WITH BLANKS CAN
*                                BE USED AS INPUT TO LOCOPT MACRO.
***********************************************************************
PRGALTD    DS     0CL12          ALTDEST INFO -
*                                THIS INFO IS ONLY VALID WHEN
*                                PRGSALTD IS ON.
PRGASTA1   DS     XL1            ALTDEST STATE 1
*          EQU    X'80'-X'01'    RESERVED
           DS     XL3            RESERVED
PRGANAME   DS     CL8            ALTDEST NAME -
*                                PADDED ON RIGHT WITH BLANKS CAN
*                                BE USED AS INPUT TO LOCOPT MACRO.
***********************************************************************
PRGOLU     DS     0CL12          DLU INFO -
*                                THIS INFO IS ONLY VALID WHEN
*                                PRGDACTV IS ON.
PRGOSTA1   DS     XL1            DLU STATE 1
PRGOTPRO   EQU    X'80'          DLU IS A TPROCESS ENTRY
PRGOPROG   EQU    X'40'          DLU IS THE INTRO MACRO PROGID
PRGOMH     EQU    X'20'          DLU IS AN MH
*          EQU    X'10'-X'01'    RESERVED
           DS     CL3            RESERVED
PRGONAME   DS     CL8            DLU NAME -
*                                PADDED ON RIGHT WITH BLANKS CAN
*                                BE USED AS INPUT TO LOCOPT MACRO.
************************************************************************
PRGADBUF   DS     A              ADDRESS OF THE FIRST UNIT OF
*                                THE FIRST BUFFER OF THE MESSAGE
************************************************************************
PRGSENSE   DS     0XL4           SNA SENSE -
*                                THIS FIELD IS USED ONLY WHEN
*                                PRGDLU IS ON
           DS     XL2            RESERVED
PRGUSER    DS     XL2            USER SENSE -
*                                THIS FIELD IS ALL ZEROS ON ENTRY
*                                TO THE PURGE EXIT.  IF THE MESSAGE
*                                IS PURGED, THEN TCAM WILL CHECK
*                                TO SEE IF THIS FIELD HAS BEEN
*                                CHANGED TO A NON ZERO VALUE.  IF
*                                SO, THEN TCAM WILL SET BIT 31 IN
*                                THE MESSAGE ERROR RECORD (ALONG
*                                WITH BITS 24 AND 27) AND PURGE
*                                THE MESSAGE USING THE VALUE IN
*                                THIS FIELD AS THE USER SENSE.
*                                THIS SENSE CAN BE INTERROGATED
*                                BY THE IEDSENSE MACRO.
PRGEND     EQU    *              END OF PARAMETER LIST
PRGLEN     EQU PRGEND-IEDPRG     LENGTH OF PARAMETER LIST.
```

RSMSG=routine

Function: Specifies the name of a user-written, closed subroutine that builds "restart in progress" messages.

When the IBM-supplied startup/restart message generation service facility is used, the name of the TCSUP macro must be specified here.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Note: If this operand is coded, the routine is given control, for each external LU, for a warm restart or a point-of-last-environment (POLE) restart.

TCAM does not automatically save and restore the registers for this routine. The user must restore any registers used.

This routine is coded and assembled as part of the MCP in the same manner as the message handlers. Its exact location is not important since it is called as a closed subroutine. See "Using Exits" in the *TCAM Installation Guide* for the correct linkages.

symbol

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

None.

REDIRECT Macro

The REDIRECT macro:

- Queues a message for an additional destination
- Specifies the name of an exit to be taken when a message is redirected
- May be specified more than once within a subgroup.

REDIRECT queues a message for a destination in addition to the destinations specified by the FORWARD macro, when errors specified by the *mask* operand are detected. The bits specified by the error mask operand are compared with the setting of the bits in the message error record for the message. If specified bits in the message error record are on, the REDIRECT macro is executed; otherwise, the REDIRECT macro is not executed.

The EXIT= operand of the REDIRECT macro allows you to provide the name of an exit to be taken when this macro executes. When control is returned to TCAM, you may either specify that the message be sent to the DEST= destination or to a destination whose name is provided by your routine, or that the message not be redirected.

The REDIRECT macro may not be used for redirecting error messages. (See the ERRORMSG macro.)

The additional destination specified may be any single, process, or cascade entry in the terminal table. A distribution list cannot be specified as the additional destination.

If the message is sent before it is copied, redirecting a message that has been queued for a destination that uses main-storage-only queuing to a destination that uses disk queuing may cause an 045-2 abend.

REDIRECT may send unsent messages to an application program, return them to the origin, or send them to the alternate destination when the intended destination is inoperative.

Note: If a message having an invalid destination field is entered, the destination is not corrected by the user exit of the FORWARD macro, and no dead-letter queue is specified by the INTRO macro, then REDIRECT cannot be used in conjunction with that message, because the message header cannot be recalled by TCAM from a destination queue.

Supported Resources and General Requirements: ALL.

Valid subgroups: Inmessage and outmessage.

NAME	OPERATION	OPERANDS
[symbol]	REDIRECT	[mask] [,CONNECT={AND}] {OR } [,DEST={destname}] {opfield } {ORIGIN } [,EXIT=rtnname]

```
CONNECT={AND}
        {OR }
```

Function: Specifies the type of logical connection to be made between the mask and the message error record.

Format: AND or OR.

Default: CONNECT=OR.

Note: AND specifies that the macro is to be executed only if *all* of the bits specified by *mask* are on in the message error record.

OR specifies that the macro is to be executed if *any* bit specified by *mask* is on in the message error record.

```
DEST={destname}
     {opfield }
     {ORIGIN  }
```

Function: Specifies the additional destination.

Format: destname, opfield, or ORIGIN. *destname* may be up to eight bytes and is the name of any single, or cascade entry in the terminal table. It must be specified within framing C", CLn", X", or XLn" characters. *opfield* is the unframed name of an option field defined by an OPTION macro and cannot be named ORIGIN.

Default: DEST=ORIGIN.

If an invalid destination is specified, REDIRECT is not executed. *opfield* refers to an option field of up to eight bytes containing the name of the destination. The additional destination is the resource:

- Specified in the option field assigned to the origin if redirect is used in an inmessage subgroup
- Specified in the option field assigned to the destination if REDIRECT is used in an outmessage subgroup.

ORIGIN specifies that the message in error is to be sent to the origin (in addition to the destinations specified in the message header).

EXIT=rtnname

Function: Specifies the name of a user-written routine that is given control when a message is to be redirected.

The IBM supplied routine DKJAZX, in conjunction with the PRIORITY macro PRISAVE=YES specification, should be specified here to preserve the message priority of the redirected message. DKJAZX requires that messages have FHPs. If the extended networking function is present, DKJAZX also checks for a message-looping condition caused by multiple executions of the REDIRECT macro for the same message and places the message on the destination queue specified by the LOOPQ= operand of the INTRO macro when looping is detected.

If a user exit modifies data in the first unit of the header buffer, then the first unit of the header buffer of the redirected message is updated. Alterations to the prefix or extra units of the header buffer will not be updated in the redirected message. The size of the buffer and the amount of data contained in the buffer should not be changed.

Format: Must conform to the rules for a V-type address constant in assembler language.

Default: None. Specification is optional.

Note: This routine must be used in conjunction with the READY macro PURGEXT routine in order to transfer or purge messages. (See the READY macro discussion in this chapter.)

The LOCOPT macro (STATION= operand) and IEDSHOW macro (TSTATUS, STATION= operand) are valid in this routine to obtain the contents of various TCAM control blocks. For detailed information, see the applicable macro description in this chapter.

TCAM automatically saves and restores registers for this routine; you may change the contents of registers 2 through 12 and 15. However, the contents of registers 13 and 14 must not be altered.

When the user routine receives control, register 1 contains the address of the header buffer, register 13 contains the address of a 18 word user save area, register 14 contains the return address of the TCAM calling routine, and register 15 contains the address of the entry point of the user routine.

When a user exit return control, TCAM expects a return code of X'00', X'04', or X'08' to be returned in register 15 and treats any other return code as if the exit were not taken. Control should be returned to TCAM through a BR 14 instruction.

When a return code of X'00' is returned, TCAM redirects the message in question to the destination specified in the DEST= operand of this macro.

When a return code of X'04' is returned, the message is not redirected.

When a return code of X'08' is returned, register 1 must point to the name of the destination that you wish the message to be directed to. The length specified in the TTABLE macro (see the MAXLEN= operand in the TTABLE macro discussion) is used to determine the length of the destination name. If the destination name cannot be found, the message is directed to the dead-letter queue. If a dead-letter queue was not specified, the message is overlaid and lost.

`mask`

Function: Specifies the 5-byte bit configuration used to test the message error record for the message. (The message error record is described in Appendix A.)

Format: Decimal or hexadecimal. If hexadecimal format is used, framing characters must be specified. If X'' is used, leading zeros must be coded. If XL5'' is used, leading zeros may be omitted.

Default: None. Specification is optional.

Note: Omitting the operand or specifying an all-zero mask causes unconditional execution.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

None.

RESTABLE Macro

The RESTABLE macro:

- Is executed in an extended networking environment to build a resource table, which is an inverted terminal name table sorted on the resource identifiers specified by you in terminal table RESOURCE option field entries
- May be issued once, after an OPEN macro and before the READY macro in the initialization section of the MCP.

In an extended networking environment, each external LU or application in a TCAM node that can originate or receive a message is assigned a unique numeric resource identifier from 100 to 65,535 (resource identifiers 1 through 99 are reserved for system use). This identification number is provided during resource definition in an option field called RESOURCE. Within any individual MCP, any particular resource identifier must be unique. When TCAM is initialized, for routing purposes, the RESTABLE macro is issued to construct an inverted terminal name table based on these resource identifications.

After successful execution of the RESTABLE macro:

1. Register 15 is set to zero
2. A sorted resource table has been constructed and its control block address placed in the TVT (at label TVTRESTB).

After unsuccessful execution of the RESTABLE macro:

1. Register 15 contains a return code of X'04' if duplicate resource identifiers were discovered (that is, more than one RESOURCE option field contained the same resource identifier) and of X'08' if every entry was without a RESOURCE option field definition
2. If register 15 contains a X'04', a sorted resource table has been constructed and its control address placed in the TVT. The duplicated resource identifier will be associated with the last resource that specified it in an option field. A warning message will also be printed on the system console identifying the duplicate resource identifiers.

Supported Resources and General Requirements: SNA, FHP.

Valid subgroup: Not applicable.

NAME	OPERATION	OPERANDS

[symbol]	RESTABLE	`[,DSECT={NO }]`
		`        {YES}`
		`[,EXTRAS={0}]`
		`         {n}`
		`[,OPTIONS=(opt1,...)]`

```
DSECT={NO }
      {YES}
```

Function: Specifies whether the dummy section describing the format of a resource table entry is to be generated.

Format: YES or NO.

Default: DSECT=NO.

Note: If YES is specified, all other operands are ignored and the dummy section describing the format of a resource table entry is generated.

```
EXTRAS={0}
       {n}
```

Function: Specifies how many extra dummy resource table entries RESTABLE should generate (to permit you to supply entries for your own special processing).

Format: *n* is a decimal integer. The maximum number of total entries in the table including any extra entries, must not exceed 65,535.

Default: EXTRAS=0.

```
OPTIONS=(opt1,...)
```

Function: Specifies which option fields' data from the terminal table entries are to be appended to each resource-table entry.

Format: opt1 is an option field name that must conform to the rules for assembler language symbols. Up to 12 option field names may be listed; if more than one name is specified, each name must be separated by a comma and the list enclosed in parentheses.

Default: None. Specification is optional.

Note: If a particular option field is not defined for any terminal-table entry, the appended field in the resource table entry contains binary zeros.

Halfword boundary alignment is automatically maintained among the resource table entries. However, boundary alignment within the option field portion of the entry is your responsibility.

In addition, the total length of each entry may not exceed 255 characters.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Example

Assume that TR3270A is a 3270 display whose RESOURCE option field contains a resource identifier of 464. At initialization, the RESTABLE macro creates an entry in the resource table for TR3270A with a resource identifier of 464, and its corresponding TNT index:

464 TNT for TR3270A

The FHPBUILD macro in the incoming group of the DMH processing an inquiry message (for example, originating from terminal TR3270A) creates a TCAM origin address field (TOAF) in the FHP comprised of the 1-byte local node identifier and the 2-byte resource identifier. When the reply to the inquiry returns from the application in the same or another TCAM node, the message can be routed to station TR3270A, using the resource identifier of 464 in the TCAM destination address field (TDAF) in the FHP.

Return Codes

One of the following return codes is set in register 15:

Code	Meaning
X'00000000'	Successful execution
X'00000004'	No RESOURCE option field definition.
X'00000008'	Duplicate RESOURCE identifiers.

If register 15 contains a return code of X'08', no resource table has been built.

SCREEN Macro

The SCREEN macro:

- The SCREEN macro conditionally modifies the write operation for 3270 Information Display System stations.

The SCREEN macro can replace the existing command in the 3270 data stream with the command specified on the SCREEN macro.

By using the SCREEN and MSGEDIT macros for 3270 stations, the message handler can provide attachment transparency for output messages from an application program that are to be sent to 3270 stations. For output from the 3270 station, a MSGEDIT macro is coded to remove the Escape character.

Supported Resources and General Requirements: SNA.

Valid Subgroup: Outheader.

NAME	OPERATION	OPERANDS
[symbol]	SCREEN	[{WDC}] {WEA} {WRE} {EAU} {WSF} [,conchars] [,BLANK={YES }] {NO } {char}

```
BLANK={YES }
      {NO  }
      {char}
```

Function: Specifies whether EBCDIC blank characters are to be ignored when encountered in the character string in the message header being compared to the string specified by the *conchars* operand or whether blanks are to be part of the header string when encountered in it. If EBCDIC blanks are to be counted as part of the header string, this operand also specifies whether some other hexadecimal character is to be ignored when encountered in the header string.

Format: YES, NO, or *char*. *char* is a single character that may be specified in either character or hexadecimal format. If character format is specified, it may be unframed or framed with C'' or CL1'' characters. If hexadecimal format is specified, it must be framed with X'' or XL1'' characters.

Default: BLANK=YES.

Note: YES specifies that the EBCDIC blank character (X'40') is to be ignored by this macro whenever it is encountered in the header character string being checked against the control character string specified by the *conchars* operand. For example, if BLANK=YES and an eight-byte field in the header is being checked by this macro, a blank appearing in the fifth byte of the field is ignored and the sixth through ninth bytes are considered to be the last four bytes of the field (assuming that no blanks are coded in the sixth through ninth bytes).

NO specifies that the EBCDIC blank character is to be treated in the same way as any other character when it is encountered by this macro in the header string being compared to the string specified by *conchars*.

char specifies that the single character replacing *char* be ignored by this macro whenever it is encountered in the header string being compared to the string specified by the *conchars* operand. That is, the macro automatically skips over the character without performing a comparison and goes on to check the next character in the header. If BLANK=*char* and *char* is not the EBCDIC blank character, the EBCDIC blank is not ignored by this macro when it is encountered in the header string, but is compared to the character in the corresponding space in the *conchars* string, in the same way as any other character.

This operand is meaningless unless the *conchars* operand is also specified.

`conchars`

Function: Specifies the character or character string that, if found in the header as the next nonblank field, executes the function.

Format: One to eight nonblank characters in character or hexadecimal format. If character format is used, the string may be unframed or framed with C'' or CLn'' characters. If hexadecimal format is used, the string must be framed with X'' or XLn'' characters.

Default: None. Specification is optional.

Note: If this operand is omitted, the SCREEN function is performed unconditionally. If the next field in the header does not match this operand, the function is not performed and the return code is set to zero.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

SCREEN

```
{WDC}
{WEA}
{WRE}
{EAU}
{WSF}
```

Function: Specifies the type of write operation.

Format: WDC, WEA, WRE, EAU, WSF.

Default: WDC.

The operation type keywords correspond to 3270 command as follows:

WDC	Write
WRE	Erase Write
WEA	Erase Write Alternate
EAU	Erase All Unprotected
WSF	Write Structured Fields.

Refer to the *3270 Component Description* manual for additional information on these commands.

For 3270 stations, if an escape character is found in the first unit's data, the adjacent command code is replaced by the remote command code specified with this operand. For SNA stations, the SCREEN macro assumes that the first data character in the first unit is the command code and replaces the character with the remote command code specified with this operand.

```
{WRE}
{WDC}
```

Function: Specifies the type of write operation for the IBM 2265 or the IBM 2260.

Format: WRE or WDC.

Default: None. This operand is optional if you are checking the type of operation in effect for the message being processed, but is required if changing the operation type.

Note: WRE specifies a write erase operation the screen before the next segment is displayed. WDC specifies a write display control operation.

A SCREEN macro specifying no WRE or WDC operation may be issued to check the type of write operation in effect for the message being processed without changing the type of operation.

Return Codes

Coding	**Meaning**
X'00000000'	Successful execution.
X'00000004'	Command requested invalid for 3270's.
X'00000008'	Escape character not found.

SENDMSG Macro

The SENDMSG macro:

- Permits you to generate and send one or more messages or a table to any external LU or application program
- Permits you to generate an extended operator command
- Operates in conjunction with the NEWMSG macro and the EXMSG macro
- Causes a fixed header prefix to be included with the generated message
- May be used more than once in a subgroup
- Requires that DLQ= be specified before INTRO execution.

You may use the SENDMSG macro in one of two ways to generate exception requests when executed in the inmessage and/or outmessage subgroups of an MH.

Using one approach, you define an exception request table in the MCP by coding a series of EXMSG macros. If the EXIT= operand is omitted from the SENDMSG macro, SENDMSG processing automatically accesses the exception request table to obtain the appropriate request. If you code EXITCDE=NEWMSG on SENDMSG, the contents of the NEWMSG option field are used as an index into the exception request table to obtain the correct request description; otherwise, the value directly specified on EXITCDE= is used as the index into the exception request table.

SENDMSG processing uses the descriptive information in the appropriate entry in the exception request table to generate the designated message. Messages generated automatically in this way have an "E" (X'C5') in the mode byte of the FHP of the generated message. They will not be retrieved by the SEND command if no destination is specified on the SEND command, but, in every other way, they are normal messages and can be subsequently processed.

All messages generated by the SENDMSG macro have a null input sequence number.

If you specify a user exit on the EXMSG macro, the exit is given control at the appropriate point during SENDMSG processing. Refer to the description of the EXMSG macro for further details.

The message currently being processed should have an FHP if the EXMSG macro describing the message to be generated specified DEST=ORIG or DEST=DAF or CPYDEST=ORIG or CPYDEST=DAF.

If there is no FHP with the current message or HEADER=NO is specified, the generated message (if any) is routed to the current terminal if DEST=ORIG is specified in the EXMSG macro. If the current terminal is an application program TPROCESS name corresponding to a PUT or WRITE DCB, the message can be routed only if the TPROCESS macro specified the name of a queue in this TCAM system that can receive replies using the ALTDEST= operand. This is the same procedure used by application programs that issue operator control commands. If EXTEXT= is specified in the EXMSG macro, no original message text is automatically included if there is no FHP with the current message or HEADER=NO is specified.

In general, the EXMSG macro should be used with SENDMSG for exception request generation. However, if your special requirements necessitate the use of an exit routine, the following discussion is a general description of SENDMSG macro operation when such a routine is specified:

- In the macro, you specify the name of the user-written exit routine via the EXIT= operand.
- SENDMSG passes a parameter list to the user-written exit routine. The parameter list contains information for use by the exit routine in constructing a message.

- The exit routine must complete the return portion of the parameter list which describes the new message's format and specifies where it is to be routed. The exit routine then passes control back to SENDMSG.
- Based on the specifications in the returned parameter list, SENDMSG schedules the message for routing.

If the user exit routine is specified and it sets the return portion of the parameter list to zero, SENDMSG treats the action as a logical no-op. The exit routine can then simply perform as a user's logical extension to the inmessage or outmessage subgroups generating no messages. The WTOMSG CSECT in the model MCPs and the KEYDAFBD CSECT in the extended networking model MCPs provide a coding example.

The date and time in the FHP of any messages generated by the SENDMSG macro (with or without EXIT) is the current date and time unless an exit routine returns an FHP address of an FHP in which the date field is already initialized.

You may wish to code a single SENDMSG exit routine in an MCP to generate all required messages. The EXITCDE= operand may be used to specify a value which is passed as an input parameter to the user exit routine. The meaning of the value depends on how the routine is coded.

Using the alternative approach, you code the EXIT= operand on the SENDMSG macro to take a user exit that is wholly responsible for generating the message.

Supported Resources and General Requirements: ALL (if APP, in inmessage subgroup only).

Valid subgroups: Inmessage or outmessage.

NAME	OPERATION	OPERANDS
[symbol]	SENDMSG	EXIT={ACCTING}
		{name }
		[,DSECT={NO }]
		{YES}
		[,EXITCDE={0 }]
		{n }
		{(n,MSG)}
		{NEWMSG }
		[,EXITTYP={V}]
		{A}
		[,HEADER={YES}]
		{NO }
		[,PRI={255}]
		{n }

```
DSECT={NO }
      {YES}
```

Function: Specifies whether the dummy section describing the input parameter list for the SENDMSG user exit routine.

Format: YES or NO.

Default: DSECT=NO.

Note: If YES is specified, all other operands are ignored and the input parameter list for the SENDMSG user exit routine dummy section is generated.

```
EXIT={ACCTING}
     {name   }
```

Function: Provides SENDMSG with the address of a user exit routine or indicates that the special ACCTING routine exit is to be taken. Omission of this operand indicates that the message identified by the EXITCDE= operand should be selected from the EXMSG table and generated by SENDMSG.

Format: *name* or ACCTING. *name* must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

name is the CSECT name of the user exit routine. (The EXITTYP= operand specifies whether a V-type or an A-type address is to be used.)

ACCTING specifies that the portion of the online accounting area that has been filled is to be sent. If ACCTING is specified, this SENDMSG macro must be preceded by a TESTBR macro specifying SPECACT=ACCTING.

- Register 1 contains the parameter list address
- Register 13 contains the address of an area to be used by the user-written routine for saving registers used by SENDMSG
- Register 14 contains the return register to SENDMSG
- Register 15 contains the user exit routine address

The user exit routine must save all of the registers passed to it and follow normal OS/VS register save-area linkage conventions. An example of exit routine register initialization follows:

```
SNDEXIT    CSECT
           USING    *,15
           USING    SNDINPTD,1
           SAVE     (14,12)
           LR       12,15           LOAD NEW BASE
           DROP     15
           USING    SNDEXIT,12      REESTABLISH BASE
                                    PERFORM OS SAVE AREA LINKAGE
```

The parameter list is divided into two parts, an *input portion* that contains information passed to the exit routine, and a *return portion* that contains data from the exit routine defining the new message to be generated. To access the parameter list symbolically, code a SENDMSG macro specifying DSECT=YES in your exit routine.

An expansion of the SENDMSG exit routine parameter list DSECT is included in *TCAM Program Reference Summary.*

If the return portion of the parameter list is not filled in, SENDMSG considers this as a request for no action (if, for example, the user exit found an insufficient buffering capacity for the table it wished to transmit).

The input portion of the parameter list contains information passed to the exit routine, including:

- The EXITCDE= value from the macro or the next message number from the NEWMSG option field
- Addresses of the message buffer, FHP (if present) and data
- External LU name and name of GROUP macro for the external LU session partner.

The return portion of the parameter list contains data from the exit routine that specifies for the new message to be generated:

- The address and length of the message
- An FHP address (optional) If one is not specified, SENDMSG code constructs the FHP.
- Whether the message is an operator control command
- Whether this is a logical reject message If the logical reject message indicator is turned on, the TCSENDBL macro increments the resource's ERRCOUNT option field, which may be displayed online by the extended Display Statistics for a Resource (DATAname) operator control command
- The message destination, which may be:
  - An entry in the routing key table
  - An entry in the terminal table
  - The message originator (based on the originator's network address in the FHP
  - The basic or extended operator control facility (if this is an operator command).

```
EXITCDE={(0)     }
        {n       }
        {(n,MSG)}
        {NEWMSG  }
```

Function: Specifies whether to use the NEWMSG option field values as indices into the EXMSG table, to use *n* as part of the parameter list passed to the user exit routine coded on the EXIT= operand, or as an index *n* into the EXMSG table.

Format: NEWMSG, *n, (n,* MSG), or 0. *n* is a decimal integer from 1 to 255. The macro expands to AL1(n) so that equated values may be used. It is your responsibility to assign this operand a value that falls within the prescribed limits.

Default: EXITCDE=0.

NEWMSG specifies that the SENDMSG macro is to see if any NEWMSG macros were executed in the MH processing path. If so, SENDMSG is to access the next number from the NEWMSG option field and use it as the EXITCDE value. SENDMSG macro processing selects and generates one message from the exception request table corresponding to each number in the NEWMSG option field. Or, if the EXIT= operand is specified, the SENDMSG macro enters the user's exit routine once for each number in the NEWMSG option field. The exit routine must construct one message at a time and then return to SENDMSG. When SENDMSG detects no more messages to be built, it passes control to the next sequential macro instruction in the current subgroup.

It is important to understand that if both EXITCDE=n and EXIT is specified, the only way a message will be sent is via the user exit routine. If EXITCDE=n is specified without the EXIT= operand, the *n* value will be used as an index into the EXMSG table.

If a single exit routine is used to generate several different messages, the EXITCDE= n specification may be used to branch to a specific section of the code. For example, the routine might (after saving the input registers) process a branch table as shown below:

```
             SR          2,2                  CLEAR A REGISTER
             IC          2,SNDXITCD           GET THE EXIT CODE
             SLL         2,2                  MULTIPLY BY 4
             B           BRANTAB(2)           BRANCH IN THE TABLE
BRANTAB      B           CODE0                EXITCDE=0
             B           CODE1                EXITCDE=1
             B           CODE2                EXITCDE=2
             .
             .
             .
             B           CODEn                etc.
```

Another way of creating a branch table is to use the CODEBR macro instruction:

```
             SR          15,15                CLEAR REGISTER 15
             IC          15,                  SNDXITCD GET THE EXIT CODE
             SLL         15,2                 MULTIPLY BY 4
             CODEBR      NEG=BANDNEWS,
                         POS=(EXITCDE1,EXITCDE2,EXITCDE3,
                           ... EXITCDEn)

EXITCDE0     EQU         *
             .
             .
             .
```

If you want to enter the user exit routine and also want to use the *n* value as an index into the EXMSG table, then (n, MSG) must be specified. However, if a message is returned from the user exit routine, that message will be sent and not a message from the EXMSG table. To ensure that *n* is used as an EXMSG table index, the user exit routine should clear the user return parameter list.

```
EXITTYP={V}
        {A}
```

Function: Indicates the location of the user-written routine specified in the EXIT= operand.

Format: V or A.

Default: EXITTYP = A.

Note: A specifies that the routine is assembled within the MCP (an A-type address constant is generated).

V specifies that the routine is a module in the load library (a V-type address constant is generated).

```
HEADER={YES}
       {NO }
```

Function: Indicates whether the current message header is to be recalled and made available to either the exit routine or to SENDMSG macro processing of EXMSG macro defined messages.

Format: YES or NO.

Default: HEADER = YES.

Note: Any value other than YES or NO is treated as the default.

This operand is ignored if EXIT = ACCTING is specified.

YES specifies that the current message header is to be recalled. If no recall is possible, the SENDMSG macro does not execute.

NO specifies unconditional execution but without header recall. It permits an exit and/or message generation in message handling situations where recall is not possible, such as with zero length buffers in an incoming DMH.

With HEADER = NO, the SENDMSG macro exit parameter list is unchanged and is the same as for recalled messages without FHPs except that the current buffer will be empty (no data, no reserves, no FHP, and the available space is set to binary zeros).

```
PRI={255}
    {n  }
```

Function: Specifies the message priority level to be assigned to the message returned by the user exit routine.

Format: *n* where *n* is an unframed decimal integer from 1 through 255; it may also be a symbol equated to such a number.

Default: PRI = 255.

Note: This operand is ignored if your exit routine returns an FHP address in the return parameter list.

The special accounting routine returns an FHP with message priority level zero specified.

Return Codes

None.

SEQUENCE Macro

The SEQUENCE macro:

- Checks the input sequence number of an incoming message
- Inserts the output sequence number of an outgoing message
- Should be specified only once in each subgroup
- Should not be used to insert a sequence number in a message containing a fixed header prefix.

If specified in an inheader subgroup, SEQUENCE scans the input sequence number field in the header of each message. If the sequence number is not one greater than the sequence number of the last message received from the same external LU or application program, an error flag is set in bit 3 or 4 of the message error record assigned to the message (depending on whether the number is high or low, respectively), and a return code indicating an error is set in register 15.

The header field for the input sequence number may contain up to four sequence characters. (Leading zeros may be omitted in the sequence number entered at the external LU.) This field must contain a decimal representation of the input sequence number and should be delimited by a blank. For example, if the sequence number is twelve, the field must contain a character 1 followed by a character 2 followed by a delimiting blank. If no delimiting blank is found, the SEQUENCE macro is not executed. At the time SEQUENCE looks at the field, the characters should have been translated into EBCDIC by a CODE macro. Reserve five bytes in your header by coding the RESERVE= operand of the GROUP macro (or the PCB macro) for insertion of the output sequence number, if used.

TCAM maintains internal counters in the terminal-table entry to keep track of the incoming and outgoing sequence numbers for each external LU and application program. If the SEQUENCE macro is issued in an inheader subgroup, the first message from an external LU or application program must contain the same input sequence number as the input counter for that external LU or application program. TCAM initially sets each input counter to 1. The next incoming message after 9999 must be numbered 1. Processing continues after the maximum number is reached.

In general, SEQUENCE in an inheader subgroup increments the input counter for each message having a correct input sequence number in the header. If, however, a CANCELMG macro cancels a message, the input sequence number is not incremented.

If specified in an outheader subgroup, SEQUENCE places an output sequence number in the header of each outgoing message handled by the MH. The five-byte output sequence number (a blank followed by four EBCDIC characters) is inserted immediately following the byte to which the scan pointer is pointing when SEQUENCE executes. When the first message is sent to an external LU or application program, a 1 is placed in the output counter for that external LU or application program; as each succeeding output message is handled, this sequence number is incremented by 1 and the resulting number inserted in the header. (A count is maintained for each external LU. A message in error routed to an external LU or application program using a REDIRECT macro retains the output sequence number placed in it.

Use of SEQUENCE is optional. If used, it must appear within an inheader or outheader subgroup. Its position must correspond to the relative position, within the header, of the sequence-number field.

TCAM increments the input sequence number counter in the terminal-table entry only if a SEQUENCE macro is issued in the inheader subgroup. TCAM increments the output sequence number counter in the terminal-table entry each time a message is sent to the external LU or application program.

SEQUENCE

Supported Resources and General Requirements: ALL.

Valid subgroups: Inheader and outheader.

NAME	OPERATION	OPERANDS
[symbol]	SEQUENCE	(no operands)

symbol

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Return Codes

One of the following return codes is set in register 15:

X'00000000'	Successful execution.
X'00000004'	Inheader Subgroup: Sequence number in message low. Outheader Subgroup: Insufficient reserve space.
X'00000008'	Inheader Subgroup: Sequence number in message high. Outheader Subgroup: Not applicable.
X'0000000C'	Inheader Subgroup: Origin unknown. Outheader Subgroup: Not applicable.
X'FFFFFFFC'	Either: • Delimiting blank not found • Zero length buffer • Normal FM data.

SETEOF Macro

The SETEOF macro:

- Sets a bit in the buffer prefix to indicate an EOF message
- Notifies the application program's access method modules to take the DCB end-of-data (EODAD) exit when the GET or READ macro following receipt of this message is executed.

The SETEOF macro identifies the last message in a data file being processed by an application program. When the application program receives a message flagged by SETEOF, the next GET or READ/CHECK macro it issues after the complete message has been received causes control to be passed to the routine whose address is specified EODAD= operand of the application program input DCB macro for the destination queue addressed by the GET or READ. Thus, by issuing a SETEOF macro, you cause the application program to stop obtaining work units from one or more destination queues and do whatever is specified by the routine located at the EODAD address.

The SETEOF macro is issued in the outheader subgroup of the outgoing group of the MH handling messages routed to an application program.

Note: In the case of multiple-buffer headers, to be effective, SETEOF must be executed for the first header buffer.

Supported Resources and General Requirements: APP.

Valid subgroup: Outheader.

NAME	OPERATION	OPERANDS

[symbol]	SETEOF	[conchars]
		[,BLANK={<u>YES</u> }]
		{NO }
		{char}

```
BLANK={YES }
      {NO  }
      {char}
```

Function: Specifies whether EBCDIC blank characters are to be ignored when encountered in the character string in the message header being compared to the string specified by the *conchars* operand or whether they are to be part of the header string when encountered in it. If EBCDIC blanks are to be counted as part of the header string, this operand also specifies whether another hexadecimal character is to be ignored when encountered in the header string.

Format: YES, NO, or *char*. *char* is a single character that may be specified in either character or hexadecimal format. If character format is specified, it may be unframed or framed with C'' or CL1'' characters. If hexadecimal format is specified, it must be framed with X'' or XL1'' characters.

Default: BLANK=YES.

Note: YES specifies that the EBCDIC blank character (X'40') is to be ignored by this macro whenever it is encountered in the header character

string being checked against the control character string specified by the *conchars* operand. For example, if BLANK=YES and an eight-byte field in the header is being checked by this macro, a blank appearing in the fifth byte of the field is ignored and the sixth through ninth bytes are considered to be the last four bytes of the field (assuming that no blanks are coded in the sixth through ninth bytes).

NO specifies that the EBCDIC blank character is to be treated in the same way as any other character when it is encountered by this macro in the header string being compared to the string specified by *conchars*.

char specifies that the single character replacing *char* to be ignored by this macro whenever it is encountered in the header string being compared to the string specified by the *conchars* operand. That is, the macro automatically skips over the character without performing a comparison and goes on to check the next character in the header. If BLANK=*char* and *char* is not the EBCDIC blank character, the EBCDIC blank is not ignored by this macro when it is encountered in the header string, but is compared to the character in the corresponding space in the *conchars* string in the same way as any other character.

This operand is meaningless unless the *conchars* operand is also specified.

`conchars`

Function: Specifies the character or character string that, if found in the header as the next nonblank field, executes the function.

Format: One to eight nonblank characters in character or hexadecimal format. If character format is used, the string may be unframed or framed with C'' or CLn'' characters. If hexadecimal format is used, the string must be framed with X'' or XLn'' characters.

Default: None. Specification is optional.

Note: If this operand is omitted, the SETEOF function is performed unconditionally. If the next field in the header does not match this operand, the function is not performed.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

None.

SETEOM Macro

The SETEOM macro:

- Allows dynamic control of the amount of data required for a logical message
- Specifies the action to be taken following the end of the message with the data in a buffer
- Allows removal of delimiting characters (end-of-message indicators)
- May be executed only once for any give input

The SETEOM macro allows you to dynamically control the amount of data in any message either by specifying the length or by specifying an end-of-message (EOM) indicator. This macro causes data in an incoming transmission sequence to be either blocked or deblocked. If an EOM indicator is found in a buffer, then two messages may be formed; that is, incoming data is deblocked to form one or more messages. If an EOM indicator is *not* found, the transmission being handled is processed by the incoming MH, and subsequent transmissions are processed in the same way until an EOM indicator is detected; that is, multiple buffers are blocked to form a single message. See the *TCAM Installation Guide* for a more complete description of how to use the SETEOM macro.

Care should be taken when PROCESS=YES is specified on SETEOM. TCAM keeps all deblocked logical messages in TCAM buffers on a SETEOM queue until the real end of chain is received. TCAM keeps the messages because it does not allow a completed logical message to continue through the MH and be queued for its destination without first ensuring that the entire chain has been received without error.

TCAM cannot pass logical messages to the MH following execution of the SETEOM macro until real SNA end of chain has been received. If an error occurs before the end of chain, there is no way to retransmit only the RU in error. Therefore, the entire chain must be retransmitted. For this reason, you should not use SETEOM PROCESS=YES unless you ensure that you will not exhaust TCAM buffers while processing long SNA chains. Instead, you should deblock SNA chains in a TCAM application program using the RECDEL operand of the TPROCESS macro. For more information, see the logical message section of the *TCAM Installation Guide.*

Supported Resources and General Requirements: SNA.

Valid subgroup: Inblock.

NAME	OPERATION	OPERANDS

[symbol]	SETEOM	[ENDCHAR={chars }]
		{opfield}
		[,LENGTH=({integer },opfield2)]
		{opfield1}
		[,PROCESS={YES}]
		{NO }
		[,REMOVE={YES}]
		{NO }

```
ENDCHAR={chars   }
        {opfield}
```

Function: Specifies the character or character string that delimits the message.

Format: *chars* may be from one to eight nonblank characters specified in character or hexadecimal format. If character format is specified, the field must be framed with C'' or CLn'' characters. If hexadecimal format is specified, the field must be framed with X'' or XLn'' characters. *n* must be the length of the characters.

Default: None. This operand must be specified if the LENGTH= operand is not specified. If the LENGTH= operand is specified, this operand is optional.

Note: *opfield* is the name of a field defined by an OPTION macro containing the character or character string used as an EOM indicator. The first byte of the option field contains the length of the delimiter, followed by the delimiter.

If both the ENDCHAR= and the LENGTH= operands are specified, the message delimiter is either the length or the characters, whichever is encountered first.

```
LENGTH=({integer },opfield2)
        {opfield1}
```

Function: Specifies the number of bytes in the message.

Format: *integer* or *opfield1* followed by *opfield2,* enclosed in parentheses. *integer* is a decimal integer that may be up to 65,535. *opfield1* is the name of a halfword option field defined by an OPTION macro containing the length, in bytes, of the message. *opfield2* is a halfword option field initially set to zero that maintains a count of the number of bytes already received.

Default: None. Must be specified if the ENDCHAR= operand is not specified. If the ENDCHAR= operand is specified, this operand is optional.

Maximum: 65,535.

Note: If both the ENDCHAR= and the LENGTH= operands are specified the message delimiter is either the length or the characters, whichever is encountered first.

```
PROCESS={YES}
        {NO }
```

Function: Specifies whether any data following an EOM indicator is to be processed or discarded.

Format: YES or NO.

Default: PROCESS=NO.

Note: YES indicates that any remaining portion of the data is to be processed. NO indicates that any remaining portion is to be discarded; that is, the original message is truncated.

When YES is specified, processing of the second portion (which is the first buffer of a new message) begins at the SETEOM macro. When the EOM indicator is encountered, a new buffer is obtained. The remaining portion of the buffer is moved into the new buffer that becomes a header buffer for the new message with the scan pointer set to the beginning of the buffer. Thus, this buffer will be the next buffer to be processed by this MH. The reserve space specified by the GROUP is reserved in this new header buffer when it is initialized. All subsequent buffers of the new message also begin processing at the SETEOM macro. If sufficient buffer units are not available to move the remaining portion of the current buffer, a return code of X'10' is set in register 15.

NO should be used when any remaining portion of the incoming data is to be discarded. This means that not only the remaining portion of the current buffer is dropped, but all the subsequent buffers, if any, of the current messages are returned to the buffer pool.

If the LOCK macro is coded for logical messages, PROCESS=NO must be specified.

```
REMOVE={YES}
       {NO }
```

Function: Removes EOM characters from buffers containing logical messages.

Format: YES or NO.

Default: REMOVE=NO.

Note: YES causes EOM characters to be removed from incoming buffers containing parts of one or more logical messages.

NO causes EOM characters to remain in the buffer in which they appear.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

One of the following return codes is set in register 15:

Code	Meaning
X'00000000'	Successful execution; message not delimited by this buffer.
X'00000004'	Successful execution; an EOM indicator delimited this message.
X'00000008'	Successful execution; the user-specified length delimited this message.
X'0000000C'	Successful execution; an EOM indicator *and* the user-specified length delimited this message.
X'00000010'	Error; no buffer units are available.
X'00000014'	Error; buffer was not processed because no option field was found.

SETSCAN Macro

The SETSCAN macro:

- Explicitly moves the scan pointer forward or backward
- Returns (in a designated register) the address of the last character of a specified character string
- Returns in register 15 the current address of the scan pointer.

The SETSCAN macro explicitly repositions the scan pointer forward or backward in the buffer, if specified. After the previous macro has executed, the scan pointer is positioned at the last character of the field acted upon by that macro. SETSCAN moves the scan pointer a specified number of nonblank characters to the byte following a specified character or the last character of a specified character string so that one or more fields are skipped.

Alternatively, the scan pointer is not moved. Instead, either a designated character string is located and the address of the last character of the string is placed in a specified register, or the current address of the scan pointer is placed in register 15.

When an outgoing message is processed by an outheader subgroup, STARTMH positions the scan pointer to the last reserve character in the buffer (beyond the FHP, if present); or, if there are no reserve characters or FHP, to the last byte of the buffer prefix. If this is not the location of the next header field to be processed, you may employ SETSCAN to move the scan pointer to the byte immediately preceding the next header field to be processed.

For more information on the use of SETSCAN and the scan pointer, see the chapter of the *TCAM Installation Guide* titled "Coding the Message Handler."

Supported Resources and General Requirements: ALL.

Valid subgroups: Inheader and outheader. Inblock, inbuffer, and outbuffer are valid subgroups only if SKIPFLD=LAST is specified.

NAME	OPERATION	OPERANDS
[symbol]	SETSCAN	{skipchars}
		{integer }
		[,BLANK={YES }]
		{NO }
		{char}
		[,MOVE={RETURN}]
		{KEEP }
		[,POINT={BACK }]
		{FORWARD}
		[,RESULT={(register)}]
		{(15) }
		[,SKIPFLD={FIRST }]
		{SOH }
		{DEL }
		{LAST }
		{integer}

```
BLANK={YES }
      {NO  }
      {char}
```

Function: Specifies whether EBCDIC blank characters are to be ignored when being counted or when encountered in the character string in the message header being compared to the string specified by the *skipchars* operand, or whether blanks are to be counted as characters or as part of the header string when encountered in it. If EBCDIC blanks are to be counted as characters or as part of the header string, this operand also specifies whether some other hexadecimal character is to be ignored when skipping or when encountered in the header string.

Format: YES, NO, or *char*. *char* is a single character that may be specified in either character or hexadecimal format. If character format is specified, it may be unframed or framed with C'' or CL1'' characters. If hexadecimal format is specified, it must be framed with X'' or XL1'' characters.

Default: BLANK=YES.

Note: YES specifies that the EBCDIC blank character (X'40') is to be ignored by the macro whenever characters are being skipped or whenever it is encountered in the header character string being checked against the skip character string specified by the *skipchars* operand. For example, if BLANK=YES is coded and an eight-byte field in the header is being checked by this macro, a blank appearing in the fifth byte of the field is ignored and the sixth through ninth bytes are considered to be the last four bytes of the field (assuming that no blanks are coded in the sixth through ninth bytes).

Notes: For 3270 input, care must be taken to ensure that data stream control characters (cursor1,cursor2) at the beginning of a message do not cause unpredictable results; that is, the cursor address returned to the host processor may contain zero, one, or two X'40's depending on the location of the operator's last entry to the hardware buffer.

NO specifies that the EBCDIC blank character is to be treated in the same way as any other character when characters are being skipped or whenever it is encountered in the header string being compared to the string specified by *skipchars*. If BLANK=NO is specified, the SKIPFLD= operand must not be specified.

char specifies that the single character replacing *char* be ignored by this macro whenever characters are being skipped or whenever it is encountered in the header string being compared to the string specified by *skipchars* operand. That is, the macro automatically skips over the character without counting or performing a comparison and goes on to check the next character in the header. If BLANK=*char* and *char* is not the EBCDIC blank character EBCDIC blank is not ignored by this macro when it is encountered in the header, but is counted or compared to the character in the corresponding space in the *skipchars* string, in the same way as any other character.

```
MOVE={RETURN}
     {KEEP  }
```

Function: Specifies whether the scan pointer is to be moved to the designated position before the next macro is issued or is to remain stationary with the specified character string being located and the main-storage address of the last character in the string being returned in a designated register.

Format: RETURN or KEEP.

Default: MOVE=KEEP.

Note: If *integer* is replaced by zero, this operand is ignored. If *integer* is specified, MOVE=KEEP is assumed by TCAM even if the macro is coded MOVE=RETURN.

KEEP specifies whether the scan pointer is to be moved to the byte following the last byte of the designated character string (if *skipchars* is coded), to the byte following the designated number of characters (if *integer* is coded) or to the byte following the position specified by the SKIPFLD= operand; the pointer remains in its new location until the next macro affecting the scan pointer is issued.

RETURN specifies that the scan pointer is not to be moved. Instead, the specified character string is located and the main-storage address of the last character in the string is returned in the register designated by the RESULT= operand.

MOVE=RETURN must not be specified if POINT=BACK is specified, or if *integer* is specified rather than *skipchars,* or if SKIPFLD=integer is specified.

```
POINT={BACK   }
      {FORWARD}
```

Function: Specifies whether the scan pointer is to be moved forward or backward.

Format: BACK or FORWARD.

Default: POINT=FORWARD.

Note: FORWARD specifies that the scan pointer is to be moved forward. BACK specifies that the scan pointer is to be moved backward.

If POINT=BACK is specified, neither *skipchars* nor MOVE=RETURN can be specified. If POINT=BACK is specified, the scan pointer cannot be moved out of the buffer in which it is located; if *integer* is greater than the number of characters preceding the scan pointer in the buffer, the macro is not executed and a return code of X'04' is placed in the rightmost byte of register 15. POINT=BACK must not be specified if the SKIPFLD= operand is specified.

```
RESULT={(register)}
       {(15)      }
```

Function: Specifies the general register into which the main-storage address of the last character of the designated character string or the last character of the current message is to be placed once the string is located.

Format: A general register, 2 through 11, or 15, enclosed in parentheses.

Default: RESULT=(15).

Note: If the desired character string is found, the address of its last character is placed in the register. If RESULT=(15) is coded and the character string is not found in this buffer (if *skipchars* is coded) or if the integer specified would take the scan pointer out of this buffer (if *integer* is specified), the macro does not execute and a return code of X'00' is placed in the rightmost byte of register 15.

If a register other than 15 is specified and the character string is not found in the buffer or the integer specified would take the scan pointer out of the buffer, the macro does not execute and a return code of X'04' is placed in the low-order byte of register 15; in this case, the contents of the specified register are unchanged.

If *integer* is coded as anything other than zero, the RESULT= operand, if coded, must be coded as RESULT=(15).

If *skipchars* is coded and MOVE=KEEP is specified (or MOVE= is omitted and the default of MOVE=KEEP is taken), the RESULT= operand, if coded, must be coded as RESULT=(15).

The actual register number must be specified or defaulted to; no symbolic names for a register are allowed on the SETSCAN macro.

```
{skipchars}
{integer  }
```

Function: Specifies the new location of the scan pointer, either a character string to be located or a number of characters to be advanced.

Format: skipchars or *integer*. *skipchars* can be one to eight nonblank characters in character or hexadecimal format. If character format is used, the string may be unframed or framed with C'' or CLn'' characters. If hexadecimal format is used, the string must be framed with X'' or XLn'' characters. *integer* is an integer in decimal format.

Default: None. This operand is required unless SKIPFLD= is coded.

Maximum: For *integer*, 255.

Note: skipchars is the character or character string the scan pointer is to be moved or whose address is to be placed in a register.

The MOVE=RETURN option, when used with *skipchars*, does not cause the scan pointer to be moved; however, the address of the last character in the character string, if found within this buffer, is placed in the register specified by RESULT=(register). With this option, SETSCAN does not operate across buffers; and if the character or character string is not found, a return code is set. (See the description of MOVE=RETURN.)

The MOVE=KEEP option, when used with *skipchars*, causes the scan pointer to be moved to the byte following the character or the last byte of the designated character string. If the character or character string designated by *skipchars* is not found, the scan pointer is set to the end of the buffer plus two. With this option, SETSCAN operates across buffers until the character or character string is found.

The main storage address of the current location of the scan pointer minus one is returned if the scan pointer is within the current buffer.

integer is the number of nonblank characters to be skipped. If an attempt is made to set the scan pointer outside of the current buffer, SETSCAN is not executed, and control passes to the next macro. In the case of multiple-unit buffers, with the scan pointer pointing to the last byte of a unit, a SETSCAN 0 returns the address minus one of the *first* available byte in the *next* unit.

If the current buffer is a zero-length buffer, minus 4 (X'FFFFFFFC') is returned in register 15.

The POINT=BACK operand cannot be specified with *skipchars*. If 0 is coded for *integer*, the MOVE= and RESULT= operands are ignored.

If *integer* is specified, MOVE=KEEP is assumed even if SETSCAN is coded MOVE=RETURN.

If the SKIPFLD= operand is specified, the *integer* or *skipchars* operand is not allowed.

```
SKIPFLD={FIRST  }
        {SOH    }
        {DEL    }
        {LAST   }
        {integer}
```

Function: Specifies the new position of the scan pointer.

Format: FIRST, SOH, DEL, LAST, or *integer*. *integer* is an unframed decimal integer.

Default: None. Specification is optional in the inheader and outheader subgroups. However, SKIPFLD=LAST is required in the inblock, inbuffer, and outbuffer subgroups.

Note: If SKIPFLD=FIRST is coded and the buffer being processed is the first in this message, the scan pointer is moved to the first data character in the buffer and, its address is returned in the register designated by the

RESULT= operand if MOVE=KEEP is specified. The first data character is considered to be the first character beyond the reserve characters in the buffer. If a fixed header prefix (FHP) is present, the scan pointer points to the beginning of the FHP after this operand executes.

If MOVE=RETURN is coded, the main-storage address of the first data character in the buffer is returned in the register designated by the RESULT= operand.

If SKIPFLD=SOH is coded and the buffer being processed is the first in this message, the scan pointer is moved to the first byte of data beyond the FHP in the buffer. The message must contain an FHP if SKIPFLD=SOH is coded.

If MOVE=RETURN is coded, the main-storage address of the first data character in the buffer is returned in the register designated by the RESULT= operand.

For messages containing an FHP, SKIPFLD=SOH should be used to locate the first data character. In this case, the first data character is the character indicated by the FHPHEADP field of the FHP. As set by the FHPBUILD macro, this is the offset from the beginning of the FHP to the first data character beyond the reserve characters in the buffer. The MODEFLD macro updates this field to point past the mode character; the KEEP=NO operand of the DESTFLD macro also updates it to point past the message destination; thereby causing these fields to be logically deleted from the message.

If SKIPFLD=DEL and MOVE=KEEP are coded, and the buffer being processed is the first in this message, the scan pointer is moved to the position indicated by the "beginning of text" offset in the FHPTEXT field of the FHP. As set by the FHPBUILD macro, this offset is to the last character of the first buffer unit. The address is returned in the register designated by the RESULT= operand.

If SKIPFLD=LAST is coded and the buffer being processed is the last in this message, the scan pointer is moved to the last character in the buffer and its address is returned in the register designated by the RESULT= operand if MOVE=KEEP is specified. If MOVE=RETURN is coded, the main-storage address of the last character in the buffer is returned in the register designated by the RESULT= operand.

If SKIPFLD=*integer* is coded, where *integer* is between 1 and 255, the scan pointer is advanced the number of fields indicated and its address is returned in the register designated by the RESULT= operand. Each such field consists of a character string delimited by blanks. The scan pointer is set to the byte following the last byte of the last field skipped. The address returned may be in following buffers if multiple buffer headers had to be processed to skip the fields. MOVE=KEEP is required, and MOVE=RETURN is not valid. The SETSCAN operands *skipchars* and *integer* cannot be coded when SKIPFLD is coded. If SKIPFLD= operand is specified, BLANK= NO or POINT= BACK cannot be specified.

symbol

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Examples

Example 1: A SETSCAN macro that causes the scan pointer to skip forward over 5 characters and remain in its new position:

```
SETSCAN 5,POINT=FORWARD,MOVE=KEEP
```

Example 2: A SETSCAN macro that results in no movement by the scan pointer, but causes the address of the character '=' (located to the right of the pointer) to be placed in register 2:

```
SETSCAN C'=',POINT=FORWARD,MOVE=RETURN,RESULT=(2)
```

Example 3: A SETSCAN macro that results in the scan pointer being moved to the last character of the message:

```
SETSCAN SKIPFLD=LAST
```

Return Codes

When the SETSCAN macro executes, registers are set as shown in Figure 2-2 on page 2-360.

Macro Operand	Return Code		User Register		Explanation
	Reg.	Contents	Reg.	Contents	
SETSCAN 0 (o length buffer check)	15	X'FFFFFFFC'			Valid return code for 0 length buffer.
SETSCAN (locate character string; MOVE= RETURN)			None or (15)	Address of last character in string	Good return.
			None or (15)	X'00000000'	Specified character string not found in this buffer.
			None or (15)	X'FFFFFFFC'	Scan pointer beyond end of buffer
	15	X'00000000'	(2-11)	Address of last character in string	Good return.
	15	X'00000004'	(2-11)	Unchanged	Specified character string not found in this buffer.
	15	X'FFFFFFFC'	(2-11)	Unchanged	Scan pointer beyond end of buffer.
SETSCAN (Move scan pointer to char string; MOVE=KEEP)			None or (15)	X'00000000'	Good return.
			(2-11)	X'00000000'	Good return.
SETSCAN (skip *n* chars)			None or (15)	Address of character skipped to	Good return.
			None or (15)	X'00000000'	Skip goes past last character in this buffer.
	15	X'00000000'	(2-11)	Address of character skipped to	Good return.
	15	X'00000004'	(2-11)	Unchanged	Skip goes past last character in this buffer.
SETSCAN (skip backward)	15	X'00000000'			Good return.
	15	X'00000004'			Skip carries backwards past beginning of buffer.
SETSCAN (find scan pointer address)	15	Address of scan pointer			Good return.
	15	X'FFFFFFFC'			Scan pointer beyond end of buffer.

Figure 2-2 (Part 1 of 2). SETSCAN Return Codes

Macro Operand	Return Code		User Register		Explanation
	Reg.	Contents	Reg.	Contents	
SETSCAN SKIPFLD=FIRST or LAST					
MOVE=KEEP	15	X'00000000'			Good return.
	15	X'FFFFFFFC'			Buffer being processed is not a header buffer (SKIPFLD=FIRST) or buffer being processed is not the last buffer (SKIPFLD=LAST).
MOVE=RETURN RESULT=15	15	Address of last character in buffer			Good return.
	15	X'FFFFFFFC'			Buffer being processed is not a header buffer.
MOVE=RETURN RESULT=USEREG	15	X'00000000'	(2-11)	Address of last character in buffer	Good return.
	15	X'FFFFFFFC'	(2-11)	Unchanged	Buffer being processed is not a header buffer.
SETSCAN SKIPFLD=SOH or DEL					
MOVE=KEEP	15	X'00000000'			Good return.
	15	X'00000008'			Buffer being processed has no fixed header prefix.
	15	X'FFFFFFFC'			Buffer being processed is not the last buffer.
	15	X'FFFFFFF8'			Buffer being processed has a fixed header prefix but no text.
MOVE=RETURN RESULT=15	15	Address of last character in buffer			Good return.
	15	X'00000008'			Buffer being processed has no fixed header prefix.
	15	X'FFFFFFFC'			Buffer being processed is not the last buffer.
	15	X'FFFFFFF8'			Buffer being processed has a fixed header prefix but no text.
MOVE=RETURN RESULT=USEREG	15	X'00000000'	(2-11)	Address of last character in buffer	Good return.
	15	X'00000008'	(2-11)	Unchanged	Buffer being processed has no fixed header prefix.
	15	X'FFFFFFFC'	(2-11)	Unchanged	Buffer being processed is not the last buffer.
	15	X'FFFFFFF8'	(2-11)	Unchanged	Buffer being processed has a fixed header prefix but no text.
SETSCAN SKIPFLD=integer					
			None	Address of character skipped to	Good return.
	15	X'FFFFFFFC'	None	X'00000000'	Skip goes past last character in this buffer
	15	X'00000000'	(2-11)	Address of character skipped to	Good return.
	15	X'FFFFFFFC'	(2-11)	Unchanged	Skip goes past last character in this buffer.

Figure 2-2 (Part 2 of 2). SETSCAN Return Codes

SPECACT Macro

The SPECACT macro:

- Permits the user to signal system-wide conditions requiring action
- Sets or tests one of up to 24 special action codes in the MCP
- Is permitted in user-written closed subroutines.

Supported Resources and General Requirements: ALL.

Valid subgroups: Inheader, inbuffer, outheader, and outbuffer.

NAME	OPERATION	OPERANDS
[symbol]	SPECACT	ACTION={ACCTING } {codevalue} [,BRANCH=name] [,SETTING={ON }] {OFF}

```
ACTION={ACCTING  }
       {codevalue}
```

Function: Specifies the special action code to be tested or set.

Format: *codevalue* or ACCTING. *codevalue* should be either a decimal integer from 1 to 24 or a symbol that has been previously equated to a decimal number from 1 to 24. It is your responsibility to assign to this operand a value that falls within the prescribed limits.

Default: None. Specification is required.

Note: ACCTING specifies the "accounting table half-full" special action code.

```
BRANCH=name
```

Function: Specifies a test of a special action code condition and provides SPECACT with a name to which to branch if the condition is true.

Format: *name* must conform to the rules for assembler language symbols.

Default: None. If BRANCH= is not specified, the macro turns the codes on and off.

Note: *name* is the name of an addressable instruction in the current control section.

```
SETTING={ON }
        {OFF}
```

Function: Specifies whether the special action code is to be set on or off; if the BRANCH= operand is coded, specifies that a test for the action code being on or off is requested.

Format: ON or OFF.

Default: SETTING=ON.

ON specifies that the code is to be turned on or tested for being on.

OFF specifies that the code is to be turned off or tested for being off.

Examples

1. Set special action code 18:

   ```
   SPECACT ACTION=18,SETTING=ON.
   ```

2. Turn off special action code 18:

   ```
   SPECACT ACTION=18,SETTING=OFF.
   ```

3. See if special action code 12 is on; if it is, branch to ACTION12:

   ```
   SPECACT ACTION=12,SETTING=ON,BRANCH=ACTION12.
   ```

4. See if special action code 12 is off; if it is off, branch to NOACT12:

   ```
   SPECACT ACTION=12,SETTING=OFF,BRANCH=NOACT12.
   ```

5. Same as previous example if using equated value: ACT12 EQU 12

   ```
   SPECACT ACTION=ACT12,SETTING=OFF,BRANCH=NOACT12.
   ```

In the inmessage and outmessage subgroups, the setting of a special action code may be tested by using the TESTBR macro.

Return Codes

None.

STARTMH Macro

The STARTMH macro:

- Establishes addressability for an MH
- Directs messages to an incoming or outgoing group as appropriate
- Is required as the first macro in every MH.

STARTMH is required and must be the first macro instruction of every MH.

Supported Resources and General Requirements: ALL.

Valid subgroup: Not applicable.

NAME	OPERATION	OPERANDS

mhname	STARTMH	`[BEXIT=(name1,name2)]`
		`[,BREG={integer}]`
		`         {1      }`
		`[,DFC={FULL   }]`
		`       {PARTIAL}`
		`       {NONE   }`
		`[,LU={YES}]`
		`      {NO }`

`BEXIT=(name1,name2)`

Function: Specifies the name of the user Bind Session and Unbind Session exit routines. For addition information, see the topic "BIND User Exits" in the *TCAM Installation Guide.*

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Note: *name1* specifies the name of the user Bind Session exit routine; *name2* specifies the name of the Unbind Session exit routine.

The STARTMH macro generates virtual address constants (VCONS) for *name1* and *name2.* Therefore, these routines must have *entry* statements if they are assembled as part of the MCP.

If the exit routines are assembled as part of the MCP, then the LOCOPT macro (STATION= operand) and IEDSHOW macro (TSTATUS,STATION= operand) are valid in these exit routines to obtain the contents of various TCAM control blocks. (For detailed information on these macros, see the macro description in this chapter.)

BEXIT= is permitted on any STARTMH macro but will only get control for SNA sessions.

TCAM does not automatically save and restore the registers for this routine. The user must restore any registers used.

```
BREG={integer}
     {1      }
```

Function: Specifies the number of base registers desired for this MH.

Format: An unframed decimal integer between 1 and 11.

Default: BREG=1.

Maximum: 11.

Note: One base register is required for each 4096 bytes in the MH. Assignment begins with register 12 and works downward. If BREG=3 is coded, for instance, registers 12, 11, and 10 are assigned as the base registers for the first three 4096-byte blocks of this message handler.

```
DFC={FULL   }
    {PARTIAL}
    {NONE   }
```

Function: Specifies which SNA DFC commands will be processed by INHDR and INBUF subgroups of a device message handler. SNA DFC command in DMH

Format: FULL, PARTIAL, or NONE.

Default: DFC=PARTIAL.

Note: FULL specifies that all TCAM-supported DFC commands are to be processed by the DMH. All other commands are responded to, by TCAM, with a "function not supported" sense code. PARTIAL specifies that Logical Unit Status Ready-to-Receive and Signal commands are to be passed to the DMH. NONE specifies that no DFC commands are to be handled in the DMH. See the *TCAM Installation Guide* for an explanation of the DFC commands supported when PARTIAL or FULL are coded.

```
LU={YES}
   {NO }
```

Function: Specifies whether this message handler is to be for a host LU.

Format: YES or NO.

Default: LU=NO.

mhname

Function: Specifies the name of the macro and of the message handler.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. This name is required.

Note: Must be the same as *mhname* specified in the MH= operand of the PCB macro (for the application program that uses this message handler) or GROUP macro (for the external LUs that use this message handler).

Return Codes

None.

TASKDEF Macro

The TASKDEF macro:

- Is used to define the MCP and system service programs
- Defines an entry in the TCAM subtask table (TST)
- Must be coded in one module, together with all other TASKDEF macros defining entries in the same TST.

The TASKDEF macro instruction is the only source statement which should be used in a TST source module, other than the END statement. TASKDEF macro instructions should be coded in the order in which TST entries are to occur.

Four control sections are produced in the assembly of a TST:

- DKJATT: The TST header and fixed-length subtask entries
- DKJATTA: The abend code lists (refer to the DUMP= operand)
- DKJATTP: The PARM character strings (refer to the PARM= operand)
- DKTATTS: The shared subpool lists (refer to SHRDSP= operand).

Supported Resources and General Requirements: ALL.

Valid subgroup: Not applicable.

NAME	OPERATION	OPERANDS
[taskname]	TASKDEF	PGM=pgmname
		[,AUTO={NO }] {YES} {MCP}
		[,BACKUP=taskname]
		[,DSECT={NO }] {YES}
		[,DUMP=([{NO }[,codes,...])] {YES}
		[,MCP={NO } {YES}
		[,MCPREL={YES} {NO }
		[,NOTIFY={ALL }] {ERROR} {NONE }
		[,OPCTL={NO }] {YES}
		[,PARM='parameter list']
		[,PARMSTD={YES}] {NO }
		[,PRIORTY=nn] 6
		[,RESTART={0 }] {n } {ALWAYS}
		[,SHRDSP=(subpool,...)]
		[,ZPROMPT={NO }] {YES}

```
AUTO={NO }
     {YES}
     {MCP}
```

Function: Specifies whether this subtask is to be started automatically by the initiator.

Format: YES, NO, or MCP.

Default: AUTO=NO.

Note: If YES is specified, the initiator issues the ATTACH macro instruction for this subtask after the MCP and operator control system service programs have been started.

The AUTO=parameter is not meaningful on a TASKDEF macro specifying MCP=YES.

If AUTO=MCP is specified, the subtask is attached during MCP initialization rather than after READY processing is complete. AUTO=MCP implies MCPREL=YES. Both operands must be specified for TCAM basic operator control. Specify AUTO=MCP *only* for the basic operator control system service program.

The following combinations of operands are invalid:

- AUTO=MCP with MCPREL=NO
- AUTO=MCP with MCP=YES
- AUTO=MCP with PARM='parm'.

```
BACKUP=taskname
```

Function: Specifies the name of a TASKDEF macro defining another subtask (which may be another version of the same program) to be started by the initiator in the event that this subtask terminates abnormally.

Format: taskname must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Note: If RESTART is also specified, this operand takes effect only after the restart count has been exhausted.

This operand may not be used to specify a backup version of the basic operator control system service program.

```
DSECT={NO }
      {YES}
```

Function: Specifies whether the dummy section describing the TCAM subtask table (TST) is to be generated.

Format: NO or YES.

Default: DSECT=NO.

Note: If YES is specified, all other operands are ignored and the dummy section describing the format of the TST is generated. The dummy section is generated once for the TST assembly. If DSECT=YES is specified, it must be specified on the first TASKDEF macro. If specified on other than the first TASKDEF macro, the operand is ignored.

```
DUMP=([{NO }] [,codes,...])]
       {YES}
```

Function: Specifies that a dump is desired in the case of abnormal termination and the abend codes for which a dump is to be provided or, alternatively, the abend codes for which a dump is not to be provided.

Format: All parameters of the operand are separated by commas and enclosed in parentheses.

Default: DUMP=NO.

Note: You must code the DUMP operand if you wish for the initiator to provide you with any abnormal termination dumps.

Although each parameter on the DUMP= operand has a default value, you must code at least one of the parameters in order to obtain any abnormal termination dumps. If you do not supply an abend code list, the dump type condition specified applies to all abend codes.

For more details, refer to the chapter titled "Obtaining and Using Dumps" in the *TCAM Diagnosis Guide.*

If YES is specified, dumps are to be provided for all abend conditions. If NO is specified, dumps are not to be provided for any abend.

The condition may be followed by a list of abend codes. Abend codes are specified in the format Sxxx or Uyyy, where *xxx* and *yyy* are system and user abend codes, respectively. System and user abend codes can be mixed in any manner within the list. If abend codes are specified following a condition YES, dumps are provided for the specified codes only. If abend codes are specified following a condition NO, abend dumps are provided for all but the specified abend conditions.

```
MCP={NO }
    {YES}
```

Function: Specifies whether this subtask is an MCP.

Format: NO or YES.

Default: MCP=NO.

Note: MCP=YES and OPCTL=YES are mutually exclusive operands, as are MCP=YES and AUTO=MCP.

```
MCPREL={YES}
       {NO }
```

Function: Specifies whether a subtask that does not use the TCAM application program interface is to be brought down automatically if the MCP ends.

Format: NO or YES.

Default: MCPREL=NO.

Note: MCPREL=YES indicates that the subtask is to be brought down automatically if the MCP ends. AUTO=MCP implies MCPREL=YES. The combination of MCPREL=YES with AUTO=NO indicates that the subtask is to be reattached only if MCP termination was abnormal.

The following combinations of operands are invalid:

- MCPREL = NO with AUTO = MCP
- MCPREL = YES with MCP = YES.

Subtasks which use the TCAM API are brought down automatically when the MCP ends; code MCPREL = NO for such subtasks.

```
NOTIFY={ALL  }
       {ERROR}
       {NONE }
```

Function: Specifies which status messages are to be supplied for this subtask.

Format: ALL, ERROR, or NONE.

Default: NOTIFY = ALL.

Note: ALL specifies that all status messages are to be issued for this subtask.

ERROR specifies that only error messages are to be displayed for this subtask.

NONE specifies that no status messages are to be displayed for this subtask.

```
OPCTL={NO }
      {YES}
```

Function: Specifies whether this subtask is a basic or extended operator control system service program.

Format: NO or YES.

Default: OPCTL = NO.

Note: MCP = YES and OPCTL = YES are mutually exclusive operands.

```
PARM='parameter string'
```

Function: Specifies the character string to be passed as a parameter list to the subtask.

Format: 'parameter string' must conform to the rules for the PARM field in the EXEC JCL statement.

Default: None. Specification is optional.

Note: On entry to the subtask, register 1 points to a fullword containing the address of the PARM string. If the subtask is the MCP, TCAM moves the contents of register 1 into the TVT field TVTXPARM. The format of the PARM string is a halfword field containing the length of the character string followed by the actual character string.

If the PARM= operand is not coded on the TASKDEF macro defining the subtask, register 1 contains zeros when the subtask is attached.

PARM=WTO is valid for IEDQFW only.

```
PARMSTD={YES}
        {NO }
```

Function: Specifies the parameter entry linkage to be used when an initiator subtask is attached.

Format: NO or YES.

Default: PARMSTD=YES, if the PARM operand is coded.
PARMSTD=NO, if the PARM operand is not coded.

Note: PARMSTD=NO is only valid when the PARM operand is not coded. Register 1 contains 0 on entry to the attached subtask.

If PARMSTD=YES is coded, standard linkage is used with register 1 containing the address of a fullword which contains the address of the parameter string consisting of a halfword length and the data.

```
PGM=pgmname
```

Function: Specifies the name of the load module for the program being defined in this macro as an initiator subtask.

Format: pgmname must conform to the rules for program names.

Default: None. Specification is required.

```
PRIORTY=nn
```

Function: Specifies the relative dispatching priority to be assigned this subtask.

Format: nn is a numeric value from 1 to 15.

Default: PRIORTY=14 for MCP=YES; PRIORTY=13 for OPCTL=YES. For all others, PRIORTY=6 (medium priority).

Note: Allowable values that a user can code are from 1 to the 12. Note that these are not absolute priorities, but rather guidelines for the initiator to determine relative priorities among its subtasks. PRIORTY=15 for the MCP and PRIORTY=13 for basic operator control are automatically assigned. If you include COMWRITE and extended operator control, these two optional initiator subtasks are automatically assigned PRIORTY=15 and PRIORTY=13, respectively.

If MCP=YES or OPCTL=YES is coded, the PRIORTY= operand is ignored if specified.

```
RESTART={0      }
        {n      }
        {ALWAYS}
```

Function: Specifies the number of times this subtask is to be restarted by the initiator in the event of consecutive abends. A value of ALWAYS implies continuous restart of the subtask, no matter how many consecutive abends occur. If a program has exhausted its restart count, it may still be restarted via a command, in which case, the restart count is "refreshed" (that is, it is reset to the original value specified by this operand).

Format: 0, *n*, or ALWAYS. *n* must be a numeric value less than or equal to 15.

Default: RESTART=0 (implying no restart).

Notes: *n* is a numerical value from 0 to 15 indicating the number of times (up to 14) that restart of the program is to be attempted after consecutive abnormal terminations. A value of 15 is equivalent to specification of RESTART=ALWAYS.

RESTART=ALWAYS implies that restart of the program is to be attempted after every abnormal termination.

When RESTART=n is coded or when an operator detaches a subtask, the restart count is not decremented for tasks abended due to MCP failures (for example, an application 046 abend). See the *TCAM Operation* manual for more information.

```
SHRDSP=(subpool,...)
```

Function: Specifies the main storage subpools a subtask will share with the TCAM initiator.

Format: Unsigned decimal values 1 to 127. The MCP task and all non-TCAM system tasks will be allowed to share subpools 1-98 and 100-127. Only the MCP task and SYSOPCTL task will be allowed to share subpool 99. Subpool 0 and any subpool greater than 127 are invalid. If more than one subpool is specified, the subpools are separated by commas and the list enclosed in parentheses.

Default: Subpool 99 is the default for the MCP task and the SYSOPCTL task. For any other task, there is no default and specification is optional.

Note: The MCP task and all non-TCAM system tasks are allowed to specify shared subpools. The maximum number of subpools that can be specified is 16. The MCP task and SYSOPCTL task are automatically given subpool 99 as a subpool to share, therefore the MCP task is only allowed to specify a maximum of 15 subpools. All of the other non-TCAM system tasks can specify a maximum of 16 subpools.

```
taskname
```

Function: Specifies the name used in initiator commands and replies that refer to a subtask.

Format: taskname must conform to rules for assembler language names (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Note: If *taskname* is not coded, the initiator subtask specified on the PGM= operand is used.

```
ZPROMPT={NO }
        {YES}
```

Function: Specifies whether message DKJ607D appears at the operator's console after an MCP CLOSE or HALT (Z TP) command successfully completes for an MCP entry in the task definition table.

Format: NO or YES.

Default: ZPROMPT=YES.

Note: If YES is specified, DKJ607D is issued after an MCP CLOSE or HALT has successfully completed.

If NO is specified, DKJ607D is not issued and the TCAM initiator terminates after the successful completion of an MCP CLOSE or HALT command.

ZPROMPT= is valid only when the TASKDEF operand MCP=YES is specified.

Examples

In addition to these examples, see also the default TCAM subtask table in the *TCAM Installation Guide.*

1. A program PROG1, to be known to the initiator as subtask ABC, is to be automatically started by the initiator. A dump is to be provided for system abend code 043, and, in the event of failure, the program is to be restarted five times:

```
ABC     TASKDEF  PGM=PROG1,AUTO=YES,RESTART=5,                     X
                 PARM='parameters for prog1',DUMP=(,S043)
```

2. Another program, PROG2, to be known as subtask DEF, is to be started on command. A is to be provided in the case of any abend conditions other than system 046:

```
DEF     TASKDEF  PGM=PROG2,AUTO=NO,                                X
                 PARM='PARM1=13,PARM2=(X,44)',DUMP=(NO,S046)
```

3. From time to time, the user wishes to start PROG2 with a different set of parameters, with a dump to be provided in the case of any abend condition other than system 046 or user 5. Establishing a second subtask definition for PROG2 will accomplish this:

```
DEF2    TASKDEF  PGM=PROG2,AUTO=NO,                                X
                 PARM='PARM1=15,PARM2=(Y,0)',                      X
                 DUMP=(NO,S046,U0005)
```

4. An MCP, NETMCP, is to be known by the taskname MCP1. Restart is to be performed twice, and a dump is desired if the MCP terminates abnormally with system 045 abend code:

```
MCP1    TASKDEF  PGM=NETMCP,MCP=YES,RESTART=2,                     X
                 DUMP=(YES,S045)
```

5. A new version of NETMCP, called NETMCP2, is being tested. A dump is to be taken for any abend conditions other than system 045, after which the MCP is to be restarted once; if it abends a second (consecutive) time, we want to start the previous version:

```
MCP2    TASKDEF  PGM=NETMCP2,MCP=YES,RESTART=2,                    X
                 DUMP=(NO,S045),BACKUP=MCP1
```

Return Codes

None.

TCGEN Macro

The TCGEN macro:

- Generates two TPROCESS macros for each transmission category
- Must be coded in the terminal table before any HOSTDEF macros
- Is used in defining an extended network.

The TPROCESS macros generated by TCGEN are needed so that the MCP in an extended networking environment can set up SNA utility sessions with the other host nodes in the network, in order to facilitate host-to-host communication.

Supported Resources and General Requirements: SNA.

Valid subgroup: Not applicable.

NAME	OPERATION	OPERANDS
[symbol]	TCGEN	NUMBER=n

`NUMBER=n`

Function: Specifies the number of transmission categories defined in the local host node.

Format: n is an unframed decimal integer from 1 to 16, the maximum number of transmission categories that may be defined in an MCP.

Default: None. This operand is required.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Return Codes

None.

TCPATH Macro

The TCPATH macro:

- Generates two TERMINAL macros representing LUs in another TCAM system for a transmission category
- Must be coded after the corresponding HOSTDEF macro
- Can generate a GROUP macro for the transmission category
- Is used in defining an extended network.

The TERMINAL macros generated by TCPATH are needed for each of the other TCAM host nodes so that the SNA utility sessions with each such node can be set up to facilitate internodal communication. One TCPATH macro should be coded for each transmission category that will be used to communicate directly (that is, without going through an intermediate host) with the other TCAM host node represented by the preceding HOSTDEF macro.

Supported Resources and General Requirements: SNA.

Valid subgroup: Not applicable.

NAME	OPERATION	OPERANDS
[symbol]	TCPATH	OPDATA=(data,...) ,QUEUES=form ,TC=n [,BUFSIZE=value] [,CLOCK=time] [,LEVEL={(integer,...)}] {(0) }]

BUFSIZE=value

Function: Specifies the output buffer size (in bytes) for the GROUP macro generated by this macro.

Format: See GROUP macro.

Default: None. Specification is optional.

CLOCK=time

Function: This operand is identical to the operand of the same name on the TERMINAL macro. Refer to the description of the TERMINAL macro for details.

Format: Two decimal integers for the hours, immediately followed by two decimal integers for the minutes. Framing characters must not be specified.

Default: None. Specification is optional.

```
LEVEL={(integer,...)}
      {(0)         }
```

Function: This operand is identical to the operand of the same name on the TERMINAL macro. Refer to the description of the TERMINAL macro for details.

Format: Unframed decimal integer.

Default: LEVEL=(0).

```
OPDATA=(data,...)
```

Function: This operand is identical to the operand of the same name on the TERMINAL macro. Refer to the description of the TERMINAL macro for details. When coded on the TCPATH macro, the data fields of this operand must be assigned certain values.

Format: The maximum length and type of data specified for each option field must correspond to the OPTION macro that defines the field.

Default: None. This operand is required.

```
QUEUES=form
```

Function: This operand is identical to the operand of the same name on the TERMINAL macro. Refer to the description of the TERMINAL macro for details.

Format: DR, DN, MO, MN, or MR.

Default: None. This operand is required.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

```
TC=n
```

Function: Specifies the transmission category for which the two TERMINAL macros are being generated.

Format: *n* is the decimal integer from 1 to 16.

Default: None. This operand is required.

Return Codes

None.

TCSEARCH Macro

The TCSEARCH macro:

- May be used to gather information about the current external session partner for use in MH processing (If issued in the inheader subgroup, the current external session partner is the message origin; if issued in the outheader subgroup, the current external session partner is the message destination.)
- May be used to locate various TCAM control blocks and table entries.

This section describes only one version of the TCSEARCH macro that is used to gather information about the current external session partner and put it in the IEDXOAF answer area associated with each TCAM option field. A complete description of the TCSEARCH macro may be found in Appendix C of this manual. The format of the IEDXOAF area is described in the description of the OPTION macro.

Supported Resources and General Requirements: ALL.

Valid subgroups: Inheader, inbuffer, outheader, and outbuffer.

NAME	OPERATION	OPERANDS:
[symbol]	TCSEARCH	ARG=CURRTERM [,ERROR=name]

ARG=CURRTERM

Function: This form of the TCSEARCH macro is used to provide consolidated terminal table entry and option field data about the current external LU or application program. The identity of the current external LU or application depends on the MH subgroup in which TCSEARCH is issued: in inheader or inbuffer, it is the message originator, and in outheader or outbuffer subgroups it is the message destination. The object of the search is a terminal name entry. The ARG=CURRTERM form of TCSEARCH places the gathered information about the current session partner into the answer area called IEDXOAF that is built by the user-coded OPTION macros to define option fields. Information returned from TCSEARCH may be used by user code or as input to other macros (a FORWARD macro having the TTCIN= operand, for example, may route on the TNT offset returned by TCSEARCH).

Format: CURRTERM.

Default: None. This operand is required.

Note: The functional description of the OPTION macro discusses the IEDXOAF control section used as the answer area in this format of TCSEARCH, and also provides coding examples for accessing the option field information returned in the answer area. The answer area is not filled unless the session partner is an external LU.

ERROR=name

Function: This branch is taken if an error occurred during the execution of the TCSEARCH macro.

Format: *name* must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

symbol

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Return Codes

See the description of the TCSEARCH macro in Appendix C of this manual.

TCSENDBL Macro

The TCSENDBL macro:

- Inserts previously scheduled fields into the first unit of the message header
- May optionally remove the FHP from a message before passing it to a destination external LU or application program
- Is not normally coded in an outheader subgroup of an internodal message handler since its functions are performed internally by the INODEMH macro (GEN=OUTHDR format)
- Requires the presence of an FHP in the message header
- May be executed only once in any path of the subgroup.

If the TCSENDBL macro is issued more than once in the outheader subgroup, it is your responsibility to supply the necessary routing logic. The TCSENDBL macro must be executed only once for each message.

The TCSENDBL macro inserts output sequence number, date, and time information into the message header, overlaying the fixed header prefix area as necessary. Any of the following fields may be scheduled by the INSRTFLD macro for insertion by the TCSENDBL macro: output sequence number, input date, input time, output date, and output time.

If previous SENDMSG macro processing has designated this message as a logical error message, TCSENDBL adds 1 to the ERRCOUNT option field for the origin of the message.

If FHP=PASS is coded, the FHP is passed along with the message to the destination. This must be done in an MH for an application program that processes the FHP with the message. If an application program does not process the FHP and does not need the insertions of output sequence number, date, and time in the message header, omit the TCSENDBL macro in the AMH.

The TCSENDBL macro may be issued in an MH which processes some messages that contain FHPs and other messages that do not; if a message does not contain an FHP, TCSENDBL does not alter the message contents but returns control to the next sequential instruction with a non-zero return code.

Supported Resources and General Requirements: ALL, FHP.

Valid subgroups: Outheader

NAME	OPERATION	OPERANDS
[symbol]	TCSENDBL	[FHP={PASS }] {<u>REMOVE</u>}

```
FHP={PASS  }
    {REMOVE}
```

Function: Specifies whether the FHP is to be passed to an application program or is to be removed, as usual, when a message is transmitted.

Format: PASS or REMOVE.

Default: FHP=REMOVE.

Note: If this operand is coded FHP=PASS, the other TCSENDBL functions are not applicable.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Return Codes

One of the following return codes is set in register 15.

Code	**Meaning**
X'00000000'	Successful execution.
X'00000004'	There is no data in the buffer.
X'00000008'	The message is a TCAM message starting with IED.
X'0000000C'	Is not a first buffer of a message
X'00000010'	There is no FHP in the message buffer.
X'00000014'	An insertion offset, as found in the FHP, is invalid.
X'00000018'	The field(s) to be inserted did not fit into the available space.
X'0000001C'	Invalid input date or time.

TCSUP Macro

The TCSUP macro:

- Builds start-of-day and restart messages based on the system operator's specification of the current startup or restart condition
- Provides a standard default facility (via the GEN=STANDARD keyword) to automatically generate macros defining a complete standard set of startup or restart conditions, LU types, and startup or restart basic message formats
- May be coded optionally once in a nonexecutable section of an MCP
- Must follow all user-coded UPCONDTN, UPMSG, and UPTYPE macros if any exist.

Supported Resources and General Requirements: ALL.

Valid subgroups: Not applicable.

NAME	OPERATION	OPERANDS

symbol	TCSUP	GEN={STANDARD}
		{TAILORED}

```
GEN={STANDARD}
    {TAILORED}
```

Function: Specifies whether you want the TCSUP macro to automatically generate the UPCONDTN, UPMSG, and UPTYPE macros necessary to provide the default standard startup/restart message generation facility.

Format: STANDARD or TAILORED.

Default: GEN=STANDARD.

Note: If GEN=STANDARD is specified or defaulted, you must not code any UPCONDTN, UPMSG, or UPTYPE macros; if GEN=TAILORED is specified, at least one of each of the UPCONDTN, UPMSG, and UPTYPE macros must be coded. If either of these conditions is violated, assembly error messages result.

If GEN=STANDARD is specified, you must specify the INMSGCT, OUTMSGCT, and ERRCOUNT option fields.

The macros generated by a TCSUP macro coded GEN=STANDARD are shown in the chapter of the *TCAM Utilities* manual titled "Startup/Restart Message Generation Service Facility." See the model MCPs for an example of a tailored message-generation facility.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Note: The TCSUP symbol must be the name specified in the READY macro for the GMMSG= and RSMSG= operands.

Return Codes

None.

TERMINAL Macro

The TERMINAL macro defines an external logical unit (LU).

The specifications supplied by the TERMINAL macro creates an entry that is included in the terminal table (discussed in "Defining Network Resources" in the *TCAM Installation Guide).*

The TERMINAL macro:

- Specifies the name of the group with which the external LU is to be associated
- Specifies the type of medium on which the queue is to be maintained
- Specifies the alternate destination to be used when a reusable disk queue is being reorganized
- Specifies the buffer size to be used when sending to this station
- Specifies the permissible priorities for messages sent to this external LU
- Specifies whether logical message definition is used
- Specifies option-field data
- Specifies whether an external LU is a basic secondary operator control station
- Specifies the available screen sizes for 3270 and 3290 display stations
- Specifies information relating to the establishment of SNA sessions for this external LU.

Supported Resources and General Requirements: SNA.

Valid subgroup: Not applicable.

NAME	OPERATION	OPERANDS
symbol	TERMINAL	QUEUES=form
		[,ACTIVE={YES}] {NO }
		[,ALTDEST=entry]
		[,BUFSIZE=integer]
		[,CINTVL=integer]
		[,CLOCK=time]
		[,GROUP=grpname]
		[,LEVEL={integer}] {0 }
		[,LMD={YES}] {NO }
		[,LUCAP={PRI}] {SEC}
		[,OPDATA=(data,...)]
		[,SCRSIZE={(h,w,ah,aw)}] {(24,80) }
		[,SECTERM={YES}] {NO }
		[,SPURGE={YES}] {NO }

		[,TCMSESN={NORMAL }]
		{LUINIT }
		{LUTERM }
		{AUTOINIT}

```
ACTIVE={YES}
       {NO }
```

Function: Specifies the initial status of the external LU.

Format: YES or NO.

Default: YES.

Note: YES specifies that the external LU is allowed to enter into LU-LU sessions with host LUs. An external LU that is allowed to enter sessions is said to be started; otherwise, the LU is stopped. The status of the external LU can be modified to started or stopped status by the basic operator commands MODIFY START or MODIFY STOP, respectively. Refer to the *TCAM Operations* manual for more information on these commands.

```
ALTDEST=entry
```

Function: Specifies the alternate destination to which a message reusable disk queue is sent at the time the zone containing the message is serviced for reuse.

Format: Must conform to the rules for assembler language symbols and specify the name of any single or process entry in the terminal table capable of accepting messages. Framing characters must not be specified.

Default: None. Specification is optional.

Note: See the *TCAM Installation Guide* for a description of servicing for reusability. When the destination queue defined by a TERMINAL macro is reorganized, unsent messages on the queue are placed on the destination queue specified by ALTDEST=. This can be the queue for the original destination or for *any* other valid destination.

If the ALTDEST= operand is omitted, messages on a reusable disk queue may be written over and lost to the system with no error indication being made. This operand is ignored unless either QUEUES=DR or QUEUES=MR is specified in this TERMINAL macro.

`BUFSIZE=integer`

Function: Specifies the buffer size for outgoing messages destined for this external LU and overrides the BUFSIZE= operand of the GROUP macro.

Format: Unframed decimal integer greater than or equal to 76.

Default: None. Specification is optional.

Maximum: 65,535.

Note: If this operand is omitted, the buffer size specified in the BUFSIZE= operand of the GROUP macro is used.

`CINTVL=integer`

Function: Specifies the period of time following which session initiation is performed, provided neither the LU nor TCAM initiated a session during this period.

Format: Unframed decimal integer.

Default: None. Specification is optional.

Maximum: 65,535 seconds.

Note: The CINTVL= and CLOCK= operands are mutually exclusive.

The initial interval starts following the initial start for the external LU, provided the LU is available to the system; that is, ACTIVE=YES and TCMSESN=AUTOINIT are coded on the TERMINAL macro for the LU.

If CINTVL=65,536 is coded, there is no periodic contact by the host processor. A contact is initiated by the host only when it has data to transmit.

`CLOCK=time`

Function: Specifies the time of day that TCAM should initiate a session with an LU.

Format: Two decimal digits for the hours, immediately followed by two decimal digits for the minutes. Framing characters must not be specified.

Default: None. Specification is optional.

Maximum: 2359 (23 for the hours, 59 for the minutes).

Note: CLOCK= must be omitted if CINTVL= is specified.

If CLOCK= and TCMSESN=AUTOINIT is specified for an LU, then the LU is not to be placed in session with a TCAM LU until the specified time of day is reached. However, an LU may initiate a session befor the time of day specified.

```
GROUP=grpname
```

Function: Specifies the name of the GROUP macro for the station including this external LU.

Format: Must be the same name as *grpname* on the GROUP macro.

Default: The name of the previous GROUP macro in the MCP.

Note: The GROUP= operand is not required except when a TERMINAL macro is coded before the first GROUP macro in the MCP.

```
LEVEL=({integer,...})
       {0           }
```

Function: Specifies the permissible transmission priority levels that may be used in the header of a message destined for this external LU.

Format: Unframed decimal integer.

Default: LEVEL=(0).

Maximum: 255.

Note: The levels must be in ascending order. For instance, if the messages being sent to a certain external LU using EBCDIC code can have priorities of 1, 3, or 5, the LEVEL= operand for this station is coded LEVEL=(241,243,245). Here, 241, 243, and 245 are the decimal equivalents of the hexadecimal EBCDIC representations of the numbers 1, 3, and 5.

While TCAM allows for 255 priority levels, the assembler has a limitation on the size of a single operand of a macro (255 characters). If three-digit levels, separated by commas, are used, the assembler character limit of 255 characters limits the TCAM priority levels to only 163.

For more information on message priority, see the chapter titled "Coding the Message Handler" in the *TCAM Installation Guide.*

```
LMD={YES}
    {NO }
```

Function: Specifies whether individual logical messages entered by this external LU may be included in multiple physical transmissions.

Format: YES or NO.

Default: LMD = NO.

Note: YES indicates that any individual logical message entered by this external LU is independant of the size of a SNA chain. If a logical message is larger than a SNA chain, the remainder of the message may be included in the external LU's next SNA chain. If a logical message is smaller than a SNA chain, other logical messages may be included with it in the SNA chain.

If this operand is omitted or if NO is specified, logical messages entered by this external LU must be entered in a single SNA chain. Logical messages are discussed in the chapter titled "Logical Messages" in the *TCAM Installation Guide*.

```
LUCAP={PRI}
      {SEC}
```

Function: Specifies the capability of this LU to run as a primary or secondary LU for sessions initiated by this host.

Format: PRI or SEC.

Default: LUCAP = SEC.

Note: PRI specifies that the LU is to run as the primary LU in an LU-LU session initiated by a TCAM LU in this TCAM system. SEC specifies that the LU is to run as a secondary LU in an LU-LU session initiated by a TCAM LU in this TCAM system.

LUCAP = PRI should be coded on the TERMINAL macro for the application program connected to another TCAM system if the application is capable of being the primary LU in a session with an LU under this TCAM system.

```
OPDATA=(data,...)
```

Function: Specifies the actual data to be inserted in the set of option fields assigned to the external LU (see the discussion of the OPTION macro) and also specifies which option fields are not to be created for this external LU.

Format: The maximum length and type of data specified for each option field must correspond to the length and type specified by the OPTION macro that defines the field, and the order in which the data for each field is specified must correspond to the order in which the OPTION macros are specified. Framing characters are not used.

Default: None. Specification is optional.

Certain option fields have reserved names and special uses in TCAM; see the chapter of this manual titled "Option Fields Reserved for Use by TCAM Functions" for details. Do not use these names for option fields unless the fields are to be used for the purposes described in that chapter.

When specifying option fields for a particular external LU, you may omit the last several option fields defined by OPTION macros by merely closing the parentheses after the data for the last field you wish to define. A comma is used:

- To delimit the data for each field.
- To indicate that no data is specified for the first or an intermediate field defined by an OPTION macro.
- To indicate that the OPDATA= operand is to be continued (if specified immediately preceding the right parenthesis).

You must specify either data and a comma, or a comma alone, for the first and each intermediate field defined by an OPTION macro (with one exception A comma alone is coded if a field other than the last is not to be assigned for this station. If the last field is not to be assigned, no data is coded for the field and the comma is also omitted. Framing characters are not coded.

Examples:

Because the operand field of a macro is limited to 255 characters, TCAM provides a facility to specify additional OPDATA= parameters if necessary. A comma placed as the last character of the OPDATA= operand; that is,

```
OPDATA=(data,data,..., data,)
```

indicates a continuation of the OPDATA= operand. The next source statement would then be coded:

```
symbol   TERMINAL OPDATA=(data,...)
```

where *symbol* is the name specified on the TERMINAL macro that specified the continuation.

There is no limit (other than the number of option fields defined) on the number of continuation statements that may be used. Device-dependent operands must be coded on the first source statement that has the OPDATA= operand.

Assume that four OPTION macros have been coded. If you want to specify all four fields for a particular external LU, you would code the OPDATA= operand of the TERMINAL macro:

```
,OPDATA=(field1,field2,field3,field4)
```

where *field1, field2, field3, field4* represent the actual initial data to be inserted into each of the four option fields. If only *field1* and *field4* are to be implemented for this external LU, you would code:

```
,OPDATA=(field1,,,field4)
```

If only *field1, field2,* and *field3* are to be implemented, you would code:

```
,OPDATA=(field1,field2,field3)
```

If only *field1* is to be implemented, you would code:

```
,OPDATA=(field1)
```

```
QUEUES=form
```

Function: Specifies where the message queues are to be maintained.

Format: DR, DN, MO, MN, or MR.

Default: None. Specification is required.

Note: For a discussion of this topic, see the chapter titled "Defining Data Sets" in the *TCAM Installation Guide.*

- DR specifies reusable disk queues.
- DN specifies nonreusable disk queues.
- MO specifies main-storage-only queues.
- MR specifies main-storage queues with reusable-disk backup.
- MN specifies main-storage queues with nonreusable-disk backup.

If the form of data set specified by this operand does not correspond to a related message queue data set specified by a DCB macro or by the MSUNITS= operand of the INTRO macro, the TCAM system terminates abnormally.

If MO is specified, the distribution list, multiple routing, and REDIRECT facilities should be used with care since one extra buffer is required to accommodate every destination other than the original destination. If anything other than MO is specified, DISK=YES must be specified (or defaulted to) on the INTRO macro.

If MO, MR, or MN is specified, the MSUNITS= operand of the INTRO macro must specify a nonzero integer.

```
SCRSIZE={(h,w,ah,aw)}
        {(24,80)    }
```

Function: Specifies the correct number of rows and characters per row for display station screens.

Format: Decimal integers from 1 to 255, enclosed in parentheses and separated by commas.

Default: SCRSIZE=(24,80). (There is no default for the *ah* and *aw* parameters.)

Maximum: 255 each.

Note: *h* specifies the number of rows and *w* specifies the number of characters per row for the primary screen size. *ah* specifies the number of rows and *aw* specifies the number of characters per row for the alternate screen size. When an alternate screen size is used, *h, w, ah,* and *aw* must be specified and *h* and *w* must be the smaller of the two screen sizes. For SNA displays, you still must code *h, w, ah,* and *aw.* if only the alternate large-screen size is to be used. For example if the desired screen size is 43x80, you specify SCRSIZE=(43,80,43,80).

Standard IBM screen sizes are 6x40, 12x40, 12x80, and 15x64 for the 2260; 12x40, 12x80, 24x132, 27x160, 32x80, 32x132, 43x80, 43x132 for the 3270; and 31x160, 50x106, 62x80, 62x132, 62x160 for the 3290.

The Bind command establishes primary and alternate screen sizes to be used during the current LU-LU session. By using the IEDLSCR macro in your Bind exit, you can cause the Bind command to specify the screen size values coded on the SCRSIZE= operand. Also, using the Bind command, you can permanently replace the values coded with the SCRSIZE= operand in the MCP.

See the *TCAM Installation Guide* for additional information about large-screen support.

```
SECTERM={YES}
        {NO }
```

Function: Specifies whether this external LU may be considered a basic secondary operator control station.

Format: YES or NO.

Default: SECTERM=NO.

Note: If YES is specified, operator commands are recognized, acted upon, and the appropriate response returned to the external LU. If an external LU other than the system console is to be the basic primary operator control station, SECTERM=YES must be specified for that external LU's TERMINAL macro.

```
SPURGE={YES}
       {NO }
```

Function: Specifies that the session purge option is to be used for this LU.

Format: YES or NO.

Default: NO.

SPURGE=YES may be coded only on terminal macros which do not have any of the following operands specified:

TCMSESN=AUTOINIT
TCMSESN=LUINIT
CLOCK=
CINTVL=

See the *TCAM Installation Guide* for a description of the possible uses for this operand.

```
symbol
```

Function: Specifies the name of the external LU.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. This name is required for all TERMINAL macros.

Note: his name can appear in an origin or destination field of a message header. The length of *symbol* cannot be greater than defined by the MAXLEN= operand of the TTABLE macro.

Care should be taken while following the naming conventions for the TERMINAL macro. For example, do not use a name on the TERMINAL macro if that name is used elsewhere as a DD name in your JCL.

SYSCON cannot be used as a name.

```
TCMSESN={NORMAL  }
        {LUINIT  }
        {LUTERM  }
        {AUTOINIT}
```

Function: Specifies how the TCAM will initiate and terminate LU-LU sessions with the external LU.

Format: NORMAL, LUINIT, LUTERM, or AUTOINIT.

Default: NORMAL

Note: NORMAL specifies that either TCAM or the external LU may initiate a session for the external LU with a host LU. TCAM will terminate a TCAM-initiated session if:

1. there are no more messages to send from the destination queue or,
2. the queue is held and SESSION=KEEP was not specified on the HOLD macro that was executed to intercept the queue.

LUINIT specifies that only the external LU can initiate a session with a host LU. This specification implies a session termination rule of LUTERM.

LUTERM specifies that either TCAM or the external LU may initiate a session for the external LU but that the external LU terminates the session. There are, however, certain situations under which TCAM will terminate a session regardless of how the TCMSESN operand is coded. For example, an outage in the VTAM network may cause session termination. Refer to the *TCAM Installation Guide* for additional information.

AUTOINIT specifies that TCAM should automatically initiate a session with the external LU. This specification implies a session termination rule of LUTERM. TCAM will attempt the session initiation whenever one of the following occurs and a session does not already exist for the external LU:

- A message is enqueued on the destination queue for the external LU.
- The external LU is already started (see ACTIVE= operand) and a MODIFY START=AMI basic operator control command causes the default host LU for the external LU to be started.

  Note: The default host LU for the external LU is defined by the IEDLUNAM option field. If there is no valid IEDLUNAM option data for the LU, then PROGID is the default host LU.
- The default host LU is already started and a MODIFY START basic operator control command is issued for the external LU.

The automatic session initiation attempt by TCAM will not be successful if the external LU is not already activated by VTAM.

The CLOCK= and CINTVL= operands are valid for NORMAL, LUTERM, and AUTOINIT specifications and invalid for LUINIT.

Return Codes

None.

TERRSET Macro

The TERRSET macro:

- Enables you to set or reset selected bits in the message error record
- Is often issued before a related ERRORMSG macro.

The TERRSET macro enables you to alter the message error record for this message by coding a 5-byte mask to set off specified bits (error flags). This is particularly useful when, for example, certain errors do not result in the loss of significant data. In such cases, rather than generating an error reject message, you may attempt to transfer and process a partial message - possibly editing a warning notation into the message text.

Supported Resources and General Requirements: ALL.

Valid subgroups: Inblock, inbuffer, inheader, outbuffer, and outheader.

NAME	OPERATION	OPERANDS

[symbol]	TERRSET	[BIT={ON }]
		{OFF}
		[,MASK={hexmask }]
		{X'0000080000'}

```
BIT={ON }
    {OFF}
```

Function: Specifies whether the message error record bits specified in the MASK= operand are to be set on or off.

Format: ON or OFF.

Default: BIT=ON.

```
MASK={hexmask       }
     {X'0000080000'}
```

Function: Provides a five-byte mask in hexadecimal format indicating which message error record bits the user wants turned on (or off, if BIT=OFF is specified). If a bit is set to 1 in the mask, the corresponding bit in the MER is turned either off or on, depending on how the BIT= operand is coded.

Format: A five-byte hexadecimal mask framed by X'' characters.

Default: MASK=X'0000080000'.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

None.

TESTBR Macro

The TESTBR macro:

- Permits dynamic variation of the message path of a message through a subgroup
- May be specified more than once within a subgroup.

TESTBR provides the facility for executing any macro within an inmessage or outmessage subgroup on a conditional basis. It expands the function of the mask operand of the different error-handling macros so that execution can be dependent upon (in addition to being dependent upon the contents of the message error record) the contents of an option field with the message origin or destination. Because TESTBR offers the ability to deviate from the sequential execution of all macros, you are able to create your own logical flow through the subgroup. The single restriction is that the instruction to which a branch is taken must be a TESTBR macro with no operands.

Depending upon the use of the operands, the macro may do one of the following:

- Do nothing but serve as a name for the next sequential instruction if a branch to it from somewhere else in the subgroup is called for
- Cause an unconditional branch to another macro within the subgroup
- Cause a similar branch if a specified condition is not filled.

The condition may be:

- The status of any bit or bits in the message error record
- The status of any bit or bits in 1 byte within any specified option field for the message origin or destination
- A special action code being currently in the off state
- A counter not having reached a certain threshold limit value.

Supported Resources and General Requirements: ALL.

Valid subgroups: Inmessage and outmessage.

NAME	OPERATION	OPERANDS
[symbol]	TESTBR	BRANCH=name
		[,CONNECT={AND}]
		{OR }
		[,{MASK=X'mertest' }]
		{OPTION=name }
		{SPECACT={action }}
		{ACCTING}
		[,MASK=X'statest']
		[,OFFSET={integer}]
		{label }
		{0 }
		[,THRESH={integer}]
		{label }
		[,TOFFSET={integer}]
		{label }

```
BRANCH=name
```

Function: Specifies the name of the macro to execute next if a branch is to be taken.

Format: *name* must conform to the rules for assembler language symbols.

Default: None. Specification is required unless *symbol* is specified; in which case, it is not allowed.

Note: *name* must be the name of a TESTBR macro instruction in the same subgroup. The next instruction to be executed must always be a TESTBR macro with a *symbol* value and no operands.

If only the BRANCH = *name* operand is specified, the branch is taken unconditionally.

```
CONNECT={AND}
        {OR }
```

Function: Specifies the type of logical connection to be made between the mask and the error record.

Format: AND or OR.

Default: CONNECT = OR.

Note: If AND is specified, a logical AND operation is performed to determine if the BRANCH = operand will be executed next. The next sequential instruction is executed only if *all* bits on in the mask are on in the message error record or option field; otherwise, a branch is taken to the location specified by BRANCH =. In assembler language terms, this is equivalent to an AND instruction followed by a BNO instruction.

If OR is specified, a logical OR operation is performed to determine if the BRANCH = operand will be executed next. The next sequential instruction is executed if *any* bit on in the mask is on in the message error record or option field; otherwise, a branch is taken to the location specified by BRANCH =. In assembler language terms, this is equivalent to an OR instruction followed by a BZ instruction.

If however, the mask operand is specified as all hexadecimal zeros, the branch is *never* taken regardless of the CONNECT = operand specification or the field being tested.

```
MASK=X'mertest'
```

Function: Specifies the 5-byte bit configuration used to test the message error record for the message.

Format: *mertest* is a hexadecimal field, 5 bytes in length, framed by X'', for a total of 13 coded characters.

Default: None. Specification is optional.

Note: If specified bits in the message error record are on, no branch is taken. (Whether all or only one of the specified bits need to be on is decided by the CONNECT= operand.)

If, however, the MASK= operand is specified as all hexadecimal zeros, the branch is never taken regardless of the CONNECT= operand specification or the field being tested.

If this operand is coded, neither OPTION= nor SPECACT= may be coded.

```
MASK= X'statest'
```

Function: Specifies the 1-byte bit configuration used to test a particular resource's option byte.

Format: *statest* is a hexadecimal field, 1 byte in length framed by X'', for a total of five coded characters.

Default: None. Specification is optional.

Note: If specified bits in the particular option field byte are on, no branch is taken. (Whether all or only one of the specified bits need to be on is decided by the CONNECT= operand.)

This operand is valid only if the OPTION= operand is specified.

```
OFFSET={integer}
       {label  }
       {0      }
```

Function: Specifies which byte within the option field is to be tested.

Format: *integer, label,* or 0. *integer* is an explicit decimal integer from 0 to 255. *label* is a symbol that has previously been equated to a decimal value from 0 to 255.

Default: OFFSET=0.

Note: This operand is valid only if the OPTION= operand is specified.

```
OPTION=name
```

Function: Specifies the name of an option field to be tested.

Format: *name* must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Note: This operand must be the name of an OPTION macro instruction.

```
SPECACT={action }
        {ACCTING}
```

Function: Designates the special action flag--set by SPECACT or ACCTING macro processing--to be tested. If the flag is not on, TESTBR will branch to the TESTBR macro specified by the BRANCH= operand. If the flag is on, the instruction following TESTBR is executed next.

Format: *action* or ACCTING. *action* is a decimal integer, or equated equivalent, with a value from 1 through 24.

Default: None. Specification is optional.

Note: *action* designates a special action flag to be tested. The TESTBR macro designated by the BRANCH= operand is the next instruction executed if the flag is off.

ACCTING specifies that the macro should test the special action flag set by the ACCTING macro (or by you at closedown via a SPECACT macro) signaling that a half-section of the system accounting area is filled.

```
symbol
```

Function: Specifies the name of the macro. When a name is specified, no action is performed by the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is required if no operands are coded; it is not allowed if any operand is coded.

Note: When a name is coded, no operand is allowed.

```
THRESH={integer}
       {label  }
```

Function: Specifies the threshold limit value to be compared with the counter byte specified by the OPTION= and OFFSET= operands.

Format: *integer* is a decimal integer from 1 to 255. *label* is a symbol that has previously been equated to a decimal value from 1 to 255. It is your responsibility to assign this operand a value that is within the prescribed limits.

Default: None. Specification is optional.

Note: The TESTBR macro increments the selected byte by one and then compares the result to the THRESH= value. If the option field byte contains a lower value than the threshold, a branch is taken to the TESTBR macro specified by the BRANCH= operand (threshold-not-reached condition). If the option field byte contains a value equal to the threshold limit value, the branch is not taken, the next sequential instruction is executed and the threshold counter byte of the user specified option field is reset to zeroes.

This operand is valid only if the OPTION= operand is specified.

```
TOFFSET={integer}
        {label  }
```

Function: Specifies the offset into the option field specified by the OPTION= operand that contains the thresh value to be tested against the counted value.

Format: integer or label integer is an decimal integer from 0 to 255. *label* is an assembler language symbol that can be used within the parentheses of a DC AL1( ) assembler language instruction.

Default: None. Specification is optional.

Note: This operand is valid only if the OPTION= operand is specified.

Return Codes

None.

TESTOPEN Macro

The TESTOPEN macro:

- Tests for the successful initialization and activation of the one or more data sets named in one TCAM OPEN macro instruction
- Provides the system console with the ddname of any data set that was not successfully initialized and activated, and allows the console operator to choose between continuing to process and abnormally terminating the MCP
- May optionally be issued after the specific OPEN macro to be tested, but before the READY macro
- May be coded more than once.

Supported Resources and General Requirements: Not applicable.

Valid subgroups: Not applicable.

NAME	OPERATION	OPERANDS

[symbol]	TESTOPEN	OPEN={name }
		{(register)}
		[,TYPE=L]

```
OPEN={name       }
     {(register)}
```

Function: Specifies the name of the OPEN macro that is to be tested.

Format: *name* or *(register) name* must conform to the rules for assembler language symbols. *(register)* is a decimal integer, within parentheses; minimum 2, maximum 12.

Default: None. Specification is required.

Note: *name* is the name of the OPEN macro to be tested.

If the register format is used, the specified register must contain the address of the OPEN macro to be tested.

If there is one list-type OPEN macro and one execute-type OPEN macro, the list-type should be indicated in this operand.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

TESTOPEN

TYPE=L

Function: Specifies that the macro is to test a list-type OPEN macro.

Format: L.

Default: None. Specification is optional.

Note: If this operand is omitted, the OPEN macro is assumed to be an execute-type macro.

Return Codes

None.

TLIST Macro

The TLIST macro:

- Defines a cascade entry or distribution entry in the terminal table
- Is optional among macros defining the terminal table.

The TLIST macro causes the name of a list of single or process entries in the terminal table, together with information about the entries in the list, to be included as an entry in the terminal table.

A distribution or cascade list consists of the names of single or process entries in the terminal table. One TLIST macro must be specified for each list to be created. TLIST terminal entries can be used only to send data from TCAM to destinations.

When a message contains the name of a distribution list as a destination code, TCAM sends the message by separate transmissions to each external LU or application program indicated by an entry in the list. Each entry in the list must have a corresponding single or process entry in the terminal table. When a message contains the name of a cascade list as a destination code, TCAM places the message on the destination queue for that valid destination in the list that has the fewest messages waiting to be sent to it. If several destinations have the same number of messages, the message is queued for the first such destination.

If any one of the destinations has disk queuing, it should be specified ahead of all destinations with main-storage queuing. If the first (or primary) destination has main-storage queuing, the message may be serviced and freed from the queue before it can be copied to all secondary destinations. This could cause an ABEND (045-2). Having disk queuing for the primary destination ensures that the message will be available for a longer period of time, even after it is marked serviced.

Supported Resources and General Requirements: ALL.

Valid subgroup: Not applicable.

NAME	OPERATION	OPERANDS

symbol	TLIST	`LIST=(entry [,...])`
		`,TYPE={D}`
		`      {C}`

`LIST=(entry,...)`

Function: Specifies the actual entries in the distribution list or cascade list being created.

Format: Each entry is the name of a single, process, or cascade entry in the terminal table. If TYPE = D, at least two entries must be specified. If TYPE = C, only one entry is required. Each entry must have a valid TCAM queue associated with it.

Default: None. This operand is required.

The name of a distribution entry in the terminal table may not be specified as an entry in a distribution list. If the list being created is a distribution list, it may contain the name of one or more cascade entries. If it is a cascade list, it may not contain the name of a cascade or distribution entry.

Because of the limitation of 255 characters in a macro operand, a facility is provided to specify additional TLIST entries if necessary. A comma placed as the last character of the entry suboperand indicates a continuation of the list. The next source statement would then be coded:

```
symbol  TLIST LIST=(entry,...)
```

where *symbol* is the TLIST name as specified on the previous TLIST macro that specified the continuation.

There is a limit of 32,767 entries in a distribution or cascade list.

```
symbol
```

Function: Specifies the name of the list.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. This name is required.

```
TYPE={D}
     {C}
```

Function: Specifies whether the list is a distribution or a cascade list.

Format: D or C.

Default: None. This operand is required.

Note: D specifies a distribution list. C specifies a cascade list.

Return Codes

None.

TPROCESS Macro

The TPROCESS macro:

- Specifies a queue to be associated with an application program
- Provides queue security for an application program.

The TPROCESS macro causes the name of a queue for an application program and associated information to be included as an entry in the terminal table. The entry produced is a process entry.

One TPROCESS macro must be included for each destination queue to which an application program can direct a GET or READ macro, and at least one must be included for each process entry to which a PUT or WRITE macro may be directed.

An operand of the TPROCESS macro specifies the name of a process control block (PCB), which establishes communication between a message handler and application. (The PCB is created by coding a PCB macro).

The maximum number of TPROCESS entries that can refer to the same process control block is 255. A GET/READ TPROCESS entry can be defined as an SNA LU.

You may specify that checkpointing of the application program is to be synchronized with that of the message control program. Synchronization of operating system and TCAM checkpoints is discussed in the *TCAM Application Programming* manual.

You specify the initial contents of the option fields for the process entry in the terminal table.

The TPROCESS macro helps connect an application program to the message control program. The GET and PUT or READ and WRITE macros issued in an application program each specify the name of a data control block created by a DCB macro issued in the application program. The DCB macro specifies (by its DDNAME= operand) a DD card. The QNAME= parameter of the DD card names a process entry. The PCB= operand of the TPROCESS macro creating this entry specifies a process control block. The MH= operand of the PCB macro creating the process control block specifies the message handler that handles messages directed to and received from the application program.

Supported Resources and General Requirements: APP.

Valid subgroup: Not applicable.

TPROCESS

NAME	OPERATION	OPERANDS
procname	TPROCESS	PCB=pcbname [,ALTDEST=entry] [,CKPTSYN={YES}] {NO } [,DATE={YES}] {NO } [,LEVEL=(integer[,...])] [,LU={YES}] {NO } [,OPDATA=(data[,,,}))] [,QBACK={YES}] {NO } [,QUEUES=form] [,RECDEL=delimiter] [,SECTERM={YES}] {NO } [,SECURE={YES}] {NO }

ALTDEST=entry

Function: If this process entry is for GETs or READs issued by an application program, this operand specifies the alternate destination to which to send the message at the time the zone containing the message is being serviced for reuse. If this process entry is for PUTs or WRITEs from an application program, this operand specifies the destination to which replies to operator control commands issued by the application program are sent.

Format: The name of any external LU or process entry in the terminal table.

Default: None. Specification is optional.

Note: For a GET or READ process entry, the entry specified may by the one created by this TPROCESS macro, thus preventing the message from being discarded from a reusable queue.

For a PUT or WRITE entry specifying SECTERM=YES, the entry specified may be a GET or READ process entry for the application program that will receive the response to the operator control commands.

CKPTSYN={YES}
{NO }

Function: Specifies whether the destination queue to which the application program directs its GETs or READs is to be restarted at the point where the last checkpoint record was created.

Format: YES or NO.

Default: CKPTSYN=NO.

Note: CKPTSYN=YES specifies that no purging or scanning of the queue is to be performed when TCAM is restarted. If an OS/VS checkpoint of the application program is used in synchronization with the TCAM checkpoint, specify CKPTSYN=YES. If this operand is omitted, the queue is scanned and updated when TCAM is restarted. When synchronization is not specified, operation following restart with scan resumes with the first unserviced message for the queue (a message is considered serviced when a GET or READ is issued for the next message from the queue). The first unserviced message is determined in the scan of the message queue done when TCAM is restarted. When not using synchronization with an OS/VS checkpoint, it is necessary to check for one duplicate message upon restart (that is, the message being processed when failure occurred).

```
DATE={YES}
     {NO }
```

Function: Specifies whether the date and time are to be recorded for each message received for the process entry.

Format: YES or NO.

Default: DATE=NO.

Note: When a message is received for the application program, TCAM records the date and time. When the application program issues a GET or a READ macro, TCAM places the recorded date and time and the source of the message in the area specified by the DTSAREA= operand of the TPDATE application program macro. This operand requires that the DATE= operand also be specified on the PCB macro for this process entry.

Recording of the date and time takes place when the FORWARD macro is executed. The conditions under which date and time are not inserted are:

- A FORWARD macro error is detected.
- An additional buffer is required but none is available.
- The message is not processed by the FORWARD macro (for example, messages generated by the ERRORMSG macro).

```
LEVEL=(integer[,...])
```

Function: Specifies the permissible message priority levels that may be used with a PRIORITY macro for a message queued on this process queue.

Format: Each integer is a decimal integer. The *integer,...* values must be specified in ascending order.

Default: None. Specification is optional.

Maximum: 255.

Note: If this operand is omitted, all messages sent to the application program by this process entry are assumed to have zero message priority. If the messages being sent to the application program through this process entry can have, for example, priorities of 1, 3, or 5, the LEVEL= operand is coded LEVEL=(241,243,245). Here, 241, 243, and 245 are the decimal equivalents of the hexadecimal EBCDIC representations of the numbers 1, 3, and 5.

For more information on message priority, see the discussion of the PRIORITY macro in this manual; also see the *TCAM Installation Guide.*

```
LU={YES}
   {NO }
```

Function: Specifies whether the TPROCESS entry should be defined as a host LU.

Format: YES or NO.

Default: LU=NO.

Note: LU=YES is valid only if the QUEUES= operand is coded.

```
OPDATA=(data,...)
```

Function: Specifies the actual data to be inserted in the set of option fields assigned to this process entry (see the discussion of the OPTION macro), and also specifies which option fields are not to be created for this process entry.

Format: The maximum length and type of data specified for each option field must correspond to the length and type specified by the OPTION macro that defines the field. The order in which the OPTION macros are specified must correspond to the values of data specified in this operand.

Default: None. Specification is optional

A comma:

- Delimits the data for each field
- Indicates that no data is specified for the first or an intermediate field defined by an OPTION macro
- Indicates that the OPDATA= operand is to be continued (if included immediately preceding the right parenthesis).

You must specify either data and a comma, or a comma alone, for the first and each intermediate field that is specified by an OPTION macro. When specifying option fields for a particular process entry, you may omit the last several option fields defined by OPTION macros by merely closing the parentheses after the data for the last field you wish to define. A comma alone is coded if a field other than the last is not to be defined for this process entry. If the last field is not to be defined, no data is coded for the field and the comma is also omitted. Framing characters are not coded.

Example: Assume that four OPTION macros have been coded. If you want to specify all four fields for a particular application program, code the OPDATA= operand of the TPROCESS macro:

```
,OPDATA=(field1,field2,field3,field4)
```

where *field1, field2, field3* and *field4* represent the actual initial data to be inserted into each of the four option fields. If only *field1* and *field4* are to be implemented for this application program, code:

```
,OPDATA=(field1,,,field4)
```

If only *field1, field2,* and *field3* are to be implemented, code:

```
,OPDATA=(field1,field2,field3)
```

If only *field1* is to be implemented, you would code:

```
,OPDATA=(field1)
```

A message processed by an application program and then sent to a destination must be handled by two sets of outgoing MH subgroups. Macros issued in the incoming subgroups handling messages coming in from an external LU update the option fields assigned to the external LU. Macros issued in the outgoing subgroups handling messages for the application program update the option fields assigned to the process entry associated with the GET or READ macro that obtains the messages for processing. Macros issued in the incoming subgroups handling messages from an application program update the option fields assigned to the process entry associated with the PUT or WRITE macro that returns messages from the application program to the MCP. Macros issued in outgoing subgroups handling messages being sent to a destination external LU update the option fields assigned to that external LU. (For a description of which message handler subgroups are required when there is an application program, see the chapter titled "Coding the Message Handler" in the *TCAM Installation Guide.)*

Because the operand field of a macro is limited to 255 characters, TCAM provides a facility to specify additional OPDATA= parameters if necessary. A comma placed as the last character of the OPDATA= operand; that is,

```
OPDATA=(data,data,... data,)
```

indicates a continuation of the OPDATA= operand. The next source statement would then be coded:

```
symbol     TPROCESS OPDATA=(data,...)
```

where symbol is the process entry name as specified on the TPROCESS macro that specified the continuation. There is no limit (other than the number of option fields defined) on the number of continuation statements used.

```
PCB=pcbname
```

Function: Specifies the name of the process control block that is associated with this process entry.

Format: Must conform to the rules for assembler language symbols.

Default: None. This operand is required.

Note: The process control block is created by a PCB macro. All TPROCESS macros issued for different application programs must not have the same PCB.

```
procname
```

Function: Specifies the name of the process entry in the terminal table.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. This name is required.

Note: The name must be the same as that entered in the QNAME= parameter of the DD statement associated with the DCB macro for an application program. SYSCON cannot be used as a name.

```
QBACK={YES}
      {NO }
```

Function: Specifies whether the application program may issue the QRESET macro.

Format: YES or NO.

Default: QBACK=NO.

Note: QBACK=YES causes the Queue Reset Executor module to be loaded into the MCP address space and a 258-byte work area to be allocated in the MCP address space. For information about TCAM's QRESET facility see "Transferring Data Between TCAM and an Application Program" in *TCAM Application Programming*.

```
QUEUES=form
```

Function: Specifies where the message queues containing messages for the application program are to be maintained (for GET/READ operations only).

Format: DR, DN, MO, MR, or MN.

Default: None. Specification is optional.

Note: DR specifies reusable disk queues. DN specifies nonreusable disk queues. MO specifies main-storage-only queues. MR specifies main-storage queues with reusable disk backup. MN specifies main-storage queues with backup on nonreusable disk.

If the form of data set specified by this operand does not correspond to a related message queue data set defined by a DCB macro or MSUNITS= operand of INTRO, the TCAM system terminates abnormally. By omitting the QUEUES= operand, you specify that this process entry is for PUTs or WRITEs from an application program; in other words, do *not* code the QUEUES= operand for PUT/WRITE operations.

If MO, MR, or MN is specified, the MSUNITS= operand of the INTRO macro must specify a nonzero integer; otherwise, the TPROCESS macro does not assemble properly. If anything other than MO is coded, DISK=YES must be specified (or defaulted to) on the INTRO macro.

```
RECDEL=delimiter
```

Function: For a process entry associated with a GET or READ macro, this operand specifies a 1-byte, nonzero, hexadecimal value used to delimit a record for the application program. For a process entry associated with a PUT or WRITE macro, this operand specifies a value to be inserted at the end of each variable-length record returned from an application program by means of a PUT or WRITE macro specifying the DCB associated (by coding the QNAME= operand of its DD card) with the process entry.

Format: A single, unframed, nonzero, hexadecimal value.

Default: None. Specification is optional.

Note: This character may be inserted periodically into a TCAM message by means of a MSGEDIT macro whose data operand is coded DELIMIT, for a process entry associated with a GET or READ. If the RECFM= operand of the input DCB macro specified by a GET or READ macro in the application program specifies V, VB, or U and if the OPTCD= operand does not have the U suboperand coded in it, the application program GET or READ considers this character to be a record delimiter. The delimiter specified by RECDEL= may be included by you in the incoming message or may be inserted by means of a MSGEDIT macro.

For a process entry associated with a PUT or WRITE macro, TCAM automatically inserts the value at the end of each variable-length record. For other than variable-length records, this operand is meaningless.

```
SECTERM={YES}
        {NO }
```

Function: Specifies whether the application program may be considered a basic secondary operator control station (so that operator commands may be sent to TCAM from the application program by using a PUT or WRITE macro).

Format: YES or NO.

Default: SECTERM=NO.

Note: This operand is meaningful only if this process entry is associated with a PUT or WRITE macro. It is ignored if coded for a process entry associated with a GET or READ macro. If this process entry is to be the basic primary operator control station, SECTERM=YES must be specified for the entry.

If SECTERM=YES is specified, the ALTDEST= operand must also be coded specifying the destination of replies to operator commands issued by the application program.

```
SECURE={YES}
       {NO }
```

Function: Specifies whether this queue is to be secured.

Format: YES or NO.

Default: SECURE=NO.

Notes:

1. *The TPROCESS macro allows you to specify in your message control program whether a queue is to be secured. A secure queue is one that an application program can open only after authorization from the system operator. When an application program attempts to open a secure queue, a WTOR is issued, which gives the operator the JOBNAME of the job trying to open the queue and the QNAME of the queue. With this information, the operator decides whether to allow the queue to be opened. If the operator does not allow the queue to be opened, the application program is abnormally terminated with a 043-06 completion code.*

2. No *message stating that an application program attempted to open a secure queue is sent to the basic primary operator control station.*

3. *The QUEUES=form operand of the TPROCESS macro must be coded if the SECURE=YES operand is used.*

4. *Only the put queue (and not the get queue) is tested for security.*

Return Codes

None.

TSPECGRP Macro

The TSPECGRP macro:

- Creates a list of GROUP macros to be handled specially by the **ONLN S** and **OFFLN S** extended operator control commands
- Is optional and is coded in the initialization section of the MCP after INTRO and before READY.

Note: The IEDTCSD macro should be included in the MCP to provide dummy sections (DSECTS) for control blocks referenced by the code generated by DAFROUTE.

Supported Resources and General Requirements: Not applicable.

Valid subgroup: Not applicable.

NAME	OPERATION	OPERANDS
[symbol]	TSPECGRP	GROUPS=(groupname,...)

`GROUPS=(groupname,...)`

Function: Specifies the GROUP macros in the terminal table to be placed in the special group list.

Format: Each entry is the name of a single GROUP macro in the terminal table.

Default: None. Specification is optional.

Note: The names specified in this list are not checked for validity at assembly.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

None.

TTABLE Macro

The TTABLE macro:

- Defines the start of the terminal table
- Names the last entry in the table
- Defines the maximum length of name entries
- Is required as the first macro defining the terminal table
- Is issued only once.

An operand of the TTABLE macro specifies the name of the last macro issued in the section of code defining the terminal table; thus, the TTABLE macro defines the beginning and end of the terminal table coding section of the MCP. The TTABLE macro must be followed immediately by the macros defining the terminal table.

Supported Resources and General Requirements: ALL.

Valid subgroup: Not applicable.

NAME	OPERATION	OPERANDS
[symbol]	TTABLE	LAST=name [,MAXLEN=integer]

`LAST=name`

Function: Specifies the name of the last entry in the terminal table (that is, the name of the last TERMINAL, TLIST, or TPROCESS macro coded).

Format: Must conform to the rules for assembler language symbols.

Default: None. This operand is required.

Note: The length of this name is the default value for the MAXLEN= operand.

`MAXLEN=integer`

Function: Specifies the maximum number of characters in the names in the terminal table.

Format: An unframed decimal integer.

Default: Length of *name* specified by the LAST= operand. Specification is optional.

Maximum: 8.

Note: If this operand is omitted, the length of the last entry is assumed. The operand is not necessary if the lengths of all terminal-table entry names are the same, or if the last entry has the greatest length.

`symbol`

Function: Specifies the name of the macro and the name of the terminal name table (an internal table associated with the terminal table).

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

None.

TYPETABL Macro

The TYPETABL macro:

- Builds a branch table that can be referred to by the MSGTYPE macro.

A given message-type table named in a MSGTYPE macro is built from one or more TYPETABL macros. Each TYPETABL macro expands into a message-type character followed by the address to be branched to for that message type. The remaining TYPETABL macros for that table must have blank name fields and follow directly behind the TYPETABL macros having the table name. Within a group of TYPETABL macros, each one with a name is the beginning of a new table.

Supported Resources and General Requirements: ALL.

Valid subgroups: Inheader and outheader.

NAME	OPERATION	OPERANDS
[symbol]	TYPETABL	conchar ,ROUTINE=name

`conchar`

Function: Specifies the single message-type character.

Format: One nonblank character in character or hexadecimal format. If character format is used, the character must be a single unframed character or framed with C'' or CL1''. If hexadecimal format is used, the character must be framed with X'' or XL1''.

Default: None. This operand is required.

`ROUTINE=name`

Function: Names a user label in the same inheader or outheader subgroup in which the MSGTYPE macro names the message-type table. This is the label to be branched to when the message-type character is found.

Format: Must conform to the rules for assembler language symbols.

Default: None. This operand is required.

Note: The LOCOPT macro (STATION=) and IEDSHOW macro (TSTATUS,STAT) are valid in this user-written routine to obtain the contents of various TCAM control blocks. For detailed information on these macros, see the applicable macro descriptions in this chapter.

`symbol`

Function: Specifies the name of the macro and the message-type table.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

None.

UNLOCK Macro

The UNLOCK macro:

- Removes an external LU from extended lock mode.

When a LOCK macro is used to lock an external LU in extended lock mode to an application program, the UNLOCK macro may be included in the MH to remove the external LU from extended lock mode. When coded in an inheader or outheader subgroup, the *conchars* operand may be used in conjunction with control characters in the message header to provide the capability of optional execution of UNLOCK. When coded in an inmessage or outmessage subgroup, the *mask* operand may be used to provide optional execution.

UNLOCK specified in an inheader or inmessage subgroup removes the lock condition from the source. In an outheader or outmessage subgroup, the lock condition for the destination is removed. When the extended lock condition is not in effect, the macro is ignored.

When the UNLOCK macro is issued in an inheader subgroup handling inquiry messages being received from an external LU in extended lock mode, the message currently being handled is routed to the destination specified in its header or by a FORWARD macro if UNLOCK is issued before the FORWARD macro is issued. If UNLOCK is issued after FORWARD, the message is routed to the application program to which the originating external LU was locked.

The UNLOCK macro may be issued immediately following an unconditional LOCK macro to remove a certain message type from lock mode before the message is queued. For example:

```
LOCK     EXTEND
UNLOCK   A
```

places the external LU in extended lock mode for all except type A messages.

```
LOCK     MESSAGE
UNLOCK   J
```

terminates message lock mode only for type J messages.

For a discussion of the lock mode and its function in a TCAM system, see the description of the LOCK macro in this manual and the section of the chapter titled "The TCAM Inquiry/Reply Facility" in the *TCAM Installation Guide.*

Supported Resources and General Requirements: SNA.

Valid subgroups: Inheader, inmessage, outheader, and outmessage.

When Used in an Inheader or Outheader Subgroup

NAME	OPERATION	OPERANDS

[symbol]	UNLOCK	[conchars]
		[,BLANK={<u>YES</u> }]
		{NO }
		{char}

When Used in an Inmessage or Outmessage Subgroup

NAME	OPERATION	OPERANDS
[symbol]	UNLOCK	[mask] [,CONNECT={<u>OR</u> }] {AND}

```
BLANK={YES }
      {NO  }
      {char}
```

Function: Specifies whether EBCDIC blank characters are to be ignored when encountered in the character string in the message header being compared to the string specified by the *conchars* operand or whether blanks to be part of the header string when encountered in it. If EBCDIC blanks are to be counted as part of the header string, this operand also specifies whether some other hexadecimal character is to be ignored when encountered in the header string.

Format: YES, NO, or *char*. *char* is a single character that may be specified in either character or hexadecimal format. If character format is specified, it may be unframed or framed with C'' or CL1'' characters. If hexadecimal format is specified, it must be framed with X'' or XL1'' characters.

Default: BLANK=YES.

Note: YES specifies that the EBCDIC blank character (X'40') is to be ignored by this macro whenever it is encountered in the header character string being checked against the control character string specified by the *conchars* operand. For example, if BLANK=YES is coded and an eight-byte field in the header is being checked by this macro, a blank appearing in the fifth byte of the field is ignored and the sixth through ninth bytes are considered to be the last four bytes of the field (assuming that no blanks are coded in the sixth through ninth bytes).

NO specifies that the EBCDIC blank character is to be treated in the same way as any other character when it is encountered by this macro in the header string being compared to the string specified by *conchars.*

char specifies that the single character replacing *char* is to be ignored by this macro whenever it is encountered in the header string being compared to the string specified by the *conchars* operand. That is, the macro automatically skips over the character without performing a comparison and checks the next character in the header. If BLANK=*char* and *char* is not the EBCDIC blank character EBCDIC blank is not ignored by this macro when it is encountered in the header string but is compared to the character in the corresponding space in the *conchars* string in the same way as any other character.

This operand is meaningless unless the *conchars* operand is also specified.

`conchars`

Function: Specifies the character or character string that, if found in the header as the next nonblank field, executes the unlock function.

Format: One to eight nonblank characters in character or hexadecimal format. If character format is used, the string may be unframed or framed with C'' or CLn'' characters. If hexadecimal format is used the string must be framed with X'' or XLn'' characters.

Default: None. Specification is optional.

Note: If this operand is omitted, the unlock function is performed unconditionally.

```
CONNECT={OR }
        {AND}
```

Function: Specifies the type of logical connection to be made between the mask and the message error record.

Format: OR or AND.

Default: CONNECT=OR.

Note: AND specifies that the macro is to be executed only if *all* of the bits specified by the *mask* are on in error record. OR specifies that the macro is to be executed if *any* bit specified by *mask* is on in the message error record.

`mask`

Function: Specifies the 5-byte bit configuration used to test the message error record for the message. (See Appendix A for a discussion of the message error record.)

Format: Decimal or hexadecimal. If hexadecimal format is used, framing characters must be specified. If X'' is used, leading zeros must be coded. If XL5'' is used, leading zeros may be omitted.

Default: None. Specification is optional.

Note: Omitting the operand or specifying an all-zero mask causes unconditional execution.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols (see the *symbol* entry in the "Glossary").

Default: None. Specification is optional.

Return Codes

None.

UPCONDTN Macro

The UPCONDTN macro:

- Defines a valid TCAM startup/restart condition
- Is optional in a nonexecutable section of the MCP and may be coded up to 255 times
- Must be coded prior to any UPMSG, UPTYPE, or TCSUP macro.

Note: For further information on this macro, refer to the "Startup/Restart Message Generation Service Facility" chapter in *TCAM Utilities*.

Supported Resources and General Requirements: Not applicable.

Valid subgroups: Not applicable.

NAME	OPERATION	OPERANDS
opreply	UPCONDTN	condno [CONTYP={SDAY }] {<u>RSTRT</u>} [DSECT={YES}] {<u>NO</u> } [EXIT={name}] {<u>NONE</u>} [EXITYP={V}] {<u>A</u>}

```
condno
```

Function: Specifies the ordinal number of the UPCONDTN macro.

Format: A positional parameter, specifying a decimal number from 1 to 255. Condition numbers must be sequential, incremented by 1 for each successive UPCONDTN macro.

Default: None. Specification is required.

```
CONTYP={SDAY }
       {RSTRT}
```

Function: Identifies the startup/restart condition as one of two basic types: start-of-day or restart.

Format: SDAY or RSTRT.

Default: CONTYP=RSTRT.

Note: CONTYP=SDAY signifies a start-of-day condition; option fields and sequence number fields are cleared if so specified in the applicable UPMSG macro.

CONTYP=RSTRT indicates a normal restart condition; option and sequence number fields are not altered.

```
DSECT={YES}
      {NO }
```

Function: Permits the system programmer to generate the internal dummy section for the UPCONDTN parameter list.

Format: YES or NO.

Default: DSECT = NO.

Note: DSECT = YES generates the dummy section. If DSECT = YES is specified, all other UPCONDTN operands are ignored.

```
EXIT={name}
     {NONE}
```

Function: Permits you to name an exit routine to be entered from startup/restart processing after the system console operator has replied to the DKJ750A WTOR message with a valid startup/restart condition. The user routine returns to startup/restart processing with either a no-change indication or a different startup/restart condition number.

Format: *name* or NONE. *name* must be a valid assembler language symbol that is either the label of an instruction assembled in the MCP or the name of a module in the system load library.

Default: EXIT = NONE.

Upon entry to the user routine:

- Register 1 contains the operator reply condition number.
- Register 13 contains the address of saved startup/restart routine registers.
- Register 14 contains the return address.
- Register 15 contains the user routine base address.

Upon exit from the user routine:

- Register 1 contains either a condition number (same or changed) or X'00', signifying that you want to bypass all startup/restart processing at this time.

```
EXITYP={V}
       {A}
```

Function: Indicates where the user routine may be found.

Format: V or A.

Default: EXITYP = A.

Note: EXITYP = A signifies that the user exit routine is a label in the MCP; an A-type address constant is generated.

EXITYP = V signifies that the user exit routine is a module in the system load library; a V-type address constant is generated.

`opreply`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is required.

Return Codes

None.

UPMSG Macro

The UPMSG macro:

- Defines startup/restart message text and specifies the labels by which messages are referred to in UPTYPE macros
- Provides facilities to insert variable data into basic message text
- Specifies if you want to clear specified option fields during start-of-day conditions
- Is optional in a nonexecutable section of the MCP and may be coded as many times as required
- Must be coded after all UPCONDTN macros and before any UPTYPE or TCSUP macros.

Note: For more information, refer to the "Startup/Restart Message Generation Service Facility" chapter of *TCAM Utilities.*

Supported Resources and General Requirements: Not applicable.

Valid subgroups: Not applicable.

NAME	OPERATION	OPERANDS
msgname	UPMSG	[DATE=integer]
		[,DESTNAM=integer]
		[,DSECT={YES}]
		{NO }
		[,EXIT={name}]
		{NONE}
		[,EXITYP={V}]
		{A}
		[,LASTIN=integer]
		[,LASTOUT=integer]
		[,NLINDIC={X'chars'}]
		{C'chars'}
		{C'//' }
		[,OPDAT=(opname,integer,...)]
		[,OPTCLR=(opname,...)]
		[,OUTSEQ=charpos]
		[,TEXT={msgtext}]
		{NONE }
		[,TIME=integer]

DATE=integer

Function: Specifies that the current date into the startup or restart message.

Format: *integer* is a decimal integer; minimum is 1, maximum is 5 bytes less than the length of the specified TEXT= operand data.

Default: None. Specification is optional.

Note: This insertion is done *only* if the message is destined for an external LU for which construction of an FHP has been requested via the FHP=YES operand of the associated UPTYPE macro. If you wish to insert a date into a message without an FHP, you must do so by issuing a DATETIME macro in the appropriate outheader subgroup.

The insertion starts at the specified offset into the message defined by the TEXT= operand. The first byte of the text is character location one. Beginning at the specified character location in the message date specified in TEXT=, space for the date insertion must be reserved in the message text by coding a filler character (for example, X'40' or C'X') for each of the six positions required for this insertion.

```
DESTNAM=integer
```

Function: Specifies that the name of the external LU to which the startup or restart message is to be sent should be inserted into the startup or restart message.

Format: *integer* is a decimal integer; minimum is 1, maximum is 1 byte less than the length of the TEXT= operand data minus the MAXLEN= value as coded in the TTABLE macro.

Default: None. Specification is optional.

Note: The insertion starts at the specified offset into the message defined by the TEXT= operand. The first byte of the text is character position one. Beginning at the specified offset into the message text by coding a filler character (for example, X'40' or C'X') for each position required for this insertion. The total number of positions that must be reserved for destination name insertion is equal to the value coded in the MAXLEN= operand of the TTABLE macro.

```
DSECT={YES}
      {NO }
```

Function: Specifies whether an internal dummy section for the UPMSG parameter list is to be generated.

Format: YES or NO.

Default: DSECT=NO.

Note: DSECT=YES generates the dummy section. If DSECT=YES is specified, all other UPMSG operands are ignored.

```
EXIT={name}
     {NONE}
```

Function: Allows you to specify the name of a user-written exit routine to be entered from startup/restart processing after a message has been selected according to the current startup/restart condition and the type of the destination external LU.

Format: *name* or NONE. *name* must conform to the rules for assembler language symbol

Default: EXIT=NONE.

EXIT=NONE signifies that you do not want exit routine processing. EXIT=*name* provides the name of the user-written exit routine where *name* is a valid assembler language symbol that is either the label of an instruction assembled in the MCP or the name of a module in the system load library. Upon entry to the user routine:

- Register 1 contains the current startup/restart condition number
- Register 2 contains the ordinal number of the destination external LU's logical station type (UPTOPT option field)
- Register 3 contains the address of selected message text in the startup/restart routine work area (The first byte contains a hexadecimal character count of the length of the text, plus one)
- Register 13 contains the address of saved startup/restart routine registers
- Register 14 contains the return address
- Register 15 contains the user routine's base address.

Upon return from the user routine:

- Register 3 contains the address of the message to be sent in the startup/restart routine work area (The user-written exit routine may have altered the message text, but not its length or the length byte which precedes it; nonstandard data insertions may have been performed by the user routine. Note that data insertions requested by the DATE=, TIME=, DESTNAM=, LASTIN=, LASTOUT=, OPDATA=, and OUTSEQ= operands are performed on the message as it is returned from the user exit; care must be taken to ensure that displacement values specified in these operands apply to the returned message if it has been altered)
- Register 3 contains X'00' which indicates that you want no message to be sent.

```
EXITYP={V}
       {A}
```

Function: Indicates where the user routine can be found.

Format: V or A.

Default: EXITYP=A.

Note: EXITYP=A indicates that the user exit routine specified by the EXIT= operand is the label of an instruction assembled with the MCP; an A-type address constant is generated.

EXITYP=V indicates that the user exit routine is a module in the system load library; a V-type address constant is generated.

If EXIT=NONE is coded, this operand is ignored.

```
LASTIN=integer
```

Function: Specifies that the input sequence number last received from the destination should be inserted into the startup/restart message.

Format: *integer* is a decimal integer; minimum is 1, maximum is 3 bytes less than the length of the message text.

Default: None. Specification is optional.

Note: The insertion starts at the specified offset into the message defined by the TEXT= operand. The first byte of the text is character position one. Beginning at the specified offset into the message text, space for the last input sequence number insertion must be reserved in the message text by coding a filler character (for example, X'40' or C'X') for each of the four positions required for insertion.

The TCAM internal input sequence number for a station is not incremented unless you execute a SEQUENCE macro in the inheader subgroup of the DMH associated with the station.

```
LASTOUT=integer
```

Function: Specifies that the output sequence number last sent to the destination is to be inserted into the startup/restart message.

Format: *integer* is a decimal integer; minimum is 1, maximum is 3 bytes less than the length of the message text.

Default: None. Specification is optional.

Note: The insertion starts at the specified offset into the message defined by the TEXT= operand. The first byte of the text is character position one. Beginning at the specified offset into the message text, space for the last output sequence number insertion must be reserved in the message text by coding a filler character (for example, X'40' or C'X') for each of four positions required for this insertion.

```
msgname
```

Function: Supplies the name by which UPMSG-constructed basic message text is referred to in the MSGLIST= operand of the UPTYPE macro.

Format: Must conform to the rules for assembler language symbols. (See the *symbol* entry in the "Glossary.")

Default: None. Specification is required.

```
NLINDIC={X'chars'}
        {C'chars'}
        {C'//'   }
```

Function: The characters specified by this operand are coded by the TEXT= operand to specify the location of new line or carriage return-linefeed symbols in the message text, and also to reserve space for these symbols.

Format: 2 bytes of data, in either character or hexadecimal format, specified with framing C'' or X'' characters.

Default: NLINDIC=C'//'.

Note: During startup or restart processing, these characters are replaced by the appropriate symbol for the destination based on your specification of the NEWLINE= operand in the UPTYPE macro for the destination's logical station type.

`OPDAT=(opname,integer,...)`

Function: Specifies that the current contents of up to ten option fields for the destination should be inserted into the startup/restart message sent to the station.

Format: (opname,integer....). opname must conform to the rules for assembler language symbols and must be the name of an OPTION macro. *integer* is a decimal integer representing the offset into the message text at which the specified option field data is to be inserted; minimum is 1; maximum is 1 byte less than the length of the message text, minus the length of the option field.

Up to 10 pairs of *opname* and *integer* suboperands may be specified.

Default: None Specification is optional.

Note: Insertion of each option field's contents is to start at a specified offset into the message defined by the TEXT= operand. The first byte of text is character location one. Beginning at the specified offset for that option, space for each insertion must be reserved in the message text by coding a filler character (for example, X'40' or C'X') for each position required within each option data insertion requested.

`OPTCLR=(opname,...)`

Function: Permits specification of those option fields associated with a startup/restart destination which are to be cleared if the UPCONDTN macro defining the current startup/restart condition specified CONTYP=SDAY and this message is selected. Character-type (C) option fields are cleared to blanks (X'40') and all other types of fields are cleared to X'00'.

Format: Up to 10 option field names conforming to the rules for assembler language symbols, separated by commas and enclosed by framing parentheses; the parentheses may be omitted if only one option field is specified.

Default: None. Specification is optional.

Note: This operand is ignored for a startup or restart message to a destination defined via an ALTNAME macro. If an option field is specified that was not initialized for a particular destination, processing continues as if the option field had not been specified.

`OUTSEQ=charpos`

Function: Permits you to specify the location in the startup/restart message where you want to insert the output sequence number associated with the message.

Format: *charpos* is a decimal integer; minimum is 0, maximum is equal to the message text length. If 0 is specified, the output sequence number appears as the first field in the output message header.

Default: None. Specification is optional.

Note: This insertion is performed *only* if the message is destined for an external LU for which construction of an FHP has been requested via the FHP=YES operand of the associated UPTYPE macro. If you wish to insert an output sequence number into a message without an FHP, you must do so by issuing a SEQUENCE macro in the appropriate outheader section.

The insertion starts at the specified character position in the message specified in the TEXT= operand of the UPMSG macro (the first position in the text is character position number one). No space should be reserved in the TEXT= operand for output sequence number insertion because the insertion is performed when the FHP is processed by TCSENDBL macro processing.

```
TEXT={msgtext}
     {NONE   }
```

Function: Specifies the basic startup/restart message text or, alternately, that no message should be sent.

Format: msgtext or NONE. msgtext is the actual message text framed by C'', CLn'', X'', or XLn'' characters *(n* is the length of the basic text). The length of the text should not exceed 254 positions (minus the length of an FHP if the message is being sent to a logical station type requiring an FHP); excess text positions are truncated.

Default: TEXT=NONE.

Note: TEXT=NONE indicates no startup or restart message should be sent; TEXT= *msgtext* supplies the basic message text.

If data insertions are requested in the UPMSG macro by the DATE=, TIME=, DESTNAM LASTIN=, LASTOUT=, and/or OPDAT= operands, space must be reserved for such insertions in the TEXT= operand; use of X'40' or C'X' is recommend to reserve each position.

The location of each requested new-line symbol is indicated in the TEXT= operand by the presence of the characters specified in the NLINDIC= operand at the appropriate position in the basic message text. If a message is being sent to a printer, one new line symbol should be indicated as the last two text positions to reposition the carriage or print element to the first position of the next line.

```
TIME=integer
```

Function: Specifies that the current time into the startup/restart message.

Format: *integer* is a decimal integer; minimum is 1; maximum is 3 bytes less than the length of the TEXT= operand data.

Default: None. Specification is optional.

Note: The insertion starts at the specified offset into the message defined by the TEXT= operand. The first byte of the text is character location one. Beginning at the specified character location in the message data specified in TEXT=, space for the time insertion must be reserved in the message text by coding a filler character (for example, X'40' or C'X') for each of the four positions required for this insertion.

Return Codes

None.

UPTYPE Macro

The UPTYPE macro:

- Defines and specifies characteristics of an external LU for the startup/restart message generation processing routine
- Is optional in a nonexecutable section of the MCP and may be specified up to 254 times
- Must be coded after all UPCONDTN and UPMSG macros
- Must be coded prior to the TCSUP macro.

Note: For further information on this macro, refer to the "Startup/Restart Message Generation Service Facility" chapter in *TCAM Utilities.*

Supported Resources and General Requirements: Not applicable.

Valid subgroups: Not applicable.

NAME	OPERATION	OPERANDS

typeid	UPTYPE	MSGLIST=(msgname,...)
		[,ALTYP={SAME }]
		{NOMSG }
		{typeid}
		[,DSECT={NO }]
		{YES}
		[,FHP={YES}]
		{NO }
		[,NEWLINE={X'4015'}]
		{X'2615'}

```
ALTYP={SAME  }
      {NOMSG }
      {typeid}
```

Function: Provides the facility for specifying an alternative external LU type or for bypassing startup/restart message generation for destinations defined by an ALTNAME macro.

Format: SAME, NOMSG, or *typeid*. *typeid* must conform to the rules for assembler language symbols.

Default: ALTYP=NOMSG.

Note: ALTYP=SAME indicates that identical startup/restart message generation processing should be performed for external LUs of this type whether the external LUs are defined by a TERMINAL macro or an ALTNAME macro.

ALTYP=*typeid* the name of another UPTYPE macro specified *typeid* is to be used as the type of logical station.

ALTYP=NOMSG indicates that no message should be generated to external LUs defined by ALTNAME macros.

```
DSECT={NO }
      {YES}
```

Function: Specifies whether an internal dummy section for the UPTYPE parameter list is to be generated.

Format: NO or YES.

Default: DSECT = NO.

Note: DSECT = YES generates the dummy section. If DSECT = YES is specified, all other UPTYPE operands are ignored.

```
FHP={YES}
    {NO }
```

Function: Specifies whether a fixed header prefix (FHP) is to be built for a startup/restart message sent to an external LU.

Format: YES or NO.

Default: FHP = YES.

Note: FHP = YES indicates an FHP is to be built for startup/restart messages destined to an external LU of this external LU type.

FHP = NO indicates that no FHP is to be built.

```
MSGLIST=(msgname,...)
```

Function: This operand associates a list of startup/restart message texts with an external LU type. Each message text must have been previously defined by an UPMSG macro, and the message list must specify a message for each one of the startup/restart conditions previously defined by an UPCONDTN macro.

Format: Must conform to the rules for assembler language symbols. You specify the names of the UPMSG macros, separated by commas and enclosed by framing parentheses.

Default: None. Specification is required.

Note: As many message names must be specified as there are UPCONDTN macros coded in the MCP; each name specifies the message to be sent to this external LU type when the originally corresponding UPCONDTN macro has been selected as the current startup/restart condition.

```
NEWLINE={X'4015'}
        {X'2615'}
```

Function: Specifies the new-line symbol used for this external LU type.

Format: X'4015' or X'2615'.

Default: NEWLINE=X'2615'.

Note: This new-line symbol is inserted into the selected startup/restart message to replace the character specified by the user in the NLINDIC= operand of the UPMSG macro defining the message.

```
typeid
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is required.

Return Codes

None.

Chapter 3. Option Fields Reserved for Use by TCAM Functions

This chapter describes certain option fields that are either required for TCAM functions or are used in the TCAM model MCP and tells how to specify each of these fields. If you intend to include functions which require any of these option fields in your MCP, you must specify the required fields via OPTION macros and initialize them via the OPDATA= operand of your TERMINAL and TPROCESS macros or operator command.

The names of option fields described in this section should be reserved for the option fields described here.

The TCAM facilities that use option fields described in this chapter are discussed in the *TCAM Installation Guide.*

ALTDEST Option Field

The ALTDEST option field supports the TCAM transfer or purge facility and the TCAM alternate path facility for internodal destinations. For details on these facilities, see the chapter of the *TCAM Installation Guide* titled, "Coding the Message Handler."

Definition:

```
ALTDEST OPTION CL8
```

Initialization: The ALTDEST option field for an external LU or application should be initialized to contain the name of the alternate destination to which the external LU or application message traffic should be sent due to execution of the REDIRECT macro (either selectively or unconditionally) in MH inmessage or outmessage subgroups.

The ALTDEST option field for an internodal destination queue should be initialized to contain the name of another internodal destination queue entry designated by the user to represent a different extended route if this extended route is not operable. The internodal MH generated by the INODEMH macro includes code to support this function.

Note that if the automatic alternate path facility is desired for an internodal destination queue terminal table-entry, the ALTDEST option field must be specified for the entry. Otherwise, the user may choose to initially specify ALTDEST contents as zero and subsequently provide an alternate name via operator command.

Special Notes:

- The Display Statistics for a Resource extended operator command may be used to provide an online display of the current contents of the ALTDEST option field for any terminal table entry.

- The Activate/Deactivate Automatic Rerouting of Messages (AUTO) extended operator command may be used to alter dynamically the contents of the ALTDEST option field for any terminal table entry.

- The model MCP definition statement for this option field specifies CL8 to allow use of any valid terminal-table-entry name; if this field will always contain entry names less than 8 characters long in any particular MCP, it may be defined as a shorter character field. The length specified must be sufficient for the longest name that can be kept in the field. (The value specified by the MAXLEN= operand of the TTABLE macro is the highest value that need ever be specified.)

COPYNAME Option Field

The COPYNAME option field is used in the model MCPs to support the IBM 3271 COPY feature.

Definition:

```
COPYNAME OPTION CL8
```

Initialization: For IBM link-attached 3277-devices only, initialize the COPYNAME option field with the terminal-table-entry name of the 3284 or 3286 printer device to which requested display hard copies are to be sent.

Special Notes:

- The 3270 copy function requires user MH code for implementation. Refer to the model MCPs for an illustration of appropriate support code.

- The model definition statement for this option field specifies CL8 to allow use of any valid terminal-table-entry name; if this field is to always contain entry names less than 8 characters long in any particular MCP, it may be defined as a shorter character field. The length specified must be sufficient for the longest name that can be kept in the field. (The value specified by the MAXLEN= operand of the TTABLE macro is the highest value that need ever be specified.)

For more information on TCAM support of the 3270 Information Display System, see the *TCAM Installation Guide.*

ERRCOUNT Option Field

ERRCOUNT may optionally be used, in conjunction with the SENDMSG and TCSENDBL macros, to maintain a count of the number of logical error messages entered by an external LU. The Display Statistics for a Resource extended operator command can provide an online display of the current contents of this option field for the resource.

Definition:

```
ERRCOUNT OPTION H
```

Initialization: The ERRCOUNT option field, if used, should be initialized to zero.

FLAG3270 Option Field

The FLAG3270 option field is used to more accurately identify the 3270 device in order to allow the user to perform a specific function for the device.

One use of the FLAG3270 option field is:

Definition:

```
FLAG3270 OPTION XL1
```

Initialization: For IBM 3270 Information Display System terminal-table-entries, the FLAG3270 option field provides terminal attributes and must be initialized as follows:

Bit	Value
0	1 for a link-attached IBM 3275 stand-alone display station
1	1 for a channel-attached device; if 0, the device is link-attached
2	1 for a printer; if 0, the device is a display
3	1 for a small capacity device (480 characters); if 0, the device has a 1920 character capacity
4	1 for truncating messages dynamically in the outheader subgroup
5-7	(Reserved for internal processing use)

The contents of this byte should be the hexadecimal representation of the applicable bits as described above.

IEDBNDPM Option Field

The IEDBNDPM option field is used to save the session parameters used for session establishment. After invoking the user bind exit routine, if one exists, the session parameters are saved in this option field.

Definition:

```
IEDBNDPM OPTION CL80
```

In the definition above, the calculated length in bytes of the IEDBNDPM option field is the length of a logon mode of 36 bytes plus the 4-byte prefix for two simultaneous LU-LU sessions where the LUs are in different TCAM systems. For more information, see the *TCAM Installation Guide.*

Initialization: The IEDBNDPM option field, if used, must be initialized to blanks.

IEDBNDSZ Option Field

The IEDBNDSZ option field is used for external LUs which may engage in multiple concurrent sessions. It contains the length of the anticipated session parameters to be saved in the IEDBNDPM option field for sessions involving that external LU.

Definition:

```
IEDBNDSZ OPTION XL1
```

Initialization: IEDBNDSZ must be initialized for LUs which may engage in multiple sessions. IEDBNDSZ contains a hexadecimal value representing the maximum length of the anticipated session parameters for sessions involving this LU.

Note: The IEDBNDSZ option field is only meaningful if the IEDBNDPM option field has been coded for that same LU and if the LU may engage in multiple concurrent sessions.

IEDLMODE Option Field

IEDLMODE is a TCAM option field which specifies the logon mode name to be used when TCAM initiates a session with that external LU.

For further information, refer to the *TCAM Installation Guide.*

Definition:

```
IEDLMODE OPTION CL8
```

Initialization: The IEDLMODE option field, if used, should be initialized to the name of the desired VTAM logon mode table entry.

Note: If the IEDLMODE name is unknown to VTAM, the session initiation request will be rejected by VTAM. When this occurs, the session initiation request will be reissued to indicate that VTAM is to use the default session parameters for the LU.

IEDLUNAM Option Field

IEDLUNAM is a TCAM option field which specifies the host LU to serve as a session partner when TCAM initiates a session with that external LU. It contains the name of either a device message handler (DMH), a TPROCESS entry, or PROGID.

Definition:

```
IEDLUNAM OPTION CL8
```

Initialization: The IEDLUNAM option field, if used, should be initialized to the name of the desired host LU.

Note: If IEDLUNAM is invalid or if IEDLUNAM is not specified, then the value specified by the PROGID operand on the INTRO macro will be used as the host LU.

INMSGCT Option Field

INMSGCT may optionally be used in conjunction with a COUNTER macro to maintain a count of the number of messages entered by a resource. If the option field used for this purpose is named INMSGCT, the Display Statistics for a Resource extended operator command can provide an online display of the current contents of this option field for a resource.

Definition:

```
INMSGCT OPTION H
```

Initialization: The INMSGCT option field, if used, must be initialized to zero.

INODEID Option Field

INODEID is used internally by the INODEMH macro processing to guard against internodal message looping conditions in an extended networking environment.

Definition:

```
INODEID OPTION AL1
```

Initialization: The INODEID option field must be initialized for internodal destination queue terminal-table entries with the node identifier of another TCAM node associated with the internodal destination queue. (A symbolic name equated to the value may be used.) Omit this option field for other types of terminal-table entries.

Special Note: Refer to the description of the INODEMH macro for an explanation of loop control techniques.

INODFLAG Option Field

INODFLAG provides attribute characteristics for internodal destination queues to be used by the INODEMH macro when coded in the outmessage subgroup for determination of error-handling procedures in an extended networking environment.

Definition:

```
INODFLAG OPTION XL2
```

Initialization: For internodal destination queue terminal-table entries, the INODFLAG option field should be initialized as follows:

Byte 0	Set to 0 (Used internally during INODEMH processing.)
Byte 1	
Bit 0	Set to 1 to prevent the queue flush function from taking effect.
Bit 1	Set to 1 for disk-queued entries only if you intend to intercept the entry's queue when errors are detected.
Bit 2	Set to 1 if you intend to perform your own error recovery procedures when errors are detected for this entry.
Bit 3	Set to 1 for disk-queued entries that carry critical internodal traffic which you want sequence-checked.
Bit 4-7	Set to 0; reserved.

The contents of byte 1 should be the hexadecimal representation of the applicable bits as described above.

For other types of terminal-table entries, omit the INODFLAG option field.

Special Note: If both bits 1 and 2 of byte 1 are set to 0 and if errors are detected, TCAM will automatically try to route the message by means of a different extended route as specified via the contents of an ALTDEST or VIANODE option field. Refer to the INODEMH functional macro description and the *TCAM Installation Guide* for further details on how the indicators in the INODFLAG option field are used.

NEWMSG Option Field

The NEWMSG option field is used by the NEWMSG, SENDMSG, and EXMSG macros to support message generation from within an MH. These macros permit the user to generate and send predefined notification messages based on the receipt and analysis of an incoming message. For more information on this facility, see the chapter titled "Coding the Message Handler" in the *TCAM Installation Guide.*

Definition:

```
NEWMSG OPTION XL4
```

Initialization: The OPDATA= operand field initializing a NEWMSG option field should be initialized to zero.

OUTMSGCT Option Field

OUTMSGCT may optionally be used in conjunction with a COUNTER macro to maintain a count of the number of messages transmitted to a destination. If the option field used for this purpose is named OUTMSGCT, the Display Statistics for a Resource extended operator command can provide an online display of the current contents of this option field for the external LU of the internodal link.

Definition:

```
OUTMSGCT OPTION H
```

Initialization: The OUTMSGCT option field, if used, must be initialized to zero.

REALNAME Option Field

For any TPROCESS entry to which the user has assigned alternate names by means of the ALTNAME macro, the REALNAME option field is required to identify the primary name for correct execution of TCAM functions.

Definition:

```
REALNAME OPTION CL8
```

Initialization: For any TPROCESS to which an alternate name has been assigned, the REALNAME option must contain the name provided by the TPROCESS macro that initially defines the entry. If this specification is *omitted,* the following event occurs:

When the Display Statistics for Resources extended operator command is entered, all the figures for the TPROCESS are given once in the normal listing and once for each defined alternate name; this multiple count is also reflected in the totals.

The REALNAME option may be omitted if no ALTNAME macro was issued for the entry.

Special Notes: The model MCP definition statement for this option field specifies CL8 to allow use of any valid terminal-table entry name; if this field always contains entry names less than 8 characters long in any particular MCP, it may be defined as a shorter character field. The length specified must be sufficient for the longest name that can be kept in the field. (The value specified by the MAXLEN= operand of the TTABLE macro is the highest value that need ever be specified.)

RESOURCE Option Field

In an TCAM extended networking environment, the RESOURCE option field is used primarily to assign a resource identifier (unique within the TCAM node represented by each MCP) that can be used as a component in a two-part TCAM address for message routing (or, secondarily, for system accounting). When the MCP is initialized, the RESTABLE macro builds the resource table based on all the RESOURCE option fields that the user has initialized for terminal-table entries. For more information on the resource table, see the *TCAM Installation Guide.*

Definition:

```
RESOURCE OPTION AL2
```

Initialization: Because the RESOURCE option field is defined as an address-type constant, you may specify the resource identifier by means of a symbolic name equated to a numeric value from 100 through 65,535. (Resource identifiers 0 through 99 are reserved for system use.) Initialize the RESOURCE option field for all terminal-table entries representing external LUs that can send or receive messages in the extended networking environment and for application program terminal-table entries (PUT/WRITE-type) that can *originate* extended networking messages.

RETURNQ Option Field

The RETURNQ option field enables the TCAM routing facility to associate the single resource identifier assigned to any application program with both a PUT/WRITE-type destination queue (to which the application transfers messages) and a GET/READ-type destination queue (from which it acquires messages).

Definition:

```
RETURNQ OPTION CL8
```

Initialization: The RETURNQ option field for any PUT/WRITE-type TPROCESS terminal-table entry representing an application program that can both send and receive messages using routing by key should contain the name of the associated GET/READ-type TPROCESS entry for the same application program.

RETURNQ may not be coded for other types of terminal-table entries.

Special Notes: Each application that can receive messages using routing by key must be assigned a single resource identifier that is unique in its own TCAM node. If the application can receive and also send messages, however, it must have at least two queues associated with it; one is designated by a GET/READ-type TPROCESS terminal-table entry, and another is designated by a PUT/WRITE-type TPROCESS entry. The RETURNQ option field provides a mechanism to associate the PUT/WRITE entry with its related GET/READ entry.

In the following examples PUTQ is a PUT/WRITE-type TPROCESS entry representing an application in an extended networking environment with a resource identifier of 201. GETQ is the associated GET/READ-type entry defining the queue from which the application program receives its input messages:

```
                      RESOURCE option             RETURNQ option

PUTQ TPROCESS         201                         GETQ
GETQ TPROCESS         (omitted)                   (omitted)
```

Note that the RESOURCE option field is initialized only for the PUTQ entry—the entry that can originate the messages. When the program generates a message, TCAM macros use this field to initialize the TCAM origin address field (TOAF) in the FHP. However, since the PUTQ entry cannot receive messages, the RETURNQ option field is used to specify the associated entry that can receive messages—the GETQ entry. When the resource table is built (by the RESTABLE macro) the table entry for this resource identifier will point to the programs' GETQ process entry. This enables routing macros to place the proper TCAM address in the TCAM destination address field (TDAF) in the FHP and place messages destined for the program on the application program's input queue.

The RETURNQ facility can also be used to permit several different external LUs to function as a pool. Assume that TERM1, TERM2, or TERM3 are names of terminals that can each enter messages randomly. Responses go to the terminal that is free to receive them: TERM1, TERM2, or TERM3

```
                    RESOURCE option      RETURNQ option

TERM1 TERMINAL      128                  TRMLIST5
TERM2 TERMINAL      129                  TRMLIST5
TERM3 TERMINAL      130                  TRMLIST5

TRMLIST5 TLIST TYPE=C,LIST=(TERM1,TERM2,TERM3)
```

In this example, the resource-table entries for resources 128, 129, and 130 all point to the TRMLIST5 terminal-table entry. Messages routed to the cascade-type distribution list TRMLIST5 are placed on the queue for either TERM1, TERM2, or TERM3 (whichever is shortest).

The model definition statement for this option field specifies CL8 to allow use of any valid terminal table entry name; if this field is to always contain entry names less than 8 characters long in any particular MCP, it may be defined as a shorter character field. The length specified must be sufficient for the longest name that can be kept in the field. (The value specified by the MAXLEN= operand of the TTABLE macro is the highest value that need ever be specified.)

SECURITY Option Field

SECURITY contains an authorization mask used by the KEYPROC macro, in conjunction with the SECURTY= operand of the KEYDEF macro, to determine if this external LU is authorized to send messages to a security-checked destination using routing by key. For more information on routing by key, see the chapter titled "Coding the Message Handler" in *TCAM Installation Guide*.

Definition:

```
SECURITY OPTION XL3
```

Initialization: For external LUs that are authorized to access destinations via security-checked keys (that is, keys for which a KEYDEF macro with a SECURTY= operand has been specified), the SECURITY option field must be initialized with a 24-bit authorization mask, specified as three hexadecimal bytes.

SESSION Option Field

The SESSION option field is used for external LUs that may initiate end-to-end sessions to contain the name of an application with which the particular external LU or application is currently in an end-to-end session. The option field is used by TCAM routing macros to perform end-to-end session management. For more information on end-to-end sessions, see the chapter titled "Coding the Message Handler" in the *TCAM Installation Guide*.

Definition:

```
SESSION OPTION CL8
```

Initialization: For external LUs that can initiate end-to-end sessions, initialize the SESSION option field to either:

- The routing key name of an application with which the external LU or application is to be automatically placed into end-to-end session when the MCP is activated, or
- An asterisk (*), indicating that the external LU or application is not to be placed into an initial end-to-end session with any application.

SNADFC Option Field

The SNADFC option field is used in the model MCPs to recognize SNA protocal differences for the TCAM V3 supported divice types, and to take action accordingly.

Initialization:

```
SNADFC OPTION  XL1
```

Initialization: The option field is initialized as follows during system generation, at session startup (via the bind exit) and while processing messages through the DMHs:

Bit 0	Set to 1 only if exception request occurs on input
Bit 1	Set to 1 only if LU wants to send
Bit 2	Set to 1 only if cancel key is pressed on output
Bit 3	Set to 1 only if LU is in “change direction” state
Bit 4-7	Reserved

SNASENSE Option Field

SNASENSE is used in the model MCPs to contain SNA sense information that is stored in the option field for analysis by user code by execution of an IEDSENSE macro. IEDSENSE does not require that the field into which it moves sense data be an option field named SNASENSE.

Definition:

```
SNASENSE OPTION XL4
```

Initialization: The SNASENSE option field, if used, must be initialized to zero.

TCSOPTS Option Field

TCSOPTS is used to contain external LU, internodal destination queue, or application-related data that is required for execution of numerous TCAM functional macros and system service capabilities. Some considerations of TCSOPTS data follow:

- Whether the user has designated the external LU or application as an TCAM extended operator control station
- Whether the user has authorized that external LU or application to enter commands requesting retrieval of disk-queued messages (via the online retrieval system service program)
- Whether, during online system operation, one of the following extended operator commands has been processed:
  - Activate/Deactivate Automatic Rerouting of Messages
  - Copy Messages
  - Purge a Queue
  - Redirect Messages on Error Conditions
  - Transfer All Messages
- Whether or not the external LU should be sent startup notification messages at system startup or restart

Definition:

```
TCSOPTS OPTION XL4
```

Initialization: The user must supply the following 4 bytes of data to initialize TCSOPTS for an external LU or application program.

Byte 0

Bit 0	Reserved; set to 0.
Bit 1	Reserved; set to 0.

Bit 2	Set to 1 if the external LU is allowed to enter retrieval requests for the online retrieval system service program
Bits 3-4	Reserved; set to 0.
Bit 5	Set to 1 if the associated external LU or application is to be in perpetual end-to-end session. This bit, in conjunction with the SESSION option field, is used to support external LUs which are in perpetual end-to-end sessions to perform one function only. For example, a user may want to excess an extended operator control station system service program and may not want to access other application programs within the system. For information on coding this bit, see the chapter titled "Coding the Message Handler" in the *TCAM Installation Guide.*
Bit 6	Reserved; set to 0.
Bit 7	Set to 1 if the external LU is an extended primary or secondary operator control station. See the discussion of the basic and extended operator control system service programs in the *TCAM Utilities* manual for information on the extended operator control system service program.
Byte 1	Reserved; set to 0.
Byte 2	Reserved; set to 0.
Byte 3	
Bit 0	Set to 1 if traffic to the external LU should be redirected by the transfer or purge facility to the external LU named in the ALTDEST option field when errors occur. This bit should be off if the ALTDEST option field is unused for the external LU. The bit can be turned on or off dynamically by operator control; therefore, in most cases, this bit should be off when specified in the OPDATA= operand. For information on the transfer/purge facility, see the chapter titled "Coding the Message Handler" in the *TCAM Installation Guide.* See also special note below.
Bit 1	Set to 1 if traffic to the external LU should be unconditionally transferred to the external LU named in the ALTDEST option field. This bit should be off if the ALTDEST option field is unused for the external LU. The bit can be turned on and off dynamically (using the Redirect Messages on Error Conditions extended operator command); therefore, in most cases, this bit should be turned off when specified in the OPDATA= operand. See also special note below.
Bit 2	Set to 1 if startup/restart message processing is to be bypassed for the external LU. For information on such processing, see "Startup/Restart Message Generation Service Facility" in *TCAM Utilities.*
Bit 3	Set to 1 if unsent messages in this queue are not to be saved by the Save Message Queues system service program. This bit is ignored on a SAVE request if the queue is specifically identified by name.
Bit 4	Set to 1 if traffic to the external LU should be copied to the external LU named in the ALTDEST option field. For information on the use of this bit, see the chapter titled "Coding the Message Handler" in the *TCAM Installation Guide.* This bit should be off if the ALTDEST option field is unused for the external LU. The bit can be turned on and off dynamically (using the Copy Messages extended operator command); therefore, in most cases, this bit should be turned off when specified in the OPDATA= operand. See also special note below.
Bit 5	Set to 1 if traffic to the external LU should be purged. For information on the use of this bit, see the chapter titled "Coding the Message Handler" in the *TCAM Installation Guide.* The bit can be turned on and off dynamically (using the Purge a Queue extended operator command); therefore, in most cases, this bit should be turned off when specified in the OPDATA= operand. See also special note below.

Bit 6	Set to 1 if information about this external LU should not be included in output from the NET extended operator command. If the external LU is perpetually in transfer mode or is always held, this bit should be on, so that the external LU is excluded from exception-to-normal status display generated by the NET command.
Bit 7	Set to 1 if this terminal entry defines an internodal destination queue.

Special Note: Note that, in byte 3 of the TCSOPTS option field, bits 0, 1, 4, and 5 serve only as flags indicating whether the respective functions should be activated. Actual rerouting depends on relevant coding within the MH outmessage subgroup for the particular external LU.

THRESH Option Field

The THRESH option field is used to maintain five different subfields:

- A user-specified threshold value representing the maximum number of messages that may be queued at one time to a main-storage-only destination queue associated with the terminal-table entry. If the KEYPROC, DAFROUTE, or INODEMH macro detects a count of unsent messages on an associated main-storage-only queue exceeding this value, the message is not placed on the queue, and control is passed to a user-specified error routine (specified by the NODEDWN= or RESDOWN= operand of the applicable routing macro).
- A user-specified threshold value representing the maximum number of receive-side physical errors detected in the incoming section of the message handler associated with the terminal-table entry, beyond which the user considers the entry to be disabled.
- A user-specified threshold value representing the maximum number of send-side physical errors, detected in the outgoing section of the message handler associated with the terminal-table entry, beyond which the user considers the entry to be disabled.
- A field used to maintain an online count of the number of receive-side physical errors detected for the associated terminal-table entry.
- A field used to maintain an online count of the number of send-side physical errors detected for the associated terminal-table entry.

Definition:

```
THRESH OPTION XL5
```

Initialization: For each terminal-table entry for which the user wants to provide the associated functions, the following THRESH option field values should be provided:

- Byte 0: A one-byte hexadecimal value specifying the main-storage-only queue threshold value.
- Byte 1: A one-byte hexadecimal value specifying the receive-side physical error threshold value.
- Byte 2: A one-byte hexadecimal value specifying the send-side physical error threshold value.
- Bytes 3 and 4: Two one-byte initial physical error count fields, each specified as hexadecimal zeros.

Special Notes:

- The THRESH option field may be omitted if the user does not want threshold checking performed for the associated terminal-table entry.
- As described above, main-storage-queue threshold checking is carried out if the user initializes the THRESH option for the destination queue terminal-table entry.
- Receive-side and send-side error threshold checking requires the user to initialize the THRESH option field data for the terminal-table entries, and also to include user-written routines in the

message handler to perform the actual threshold checking and subsequent action, if the threshold is reached. Receive-side error threshold checking is performed in the incoming section of the MH; send-side error threshold checking is performed in the outgoing section of the MH. Consult the terminal message handler of the model MCP for an example of error threshold checking.

TRACEIT Option Field

The TRACEIT option field is used in the model MCPs. It is used with the buffer trace MH macro, IEDBUFTR. If the option field is X'01', a buffer trace entry will be created when the buffer trace is running.

Definition:

```
TRACEIT OPTION XL1
```

Initialization: The field is initialized during system generation and can be modified via the insert-option-field-data command.

UPTOPT Option Field

The UPTOPT option field is used in conjunction with the TCAM startup/restart message generation facility to identify the external LU type of the associated terminal-table entry. For information on this facility, see "Startup/Restart Message Generation Facility" in *TCAM Utilities.*

Definition:

```
UPTOPT OPTION AL1
```

Initialization: The UPTOPT option field should be initialized with the name of an UPTYPE macro instruction if user-tailored startup/restart message generation is required. If the UPTOPT option field is not initialized for any external LU, that external LU assumes the logical terminal type defined by the first UPTYPE macro coded in the MCP.

If standard TCAM startup/restart processing is used, the UPTOPT option macro should not be coded.

VIANODE Option Field

The VIANODE option field for internodal destination queues designates a secondary alternate internodal destination queue for use if both this entry and its primary alternate (as specified by the ALTDEST option field) are inoperative.

Definition:

```
VIANODE OPTION CL8
```

Initialization: For internodal destination queue terminal-table entries for which the TCAM automatic alternate extended routing facility is desired, the VIANODE option field should contain the name of another terminal-table entry for an associated internodal destination queue that will serve as a second alternate extended route to the TCAM node with which this entry connects. VIANODE should be omitted for other types of terminal-table entries.

Special Notes:

- The model MCP definition statement for this option field specifies CL8 to allow use of any valid terminal-table entry name; if this field will always contain entry names less than 8 characters long in any particular MCP, it may be defined as a shorter character field. The length specified must be sufficient for the longest name that can be kept in the field. (The value specified by the MAXLEN= operand of the TTABLE macro is the highest value that need ever be specified.)

Appendix A. Message Error Record

TCAM assigns a 5-byte *message error record* to each message for the duration of its processing by the input or output subgroups of a message handler. Each of the 40 bits of the message error record (except reserved and unused bits) indicates the presence (when 1) or the absence (when 0) of a specific error condition that has affected or may affect successful processing or transmission of the message. Some of the errors that may be recorded in the message error record are transmission errors Some are due to mistakes in entering a message (wrong sequence number, invalid origin code); and some are due to a shortage of system resources (insufficient number of buffers, insufficient space in a main-storage-only message queue data set). For more information on specific errors, refer to the applicable section of the *TCAM Installation Guide.*

You may code one or several error-handling macros in your message handler; among these are CANCELMG, ERRORMSG, HOLD, IEDHALT, MSGGEN, and REDIRECT. CANCELMG may be coded in the inmessage or outheader subgroup, HOLD may be coded in the inheader, inmessage, outheader, or outmessage subgroup, IEDHALT may be coded in the inheader, inmessage, or outmessage subgroup, while the others may be coded in either the inmessage or outmessage subgroup. These macros each have an optional 5-byte error mask operand, which may test the message error record, so that the error-handling macro for which the mask is coded is executed only if the errors specified in the mask have occurred. When error-handling macros are coded in an inmessage subgroup, they test the message error record after the message is received from a external LU or application program; in the outmessage subgroup, they test the message error record after the message is sent to a external LU.

SNA Sense Data

For SNA logical units, part of the SNA sense data sent or received during the current operation is made available in the message error record as follows:

- One bit in the message error record is used for each of the possible SNA major sense data codes (path error, RH usage error, state error, etc.) as given in the first byte of SNA sense data.

Bit	**Meaning**
13	Request reject error
21	Request error
23	Path error
29	RH usage error
30	State error

- The second byte of SNA sense data (SNA sense modifier byte) is an 8-bit binary modifier that further defines the error condition. This byte is moved into bits 32 through 39 of the message error record. Note that this is an 8-bit binary number further defining the error condition; it is not 8 independent bits.

- The third and fourth bytes of the SNA sense data (the user sense bytes) are not available in the message error record. However, bit 31—user sense received—is set in the message error record whenever nonzero user sense data is received in these bytes. The IEDSENSE macro may be used to obtain the sense value.

Testing the Message Error Record With an Error Mask

Depending on how the sense-modifier byte (see the "SNA Sense Data" section in this appendix) is coded, the testing of the message error record is done one of two ways:

1. If the sense-modifier byte in the error mask is coded as X'00' any major SNA sense codes in the error mask are interpreted as being general. For example, if bit 30 is on in the error mask and the sense-modifier byte is X'00', the mask tests for any type of state error, regardless of sense-modifier value. In this case, the 40 message error record bits are completely independent of each other. They are tested by simply combining (by AND) a copy of the message error record and an error mask and checking the result as follows:

Result of AND Operation	**Conclusion**
Zero	None of the error mask bits were on in the message error record.
Same as error mask	All of the error mask bits were on in the message error record.
Nonzero (but not equal to the error mask).	Some of the error mask bits were on in the message error record.

   The decision to execute or not execute the macro depends on the results of the error mask test and on how the CONNECT= operand on the macro was coded. The logic of that decision is shown in the following table where the top row indicates how many of the error mask bits matched the corresponding bits in the message error record:

	NONE	**SOME**	**ALL**
CONNECT=OR	NO	YES	YES
CONNECT=AND	NO	NO	YES
CONNECT=NAND	YES	NO	NO

2. If a nonzero sense-modifier value is coded in the error mask, then any SNA major sense codes in the error mask are interpreted as being specific in meaning. For example, if bit 30 is on in the error mask and the sense-modifier byte contains the value X'03', then the mask tests for the specific state error with a sense-modifier byte of X'03'. All other types of state errors are ignored, as are any other major error types with sense-modifiers of X'03'.

Figure A-1 on page A-3 illustrates these rules for SNA logical units.

User-Coded Error Mask	Message Error Record	How many of the error mask bits matched the MER bits?	Macro Executed CONNECT= OR	AND	NAND
X'0000004200' • Text error • Any type of state error	X'0000008202'	Some (This is a state error type.)	YES	NO	NO
X'0000000403' • RH usage error with sense modifier = X'03'	X'0000008403'	All	YES	YES	NO
X'0000008403' • Selection error • RH usage error with sense modifier = X'03'	X'0000004402'	None (This is an RH usage error, but not the specific one tested for.)	NO	NO	YES
X'0004008003' • Selection error • Request reject with sense modifier = X'03'	X'0000004403'	None (The matching sense modifier bytes are ignored because the major SNA sense bits don't match.)	NO	NO	YES
X'0000050000' • Any type of request or path error	X'0000010004'	Some	YES	NO	NO

Figure A-1. Rules for SNA logical Units

Note: When an error occurs during initiation of an SNA session, it is shown in the message error record if SNA sense data is present. SNA sense data associated with any error buffers passed to the user's message handler can be examined by issuing the IEDSENSE macro.

In addition to SNA sense data, session-initiation errors are indicated by the combination of bits 24 and 26 in the message error record.

Message Error Record Bit Definitions

The meaning of each bit in the message error record follows. Bit 0 is the leftmost bit and bit 39 is the rightmost bit in each error record. The following notes will be referenced by some of these bit meanings to provide additional information which applies to more than one bit.

Notes:

1. *Input from non-SNA devices as well as SNA devices will not be queued if the MH attempts to queue the input for a reusable queue or main storage queue while the queues are congested. Instead, the incoming message lost MER bit will be set to notify the MH that data will be lost.*

2. *When error notification to the DMH is required and SNA sense data is provided by VTAM, the appropriate MER SNA sense value is set to reflect the SNA sense data provided by VTAM.*

3. *When error notification to the DMH is required and SNA sense data is not provided by VTAM:*

   - *bit 23 of the MER will be set to indicate a SNA major and minor sense value of X'8000'*

   - *the RTNCD information will be stored in the user sense field of the X'8000' SNA field and will result in the following MER bit settings:*

     Retriable completion - bit 5 of the MER is set
     Data integrity error - bit 7 of the MER is set
     Environment error - bit 10 of the MER is set
     Logic error - bit 11 of the MER is set.

   - *SNA sense data associated with any error buffers passed to the user's message handler may be examined by issuing the IEDSENSE macro.*

4. *When a message handler violates the session protocol by using a DFC command which is incompatible with the FM profile, the MER will be updated to indicate the error with an SNA sense code of X'1003' (function not supported).*

5. *For more information on return code (RTNCD-FDBK2) combinations, refer to "Appendix B" of the VTAM Programming manual.*

Bit	*Meaning*
0	*Header error* The scan pointer has reached the end of the last segment in the message, but the end of the inheader or outheader subgroup has not been reached.
1	*Invalid origin code* The ORIGIN macro found that the origin field in the incoming header contained a code that did not correspond to any external LU name in any group.
2	*Unused*
3	*Sequence number high or not a valid number* The SEQUENCE macro found a message sequence number that is not a valid decimal integer or is higher than the expected number for the next message originating from the resource. When this error is detected, the expected sequence number is not changed. If you do not cancel the message, the same sequence number may appear in more than one message.

4 *Sequence number low*
The SEQUENCE macro found a message sequence number lower than the expected number for the next message originating from the resource. You may inadvertently use the same message number in more than one message. This bit can detect such an error, thus allowing you to resend the corrected message.

5 *Retriable completion*
TCAM issued a VTAM application macro that resulted in a retriable (temporary) error condition. RTNCD X'08' was received. TCAM has retried the macro up to a retry limit of three (3), and is treating the condition as a permanent error (see note 3).

6 *Insufficient buffers*
The TCAM buffer-assignment routine was unable to provide sufficient buffers for the incoming message. Infrequent occurrences of this condition may be corrected by requesting the origin resend the message. Frequent occurrences of this condition indicate that TCAM should be redefined with a larger number of buffer units.

7 *Data integrity error*
TCAM issued a VTAM macro for a request that was cancelled due to another operation being performed which negated the original request. RTNCD X'0C' was received (see note 3).

8 *MSMIN passed*
The percentage of the number of units specified by the MSUNITS= operand of the INTRO macro that are queued in the main-storage message queue data set has fallen to or below the number specified by the MSMIN= operand of INTRO.

9 *MSMAX passed*
The percentage of the number of units specified by the MSUNITS= operand of the INTRO macro that are queued in the main-storage message queue data set has risen to or above the number specified by the MSMAX= operand of INTRO; indicates that the message queues are almost full.

10 *Environment Error*
The connectivity of the session has been disrupted. RTNCD X'10' was received (see note 3).

11 *Logic error*
Either:

- An invalid sequence of VTAM application macros was issued
- The issuing of a VTAM macro was incorrect for that particular session
- The information comprising the macro invocation was either invalid or incomplete.

RTNCD X'14' or X'18' was received (see note 3).

12 *ACB closed*
TCAM issued a VTAM application macro associated with an ACB that was currently closed.

13 *Request reject*
A request RU was delivered to the intended logical unit; although it was understood and supported, the request could not be executed.

14 *Invalid destination code*
A destination specified in the FORWARD macro is invalid because it does not have a matching entry in the terminal name table.

15 *Software error*
AN SRB exit has abended. These SRB exits are scheduled by VTAM when an event has completed.

16 *Incoming message lost*
An incoming message has been lost due to lack of space in a main-storage-only or reusable-disk message queue data set (see note 1).

17 *Unused*

18 *Unused*

19 *Unused*

20 *User error*
This bit may set by you to indicate a logical error condition of your choosing. The bit is set by means of a TERRSET macro issued in a message handler.

21 *Request error*
A request RU was delivered to the intended logical unit but could not be interpreted or translated.

22 *Threshold reached (input only)*
For input operations, the threshold has been reached or the main-storage queue is full on a main-storage-only queue or the reusable disk queue is in a FULL condition for a destination using the reusable disk queue.

23 *Path error*
A request RU could not be delivered to the intended destination due to physical path outage or transmission header error.

24 *Error while attempting to send or receive messages*

- Indicates, when bits 26 and 27 are both off, that an error has occurred on a Bid command issued prior to sending a message to an LU.
- Indicates, when bit 26 is also set, that an error occurred during attempted session initiation. The LU data traffic state is not active. (Note that in this situation, SNA sense data will always be shown in the message error record *along with* bits 24 and 26.
- Indicates, when bit 27 is also set, that an outbound message has been either purged by a user purge exit or canceled by a CANCELMG macro in an outheader subgroup.

25 *Error during text transfer*
An error occurred during transfer of data. This bit is also set if an abnormally terminating application program has sent a partial message to the MCP with a PUT or WRITE macro, since TCAM automatically sends this partial message to the requested destination.

26 *Used in conjunction with bit 24 to indicate error occurred during session initiation.*

27 *Used in conjunction with bit 24 to indicate message has been purged or cancelled by the CANCELMG macro processing.*

28	*Message reorganized bit.* Indicates that while the output message was being sent, it was moved on the reusable message queue data set during a disk reorganization.
29	*RH Usage Error* Indicates that the value of a field or combination of fields in the RH violates architectural rules or BIND options previously selected.
30	*State error* Indicates a SNA sequence number error, or an RH or RU which is not allowed for the receiver's current session control or data flow control state.
31	*User sense received* Indicates that some user sense data was received (that is, the third and fourth bytes of SNA sense data were nonzero).

Sense Byte

This byte contains the second byte of SNA sense data. The byte contains an 8-bit binary number and the individual bits have no significance.

32-39 *Contains the second byte of the SNA sense data*

Appendix B. Internal and Transmission Code Charts

This appendix includes two sets of charts. Figure B-1 comprises four charts that include character sets and hexadecimal code for the extended binary coded decimal interchange code (EBCDIC) along with USASCII 7-bit and ASCII 8-bit representations that correspond to EBCDIC. Figures B-2 and B-3 comprise the second set of code charts; these figures illustrate in collating sequence (from hexadecimal 00 to hexadecimal FF) the valid hexadecimal representations of graphic and control characters for USASCII 7-bit and ASCII 8-bit code.

The first set of charts (Figure B-1) is based on the collating sequence of the EBCDIC that is used internally by the host processor (see column 3).

Arrangement of Charts

Three columns are associated with each entry in Figure B-1. For instance, columns 1, 2, and 3 are associated with the EBCDIC entry, and columns 4, 5, and 6 are associated with the USASCII entry. The columns on the extreme left and right ends of Figure B-1 contain reference numbers to designate rows. These numbers can be used in conjunction with the column numbers to designate a particular entry on the chart.

The arrangement of the charts in Figures B-2 and B-3 is based on the collating sequence of the hexadecimal representations for the codes.

Thus, columns 1 through 3 in Figure B-1 (in conjunction with the columns that correspond to the device that originally entered the message) may be used for decoding messages in a dump when those messages have already been translated by the appropriate translation table; if a message was entered by a device whose code is EBCDIC then columns 1 through 3 may be used for code translation *and* internal System/370 translation.

Conventions Used in Code Charts

In the code columns for the various devices in Figure B-1, some hexadecimal representations appear in parentheses while others do not. Where parentheses are used, only outgoing translation is performed by the translation table that corresponds to the device type for that column. For example, the letter "w" in internal System/370 code (EBCDIC) is represented by the bit pattern that corresponds to a hexadecimal A6 (see locations 166/1 and 166/3).

Nonequivalent Characters

The same or similar functions have different names among the various station types; for example, CR (carriage return) and NL (new line) are equivalent in row 13.

Substitutions

Where blank positions appear in the character columns of the charts, there is no equivalent internal EBCDIC character. Where these blanks appear, the SUB character is to be assumed (they were omitted to make the charts more readable). That is, in each translation table that handles incoming messages, each position representing an invalid transmission code bit pattern is translated to the EBCDIC 3F for the SUB character. In each translation table that handles outgoing messages, the transmission code bit pattern for a substitute graphic is contained in each of the following positions:

- Each position that represents an invalid EBCDIC bit pattern (a pattern to which no EBCDIC characters have been assigned)
- Each position that represents a bit pattern for a character having no equivalent in the destination station's character set

General Notes

Standard abbreviations represent the control characters. The full names of the characters are given in the following "Control Characters" section. For descriptions of these characters, see the reference manuals for the various stations.

Control Characters

ACK	Positive acknowledgment
BEL	Bell
BS	Backspace
BYP	Bypass
CAN	Cancel
CC	Cursor control
CR	Carriage (carrier) return
{CU1}	
{CU2}	Reserved for customer use
{CU3}	
{DC1}	
{DC2}	Device link escape
{DC4}	
DEL	Delete
DLE	Data link escape
DS	Digit select
EM	End of medium
ENQ	Enquiry
EOA	End of address (same as D)
EOB	End of block (same as B)

EOC	End of card
EOFC	End of first card
EOM	End of message
EOT	End of transmission (same as C)
ETB	End transmission block
ETX	End of text
FF	Forms feed
FIGS	Figures shift
FS	Field separator
HT	Horizontal tabulate
IFS	Interchange file separator
IGS	Interchange group separator
IL	Idle
IRS	Interchange record separator
IUS	Interchange unit separator
LC	Lowercase shift
LF	Line feed
LF-CR	Line feed-carriage return
LTRS	Letters shift
MZ	Minus zero
NAK	Negative acknowledgment
NL	New line
NUL	Null
PF	Punch off
PN	Punch on
PRE	Prefix
PZ	Plus zero
RES	Restore
RM	Record mark
RS	Reader stop
SI	Shift in
SM	Set mode
SMI	Start manual input
SMM	Start manual message
SO	Shift out
SOH	Start of header
SOS	Start of significance
SP	Space
STX	Start of text
SUB	Substitute
SYN	Synchronous idle
Tab	Tabulate (horizontal)
TM	Tape mark
TpAuxOff	Tape auxiliary off
TpAuxOn	Tape auxiliary on
UC	Uppercase shift
VT	Vertical tabulate
WRU	"Who Are You?"
XOFF	Transmitter off
XON	Transmitter on
Y	Positive response to polling, addressing or LRC/VRC

Ref.	Internal S/370 Code (EBCDIC)			USASCII			ASCII-8			Ref
	Character		Code (Hex)	Character		Code (Hex)	Character		Code (Hex)	
	Graphic	Control		Graphic	Control		Graphic	Control		
	1	2	3	4	5	6	7	8	9	
0		NUL	00		NUL	00		NUL	00	0
1		SOH	01		SOH	01		SOH	01	1
2		STX	02		STX	02		STX	02	2
3		ETX	03		ETX	03		ETX	03	3
4		PF	04						9C	4
5		HT	05		HT	09		HT	09	5
6		LC	06						86	6
7		DEL	07		DEL	7F		DEL	7F	7
8		GE	08						97	8
9		RLF	09						8D	9
10		SMM	0A						8E	10
11		VT	0B		VT	0B		VT	0B	11
12		FF	0C		FF	0C		FF	0C	12
13		CR	0D		CR	0D		CR	0D	13
14		SO	0E		SO	0E		SO	0E	14
15		SI	0F		SI	0F		SI	0F	15
16		DLE	10		DLE	10		DLE	10	16
17		DC1	11		DC1	11		DC1	11	17
18		DC2	12		DC2	12		DC2	12	18
19		TM	13		DC3	13		DC3	13	19
20		RES	14						9D	20
21		NL	15		LF	(0A)			85	21
22		BS	16		BS	08		BS	08	22
23		IL	17		NUL	(00)			87	23
24		CAN	18		CAN	18		CAN	18	24
25		EM	19		EM	19		EM	19	25
26		CC	1A						92	26
27		CU1	1B						8F	27
28		IFS	1C		FS	1C		FS	1C	28
29		IGS	1D		GS	1D		GS	1D	29
30		IRS	1E		RS	1E		RS	1E	30
31		IUS	1F		US	1F		US	1F	31
32		DS	20						80	32
33		SOS	21						81	33
34		FS	22						82	34
35		WUS	23						83	35
36		BYP	24						84	36
37		LF	25		LF	0A		LF	0A	37
38		ETB	26		ETB	17		ETB	17	38
39		ESC	27		ESC	1B		ESC	1B	39
40		SA	28						88	40
41			29						89	41
42		SM	2A						8A	42
43		CU2	2B						8B	43
44		MFA	2C						8C	44
45		ENQ	2D		ENQ	05		ENQ	05	45
46		ACK	2E		ACK	06		ACK	06	46
47		BEL	2F		BEL	07		BEL	07	47
48			30						90	48
49			31						91	49
50		SYN	32		SYN	16		SYN	16	50
51		IR	33						93	51
52		PN	34						94	52
53		RS	35						95	53
54		UC	36						96	54
55		EOT	37		EOT	04		EOT	04	55
56		SBS	38						98	56
57		IT	39						99	57
58		RFF	3A						9A	58
59		CU3	3B						9B	59
60		DC4	3C		DC4	14		DC4	14	60
61		NAK	3D		NAK	15		NAK	15	61
62			3E						9E	62
63		SUB	3F		SUB	1A		SUB	1A	63

Figure B-1 (Part 1 of 4). TCAM Internal and Device Codes

	Internal S/370 Code (EBCDIC)			USASCII			ASCII-8			
	Character			Character			Character		Code (Hex)	
	Graphic	Control		Graphic	Control		Graphic	Control		
Ref.	1	2	3	4	5	6	7	8	9	Ref.
64	SP		40	SP		20	SP		20	64
65			41						A0	65
66			42						A1	66
67			43						A2	67
68			44						A3	68
69			45						A4	69
70			46						A5	70
71			47						A6	71
72			48						A7	72
73			49						A8	73
74	¢		4A	[		5B	[		5B	74
75	.		4B	.		2E			2E	75
76	<		4C	<		3C	<		3C	76
77	(		4D	(		28	(		28	77
78	+		4E	+		2B			2B	78
79	|		4F	!		21	!		21	79
80	&		50	&		26	&		26	80
81			51						A9	81
82			52						AA	82
83			53						AB	83
84			54						AC	84
85			55						AD	85
86			56						AE	86
87			57						AF	87
88			58						B0	88
89			59						B1	89
90	!		5A	]		5D	]		5D	90
91	$		5B	$		24	$		24	91
92	*		5C	*		2A	*		2A	92
93	)		5D	)		29	)		29	93
94	;		5E	;		3B	;		3B	94
95	¬		5F	∧		5E	∧		5E	95
96	-		60	-		2D	-		2D	96
97	/		61	/		2F	/		2F	97
98			62						B2	98
99			63						B3	99
100			64						B4	100
101			65						B5	101
102			66						B6	102
103			67						B7	103
104			68						B8	104
105			69						B9	105
106	¦		6A	¦		7C	¦		7C	106
107	,		6B	,		2C	,		2C	107
108	%		6C	%		25	%		25	108
109	_		6D	_		5F	_		5F	109
110	>		6E	>		3E	>		3E	110
111	?		6F	?		3F	?		3F	111
112			70						BA	112
113			71						BB	113
114			72						BC	114
115			73						BD	115
116			74						BE	116
117			75						BF	117
118			76						C0	118
119			77						C1	119
120			78						C2	120
121	`		79	`		60	`		60	121
122	:		7A	:		3A	:		3A	122
123	#	EOA	7B	#		23	#		23	123
124	@		7C	@		40	@		40	124
125	'		7D	'		27	'		27	125
126	=		7E	=		3D	=		3D	126
127	"		7F	"		22	"		22	127

Figure B-1 (Part 2 of 4). TCAM Internal and Device Codes

	Internal S/370 Code (EBCDIC)			USASCII			ASCII-8			
	Character		Code (Hex)	Character		Code (Hex)	Character		Code (Hex)	
	Graphic	Control		Graphic	Control		Graphic	Control		
Ref.	1	2	3	4	5	6	7	8	9	Ref
128			80					DEL	C3	128
129	a		81	a		61	a		61	129
130	b		82	b		62	b		62	130
131	c		83	c		63	c		63	131
132	d		84	d		64	d		64	132
133	e		85	e		65	e		65	133
134	f		86	f		66	f		66	134
135	g		87	g		67	g		67	135
136	h		88	h		68	h		68	136
137	i		89	i		69	i		69	137
138			8A						C4	138
139			8B						C5	139
140			8C						C6	140
141			8D						C7	141
142			8E						C8	142
143			8F						C9	143
144			90						CA	144
145	j		91	j		6A	j		6A	145
146	k		92	k		6B	k		6B	146
147	l		93	l		6C	l		6C	147
148	m		94	m		6D	m		6D	148
149	n		95	n		6E	n		6E	149
150	o		96	o		6F	f		6F	150
151	p		97	p		70	p		70	151
152	q		98	q		71	q		71	152
153	r		99	r		72	r		72	153
154			9A						CB	154
155			9B						CC	155
156			9C						CD	156
157			9D						CE	157
158			9E						CF	158
159			9F						D0	159
160			A0						D1	160
161	~		A1	~		7E	~		7E	161
162	s		A2	s		73	s		73	162
163	t		A3	t		74	t		74	163
164	u		A4	u		75	u		75	164
165	v		A5	v		76	v		76	165
166	w		A6	w		77	w		77	166
167	x		A7	x		78	x		78	167
168	y		A8	y		79	y		79	168
169	z		A9	z		7A	z		7A	169
170			AA						D2	170
171			AB						D3	171
172			AC						D4	172
173			AD						D5	173
174			AE						D6	174
175			AF						D7	175
176			B0						D8	176
177			B1						D9	177
178			B2						DA	178
179			B3						DB	179
180			B4						DC	180
181			B5						DD	181
182			B6						DE	182
183			B7						DF	183
184			B8						E0	184
185			B9						E1	185
186			BA						E2	186
187			BB						E3	187
188			BC						E4	188
189			BD						E5	189
190			BE						E6	190
191			BF						E7	191

Figure B-1 (Part 3 of 4). TCAM Internal and Device Codes

Ref.	Internal S/370 Code (EBCDIC)			USASCII			ASCII-8			Ref.
	Character		Code (Hex)	Character		Code (Hex)	Character		Code (Hex)	
	Graphic	Control		Graphic	Control		Graphic	Control		
Ref.	1	2	3	4	5	6	7	8	9	Ref.
192	{	PZ	C0	{		7B	{		7B	192
193	A		C1	A		41	A		41	193
194	B		C2	B		42	B		42	194
195	C		C3	C		43	C		43	195
196	D		C4	D		44	D		44	196
197	E		C5	E		45	E		45	197
198	F		C6	F		46	F		46	198
199	G		C7	G		47	G		47	199
200	H		C8	H		48	H		48	200
201	I		C9	I		49	I		49	201
202			CA						E8	202
203			CB						E9	203
204	⑀		CC						EA	204
205			CD						EB	205
206	⑂		CE						EC	206
207			CF			5C			ED	207
208	}	MZ	D0	}		7D	}		7D	208
209	J		D1	J		4A	J		4A	209
210	K		D2	K		4B	K		4B	210
211	L		D3	L		4C	L		4C	211
212	M		D4	M		4D	M		4D	212
213	N		D5	N		4E	N		4E	213
214	O		D6	O		4F	O		4F	214
215	P		D7	P		50	P		50	215
216	Q		D8	Q		51	Q		51	216
217	R		D9	R		52	R		52	217
218			DA						EE	218
219			DB						EF	219
220			DC						F0	220
221			DD						F1	221
222			DE						F2	222
223			DF						F3	223
224	\	RM	E0	\		5C	\		5C	224
225			E1						9F	225
226	S		E2	S		53	S		53	226
227	T		E3	T		54	T		54	227
228	U		E4	U		55	U		55	228
229	V		E5	V		56	V		56	229
230	W		E6	W		57	W		57	230
231	X		E7	X		58	X		58	231
232	Y		E8	Y		59	Y		59	232
233	Z		E9	Z		5A	Z		5A	233
234			EA						F4	234
235			EB						F5	235
236	⑁		EC						F6	236
237			ED						F7	237
238			EE						F8	238
239			EF						F9	239
240	0		F0	0		30	0		30	240
241	1		F1	1		31	1		31	241
242	2		F2	2		32	2		32	242
243	3		F3	3		33	3		33	243
244	4		F4	4		34	4		34	244
245	5		F5	5		35	5		35	245
246	6		F6	6		36	6		36	246
247	7		F7	7		37	7		37	247
248	8		F8	8		38	8		38	248
249	9		F9	9		39	9		39	249
250	|		FA						FA	250
251			FB						FB	251
252			FC						FC	252
253			FD						FD	253
254			FE						FE	254
255			FF					EQ	FF	255

Figure B-1 (Part 4 of 4). TCAM Internal and Device Codes

' USASCII CODE

S/370 Byte (hex)	Graphic	Control	S/370 Byte (hex)	Graphic	Control	S/370 Byte (hex)	Graphic	Control	S/370 Byte (hex)	Graphic	Control
00		NUL	40			80			C0		
01		SOH	41	A		81			C1		
02		STX	42	B		82			C2		
03		ETX	43	C		83			C3		
04		EOT	44	D		84			C4		
05		ENQ	45	E		85			C5		
06		ACK	46	F		86			C6		
07		BEL	47	G		87			C7		
08		BS	48	H		88			C8		
09		HT	49	I		89			C9		
0A		LF	4A	J		8A			CA		
0B		VT	4B	K		8B			CB		
0C		FF	4C	L		8C			CC		
0D		CR	4D	M		8D			CD		
0E		SO	4E	N		8E			CE		
0F		SI	4F	O		8F			CF		
10		DLE	50	P		90			D0		
11		DC1	51	Q		91			D1		
12		DC2	52	R		92			D2		
13		DC3	53	S		93			D3		
14		DC4	54	T		94			D4		
15		NAK	55	U		95			D5		
16		SYN	56	V		96			D6		
17		ETB	57	W		97			D7		
18		CAN	58	X		98			D8		
19		EM	59	Y		99			D9		
1A		SUB	5A	Z		9A			DA		
1B		ESC	5B	[		9B			DB		
1C		FS	5C	\		9C			DC		
1D		GS	5D	!		9D			DD		
1E		RS	5E			9E			DE		
1F		US	5F	-		9F			DF		
20		SP	60	[		A0			E0		
21	¦		61	a		A1			E1		
22	"		62	b		A2			E2		
23	'		63	c		A3			E3		
24	$		64	d		A4			E4		
25	%		65	e		A5			E5		
26	&		66	f		A6			E6		
27	'		67	g		A7			E7		
28	(		68	h		A8			E8		
29	)		69	i		A9			E9		
2A	.		6A	j		AA			EA		
2B	+		6B	k		AB			EB		
2C	'		6C	l		AC			EC		
2D	-		6D	m		AD			ED		
2E	.		6E	n		AE			EE		
2F	.		6F	o		AF			EF		
30	0		70	p		B0			F0		
31	1		71	q		B1			F1		
32	2		72	r		B2			F2		
33	3		73	s		B3			F3		
34	4		74	t		B4			F4		
35	5		75	u		B5			F5		
36	6		76	v		B6			F6		
37	7		77	w		B7			F7		
38	8		78	x		B8			F8		
39	9		79	y		B9			F9		
3A	:		7A	z		BA			FA		
3B	;		7B	{		BB			FB		
3C	<		7C	|		BC			FC		
3D			7D	}		BD			FD		
3E	>		7E	~		BE			FE		
3F	?		7F		DEL	BF			FF		

Figure B-2. USASCII Code

ASCII - 8

S/370 Byte (hex)	Graphic	Control	S/370 Byte (hex)	Graphic	Control	S/370 Byte (hex)	Graphic	Control	S/370 Byte (hex)	Graphic	Control
00		NUL	40	@		80			C0		
01		SOH	41	A		81			C1		
02		STX	42	B		82			C2		
03		ETX	43	C		83			C3		
04		EOT	44	D		84			C4		
05		ENQ	45	E		85			C5		
06		ACK	46	F		86			C6		
07		BEL	47	G		87			C7		
08		BS	48	H		88			C8		
09		HT	49	I		89			C9		
0A		IF	4A	J		8A			CA		
0B		VT	4B	K		8B			CB		
0C		FF	4C	L		8C			CC		
0D		CR	4D	M		8D			CD		
0E		SO	4E	N		8E			CE		
0F		SI	4F	O		8F			CF		
10		DLE	50	P		90			D0		
11		DC1	51	Q		91			D1		
12		DC2	52	R		92			D2		
13		DC3	53	S		93			D3		
14		DC4	54	T		94			D4		
15		NAK	55	U		95			D5		
16		SYN	56	V		96			D6		
17		ETB	57	W		97			D7		
18		CAN	58	X		98			D8		
19		EM	59	Y		99			D9		
1A		SUB	5A	Z		9A			DA		
1B		ESC	5B	[		9B			DB		
1C		FS	5C	\		9C			DC		
1D		GS	5D	]		9D			DD		
1E		RS	5E	Λ		9E			DE		
1F		US	5F	–		9F			DF		
20		SP	60	‘		A0			E0		
21	!		61	a		A1			E1		
22	"		62	b		A2			E2		
23	#		63	c		A3			E3		
24	$		64	d		A4			E4		
25	%		65	e		A5			E5		
26	&		66	f		A6			E6		
27	'		67	g		A7			E7		
28	(		68	h		A8			E8		
29	)		69	i		A9			E9		
2A	*		6A	j		AA			EA		
2B	+		6B	k		AB			EB		
2C	,		6C	l		AC			EC		
2D	-		6D	m		AD			ED		
2E			6E	n		AE			EE		
2F	/		6F	o		AF			EF		
30	0		70	p		B0			F0		
31	1		71	q		B1			F1		
32	2		72	r		B2			F2		
33	3		73	s		B3			F3		
34	4		74	t		B4			F4		
35	5		75	u		B5			F5		
36	6		76	v		B6			F6		
37	7		77	w		B7			F7		
38	8		78	x		B8			F8		
39	9		79	y		B9			F9		
3A	:		7A	z		BA			FA		
3B	;		7B	{		BB			FB		
3C	<		7C	¦		BC			FC		
3D	=		7D	}		BD			FD		
3E	>		7E	~		BE			FE		
3F	?		7F		DEL	BF			FF		EO

Figure B-3. ASCII 8-bit Code

Appendix C. Internal TCAM Macros Available to the User

This appendix describes specialized macro instructions which are normally issued by TCAM code, but which are included in the TCAM macro library and may be invoked by the user.

If any TCAM macros are included in your application program, you must concatenate SYS1.TELCMLIB to your SYSLIB DD statement when you link-edit.

DATIM Macro

The DATIM macro:

- Edits input and current dates and times into a printable expression
- May be issued in the inheader, inbuffer, outheader, and outbuffer subgroups of an MH subgroup or in an initiator-attached or MCP-attached application program
- May be used more than once within a subgroup or in an initiator-attached or MCP-attached application program.

DATIM edits any combination of input and current dates and times into one expression in a printable format. If the input date or time is to be edited, it must be in the same format as contained in the FHP (FHPIDATB and FHPITIMB fields).

The input date is contained in two bytes, of which the first four bits are an index to the year (with 1979 as the base year), and the remaining 12 bits are the day of the year (expressed in Julian form). The Julian date is in packed decimal format with one digit contained in four bits. The input time is contained in two bytes in packed decimal format with one digit contained in four bits. The format is HHMM or hours, hours, minutes, minutes, and is based on a 24-hour clock (for example, 1420).

The input date or time to be edited may be passed to the DATIM routine through the FHP or DATIM work area. The third positional operand, FHP or LOCAL, specifies the location of the input date and time. If the current date or time is to be edited, the DATIM routine gets the current system date or time from the operating system before editing.

The user has the option to decide, either when the MCP is assembled or executed, which dates and times are to be edited by the DATIM macro. If the user knows at assembly which dates and times are to be so edited, he should code the keyword operands (INDATE=, INTIME=, CURDAT=, CURTIM=), but *not* code the second positional operand *(swlbl)*. If the user does not know at assembly which dates and times are to be edited, or if the specific dates and times to be edited will vary during execution, then the second positional operand *(swlbl)* must be coded but the keyword operands as specified above must *not* be coded. It is the user's responsibility to move the appropriate date and time indications into the *swlbl* code byte before the DATIM macro executes.

Note: Normally the user codes INSRTFLD and TCSENDBL macros when registering a date or time to be inserted into the message.

The user's registers are saved and restored by DATIM.

Supported Resources and General Requirements: ALL.

Valid subgroups: Inheader, inbuffer, outheader, outbuffer.

Dates and times not known:

NAME	OPERATION	OPERANDS
[symbol]	DATIM	wrklbl,swlbl [,{FHP }] {LOCAL}

Dates and times known:

NAME	OPERATION	OPERANDS
[symbol]	DATIM	wrklbl, [,{FHP }] {LOCAL} [,CURDAT={NO }] {YES} [,CURTIM={NO }] {YES} [,INDATE={NO }] {YES} [,INTIME={NO }] {YES}

```
CURDAT={NO }
       {YES}
```

Function: Specifies whether the current date is to be edited.

Format: NO or YES.

Default: NO.

```
CURTIM={NO }
       {YES}
```

Function: Specifies whether the current time is to be edited.

Format: NO or YES.

Default: NO.

```
{FHP  }
{LOCAL}
```

Function: Specifies the location of the input date and input time to be edited.

Format: FHP or LOCAL.

Default: FHP.

Note: This operand is meaningless if neither the input date nor the input time are to be edited.

FHP specifies that the input date and input time are located in the FHP fields, FHPIDATB and FHPITIMB. Register 6 must contain the buffer address of the header segment containing the FHP to be edited. The required buffer address is easily obtainable from the fullword field named IEDADBUF, which is addressable within the MCP.

LOCAL specifies that the input date and input time are located in the DATIM work area. (See the *wrklbl* operand explanation, function 2.) When DATIM is issued in an application, LOCAL must be specified because the message is longer in an TCAM buffer.

```
INDATE={NO }
       {YES}
```

Function: Specifies whether the input date is to be edited.

Format: NO or YES.

Default: NO.

```
INTIME={NO }
       {YES}
```

Function: Specifies whether the input date is to be edited.

Format: NO or YES.

Default: NO.

```
swlbl
```

Function: Specifies the name of a code byte generated by DATIM. Allows the user to refer to this code byte.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional but is ignored if any of the following operands is used: INDATE=, INTIME=, CURDAT=, CURTIM=.

swlbl must be a unique label within the assembly, so if DATIM is coded more than once in an assembly, the *swlbl* operands in each macro, if coded, must be different.

The *swlbl* operand allows the user to determine at execution which dates and times are to be edited by DATIM. The code byte is generated as a hexadecimal byte of zeros. Then, prior to executing DATIM, the user must turn on those bits in the code byte which represent the dates and times to be edited. The following chart shows the association of bits and options.

Bit	**Option**
X'08'	INDATE=YES
X'04'	INTIME=YES
X'02'	CURDAT=YES
X'01'	CURTIM=YES

symbol

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

wrklbl

Function: Specifies the name of the work area generated by DATIM. Allows the issuer to refer to the DATIM work area.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is required.

wrklbl must be a unique label within an assembly; so if DATIM is coded more than once in an assembly, the *wrklbl* operands in each macro must be different.

The user should refer to the work area for the following functions:

1. After the execution of DATIM, the DATIM work area contains the edited dates and times in one printable expression. The length of the expression in the work area is the first byte of the work area, and the date begins in the second byte of the work area.
2. If the LOCAL operand is specified and the input date and time are to be edited, the DATIM work area must contain the date and time before executing DATIM. If the input date is to be edited, it should be moved into *wrklbl*1 (for a length of two bytes). If the input time is to be edited, it should be moved into *wrklbl*3 (for a length of two bytes). The first area (the length of the byte) may be ignored.

Examples

1.

```
DATIM WORKAR,,,INTIME=YES
```

The input time, which is located in the FHP, is to be edited. (Register 6 contains the buffer address of the header segment of the message.)

The edited input time is returned in the DATIM work area at location WORKAR1 for a length specified in WORKAR.

2.

```
DATIM WORKAR2,SWITCH,LOCAL
```

The issuer indicates which dates and times are to be edited in the SWITCH code byte. If either the input time or date is to be edited, it has to be moved into the DATIM work area (WORKAR23 for time and WORKAR21 for date). After DATIM executes, the printable expression is returned to the user in the DATIM work area beginning at WORKAR21 for the length specified at WORKAR2

The following examples show various edited date and time expressions:

1. One date and one time:

   101579/1258

2. One date and two times:

   101579/1258-1300

3. Two dates and two times:

   101579/1758-101679/0851

Return Codes

None.

IEDLOCCB Macro

The IEDLOCCB macro provides a common mechanism to determine the address of the AVT or the TCX. TCAM uses either the CVT (CVTAQAVT) or the ASCB (ASCBTCME) for locating these control blocks.

The AVT is in TCAM's address space. You code AVT on the CBNAME operand if you are issuing the IEDLOCCB macro in the same TCAM address space. The TCX is in common storage (CSA) and can be addressed from any address space. In this case, you code CBNAME=TCX.

Supported Resources and General Requirements: ALL.

Valid subgroups: N/A.

NAME	OPERATION	OPERANDS

[symbol]	IEDLOCCB	CBNAME={AVT }
		{TCX }
		[,REG1=n]
		[,RESULT={<u>ADDR</u>}]
		{WORD}

```
CBNAME={AVT }
       {TCX }
```

Function: Specifies the control block that TCAM is to determine the address of.

Format: AVT or TCX.

Default: None. Specification is required.

```
REG1=n
```

Function: Specifies the register that TCAM uses to return the address of the desired control block.

Format: *n* is the number or symbolic name of a general register that is to receive the address of the requested control block.

Default: REG1=R1

Range: 1 to 15.

```
RESULT={ADDR}
       {WORD}
```

Function: Specifies whether the high-order byte of the register specified by REG1 is to be cleared.

Format: ADDR or WORD.

Default: RESULT = ADDR.

Note: ADDR specifies that the high-order byte is to be cleared. WORD specifies that the high-order byte will not be changed. The RESULT operand applies only when CBNAME = TCX is coded.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Return Codes

None.

IEDWTO Macro

The IEDWTO macro is an TCAM macro used for directing output in an TCAM environment. It is used in place of the execute form of the operating system WTO macro for routing messages to the TCAM console.

Supported Resources and General Requirements: All.

Valid subgroups: All.

NAME	OPERATION	OPERANDS

[symbol]	IEDWTO	`{((register))}`
		`{label        }`
		`[,AVTADDR={AVT address}]`
		`          {(register   )}`
		`[,TDHADDR={ATTH address}]`
		`          {(register)   }`

```
AVTADDR={AVT address}
        {(register) }
```

Function: Specifies the AVT address used to aid in locating the TASKDEF macro control block (ATTH) header.

Format: *AVT address* or *register* containing the AVT address. *AVT address* must be the 3-byte hexadecimal address of the AVT. *register* must be specified within parentheses.

Default: None. Specification is optional.

Range: *register* must be general register 1 through 15.

```
{label       }
{((register))}
```

Function: Sends the associated WTO macro list form to the console whose ID is present in the TASKDEF macro control block header field ATTHCNID. If the console ID at this location is zero, the values specified on the ROUTCDE parameter of the associated WTO macro list form are used to route the message. ROUTCDE parameters on the list form are ignored if a console ID is present in the TASKDEF macro control block header; however, if a ROUTCDE parameter of 5, 6, and/or 11 (representing the tape library, disk library, or programmer information, respectively) is specified, the list form message is routed to those consoles as well.

Format: *label* or *register*. *label* must conform to the rules for assembler language symbols. *register* must be specified within *double* parenthesis.

Default: None. Specification is required.

Range: *register* must be general register 1 through 15.

```
TDHADDR={ATTH address}
        {(register)  }
```

Function: Specifies the TASKDEF macro control block (ATTH) address.

Format: *ATTH address* or *register* containing the ATTH address. *ATTH address* must be the 3-byte hexadecimal address of the ATTH. *register* must be enclosed within parenthesis.

Default: None. Specification is optional.

Range: *register* must be general register 1 through 15.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Examples

```
     IEDWTO MESSAGE1
     IEDWTO ((REG7))
     IEDWTO MESSAGE2,AVTADDR=(RAVT)
     IEDWTO MESSAGE3,TDHADDR=ATTHDR
```

Return Codes

None.

IEDWTOR Macro

The IEDWTOR macro is an TCAM macro used for directing output in a TCAM environment. It is used in place of the execute form of the operating system WTOR macro for routing messages to and soliciting replies from the TCAM console.

Supported Resources and General Requirements: All.

Valid subgroups: All.

Name	Operation	Operands

`[symbol]`	`IEDWTOR`	`{((register))}`
		 `{label       }`
		 `[,AVTADDR={AVT address}]`
		 `          {(register )}`
		 `[,TDHADDR={ATTH address}]`
		 `          {(register)  }`

```
AVTADDR={AVT address}
        {(register) }
```

Function: Specifies the AVT address used to aid in locating the TASKDEF macro control block (ATTH) header.

Format: *AVT address* or *register* containing the AVT address. *AVT address* must be the 3-byte hexadecimal address of the AVT. *register* must be specified within parenthesis.

Default: None. Specification is optional.

Range: *register* must be general register 1 through 15.

```
{label       }
{((register))}
```

Function: Sends the associated WTOR macro list form to the console whose ID is present in the TASKDEF macro control block header field ATTHCNID. If the console ID at this location is zero, the values specified on the ROUTCDE parameter of the associated WTOR macro list form are used to route the message. ROUTCDE parameters on the list form are ignored if a console ID is present in the TASKDEF macro control block header; however, if a ROUTCDE parameter of 5, 6, and/or 11 (representing the tape library, disk library, or programmer information, respectively) is specified, the list form message is routed to those consoles as well.

Format: *label* or *register*. *label* must conform to the rules for assembler language symbols. *register* must be specified within *double* parenthesis.

Default: None. Specification is required.

Range: *register* must be general register 1 through 15.

```
TDHADDR={ATTH address}
        {(register)  }
```

Function: Specifies the TASKDEF macro control block (ATTH) address.

Format: *ATTH address* or *register* containing the ATTH address. *ATTH address* must be the 3-byte hexadecimal address of the ATTH. *register* must be specified as REG1 through REG15.

Default: None. Specification is optional.

Range: *register* must be general register 1 through 15.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Examples

```
     IEDWTOR MESSAGE1
     IEDWTOR ((REG7))
     IEDWTOR MESSAGE2,AVTADDR=(RAVT)
     IEDWTOR MESSAGE3,TDHADDR=ATTHDR
```

Return Codes

None.

LOOPCTL Macro

The LOOPCTL macro:

- Is optional in the inheader or outheader MH subgroups
- Controls message looping in an extended networking environment
- Is issued internally by the INODEMH macro in the internodal message handler and would not normally be issued by the user

When three or more host nodes are connected in a extended network, messages may be routed from one host node to another via intermediate host nodes. Each node attempts to send the message to the destination node via a primary extended route. If the primary extended route is unavailable, the MCP may be coded to attempt, automatically, to use a different extended route.

If several simultaneous failures occur, a message could begin looping among several host nodes.

The LOOPCTL macro tests a counter (FHPLOOPC) in the FHP control area built and maintained in the message buffer during its passage through the network. If a threshold limit value is exceeded, the user can initiate action to remove the message from the system (for example, queue it to a destination queue for display to a communications operator). The loop control counter can be characterized as a record of the number of times any message handler has routed the individual message.

Supported Resources and General Requirements: SNA, FHP.

Valid Subgroups: Inheader and outheader.

NAME	OPERATION	OPERANDS

[symbol]	LOOPCTL	NOLOOP=name
		[,THRESH={n }]
		{TCSINTR}

NOLOOP=name

Function: Specifies the name of the instruction to which LOOPCTL will pass control if *no* loop condition is detected for this message.

Format: *name* is the name of an assembler language instruction within this inheader or outheader subgroup of the MCP.

Default: None. Specification is required.

Note: If a loop condition is not detected, the message loop counter in the FHP is incremented by one and the NOLOOP= branch is taken.

symbol

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

```
THRESH={n       }
       {TCSINTR}
```

Function: Specifies the maximum number of message handlers by which the message may be routed before a loop condition is assumed to exist.

Format: *n* or TCSINTR. *n* is a decimal integer from 2 to 254.

Default: THRESH=TCSINTR

Note: If TCSINTR is specified, the value stipulated in the INTRO macro LOOPCNT= operand is used as the THRESH= value.

Return Codes

None.

SETRESRV Macro

The SETRESRV macro:

- Is internally issued by the INODEMH macro (GEN=INHDR) and normally is not explicitly issued
- Sets the field in the LCB which indicates the number of reserved bytes in the buffer
- Uses the FHPHEADP value to calculate the value to which it sets the LCB reserved byte count.

TCAM macros assume that if a fixed header prefix (FHP) is present, it has been built in reserved bytes in the message buffer. Since the FHP occupies reserved bytes, it is protected from alternation by operations such as MSGEDIT network, they are received as data and occupy the first data positions in the buffer. The SETRESRV macro initializes the reserve count field of the line control block to make it appear as though the positions occupied by the FHP were specified as reserved bytes.

Supported Resources and General Requirements: ALL, FHP.

Valid Subgroups: Inheader.

NAME	OPERATION	OPERANDS
[symbol]	SETRESRV	(no operands)

symbol

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Return Codes

None.

TCBINCNV Macro

The TCBINCNV macro:

- Converts a halfword binary number to the equivalent printable decimal value expressed as a 5-byte EBCDIC number with leading zeros (if any) replaced by the EBCDIC blank character (X'40')
- Converts a halfword binary number to the equivalent hexadecimal value expressed as a 4-byte EBCDIC field
- Converts an EBCDIC decimal number (up to 5 bytes long) to the equivalent halfword binary number
- May be issued in inblock, inbuffer, outheader, and outbuffer MH subgroup or in an application program.

Supported Resources and General Requirements: ALL.

Valid Subgroups: Inblock, inbuffer, outheader, outbuffer.

NAME	OPERATION	OPERANDS

[symbol]	TCBINCNV	CONVTO={DEC}
		{HEX}
		{BIN}
		,INPUT={(reg)}
		{name }
		,OUTPUT={(reg)}
		{name }

```
CONVTO={DEC}
       {HEX}
       {BIN}
```

Function: Specifies the type of conversion to be performed.

Format: DEC, HEX, or BIN.

Default: DEC.

Note:

DEC - converts a halfword binary number to the equivalent decimal value as a 5-byte EBCDIC number, with any leading zeros replaced by EBCDIC blank characters (X'40'), in the output area specified by the user.

HEX - Converts a halfword binary number to the equivalent hexadecimal value expressed as a 4-byte EBCDIC field in the output area specified by the user.

BIN - Converts an EBCDIC decimal numeric field to the equivalent halfword binary number.

```
INPUT={(reg)}
      {name }
```

Function: Provides the address of the field containing the input value.

Format: *(reg)* or *name (reg)* is a general register enclosed in parentheses and is specified as an explicit decimal integer from 2 through 12 or a symbol that has previously been equated to a decimal value from 2 through 12. It is the user's responsibility to assign a value that is within the prescribed limits. *name* must conform to the rules for assembler language symbols.

Default: None. Specification is required.

Note:

If CONVTO=DEC or CONVTO=HEX is specified, (reg) contains the address of the halfword-aligned area for binary input; if CONVTO=BIN is specified, *(reg)* contains an address pointer to the first EBCDIC decimal character of an input string to be converted from decimal to binary.

name is the name of a halfword-aligned area for binary input (if CONVTO=DEC or CONVTO=HEX is specified) or the name of an area CONVTO=BIN is specified).

If CONVTO=BIN, the input decimal number to be converted is considered to be delimited by the first nonnumeric character found, or by a maximum of 5 characters, whichever occurs first.

```
OUTPUT={(reg)}
       {name }
```

Function: Provides the address of an area that will contain the converted value.

Format: *(reg)* or *name*. *(reg)* is a general register enclosed in parentheses and is specified as an explicit decimal integer from 2 through 12 or as symbol that has previously been equated to a decimal value from 2 through 12. It is the user's responsibility to assign this operand a value within the prescribed limits. *name* must conform to the rules for assembler language symbols.

Default: None. Specification is required.

Note:

(reg) contains the beginning address of the area where the converted output is to be placed.

name for binary to decimal conversion is the name of a field 5 bytes long. *name* for binary to hexadecimal conversion is the name of a field 4 bytes long. *name* for decimal to binary conversion is the name of a halfword-aligned area.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is required.

Return Codes

None.

TCSEARCH Macro

The TCSEARCH macro:

- May be used to gather information about the current external session partner for use in MH processing. If issued in the inheader or inbuffer MH subgroups, the external session partner is the message origin; if issued in the outheader or outbuffer subgroups, the external session partner is the message destination.
- May be used to locate a control block or table entry for any of the following entities:
  - The current external LU
  - A specific group defined in this MCP or, successively, each group defined in this MCP
  - A specific external LU or, successively, each external LU
- When getting information about each successive entity of a particular type, or each successive TCAM terminal table (TNT) entry, the macro is usually executed in a loop. The found TNT entries are returned in alphabetical order.
- May also be used to perform a binary search of the node table (if the NODETABL macro was issued in the initialization section of the MCP), the routing key table (if the KEYTABLE macro was issued), the resource table (if the RESTABLE macro was issued), TNT, or a user table (if the user has previously built a table definition control block for the specified table).
- May be used to identify the type of any entry in the TCAM terminal name table (TNT), and branch to a user routine based on that type.
- May be specified in inheader, inbuffer, outheader, or outbuffer MH subgroups or in a separately assembled program.

Several formats are presented to show which operands apply to various forms of the SRCHTYP= operand. The formats can be shown as one. The operands that specify conditional branch points are listed following the other operands.

Supported Resources and General Requirements: ALL.

Valid Subgroups: Inheader, inbuffer, outheader, and outbuffer.

To locate a control block or table entry for the external session partner:

NAME	OPERATION	OPERANDS
[symbol]	TCSEARCH	ARG=CURRTERM
		[,ERROR=name]
		[,EXDATA={YES}]
		{LTD}

To locate a control block or table entry for a group:

NAME	OPERATION	OPERANDS
[symbol]	TCSEARCH	SRCHTYP=GROUP
		ARG={(reg)}
		{name }
		{NEXT }
		[,ANSWER={(reg)}]
		{name }
		[,ELSEGO=name]
		[,ERROR=name]
		[,EXDATA={YES}]
		{NO }
		[,GROUP=name]
		[,NOMORE=name]

To locate a control block or table entry for a logical unit:

NAME	OPERATION	OPERANDS
[symbol]	TCSEARCH	SRCHTYP=LUNT
		,ARG={(reg)}
		{name }
		{NEXT }
		[,ANSWER={(reg)}]
		{name }
		[,ELSEGO={name}]
		{NEXT}
		[,ERROR=name]
		[,EXDATA={YES}
		{NO }
		[,INGROUP={(reg)}]
		{name }
		[,NOMORE=name]
		[,OPTION={0 }]
		{n }
		{ALL }
		{(option1,option2,...)}

TCSEARCH

To identify the type of an entry in the terminal name table:

NAME	OPERATION	OPERANDS
[symbol]	TCSEARCH	SRCHTYP=TNT
		,ARG={(reg)}
		{name }
		{NEXT }
		[,ANSWER={(reg)}]
		{name }
		[,CASCLST=name]
		[,DISTLST=name]
		[,ELSEGO={name}]
		{NEXT}
		[,ERROR=name]
		[,EXDATA={YES}
		{NO }
		[,GETPROC=name]
		[,GROUP=name]
		[,LOGTYPE=name]
		[,LUNT=name]
		[,NOMORE=name]
		[,OPTION={0 }]
		{n }
		{ALL }
		{(option1,option2,...)}
		[,PUTPROC=name]
		[,UNKNOWN=name]

To perform a binary search of a node table, the routing key table or the resource table:

NAME	OPERATION	OPERANDS
[symbol]	TCSEARCH	SRCHTYP={KEY }
		{NODE }
		{RESOURCE}
		,ARG={(reg)}
		{name }
		{NEXT }
		[,ANSWER={(reg)}]
		{name }
		[,ELSEGO={name}]
		{NEXT}
		[,ERROR=name]
		[,EXTRAS={FIRST}
		{LAST }
		{NO }
		[,NOMORE=name]

To locate a user table definition control block:

Name	**Operation**	**Operands**
[symbol]	TCSEARCH	SRCHTYP={(reg)}
		{name }
		,ARG={(reg)}
		{name }
		{NEXT }
		[,ANSWER={(reg)}]
		{name }
		[,ELSEGO={name}]
		{NEXT}
		[,ERROR=name]
		[,EXTRAS={FIRST}
		{LAST }
		{NO }
		[,NOMORE=name]

To generate DSECTS describing the format of the table definition control block and answer area:

Name	**Operation**	**Operands**
[symbol]	TCSEARCH	DSECT={YES}
		{NO }

```
ANSWER={(reg)}
       {name }
```

Function: Specifies an area to contain the results of the search.

Format: *(reg) or name (reg)* is a general register enclosed in parentheses and can be specified as an explicit decimal integer from 2 through 12, or a symbol that has previously been equated to a decimal value from 2 through 12. *name* must conform to the rules for assembler language symbols.

Default: None. Required if ARG=NEXT or EXDATA=YES is specified. If ARG=CURRTERM is coded, ANSWER=IEDXOAF is assumed.

Note:

(reg) contains the address of a fullword-aligned main-storage area.

name is the symbolic name of a fullword-aligned main-storage area.

The "TCSEARCH Answer Area" section provides a detailed description of the size, format and content of the answer area.

```
ARG=CURRTERM
```

Function: This form of the TCSEARCH macro is used to provide consolidated terminal-table entry and option field data about the external session partner or application. The identity of the external session partner or application program depends on the MH subgroup in which TCSEARCH is issued: in inheader or inbuffer subgroups, it is the message originator, and in outheader or outbuffer subgroups, it is the message destination. The object of the search is a TCAM terminal name table entry. The ARG=CURRTERM form of TCSEARCH places the gathered information about the external session partner entry into the answer area called IEDXOAF that is built by the user-coded OPTION macros used to define option fields in TCAM. Information returned from TCSEARCH may be used by user code or as input to other macros (a FORWARD macro with the TTCIN= operand, for example, may route on the TNT offset returned by TCSEARCH).

Format: CURRTERM.

Default: None. Either ARG=CURRTERM or ARG=(reg), name, or NEXT must be specified.

Note: The user may *not* specify any other keyword operands in this format of TCSEARCH, except ERROR=name and EXDATA=YES or LTD. The processing that results from specification of ARG=CURRTERM, however, is as though the other operands had been coded as follows:

```
SRCHTYP=TNT,EXDATA=LTD,OPTION=ALL,ANSWER=IEDXOAF
```

The functional description of the OPTION macro in Chapter 2 discusses the IEDXOAF control section used as the answer area in this format of TSEARCH, and also provides coding examples for accessing the option field information returned in the answer area. The IEDXOAF control section conforms to the format of an extended TCSEARCH answer area, described in the section "The TCSEARCH Answer Area."

```
ARG={(reg)}
    {name }
    {NEXT }
```

Function: Identifies the search argument that the user wants to match in the table or control block chain being searched.

Format: name, (reg), or NEXT. *name* must conform to the rules for assembler language symbols. *(reg)* must be specified between framing parentheses, either as a decimal integer from 2 through 12, or as a symbol that has previously been equated to a value from 2 through 12.

Default: None. Specification is required unless ARG=CURRTERM is coded.

name is the symbolic name of a main storage field. Note that the name may not be NEXT. The contents of the main storage field can be:

- The actual search argument that the user wants to match with an entry in the table or control block chain specified by the SRCHTYP operand.
- Two bytes of hexadecimal zeros, followed by a 2-byte terminal name table (TNT) index. A TNT index may be specified for GROUP, LUNT, and TNT. *name* is assumed to contain a TNT index if the first two bytes are hexadecimal zeros.

The (reg) operand specifies a general register that contains one of the following fields in the low-order bytes:

- The 3-byte main storage address of a field containing the search argument
- A 2-byte terminal name table (TNT) index. A TNT index may be specified for SRCHTYP=GROUP, LUNT, or TNT. *(reg)* is assumed to contain a TNT index if the high-order two bytes of the specified general register contain hexadecimal zeros
- NEXT specifies that this macro call is being issued within a loop to locate each consecutive entry in the table specified, or each consecutive control block in the control block chain specified by SRCHTYP. The search proceeds sequentially and each macro call locates the next table entry or control block. A request for the *first* entry in the table or the *first* control block in a chain is indicated by placing a zero in the first byte of the answer area. It is the user's responsibility to ensure that the first byte of the answer area is zero before beginning a loop. Data maintained in the answer area is used to locate the next table entry or control block on subsequent calls; therefore, the user must ensure that the contents of the answer area is preserved unchanged between calls within the loop. After the last entry in the table or control block chain is located, the next call results in unsuccessful execution of the TCSEARCH macro, and a return code indicating "no more items" is set. The first byte of the answer area is reset to zero, except in the case where the SRCHTYP= operand is qualified by an INGROUP operand, and the answer area specified by ANSWER= is the same as the one specified by the INGROUP= operand.

  If the user provides the search argument in a main-storage field using either the ARG=*name* or ARG=*(reg)* operands, the argument supplied must have the same length as the field against which it is being matched. If the argument is shorter than the comparison field length, it must be padded with either space characters (X'40') or hexadecimal zeros. The pad character must be the same type as was used when the comparison field was defined.

```
DSECT={YES}
      {NO }
```

Function: Generates the dummy sections describing the format of the table definition control block and answer area.

Format: YES or NO.

Default: DSECT=NO.

Note: If YES is specified, no other keywords are permitted and the dummy sections describing the formats of the table definition control block and the answer area are generated.

```
EXDATA={NO }
       {YES}
       {LTD}
```

Function: Specifies whether to provide additional data, besides the address of the search object, when SRCHTYP=GROUP, LUNT, or TNT is specified, or when ARG=CURRTERM is specified.

Format: YES or NO.

Default: The default is conditional. If ARG=CURRTERM is specified, the default for this operand is LTD. For all other cases, the default is NO.

Note: NO indicates that only the address of the search object is to be returned in the answer area. Refer to the ANSWER= and SRCHTYP= operand descriptions for a discussion of data that is returned. The answer area may be four, eight, or twelve bytes long.

YES means the answer area contains several data items in addition to the address of the search object. Refer to the ANSWER= operand description for a discussion of data that is returned. The answer area must be at least 116 bytes long.

LTD may be specified *only* for ARG=CURRTERM. LTD means the answer area contains several data items related to the currently connected terminal, in addition to its TNT index. The ANSWER= operand description discusses the data that is returned.

EXDATA=YES is not allowed for SRCHTYP=*(reg), name,* KEY, NODE, or RESOURCE.

```
EXTRAS={FIRST}
       {LAST }
       {NO   }
```

Function: Specifies whether provision has been made for extra unsorted entries to be added to the table and, if there are extras, how to process them.

Format: FIRST, LAST, or NO.

Default: EXTRAS = NO.

Note: If FIRST is specified, TCSEARCH sequentially searches the extra entries *before* binary searching the main table.

If LAST is specified, the sequential search of the extra entries is performed *after* the binary search.

If NO is specified, extra entries are not scanned.

This operand is applicable only when searching the resource table (SRCHTYP = RESOURCE), or a user-defined table (SRCHTYP = *(reg)* or (*name)*. The associated table defined control block indicates whether any extra entries have been built for the table being searched. This facility accommodates user-supplied modification and addition of table entries without complete table reorganization.

```
INGROUP={(reg)}
        {name }
```

Function: Qualifies or limits the search for logical units to those associated with a specific GROUP definition.

Format: (reg) or *name* (reg) must be specified between framing parentheses either as a decimal integer from 2 through 12, or as a symbol that has previously been equated to a value from 2 through 12. *name* must conform to the rules for assembler language symbols.

Default: None. Specification is optional when SRCHTYP = LUNT is specified.

Notes:

1. *The (reg) suboperand specifies a general register that contains in the low-order 3 bytes the main-storage address of an answer area that was filled in be a previous TCSEARCH macro with the SRCHTYP = GROUP operand.*

2. *name is the symbolic name of answer area that was filled in by a previous TCSEARCH macro with the SRCHTYP = GROUP operand.*

3. *If the user codes a TCSEARCH loop to locate external LUs within a group and the INGROUP operand is specified, and if it is within another TCSEARCH loop to locate groups, the two TCSEARCH macros may specify the same answer area only if both specify EXDATA = YES.*

```
OPTION={0                  }
       {n                  }
       {ALL                }
       {(option1,option2,...)}
```

Function: If EXDATA=YES, this operand specifies the number of option-field addresses to return to the answer area and, optionally, which option field addresses to return.

Format: *n*, ALL, or *(option list)*. *n* is either a decimal integer from 0 to 255 or a symbol that has been previously equated to a decimal value from 0 to 255.

(option list) must conform to the rules for assembler language symbols and must be the name of an option field defined by an OPTION macro.

Default: OPTION=0.

Note: *n* is the number of option field addresses to return in the answer area; the addresses of the *n* first option fields defined in the MCP (for the found terminal table entry) are returned.

ALL specifies that the addresses of all the option fields that were defined in the MCP should be returned.

(option list) specifies the names of those options requested in the order in which their addresses should be returned in the answer area.

An answer area must be specified which is large enough to contain the requested number of option field addresses.

This operand is not allowed for SRCHTYP=GROUP, *(reg), name,* KEY, NODE, or RESOURCE.

```
SRCHTYP={GROUP   }
        {KEY     }
        {LUNT    }
        {name    }
        {NODE    }
        {RESOURCE}
        {TNT     }
        {(reg)   }
```

Function: Specifies the entity being searched for, or identifies the control block chain to be searched, or identifies a system table to be searched, or provides the address of a definition control block describing a user table to be assumed.

Format: GROUP, KEY, LUNT, *name,* NODE, RESOURCE, TNT, or *(reg)*. *name* must conform to the rules for assembler language symbols. *(reg)* must enclosed in parentheses and can be specified as an explicit decimal integer from 2 through 12 or a symbol that has been previously equated to a decimal value from 2 through 12.

Default: None. Specification is required unless ARG=CURRTERM is coded, in which case SRCHTYP=TNT is assumed.

Depending upon the value of the SRCHTYP= operand, one of the following TCAM system tables or chains of control blocks is to be searched.

If GROUP is specified, the object of the search is the TNT associated with the group designated by the ARG= operand. The search argument may be either an eight byte group name or a TNT index. If ARG=NEXT is specified, the search object is the TNT entry associated with the next group.

If KEY is specified, the object of the search is a key table entry designated by the ARG= operand. The search argument is the name of a routing key and is eight bytes long. If ARG=NEXT is specified, the search object is the next entry in the key table.

If LUNT is specified, the object of the search is the TNT entry defining the specific external LU designated by the ARG= operand. The search argument may be either an eight-byte LU name or a TNT index. If ARG=NEXT is specified, the search object is the next entry in the TNT which defines an LU.

name is the name of a user-table-definition control block.

If NODE is specified, the object of the search is a node-table entry designated by the ARG= operand. The search argument is the node identifier and is one-byte long. If ARG=NEXT is specified, the search object is the next entry in the node table.

If RESOURCE is specified, the object of the search is a resource table entry designated by the ARG= operand. The search argument is the resource identifier and two-bytes long. If ARG=NEXT is specified, the search object is the next entry in the resource table that defines an external LU.

If TNT is specified, the object of the search is the TNT entry designated by the ARG= operand. The search argument may be either an eight byte name or a TNT index. If ARG=NEXT is specified, the search object is the next entry in the TNT. INGROUP may be coded to locate external LUs in a particular group definition.

(reg) is a general register, 2 through 12, containing the address of a user table definition control block.

The format and contents of the table definition control block that the user must supply are described by the DKJBTABD dummy section.

Branch Operands

The following optional operands specify the names of conditional branch points to be taken depending upon the type of entity represented by the table entry or control block that is located. If no branch is specified for the type of entity that is located, the branch point specified by the ELSEGO= operand is taken. If ELSEGO= is not specified, control is returned to the next sequential instruction. The branch operands are listed in the order in which they are selected.

`CASCLST=name`

Condition: This branch is taken if the entity is located is a cascade list.

`DISTLST=name`

Condition: This branch is taken if the entity that is located is a distribution list.

`ELSEGO={name}`

{NEXT}

Condition: This operand specifies the default branch point to be given control where there is no specific branch point coded corresponding to the condition found.

If ELSEGO=*name* is specified, *name* is the symbolic address of the default branch point.

If ELSEGO=NEXT is specified the default branch point is back to the next iteration of this TCSEARCH macro. ELSEGO=NEXT may be specified only if ARG=NEXT is also specified.

If the ELSEGO= operand is omitted, the default branch point is the next sequential instruction after TCSEARCH.

`ERROR=name`

Condition: This branch is taken if an error occurred during the execution of the TCSEARCH macro or if no entry is found that matches the specified ARG=. Refer to the next section entitled "Return Codes" for additional information.

`GETPROC=name`

Condition: This branch is taken if the found entity is an application GET interface.

`GROUP=name`

Condition: This branch is taken if the found entity is a group entry.

`LOGTYPE=name`

Condition: This branch is taken if the found entity is a log device.

`LUNT=name`

Condition: This branch is taken if the found entity is an SNA logical unit.

`NOMORE=name`

Condition: Applies only when ARG=NEXT is specified. This branch is taken if all the table entries or control blocks for the specified SRCHTYP= have already been found and the previous execution of the TCSEARCH macro located the last one.

`PUTPROC=name`

Condition: This branch is taken if the found entity is an application PUT interface.

`UNKNOWN=name`

Condition: This branch is taken if the found entity is of an unknown type.

The TCSEARCH Answer Area

Answer Area Size

The size of the answer area is dependent upon the specification of four TCSEARCH operands: SRCHTYP=, ARG=, EXDATA=, and OPTION=. A 1-word answer area is required if EXDATA=NO and ARG=*(reg)* or *name* are specified. A 2-word answer area is required if SRCHTYP= specifies KEY, NODE, RESOURCE or a user table, and EXDATA=NO and ARG=NEXT are specified. A 3-word answer area is required for other values of SRCHTYP= with EXDATA=NO and ARG=NEXT coded. A 116-byte answer area is required if EXDATA=YES and OPTION=0 are specified.

An additional 4 bytes are required for each option field requested by coding the OPTION= operand (with a non-zero value).

It is the user's responsibility to ensure that the answer area specified by the ANSWER= operand is long enough to contain the data that TCSEARCH will provide.

Answer Area Format

The following description applies to all of the answer area sizes discussed above; a 1-word answer area contains only the first 4 bytes of data described below, and a 2-word answer area contains only the first 8 bytes.

In the case of an extended answer area (116 bytes or longer), different sections within the answer area are filled in, or not, depending upon the SRCHTYP= operand that was specified and the type of entity actually located by TCSEARCH.

The order of the SRCHTYP= operands in the following description is important; for each value of SRCHTYP, the corresponding section of the answer area is filled in, and other sections below it may be filled in, if applicable data is available for the type of entity located.

For all values of the SRCHTYP= operand:

Offset	Length In Bytes	Contents		
0	1	Flag Byte		
		Bit 0:	B'1'	This is a long answer area.
		Bit 2-3:		Reserved
		Bit 4:		ARG=NEXT and a GROUP, TNT, NODE, KEY, RESOURCE, *(reg)* or *name* loop is in process.
		Bit 7:	Bit'1'	ARG=NEXT and a LUNT loop is in process.
1	3	The address of the found entity: • If SRCHTYP=GROUP, the address of the TNT entry for the group or the address of the DCB entry for the GROUP. • If SRCHTYP=TNT, the address of the TNT entry, specified by ARG=. • If SRCHTYP=NODE, KEY, RESOURCE, *(reg)*, or *name,* the address of the node-table, key-table or resource-table entry of the user table entry specified by ARG=.		
4	4	Reserved		

For a 3-word answer area:

Offset	Length In Bytes	Contents
8	4	The answer of the SIT entry corresponding to the found TNT entry.

For SRCHTYP=LUNT:

Offset	Length In Bytes	Contents
8	1	The length of the external LU name.
9	3	The address of the TNT entry.
12	2	The index of the TNT entry.
14	3	The address of the queue control block (QCB).
17	3	The address of the terminal table entry (TTE).
20	8	The name of the external LU.

For SRCHTYP=GROUP:

Offset	Length In Bytes	Contents
28	1	The length of the group name.
29	3	The address of the TNT entry.
32	2	The index of the TNT entry.
34	2	Reserved.
36	1	Unused.

Offset	Length In Bytes	Contents
37	3	The address of the terminal table entry (TTE), which, in this case, is an LGB.
40	8	The name of the group.

For all values of the SRCHTYP=operand:

Offset	Length In Bytes	Contents
48	3	Address of the SIT entry corresponding to the found TNT entry.
51	1	The number of option fields initialized for anything in the TNT, which will always be less than or equal to the value specified or implied by the OPTION= operand.
52-115		Reserved.

For a non-zero OPTION= value:

Offset	Length In Bytes	Contents
116	4n	For OPTION=*n,* a list of *n* 4-byte option field address(es), associated with the found terminal table entry. The first *n* option fields defined in this MCP are returned in the answer area. A zero will be supplied instead of an address for an option field which is not defined in the MCP or not initialized for this station. If OPTION=*(list)* is specified, only the address of the *n* option fields named in the list are returned in the order in which they are named.

The following list specifies a value of the SRCHTYP= oeprand for each conditional branch operand. This SRCHTYP= value indicates one or more sections of an extended answer area that are to be filled in (if EXDATA=YES) when that branch condition occurs.

Branch Operand	Answer Area Section
LOGTYPE	LUNT portion of the answer area
DISTLST	LUNT portion of the answer area
CASCLST	LUNT portion of the answer area
GROUP	GROUP
PUTPROC	LUNT portion of the answer area
GETPROC	LUNT portion of the answer area
LUNT	LUNT GROUP

Note: If EXDATA=LTD (for ARG=CURRTERM), only the TERM portion of the answer area is filled in.

Examples

1. Find the data for the current external LU:

```
        .
        .
        TCSEARCH ARG=CURRTERM,ERROR=NOGOOD
        .
        .
```

2. Find the data for a particular external LU:

```
        .
        .
        TCSEARCH ANSWER=AREA,SRCHTYP=LUNT,ARG=THETERM,            X
                 EXDATA=YES,OPTION=(TCSOPTS,ALTDEST),             X
                 ERROR=ERROR
        .
        .
THETERM DC C'CHICAGO' TERMINAL NAME
AREA    DC 31F'0'     ANSWER AREA
```

3. Locate each node-table entry:

```
        .
        .
        MVI AREA,0             INDICATE BEGINNING OF LOOP
LOOP    TCSEARCH ANSWER=AREA,SRCHTYP=NODE,ARG=NEXT,               X
                 ERROR=ERROR
        B        LOOP          MORE ENTRIES
        .
        .
AREA   DC       2F'0' ANSWER AREA
```

4. Find a particular group:

```
        .
        .
        LA       R2,=C'GROUP12'
        TCSEARCH ANSWER=AREA,SRCHTYP=GROUP,ARG=(R2),              X
                 ERROR=ERROR
AREA    DS F ANSWER AREA
```

5. Find all group and all LUNTs with each group:

```
        .
        .
        MVI        ANSWER,0         INDICATE START OF LOOP
GROUP   TCSEARCH   SRCHTYPE=GROUP,ANSWER=ANSWER,ARG=NEXT,         X
                   EXDATA=YES,NOMORE=YES,ERROR=ERROR
        .
        .
LUNT    TCSEARCH   SRCHTYPE=LUNT,ARG=NEXT,ANSWER=ANSWER,          X
                   EXDATA=YES,INGROUP=ANSWER,                     X
                   NOMORE=GROUP,ERROR=ERROR,
        .
        .
        B          LUNT             GET NEXT LUNT IN GROUP
ANSWER  DS         AL116            ANSWER AREA
        .
        .
```

```
6.  Find all 'single' type stations:
         .
         .
         TCSEARCH    ARG=NEXT,SRCHTYP=TNT,ANSWER=ANSWER,          X
                     EXDATA=YES,ERROR=ERROR,                      X
                     LUNT=HANDLE,                                 X
                     OPTION=4,                                    X
                     ELSEGO=NEXT
         .
         .
ANSWER   DC          33F'0'

7.  Process all items named in TNT, except GET and
    PUT TPROCESS entries:
         .
         .
LOOP     TCSEARCH    ARG=NEXT,SRCHTYP=TNT,ANSWER=ANSWER,          X
                     EXDATA=YES,ERROR=ERROR,                      X
                     PUTPROC=LOOP,                                X
                     GETPROC=LOOP,                                X
                     ELSEGO=HANDLE
         .
         .
```

Note: The TABLE= and TDATA= operands of TCSEARCH are supported for migration purposes. The answer area generated by a TCSEARCH macro with these operands will be compatible with the format supported by the previous version of the macro. However, use of these operands in new code is not recommended.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	Successful execution.
X'00000004'	No match found.
X'00000008'	SRCHTYP=TNT, KEY, NODE, or RESOURCE specified and table definition control block address not found in the TVT indicating one of the following errors: • INTRO, KEYTABLE, NODETABL, or RESTABLE macro not executed at initialization • No table definition control block found for user table • Register form of operand specified and register contained zeros
X'0000000C'	End-of-loop condition indicated when ARG=NEXT specified; no more items. All table entries or control blocks specified by SRCHTYP= have already been found and the previous execution of TCSEARCH macro located the last one.
X'00000010'	Results are unreliable.
X'00000014'	SRCHTYP, GROUP, or LUNT specified and table entry or control block matching the search argument was found, but was not of the type specified by SRCHTYP=. The first word of the answer area is filled with address of the terminal name table entry that matches the argument but is of the wrong type. Extended answer area not filled in.

TCSSORT Macro

The TCSSORT macro:

- Internally sorts a user-specified main-storage-resident table for which a table definition control block has been constructed
- May be specified either in an MCP or in a separately assembled program; normal OS/VS register-saving conventions are followed
- May optionally be issued one or more times

Supported Resources and General Requirements: ALL

Valid Subgroups: Inheader, inbuffer, outheader, and outbuffer.

NAME	OPERATION	OPERANDS

[symbol]	TCSSORT	[DSECT={<u>NO</u> }]
		{YES}
		[TABLE={(reg)}]
		{name }

```
DSECT={NO }
      {YES}
```

Function: Specifies if the dummy section describing the table-definition control block is to be generated.

Format: YES or NO.

Default: DSECT=NO

Note: If YES is specified, the other operand is ignored and the dummy section describing the format of the table definition control block is generated.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

```
TABLE={(reg)}
      {name }
```

Function: Provides the address of the table definition control block for the table to be sorted. The DKJSORTD dummy section describes the required format of a table definition control block.

Format: *(reg)* or *name*. *(reg)* must be enclosed in parentheses and can be specified as an explicit decimal integer from 2 through 12 or as a symbol that has been previously equated to a decimal value from 2 through 12. *name* must conform to the rules for assembler language symbol

Default: None. Specification is optional.

Note: *(reg)* is a general register, 2 through 12, that contains the address of the table definition control block.

name is the name of the control block. A table definition control block defining a table for the TCSEARCH macro—the binary search facility—is also a valid input to the TCSSORT macro; they are compatible.

Return Codes

None.

Appendix D. Aids for TCS-Brokerage Users

The option field and macros described in this section facilitate migration from TCS-Brokerage to TCAM.

TCSBKG Option Field

This option field is optional. If provided, it should be two bytes long, and the contents are as follows:

Byte 0:	Bits 0-1	Reserved; set=0
	Bit 2	If set=1, this external LU can enter broadcast mode messages.
	Bit 3	If set=1, this is a floor external LU.
	Bit 4	If set=1, this external LU is kept in its assembled mode.
	Bit 5	If set=1, this external LU can enter orders.
	Bit 6	Set by TCS to indicate that a KAB or skipped sequence message should be sent. The user should initialize this bit to 0.
	Bit 7	If set=1, this external LU can only accept messages originated by an application program

The contents of the first byte should be the hexadecimal representation of the binary total of applicable bits as described above.

Byte 1:	The implied mode for any messages entered from the external LU, unless the message itself contains a mode character

The contents of the second byte should be specified as the hexadecimal representation of a character used by the MODEFLD macro.

Base Year For FHP Dates In yddd Format

Computing the current year from the base year for TCS (1970) and TCAM (1979) is performed as follows:

To determine the current year, a value that is to be added to the 4-bit year value in the FHP (the year's last two digits only) is found in the halfword at offset X'04' in the TVT. Its table name is TVTBASYR. For the base year 1979, its value is X'004F' and for the base year 1970, its value is X'0046'. This value is established by the internal macro $TSCINTR. It is used by all TCAM functions that build, update, or use the FHP date field and the current message handling date field in the TVT. It is also used by TCAM functions that format dates in the *yddd* format.

No support is provided in TCAM to change the base year since no checking is performed to ensure that the base year falls within permissible limits, is the same before and after a restart, is the same on all systems in an extended network. However, TCAM FHP-based date support is designed to function correctly with any base year value (00 to 99) that is equivalent to the current year or any of the 15 prior years, century crossing included, until March 1, 2100 (when leap year calculations will fail).

DESTAB Macro

The DESTAB macro:

- Names a pseudo-destination table that may contain the names of pseudo-destination (that is, not terminal-table entries), the names of corresponding real destinations, and user routine names, as specified by subsequent REALDEST macro instructions
- Is required for each different name used in the TABLE= operand of all DESTFLD macro instructions
- Is optional in any nonexecuted section of the MCP, but may not be coded before the INTRO macro instructions
- Must be followed immediately by at least one REALDEST macro instruction
- May be issued more than once.

Supported Resources and General Requirements: Not applicable.

Valid subgroups: Not applicable.

NAME	OPERATION	OPERANDS
tabname	DESTAB	(no operands)

`tabname`

Function: Specifies the name of the pseudo-destination table.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is required.

Return Codes

None.

DESTFLD Macro

The DESTFLD macro:

- Validates destination fields in the message header
- Allows use of the pseudo-destination names (that are converted to real destination names)
- Is used as defined in Chapter 2, except for the additional operands shown later in this macro description.

The DESTFLD macro does destination scanning, table lookups, validity checking, and queue scheduling for destinations that are in the same TCAM system. The KEYPROC, DAFROUTE, or FORWARD macros should be used for routing to destinations on other TCAM systems.

The DESTFLD macro provides a facility for forwarding the message, based on a table search for a pseudo-destination name field, if one appears in the message. As an example of the practical use of this facility, it can control message routing to an application program.

This table search can be used to permit a user-coded routine to selectively route messages, based upon user analysis of their content. See the REALDEST macro instruction in this appendix for complete information about the requirements on the user-coded routine.

If a table is referred to by the DESTFLD macro, it is searched to determine the real destination of a message, based on the contents of a message header destination field. If the destination in the message header is a pseudo-destination (that is, not an entry in the terminal table), the pseudo-destination table referred to must provide a real destination for the message, or the message is rejected. If the destination in the message header is a real destination (that is, an entry in the terminal table), a pseudo-destination table may or may not be referenced. If a table is referred to, the table search may provide a real destination for the message; if it does not, the destination in the message header is used as the real message destination.

All real destinations specified in the message header are matched against the physical origin of the message; if any destination *and* the origin are stock exchange floor stations (as indicated in the TCSBKG option field), the message is rejected.

The message may also be rejected if it does not come from an application program; any real destination specified in the message header can be specified (in the TCSBKG option field) to accept only messages originated by an application program.

Supported Resources and General Requirements: ALL.

Valid Subgroups: Inheader.

NAME	OPERATION	OPERANDS
[symbol]	DESTFLD	[REAL={YES}] {NO } [,TABLE=name] [,TLIST={YES}] {NO }

```
REAL={YES}
     {NO }
```

Function: Specifies whether the message header contains the name of a real destination or a pseudo-destination.

Format: YES or NO.

Default: REAL=YES

YES indicates that the destination is real and is a terminal-table entry. NO indicates that the destination is not real, but is, rather, a pseudo-destination.

If REAL=NO is coded, the TABLE= operand is required; if more than one destination is used in the message or if a destination in the message is not found in the referenced table, the message is rejected.

If REAL=YES is coded, or if this operand is omitted and the TABLE= operand:

- Is not coded, the destinations must be real or the message is rejected
- Is coded, but the destination in the message is not found in the referenced table, then the destination in the message must be real, or the message is rejected

```
TABLE=name
```

Function: Specifies the name of a pseudo-destination table to be used for determining the real destination of the message.

Format: *name* must conform to the rules for assembler language symbols.

Default: None. Required if REAL=NO is coded; otherwise it is optional.

Note:

name is the name of the DESTAB macro for the table being referenced.

If this operand is omitted, it is assumed that all destinations in the message are real. If this operand is coded, any EOA=operand is ignored; thus the destination in the message must be a single entry followed by a blank.

```
TLIST={YES}
      {NO }
```

Function: Indicates that any destination specified in the header may be the name of a distribution list.

Format: YES or NO.

Default: TLIST=NO.

Return Codes

After unsuccessful completion of DESTFLD processing, register 15 is set to a return code of X'00000020' if the destination station does not have a TCSBKG or TCSOPTS option field associated with it.

FHPBUILD Macro

The FHPBUILD macro:

- Constructs and initializes a fixed header prefix and inserts it into the message header
- Is used as defined in Chapter 2, except for the additional operand shown below.

The FHPBUILD macro instruction constructs and initializes the fixed header prefix within the reserved section of the header buffer.

The macro determines the offset to the end of the message or the end of the first buffer unit—whichever is found first—and uses this determination as the offset to the end of the header for insertion in the FHP. This occurs as a default in the case of an omitted MODEFLD macro that normally determines the offset to the real end of the header.

The FHPBUILD macro requires that the message mode character, if present, be the first nonblank character after the current position of the scan pointer when the macro is executed. At least one blank must follow the mode character. FHPBUILD processing skips over all preceding blanks.

After successful completion of FHPBUILD processing, the scan pointer is:

- Located at the space character following the mode character if MODE=YES is specified and the mode character was not inserted in the FHP before execution
- Not moved if MODE=NONE is specified or the mode character was already inserted in the FHP before execution

Supported Resources and General Requirements: ALL.

Valid subgroups: Inheader.

NAME	OPERATION	OPERANDS
[symbol]	FHPBUILD	[MODE={YES }] {NONE}

```
MODE={YES }
     {NONE}
```

Function: Indicates if the mode byte (FHPMODEA) should be initialized in the FHP.

Format: YES or NONE

Default: MODE=NONE

Return Codes

See Chapter 2.

INSRTFLD Macro

The INSRTFLD macro:

- Is issued to schedule the insertion of data in the outgoing message header
- Is used as defined in Chapter 2, except for the additional operand shown below.

The INSRTFLD macro is used within the inheader subgroup of an MH to schedule the insertion of specific data into the outgoing message header.

Supported Resources and General Requirements: ALL, FHP.

Valid subgroups: Inheader.

NAME	OPERATION	OPERANDS

[symbol]	INSRTFLD	TRAILER={NO }
		{YES}

```
TRAILER={NO }
        {YES}
```

Function: Indicates that a standardized security order trailer should be constructed by the TCSENDBL macro when the outheader subgroup is executed.

Format: NO or YES.

Default: TRAILER=NO.

Note: The format of the inserted trailer is described under the TCSENDBL macro instruction.

Return Codes

See Chapter 2.

MODEFLD Macro

The MODEFLD macro:

- Causes dynamic variation of the path of a message through an MH
- Changes the mode of the origin to the mode used in the message, through an MH
- Finds the end of the message header (start of the message text)
- Checks that the message header does not extend beyond the end of the first buffer unit
- Checks that messages that are security orders are entered from an appropriate external LU
- Checks that broadcast messages are entered from an appropriate external LU
- May be issued more than once

The message-mode character, which is assumed to be located in the FHP (placed there either by the FHPBUILD macro instruction or by user code), is compared with the character specified in the MODEFLD macro itself.

If the mode character does not match, all subsequent instructions up to the next MODEFLD macro or delimiter macro—whichever comes first—are not executed. The last functional macro in any inheader subgroup using MODEFLD should be a MODEFLD macro with MODE=BAD specified, that sets an error code.

If the macro is coded with ROUTONL=YES, or if the mode character does not match, processing by MODEFLD is now completed.

When other functions are to be performed, the remaining operands specify what should be done. If SETMODE=YES is coded (or the operand is omitted), the message-mode character is inserted into the TCSBKG option field for the message origin as the new external LU mode; this occurs only if the TCSBKG option field indicates that the external LU is subject to station mode changes. This function permits the station operator to omit the mode character in subsequent messages of the same type.

The macro instruction is not dependent on the location of the scan pointer, although the scan pointer normally would be at the start of the header. The position of the scan pointer is not changed by this macro instruction. Depending on the specification in the macro instruction, the header may be delimited by a fixed number of header fields, by a specified character, or by a character string (the first time that the character or character string is encountered in the message, or a particular subsequent appearance); or it may consist of an exact number of characters; or it may be assumed to stretch exactly to the end of the first buffer unit. The offset to the end of the message header (start of message text) is inserted into the FHP (FHPTEXTP). This offset is used by the following macro instructions if they are coded as indicated:

DESTFLD	EOA=EOH
ORGFLD	DELEOH=YES
SEQFLD	DELEOH=YES
SETSCAN	SKIPFLD=LAST

If the pointer to the end of the message header is not set by MODEFLD processing or by user code before any one of the above instructions is executed, then the offset to the end of the header is used as it was initialized by the FHPBUILD macro; namely, it points to the end of the message or to the end of the first buffer unit, whichever comes first.

A MODEFLD macro that specifies the mode character used in security orders should be coded with a TEST=ORDER operand. This ensures that the order will be rejected if it is entered from any station not allowed to enter security orders, according to the TCSBKG option field as specified in the OPDATA= operand of the TERMINAL macro.

Likewise, a MODEFLD macro that specified the mode character used in broadcast messages should be coded with a TEST=BCAST operand. This ensures that the message is rejected if it is entered from any station not allowed to enter broadcast messages, according to the TCSBKG option field as specified in the OPDATA= operand of the TERMINAL macro.

Supported Resources and General Requirements: ALL, FHP.

Valid subgroups: Inheader.

NAME	OPERATION	OPERANDS

[symbol]	MODEFLD	DEL={chars}
		{EOU }
		{fld }
		,MODE={char}
		{BAD }
		,SETTEXT=num
		[,MAXSIZE={NUM}]
		{35 }
		[,ROUTONL={YES}]
		{NO }
		[,SETMODE={YES}
		{NO }
		[,TEST={BCAST}]
		{ORDER}
		[,TIMES={num}]
		{1 }

```
DEL={chars}
    {EOU  }
    {fld  }
```

Function: Specifies the end of the message header and the start of the message text.

Format: chars, EOU, or *fld. chars* is a character or character string composed of from one to eight characters in either character or hexadecimal notation; *chars* must be framed with either C'' or X'' characters. *fld* is an unframed decimal integer from 1 to 100.

Default: None. Specification is required unless ROUTONL=YES, MODE=BAD, or SETTEXT=*num* is coded.

Note:

chars specifies that the header is delimited by the specified character or string of characters. Unless the TIMES= operand is coded, the header is considered to end at the first appearance of the character or character string in the header.

EOU specified that the message contains no header distinct from text. The end-of-header pointer (FHPTEXTP) is set to the last data position in the first buffer unit.

fld specifies the maximum length of the header expressed in number of fields. A field is presumed to be any number of nonblank characters delimited by one or more blank characters.

```
MAXSIZE={num}
        {35 }
```

Function: Specifies the maximum number of characters allowed in a message header.

Format: *num* is an unframed decimal integer: minimum 1, maximum 203.

Default: MAXSIZE=35.

Note: This operand is ignored unless DEL=*fld* or DEL=*chars* is coded.

```
MODE={char}
     {BAD }
```

Function: Controls the path of a message through an MH by checking the message mode field in its FHP.

Format: *char* or BAD. *char* is one character in either character or hexadecimal notation and must be framed with C″ or X″ characters.

Default: None. Either MODE=*char* or MODE=BAD must be specified.

Note:

char specifies the character to be compared with the message mode field in the FHP. If the specified character matches the field in the FHP, all instructions between this MODEFLD macro and the next MODEFLD macro or delimiter macro are executed. If the specified character does not match the FHP mode field, control is passed immediately to the next MODEFLD macro or delimiter macro.

BAD specifies that the message mode field in the FHP is invalid; that is, it could not be matched to any mode character specified in a MODEFLD macro executed previously in the MH path. A MODEFLD macro coded with MODE=BAD sets an error return code for invalid mode.

The last instruction to be executed in an inheader subgroup that uses any MODEFLD macros should normally be a MODEFLD macro specifying MODE=BAD.

```
ROUTONL= {YES}
         {NO }
```

Function: Specifies whether MODEFLD is to be used for path selection only.

Format: YES or NO.

Default: ROUTONL=NO.

Note:

YES specifies that MODEFLD processing is not to set the FHP pointer to the end of the message header and that the macro is to be used only for the mode character match for path selection within the MH. The end of header must be set by user code or by other means. ROUTONL=YES is invalid if the DEL= or SETTEXT= operand is used or if MODE=BAD is coded.

NO specifies that other MODEFLD functions besides path selection are to be performed.

```
SETMODE={YES}
        {NO }
```

Function: Specifies whether the station mode entry in the station TCSBKG option field should be set to the mode in the processed FHP.

Format: YES or NO.

Default: SETMODE=YES.

Note: YES indicates that the station mode is to be set to the mode in the processed message FHP if the station mode may change according to specification in bit four of the first byte of the TCSBKG option field. NO indicates that the station mode is to remain unchanged. An MH for an application program would probably be coded with SETMODE=NO.

`SETTEXT=num`

Function: Indicates that the header is to be delimited by the number of positions specified.

Format: *num* is an unframed decimal integer: minimum 1, maximum 202.

Default: None. Specification is required unless ROUTONL=YES or MODE=BAD are coded or the DEL= operand is used; in these cases, this operand is not allowed.

Note: If *num* is specified, the end-of-header pointer (FHPTEXTP—also referred to as the text pointer) is set either to the number of positions specified, beginning with the first nonblank position beyond the current scan pointer setting, or to the last byte of data in the first buffer unit—whichever is first.

`symbol`

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

```
TEST={ORDER}
     {BCAST}
```

Function: Specifies that the message handled by this macro instruction is a security order (TEST=ORDER) or a broadcast message (TEST=BCAST), and that it may be entered only by a station allowed to do so.

Format: ORDER or BCAST.

Default: None. Specification is optional.

Note:

This operand is ignored if ROUTONL=YES or MODE=BAD is coded.

If TEST=ORDER is specified, a return code of X'32' is set—provided that bit five of the first byte of the TCSBKG option field for the physical origin station is not on. If TEST=BCAST is specified, a return code of X'36' is set—provided that bit two of the first byte of the TCSBKG option field for the physical origin field is not on.

```
TIMES={num}
      {1  }
```

Function: Specifies the number of times that a given character string must be encountered in a message before the header is delimited. If, as an example, the DEL= operand is coded DEL=C'AB' and TIMES=3 is also specified, then the header is delimited the *third* time the character string AB is found in the message.

Format: *num* is an unframed decimal integer: minimum 1, maximum 10.

Default: TIMES=1.

Note: This operand is valid only if DEL=*chars* is coded.

Return Codes

Code	Meaning
X'00000000'	Successful execution
X'00000004'	No data in buffer
X'00000008'	Scan pointer points to data location outside the first buffer unit
X'0000000C'	Either: • Not the first buffer of a message • Duplicate buffer
X'00000010'	No FHP in message buffer
X'00000014'	End of header not found within the first message buffer unit, or within the maximum number of characters allowed in header as specified by the MAXSIZE= parameter
X'00000018'	Message origin not indicated in TCAM message buffer prefix
X'0000001C'	Mode character invalid (MODEFLD with MODE=BAD executed)
X'00000020'	Security order entered from unauthorized station
X'00000024'	Broadcast message entered from unauthorized station
X'FFFFFFFC'	Scan pointer points beyond message data in buffer

ORGFLD Macro

The ORGFLD macro:

- Validates the origin field in the message header
- Requires the presence of an FHP in the header being processed
- Will execute correctly only when the scan pointer is located within the first buffer unit of the first buffer containing the message
- Provides specialized support for brokerage applications but may be used to process any message containing an FHP
- May be issued more than once but should be executed only once for any particular message
- Requires a SEQFLD macro to be coded within the MH.

The ORGFLD macro expects the scan pointer to be set to the last byte of the field preceding the origin field or to a space character preceding the origin field in the message header. The origin field must be located in the first buffer unit.

If TRUST=YES is coded, a message may be entered with an origin field in the message header which specifies an origin other than the name of the station from which the message is entered. The specified origin is considered to be the origin for logical matters such as sequence number validation, while the physical station whose TCAM network address is in the TOAF field of the FHP is considered to be the origin for matters such as error message routing. The origin specified in the header must be the name of a station or LU defined by a TERMINAL macro.

If a SEQFLD macro instruction is executed previous to this macro, the sequence number validation has been postponed and takes place when the ORGFLD macro executes. Where the validation takes place is not relevant to you unless you choose to issue additional code that would be affected by this validation process.

After successful execution of the ORGFLD macro, register 15 is set to 0 and the scan pointer is advanced to the last character of the origin field in the message header. If ORGFLD does not execute, the location of the scan pointer is unpredictable and register 15 is set as shown in "Return Codes".

Supported Resources and General Requirements: ALL, FHP.

Valid subgroups: Inheader.

NAME	OPERATION	OPERANDS

[symbol]	ORGFLD	[DELCHAR={<u>NONE</u>}]
		{char}
		[,DELEOH={YES}]
		{<u>NO</u> }
		[,TRUST={YES}]
		{<u>NO</u> }

```
DELCHAR={NONE}
        {char}
```

Function: Specifies the character which delimits the origin field in the message.

Format: None or *char*. *char* is a character in either character or hexadecimal format. If character format is used, the character must be framed by C''. If hexadecimal format is used, the character must be framed by X''.

Default: DELCHAR=NONE.

Note:

Coding DELCHAR=NONE is the same as if this operand had been omitted.

The origin field is delimited by the first encountered of three factors:

1. A blank character
2. A character specified in this operand
3. The end of the header data in the first buffer is coded.

```
DELEOH={NO }
       {YES}
```

Function: Specifies whether the origin field in the message may be delimited by the end of header data in the first buffer unit as determined by the text offset in the FHP.

Format: YES or NO.

Default: DELEOH=NO.

Note:

The origin field is delimited by the first encountered of three factors:

1. A blank character
2. A character specified in the DELCHAR= operand, if any
3. The end of the header if this operand is specified.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

```
TRUST={NO }
      {YES}
```

Function: Specifies whether the stated origin in the message header (the logical origin) may differ from the name of the LU physically entering the message.

Format: YES or NO.

Default: TRUST=NO.

Note:

TRUST=YES specifies that the stated origin may be any valid LU defined in the system. The stated origin is then used to determine input sequence number validity.

This operand is provided so that a second LU can enter messages for an inoperative LU; the messages appear to the recipient at the destination as if they were entered from their stated origin. Error messages would, nevertheless, be routed to the physical origin.

If you intend to use the TCAM retrieval-by-stated-origin facility, you must issue the ORGFLD macro to set internal fixed-header-prefix fields. These fields define the message header origin field to TCAM. When retrieving messages using the SEND command, an additional operand of SEND is ORIG=. ORIG specifies the origin name of the messages to be retrieved.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	Successful execution
X'00000004'	No data in message buffer
X'00000008'	Scan pointer is located within the data in the buffer but outside the first unit
X'0000000C'	Either: • Not the first buffer of a message • Duplicate header
X'00000010'	No FHP in message buffer
X'0000002C'	Origin name invalid
X'00000030'	No name found before the origin field delimiter
X'FFFFFFFC'	Scan pointer located beyond the data in the buffer

REALDEST Macro

The REALDEST macro:

- Creates one entry in a pseudo-destination table
- Must—either alone or together with other REALDEST macro instructions—immediately follow a DESTAB macro instruction, and may not be issued anywhere else
- Is optional in any nonexecuted section of the MCP, but may not be coded before the INTRO macro instruction
- May be issued more than once.

The DESTAB macro defines a pseudo-destination table which is used by the DESTFLD macro to resolve pseudo-destinations. The REALDEST macro defines an entry in the pseudo-destination table. A table may contain multiple entries for routing and is constructed by specifying one REALDEST macro for each table entry, immediately following a DESTAB macro.

A REALDEST macro should be issued for each valid pseudo-destination in the system. When the DESTFLD macro refers to the pseudo-destination table named in the DESTAB macro, the names of the subsequent REALDEST macros are scanned for a match to the destination used in the message. A REALDEST macro with a blank name field is regarded as a match in all cases.

Supported Resources and General Requirements: Not applicable.

Valid subgroups: Not applicable.

NAME	OPERATION	OPERANDS
[pseudo]	REALDEST	[routname,{A}]
		{V}
		[statname]
		,END

`END`

Function: Flags this REALDEST macro as the last entry in a destination table.

Format: END.

Default: None. Must be specified in the last REALDEST macro following a DESTAB macro.

`pseudo`

Function: Specifies the pseudo-destination name for which this REALDEST macro supplies the real destination.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

Note: If this operand is omitted, the entry is considered an unconditional match for any header destination field.

`routname`

Function: Specifies a user-written closed subroutine that determines a real destination associated with the pseudo-destination name.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is required if the *statname* operand is not specified.

routname specifies the name of a user-written closed subroutine that is given control when the DESTFLD routine processes this REALDEST macro. If *routname* is specified, *statname* may not be specified.

The user routine normally determines the message destination, based on the destination field in the message. The routine can also examine other message header fields, either to determine the message destination or for other user processing (such as assigning a mode to the message, based on the destination). The routine may not change header fields that have already been processed by macros that cause information to be placed in the FHP, because the FHP information for those fields would then be invalid.

The user routine may change the contents of registers 0, 1, and 10 without saving registers. If the user routine saves and restores registers 2 through 13, then the contents of those registers may be changed.

When the user routine receives control, register 4 contains the address of a doubleword containing the pseudo-destination, as specified in the message header, left-aligned and padded with blanks. Register 6 contains the address of the header buffer; register 9 contains the address of the address vector table. Register 14 contains the return address for the calling routine. Register 15 contains the address of the entry point for the user routine. *The user must establish addressability for his own routine.*

The user routine should return control to the DESTFLD macro by branching on register 14 plus a certain displacement. The displacement specified indicates the action to be taken by the DESTFLD macro.

The displacements and their meanings are:

Displacement	**Meaning**
0	The user routine is returning the address of a destination name in register 1 and its length in register 0. If this destination is valid, the message is queued to it; otherwise, the message is rejected.
4	The table search is to continue at the next entry in the destination table.
8	The message is to be returned to its originator unless it is a process queue, in which case the destination is to be considered invalid.
12	The destination is to be considered invalid. The user routine has set register 0 to valid DESTFLD return code.
16	The user routine is returning the address of a destination name in register 1 and its length in register 0. The message is queued to both the returned destination and the message specified in the message header, if both destinations are valid entries in the terminal table. If either destination is not valid, the message is rejected for invalid destination and is not queued to either destination. The associated DESTFLD macro must not include the REAL=NO operand; if it does, the message destination is rejected as invalid.
20	The destination in the message header is to be considered invalid.
24	The message is to queued to the destination in the message header if REAL=YES is coded; otherwise, it is to be considered invalid.
28	No message is to be queued and no further action should be taken. Processing should continue with the next sequential instruction in the MH.

```
{A}
{V}
```

Function: Indicates that the preceding operand is a *routname* entry as opposed to a *statname* entry. It also indicates the location of the routine named by the *routname* operand.

Format: A or V.

Default: None. Specification is required if *routname* is specified.

Note:

A specifies that the previous operand is a routine name and that the user-written routine is assembled within the MCP. (An A-type address constant is generated.)

V specifies that the previous operand is a routine name and that the user-written routine is in the load library. (A V-type address constant is generated.)

`statname`

Function: Specifies a real destination associated with the pseudo-destination name.

Format: *statname* must conform to the rules for assembler language symbols.

Default: None. Specification is required if the *routname* operand is not specified.

Note: *statname* is the name of a single, distribution list, cascade list, or process entry in the terminal table. If *statname* is specified, *routname* may not be specified.

Return Codes

None.

SENDMSG Macro

The SENDMSG macro:

- Permits you to generate and send one or more messages or a table to any external LU or application program in the network
- Is used as defined in chapter 2, except for the modified operands shown below

A SENDMSG exit routine named DKJKABX is provided by TCAM to send a skipped-sequence-number message to one of two designated destinations, as requested by the SENDMSG EXITCDE= operand.

The exit routine DKJKABX contains the kill-and-blank (KAB) message text itself.

The KAB message text is as follows:

SKIP NBR xxxx (THRU yyyy)

xxxx	= lowest skipped sequence number - external LU's next expected input sequence number
yyyy	= highest skipped sequence number - sequence number in input message less 1 (appears only if there is more than one skipped sequence number)

Kill-and-blank processing by the SEQFLD macro sets on the TKABBIT in the TCSBKG option field.

If this KAB bit is on, the user should invoke the SENDMSG exit routine in the inmessage section of the MH to generate the skipped-sequence-number messages.

The following code illustrates this function:

```
* TEST if KAB MESSAGE IS TO BE GENERATED
       TESTBR      OPTION=TCSBKG,
                   MASK=AL1(TKABBIT),
                   BRANCH=NOKAB

* KAB MESSAGE TO BE GENERATED
SENDMSG    EXITCDE=1,EXIT=DKJKABX,EXITTYP=V      SEND KAB MESSAGE TO LOG

SENDMSG    EXITCDE=2,EXIT=DKJKABX,EXITTUP=V      SEND KAB TO ORIGINATOR
                                                 AND UPDATE NEXT EXPECTED
                                                 SEQUENCE NUMBER

NOKAB TESTBR
```

Supported Resources and General Requirements: ALL.

Valid subgroups: Inmessage or outmessage.

NAME	OPERATION	OPERANDS

SENDMSG	EXIT=DKJKABX	,EXITCDE={1}
		{2}

```
EXIT=DKJKABX
```

Function: Provides skipped-sequence-number message to be sent following KAB processing by the SEQFLD macro.

Format: DKJKABX.

Default: None. Specification is required if a skipped-sequence-number message is to be sent.

```
EXITCDE={1}
        {2}
```

Function: Indicates a value to be passed as part of the parameter list to the DKJKABX routine.

Format: 1 or 2.

Default: None. Specification is required.

Note:

1 indicates that the skipped-sequence-number message is to be sent to the destination pointed to by a key table entry generated by a KEYDEF macro which specified KEY=CL8'LOG'.

2 indicates that the skipped-sequence-number message is to be sent to the origin and that the next-expected input sequence number for this external LU is to be updated to one more than the number in the message whose processing triggered the SENDMSG macro. The appropriate bit in the FHP is set on to notify the TCSENDBL macro routine to add one to the ERRCOUNT option field for the external LU.

Return Codes

None.

SEQFLD Macro

The SEQFLD macro:

- Validates the input sequence number field by comparing it with the next-expected input sequence number either for the physical source of the message or for the logical message origin as indicated by the message header origin field (if TRUST=YES is coded in the ORGFLD macro and logical origin differs from physical source).
- Permits, at your option, the automatic replacement of a zero input sequence number for the logical message origin, as specified by the message header field, or for the physical message source.
- Updates a pointer to the input sequence number in the FHP.
- Requires the presence of an FHP in the header being processed.
- Executes correctly only when the scan pointer is located within the first buffer unit of the first buffer containing the message.
- Provides specialized support for brokerage applications but may be used to process any message containing an FHP.
- Allows you to specify that a valid input sequence number may exceed the next-expected input sequence number for the physical message source by a user-specified range of numbers. If input sequence numbers are skipped by use of this facility, the SEQFLD macro automatically sets an indicator that may be interrogated in the inmessage subgroup (if it is desired to have a message generated to indicate which sequence numbers have been skipped). This notification of skipped sequence numbers is also referred to as a "KAB" message, since it informs the operator which input sequence numbers have been "killed and blanked" and are not to be found in the system.
- Must be coded if retrieval of messages by input sequence number is to be accomplished via the online retrieval system service program.
- May be issued more than once.

Valid input sequence numbers range from 1 to 9999, plus, at your option, 0. Automatic wraparound from 9999 to 1 is provided by the SEQFLD macro as it is in the SEQUENCE macro. This wraparound should be avoided because it would cause the online retrieval system service program to yield unpredictable results.

If the message origin has not yet been validated when SEQFLD executes, but an ORGFLD macro is to be executed later for this purpose then the sequence number validation may be completely transparent to the MCP coder who does not add user code related to this fact between the two macros.

The postponement takes place if the SEQFLD macro instruction is coded with ORIGIN=YES. If the feature of logical message origin is to be employed, as previously mentioned, and the ORGFLD macro is executed after the SEQFLD macro, this specification is required. If the order of the two macros is reversed, entering that operand does not negatively affect the performance of any other feature or macro. Consequently, it is advisable always to code the SEQFLD macro with ORIGIN=YES if the ORGFLD macro for the same messages is coded with TRUST=YES.

The SEQFLD macro expects the scan pointer to point to either the last byte of the field preceding the input sequence number field or to a blank preceding the input sequence number field.

If the zero sequence number replacement function is used, up to three character positions may be required to insert the current number: zero occupies one position and a valid input sequence number may extend to four positions. Sufficient reserve characters must be present to accommodate these extras.

SEQFLD

The SEQFLD macro provides enhancements to the functions performed by the SEQUENCE macro; either of these macros, but not both, may be issued in any given path through the MCP.

After successful execution of the SEQFLD macro:

1. Register 15 is set to a return code of 0.
2. The scan pointer is advanced to the last character of the input sequence number in the message header.
3. If the input sequence number was zero, it has been replaced by the next-expected input sequence number for the origin.
4. If a zero input sequence number has been replaced with the next-expected input sequence number, causing a move of other header entries, FHP offsets (which are pointing to header fields) have been altered as necessary to reflect the data insertion.

After successful execution of the SEQFLD macro, the location of the scan pointer is unpredictable if the return code is X'04', X'08', X'0C', X'10', or X'FC'. (See "Return Codes" at the end of this macro description.)

Supported Resources and General Requirements: ALL, FHP.

Valid subgroups: Inheader.

NAME	OPERATION	OPERANDS

[symbol]	SEQFLD	[DELCHAR={NONE}]
		{char}
		[,DELEOH={YES}]
		{NO }
		[,KAB={integer}]
		{9 }
		[,ORIGIN={YES}]
		{NO }
		[,ZERO={chars}]
		{ALL }
		{NONE }

```
DELCHAR={char}
        {NONE}
```

Function: Specifies the character that delimits the input sequence number in the message header.

Format: NONE or *char*. *char* is a single character in either character or hexadecimal format. If character format is used, the character must be framed by C''. If hexadecimal format is used, the character must be framed by X''.

Default: DELCHAR=NONE.

Note: The input sequence number is delimited by the first encountered of these three factors:

- A blank character
- A character specified in this operand
- The end of header data in the first buffer unit if so specified in the DELEOH= operand.

Coding DELCHAR=NONE is allowed; the meaning is the same as if this operand had been omitted.

```
DELEOH={YES}
       {NO }
```

Function: Specifies that the input sequence number field in the message may be delimited by the end of header data in the first buffer unit as determined by the text offset in the FHP.

Format: YES or NO.

Default: DELEOH=NO.

```
KAB={integer}
    {9      }
    {NO     }
```

Function: Specifies the value by which a valid input sequence number may be higher than the next-expected input sequence number for the physical origin. Any number is considered valid that is not lower than the next-expected input sequence number and not higher than the next-expected number plus the KAB value.

Format: *integer,* 9, or NO. *integer* is an unframed decimal integer; minimum 9, maximum 255.

Default: KAB=9.

Note:

NO specifies that no skipping of input sequence numbers is allowed. It is synonymous with KAB=0.

If an otherwise high input sequence number is valid because it falls within the specified KAB range, a skipped sequence number message must be generated by coding the inmessage section accordingly. Otherwise, the input sequence counter is not updated. The message notifies the originator that the intervening sequence number or numbers were omitted.

If the stated origin is not the same as the physical origin, skipping of input sequence numbers is not allowed.

```
ORIGIN={YES}
       {NO }
```

Function: Specifies that this MH path includes validation of a stated origin field via an ORGFLD macro.

Format: YES or NO.

Default: ORIGIN=NO.

Note:

If the TRUST= operand of the ORGFLD macro in this MH path is coded TRUST=NO, then this operand may be omitted to optimize processing time.

Coding ORIGIN=NO is allowed; the meaning is the same as if this operand had been omitted.

```
symbol
```

Function: Specifies the name of the macro.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is optional.

```
ZERO={chars}
     {ALL  }
     {NONE }
```

Function: Specifies the message mode types for which a zero input sequence number is accepted as valid, and is replaced with the next-expected input sequence number for the stated origin.

Format: *chars,* ALL, or NONE. *chars* is a character string composed of one to eight nonblank characters in either character or hexadecimal format. If character format is used, the string must be framed with C'' characters. If hexadecimal format is used, the string must be framed with X'' characters.

Default: ZERO=NONE.

Return Codes

One of the following return codes is set in register 15:

Code	**Meaning**
X'00000000'	Successful execution
X'00000004'	No data in the buffer
X'00000008'	Scan pointer is located within the data in the buffer but outside the first unit
X'0000000C'	Either: • Is not the first buffer of a message • Is a duplicate header
X'00000010'	No FHP in the message buffer
X'00000014'	Input sequence number is too high
X'00000018'	Input sequence number is too low
X'0000001C'	No message origin is indicated in the TCAM buffer prefix
X'00000020'	Sequence number field is above 9999, not all numeric, or not found before the specified sequence number delimiter
X'00000024'	Origin not set
X'00000028'	Insufficient reserved character positions to replace a zero input sequence number with the next-expected input sequence number for the message origin
X'FFFFFFFC'	Scan pointer is located beyond the data in the buffer

TCSENDBL Macro

The TCSENDBL macro:

- Inserts previously-scheduled fields into the message header
- May optionally remove the FHP from a message before passing it to a destination station or application program
- Is used as defined in Chapter 2, except for the additional operand defined below.

The TCSENDBL macro inserts output sequence number, date, and time information into the message header, overlaying the fixed header prefix area as necessary. Any of the following fields may be scheduled by the INSRTFLD macro for insertion by the TCSENDBL macro: output sequence number, input date, input time, output date, and output time.

Supported Resources and General Requirements: ALL, FHP.

Valid subgroups: Outheader.

Name	Operation	Operands

	`TCSENDBL`	`[TRAILDT=({CURDAT } ]`
		`{,CURTIM}`
		`{,INDATE}`
		`{,INTIME})`

```
TRAILDT=({CURDAT }
         {,CURTIM}
         {,INDATE}
         {,INTIME})
```

Function: Specifies the date and time expression to be included in the trailer.

Format: Any combination of the following: CURDAT, CURTIM, INDATE, and INTIME, separated by commas and, unless only one entry is used, enclosed in parentheses.

Default: Specification is optional. If TRAILOT= is not coded, no dates and times are included in the trailer.

Note:

CURDAT specifies that the current date should be included in the trailer date and time expression in the trailer.

CURTIM specifies that the current time should be included in the trailer date and time expression in the trailer.

INDATE specifies that the input date should be included in the trailer date and time expression in the trailer.

INTIME specifies that the input time should be included in the trailer date and time expression in the trailer.

Return Codes

See Chapter 2.

UPCONDTN Macro

The UPCONDTN macro:

- Defines a valid TCAM startup or restart condition
- Is optional in a nonexecuted section of the MCP and may be coded up to 255 times
- Must be coded before any UPMSG, UPTYPE, or TCSUP macro.

Name	Operation	Operands

`opreply`	`UPCONDTN`	`condno`
		`[CONTYPE=EMER]`

`condno`

Function: Specifies the ordinal number of the UPCONDTN macro.

Format: A decimal number from 1 to 255. Condition numbers must be sequential, incremented by 1 for each successive UPCONDTN macro. A positional parameter.

Default: None. Specification is required.

`CONTYP=EMER`

Function: Identifies the startup or restart condition.

Format: CONTYP=EMER.

Default: None. Specification is optional.

Note: CONTYP=EMER indicates an emergency restart condition. The terminal next-expected sequence number fields are set to 1 automatically, but the first input sequence number received from each station is considered valid and is accepted. The exception is a zero input sequence number, which is invalid as the first input sequence number after an emergency restart.

`opreply`

Function: Specifies a valid operator response to the TCAM startup message DKJ750A.

Format: Must conform to the rules for assembler language symbols.

Default: None. Specification is required.

Return Codes

None.

Glossary

This glossary includes terms and definitions from the *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems,* GC20-1699. Definitions from the *American National Dictionary for Information Processing* are identified by an asterisk (*).

Reference Words Used in the Entries

The following reference words are used in this glossary.

Contrast with. Refers to a term that has an opposed or substantively different meaning.

Deprecated term for. Indicates that the term should not be used. It refers to a preferred term, which is defined.

See. Refers to multiple-word terms that have the same last word.

See also. Refers to related terms that have similar (but not synonymous) meanings.

Synonym for. Appears in the commentary of a less desirable or less specific term and identifies the preferred term that has the same meaning.

Synonymous with. Appears in the commentary of a preferred term and identifies less desirable or less specific terms that have the same meaning.

ACB. In VTAM, access method control block.

accept. In a VTAM application program, to accept a CINIT request from a system services control point (SSCP) to establish a session with a logical unit; the application program acts as the primary end of the session. Contrast with *acquire (1).*

Note: The accept process causes a BIND request to be sent from the primary end of the session to the logical unit that will act as the secondary end of the session, requesting that the session be established and passing session parameters. For example, the session-initiation request that originally caused the SSCP to send the CINIT request may have resulted from a logon by the terminal operator, from a macro instruction issued by a VTAM application program, or from a VTAM operator command.

access method. A technique for moving data between main storage and input/output devices. See *Basic Direct Access Method, Basic Sequential Access Method, TCAM (Version 2 and previous releases), and VTAM.*

access method control block (ACB). A control block that links an application program to VSAM or VTAM.

access method interface (AMI). The TCAM function for managing communication on the access method control block (ACB) interface between TCAM and VTAM.

ACF. Advanced Communications Function.

ACF/TCAM. Advanced Communications Function for TCAM. Synonym for *TCAM.*

ACF/VTAM. Advanced Communications Function for VTAM. Synonym for *VTAM.*

acquire. (1) In VTAM, the operation in which an authorized VTAM application program initiates and establishes a session with another logical unit; the application program acts as the primary end of the session. Contrast with *accept.* (2) In relation to VTAM resource control, to take over resources (communication controllers or other physical units) that were formerly controlled by a data communication access method in another domain, or to assume control of resources that were controlled by this domain but released. Contrast with *release.* See also *resource takeover.*

active. In VTAM, the status of a resource that makes the resource known to VTAM (for major nodes) or makes it available for use in the network (for minor nodes). For a logical unit (LU) minor node, it also enables the LU to participate in LU-LU sessions. Contrast with *inactive.*

address space. The area of virtual storage that is available for a particular job.

Advanced Communications Function (ACF). A group of IBM program products, principally VTAM, TCAM, NCP, and SSP.

Advanced Communications Function for the Network Control Program (NCP). An IBM program product that provides communication controller support for single-domain, multiple-domain, and interconnected network capability. Programs (SSP)

Advanced Communications Function for Systems Support Programs (SSP). An IBM program product made up of a collection of utilities and small programs. SSP is required for the operation of the NCP.

Advanced Communications Function for TCAM (TCAM). An IBM program product that provides queued message handling. TCAM, Versions 1 and 2, are telecommunications access methods, but TCAM, Version 3, is a message handling subsystem.

Advanced Communications Function for VTAM (VTAM). An IBM program product that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability. VTAM runs under MVS (OS/VS1 and OS/VS2), VSE, and VM/SP. It supports direct control application programs and subsystems such as VSE/POWER.

affinity-based routing. Message routing in which a temporary relationship, or routing affinity, is established between a source and a destination; all messages from the source are routed to the destination for the duration of the relationship. See also *invariant routing, transaction-based routing, routing by destination, routing by key*.

AMH. Application message handler.

AMI. Access method interface.

application message handler (AMH). A user-defined routine that processes messages that are received by the message control program (MCP) from an application program or that are sent by the MCP to an application program. See *message handler, device message handler, internodal message handler*. See also *Message control program*.

application program. (1) A program written for or by a user that applies to the user's work. (2) A program used to connect and communicate with resources in a network, enabling users to perform application-oriented activities.

asynchronous. Without regular time relationship; unexpected or unpredictable with respect to the execution of a program's instructions.

automatic purge/copy/redirect. A collection of message-handler and extended operator control functions that permits messages to be conditionally or unconditionally redirected to another destination, copied to another destination, or purged (that is, not sent to any destination).

Basic Direct Access Method (BDAM). An access method used to directly retrieve or update particular blocks of a data set on a direct access device.

basic information unit (BIU). In SNA, the unit of data and control information that is passed between half-sessions. It consists of a request/response header (RH) followed by a request/response unit (RU).

basic operator command. An operator command directed to the basic operator control system service program. Synonymous with *basic operator control command.*

basic operator control. The function of a particular system service program that processes a set of basic operator commands. These commands allow the operator to determine the status of the TCAM system and to alter, start, and stop TCAM and its resources by entering appropriate commands from either the system console or a basic operator control station. The basic operator control system service program is required in order to execute a TCAM message control program (MCP).

basic operator control command. Synonym for *basic operator command.*

basic operator control station. A system console, external logical unit (LU), or application program that is authorized to enter operator commands to be executed by the basic operator control system service program.

basic primary operator control station. A basic operator control station that is sent all TCAM error-recovery messages and TCAM reply messages to basic operator commands. See *basic secondary operator control station, extended primary operator control station, extended secondary operator control station.*

basic secondary operator control station. A basic operator control station that is sent only the reply messages to basic operator commands entered from it. See *basic primary operator control station, extended primary operator control station, extended secondary operator control station.*

Basic Sequential Access Method (BSAM). An access method for storing or retrieving data blocks in a continuous sequence, using either a sequential access or direct access device.

BDAM. Basic Direct Access Method.

begin bracket. In SNA, the value (binary 1) of the begin-bracket indicator in the request header (RH) of the first request in the first chain of a bracket; the value denotes the start of a bracket. Contrast with *end bracket.* See also *bracket.*

bidder. In SNA, the LU-LU half-session defined at session activation as having to request and receive permission from the other LU-LU half-session to begin a bracket. Contrast with *first speaker.* See also *bracket protocol.*

binary synchronous communication (BSC). Communication using binary synchronous line discipline. See also *binary synchronous transmission.*

binary synchronous transmission. Data transmission in which synchronization of characters is controlled by time signals generated at the sending and receiving stations. Contrast with *start-stop transmission, synchronous data link control.*

bind image. Synonym for *logon mode.*

bind image table. Synonym for *logon mode table.*

BIU. Basic information unit.

BIU segment. In SNA, a portion of a basic information unit (BIU) that is contained within a path information unit. It consists of either a request/response header (RH) followed by all or a portion of a request/response unit (RU), or of only a portion of an RU. See also *segment.*

bracket. In SNA, one or more chains of request units (RUs) and their responses that are exchanged between two LU-LU half-sessions and that represent a transaction between them. A bracket must be completed before another bracket can be started. Examples of brackets are data base inquiries/replies, update transactions, and remote job entry output sequences to work stations. See *begin bracket, end bracket.* See also *RU chain.*

bracket protocol. In SNA, a data flow control protocol in which exchanges between two LU-LU half-sessions are achieved through the use of brackets, with one logical unit (LU) designated at session initiation as the first speaker and the other LU as the bidder. The bracket protocol involves bracket initiation and termination rules. See also *bidder, first speaker.*

bracket state manager. A TCAM routine that enforces the bracket protocol by making proper bracket state changes and detecting bracket errors.

broadcast. The simultaneous transmission of data to a number of destinations.

BSAM. Basic Sequential Access Method.

BSC. Binary synchronous communication.

buffer. (1) * A routine or storage area used to compensate for a difference in rate of flow of data, or time of occurrence of events, when transferring data from one device to another. (2) An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written.

buffer list. In VTAM, a contiguous set of control blocks (buffer list entries) that allow an application program to send function management data (FMD) from a number of discontiguous buffers with a single SEND macro instruction.

buffer prefix. A control area within each buffer that contains buffer control information. A user must allow room for the buffer prefix when specifying buffer size.

buffer unit. The smallest block of main storage from which TCAM buffers and main storage message queues can be built. Synonymous with *main storage unit.*

buffer-unit pool. All the buffer units in a particular TCAM system.

cascade entry. A terminal-table entry associated with a cascade list.

cascade list. A list of pointers to single entries. When a cascade entry is named as the destination for a message, the message is sent to the valid entry in the list with the fewest messages queued for it.

chain. See *RU chain.*

channel program block (CPB). A TCAM control block used in the transfer of data between buffer units and message queues maintained on disk.

checkpoint data set. An optional TCAM data set that contains the checkpoint records used to reconstruct the message control program (MCP) environment after closedown or system failure when the TCAM checkpoint/restart service facility is used.

checkpoint records. Records that contain the status of a job and the system at the time the records are written by the checkpoint routine. These records provide the information necessary for restarting a job without having to return to the beginning of the job. There are four types: checkpoint request record, control record, environment record, and incident record.

checkpoint request record. A checkpoint record taken as a result of the execution of a CKREQ macro instruction in an application program; the record contains the status of a single destination queue for the application program. See *control record, environment record, incident record.*

checkpoint/restart service facility. A TCAM service facility that records the status of the TCAM system at designated intervals or following certain events. After system failure, the TCAM system can be restarted and can continue without loss of messages.

CIB. Command input buffer.

clear data. Data that is not enciphered.

clear session. A session in which only clear data is transmitted or received. Contrast with *cryptographic session.*

closed subroutine. A subroutine of which one replica suffices for the subroutine to be linked by calling sequences for use at more than one place in a computer program. Contrast with *open routine.*

closedown. The orderly termination of the message control program. See *flush closedown, quick closedown.*

cold restart. Startup of a message control program (MCP) following either a flush closedown, a quick closedown, or a system failure. A cold restart ignores the previous environment; that is, the MCP is started as if this were the initial startup. A cold restart is the only type of restart possible when no checkpoint/restart service facility is used. Contrast with *warm restart.*

command. (1) In SNA, any field set in the transmission header, request header (RH), and sometimes portions of a request unit (RU), that initiates an action or begins a protocol; for example, (a) Bind Session (session-control request unit), a command that activates an LU-LU session; (b) the change-direction indicator in the RH of the last RU of a chain; (c) the virtual route reset window indicator in an FID4 transmission header. (2) Loosely, a request unit. (3) In SDLC, the control information (in the C-field of the link header) sent from the primary station to the secondary station. (4) In TCAM, an operator control command.

command input buffer (CIB). A buffer-like area that contains operator commands entered at the system console. Main storage space is allocated for it dynamically and is freed once the operator command contained within the CIB has been processed. Only one CIB need be specified for operator commands entered from the system console.

communication common carrier. In the USA and Canada, a public data transmission service that provides the general public with transmission service facilities; for example, a telephone or telegraph company.

COMWRITE. An IBM-supplied subtask of the TCAM initiator that formats and writes trace records to the COMWRITE data set.

COMWRITE data set. A data set on a sequential storage device in which trace information is written.

connection point manager. In SNA, a component of the transmission control layer that (a) performs session-level pacing of normal-flow requests; (b) checks sequence numbers of received response units; (c) verifies that request units do not exceed maximum permissible size; (d) routes incoming request units to their destinations within the half-session; and (e) enciphers and deciphers function management data (FMD) request units when cryptography is selected. The sending connection point manager within a half-session builds the request/response header for outgoing request units, and the receiving connection point manager interprets the request headers that precede incoming request units.

control record. A checkpoint record included in a checkpoint data set that keeps track of the correct environment records, incident records, and checkpoint request records to use for restructuring the message control program environment during restart. See *environment record, incident record, checkpoint request record.*

conversational mode. A mode in which the next message received by an external logical unit (LU) after it enters an inquiry message is a reply to that message. See *lock mode.*

CPB. Channel program block.

cryptographic. Pertaining to the transformation of data to conceal its meaning.

cryptographic session. An LU-LU session in which a function management data request may be enciphered before it is transmitted, and deciphered after it is received. Contrast with *clear session.* See also *mandatory cryptographic session, selective cryptographic session.*

DASD. Direct access storage device.

data control block (DCB). A control block used by access method routines in storing and retrieving data.

Data Encryption Standard (DES) algorithm. A cryptographic algorithm designed to encipher and decipher 8-byte blocks of data using a 64-bit cryptographic key, as specified in the *Federal Information Processing Standard Publication 46,* January 15, 1977.

data flow. (1) In SNA: any of four flows in a given session: primary-to-secondary flow, secondary-to-primary flow, normal flow, or expedited flow. (2) The type of route or extended route that a message takes from its origin to its destination, including the host nodes that process the message while it is enroute to its destination. See *level 1 data flow, level 2 data flow, level 2+ data flow, level 3 data flow.*

data flow control (DFC) layer. In SNA, the layer within a half-session that (a) controls whether the half-session can send, receive, or concurrently send and receive request/response units (RUs); (b) groups related RUs into RU chains; (c) delimits transactions via the bracket protocol; (d) controls the interlocking of requests and responses in accordance with control modes specified at session activation; (e) generates sequence numbers; and (f) correlates requests and responses.

data set. (1) The major unit of data storage and retrieval in the operating system, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access. (2) * Deprecated term for *modem.*

data staging. An extended networking technique in which high-volume, low-priority message traffic is moved from one TCAM node to another, progressively approaching the destination TCAM. Data staging allows such traffic to be moved at a convenient time to avoid overloading the network to protect response times for high-priority inquiries.

DCB. Data control block.

dead-letter queue. In TCAM, a queue containing messages that could not be placed in the appropriate destination queue.

decipher. To convert enciphered data into clear data. Contrast with *encipher.* Synonymous with *decrypt.*

decrypt. To convert encrypted data into clear data. Contrast with *encrypt.* Synonym for *decipher.*

definite response. In SNA, a form of response requested in the request header (RH) for a request unit (RU). The receiver is requested to return a response indicating whether the request is acceptable as received. Contrast with *exception response, no response.*

delayed-request mode. In SNA, an operational mode in which the sender may continue sending request units on the normal flow after sending a definite-response chain on that flow, without waiting to receive the response to that chain. Contrast with *immediate-request mode.* See also *delayed-response mode.*

delayed-response mode. In SNA, an operational mode in which the receiver of normal-flow request units can return responses to the sender in a sequence different from that in which the corresponding request units are sent. Contrast with *immediate-response mode.* See also *delayed-request mode.*

delimiter macro instruction. A TCAM message-handler macro instruction that classifies and identifies sequences of functional message-handler macro instructions and directs control to the appropriate sequence of functional macro instructions. Contrast with *functional macro instruction.*

DES. Data Encryption Standard. See also *Data Encryption Standard algorithm.*

DES algorithm. Data Encryption Standard algorithm.

destination. An external logical unit (LU) or application program to which a message or other data is directed.

destination queue. A queue on which messages bound for a particular destination are placed after being processed by the incoming group of the message handler. See also *process queue.*

device message handler (DMH). A user-written routine defined in a TCAM message control program (MCP) that processes messages being received from or sent to an external logical unit (LU). See also *application message handler, internodal message handler, message handler.*

DFC. Data flow control.

direct access storage device (DASD). A storage device in which the access time is effectively independent of the location of the data.

distribution entry. A terminal-table entry associated with a distribution list.

distribution list. A list of pointers to single or cascade entries. When a distribution entry is named as the destination for a message, the message is sent as separate transmissions to all items in the list.

DMH. Device message handler.

domain. In SNA, a system services control point (SSCP) and the physical units (PUs), logical units (LUs), links, link stations, and all the associated resources that the SSCP has the ability to control by means of activation requests and deactivation requests. Synonymous with *single-domain network.*

dynamic accounting facility. A TCAM service facility that gathers resource utilization data for processing by user-supplied applications.

EBCDIC. Extended binary-coded decimal interchange code.

encipher. (1) To scramble data or convert it, prior to transmission, to a secret code that masks the meaning of the data to any unauthorized recipient. (2) In VTAM, to convert clear data into enciphered data. Contrast with *decipher.* Synonymous with *encrypt.*

enciphered data. Data that is intended to be illegible to all except those who legitimately possess the means to reproduce the clear data.

encrypt. In VTAM, to convert clear data into enciphered data. Contrast with *decrypt.* Synonym for *encipher.*

end bracket. In SNA, the value (binary 1) of the end bracket indicator in the request header (RH) of the first request of the last chain of a bracket. The value denotes the end of the bracket. Contrast with *begin bracket.* See also *bracket.*

end-of-address (EOA) character. A character that must be placed in a message if the system is to route that message to several destinations. The character must immediately follow the last destination coded in the message header.

end-to-end session. In TCAM a logical connection in which an affinity-based routing relationship has been established between a source and a destination. Either the source or destination can be either a logical unit (LU) or an application program. End-to-end sessions require routing by key.

end user. The ultimate source or destination of application data flowing through an SNA network. An end user may be an application program or a terminal operator.

environment record. A checkpoint record of the total TCAM system at a single point in time. See *checkpoint request record, checkpoint/restart service facility, control record, incident record.*

EOA. End-of-address. See *end-of-address character.*

error record. Five bytes assigned to each message processed by a message handler. These bytes indicate physical or logical errors that have occurred during transmission or during subsequent processing or queuing of the message. In addition, a message error record may be the created when a session cannot be established. Error records are checked by error-handling macro instructions in the in-message and out-message subgroups of a message handler. Synonymous with *message error record.*

error-recovery procedures. A set of internal TCAM routines that attempt to recover from transmission errors.

exception request. In SNA, a request that replaces another message unit in which an error has been detected.

exception response. In SNA, a value in the form-of-response requested field of a request header. The receiver is requested to return a response only if the request is unacceptable as received or cannot be processed. That is, a negative response, but not a positive response, may be returned. Contrast with *definite response, no response.* See *negative response.*

expedited flow. In SNA, a data flow designated in the transmission header that is used to carry network control, session control, and various data flow control request/response units (RUs). The expedited flow is separate from the normal flow (which carries primarily end-user data) and can be used for commands that affect the normal flow on the path. Contrast with *normal flow.* See also *isolated pacing response.*

extended lock mode. A type of lock mode in which an external logical unit (LU) remains in lock mode for the duration of several inquiry/reply cycles. Contrast with *message lock mode.* See *lock mode.*

extended network. A network that includes two or more TCAM systems using extended networking facilities.

extended networking. A TCAM function that uses a collection of TCAM macro instructions, system service programs, and message-handler facilities to simplify TCAM system definition, management, and error recovery in a network with two or more TCAM systems.

extended operator command. An operator command directed to the extended operator control system service program. Synonymous with *extended operator control command.*

extended operator control. The function of a particular system service program that processes a set of extended operator commands. These commands are not required in order to control a TCAM system, but are useful in some environments. The extended operator control system service program is required if the message control program (MCP) uses one or more of the following functions: (a) extended networking, (b) online retrieval system service program, or (c) automatic purge/copy/redirect.

extended operator control command. Synonym for *extended operator command.*

extended operator control station. A system console, external logical unit (LU), or application program that is authorized to enter extended operator commands. See *basic operator control station, extended primary operator control station, extended secondary operator control station.*

extended primary operator control station. An extended operator control station that receives the extended operator control startup and closedown messages; responses to extended operator commands entered from it; responses to extended operator commands that successfully modify the TCAM system; and, optionally, the online retrieval system service program startup and closedown messages (if online retrieval is part of the TCAM system). See *basic primary operator control station, basic secondary operator control station, extended secondary operator control station.*

extended secondary operator control station. An extended operator control station that enters extended operator commands and receives the responses made to those commands. See *basic primary operator control station, basic secondary operator control station, extended primary operator control station.*

extended route. In TCAM extended networking, a series of one or more routes that involves an intermediate TCAM node.

external LU. A logical unit (LU) that communicates with a TCAM message control program (MCP) through VTAM. Each external LU is defined to the MCP with a TERMINAL macro instruction.

FHP. Fixed header prefix.

first speaker. In SNA, the LU-LU half-session defined at session activation as (a) able to begin a bracket without requesting permission from the other LU-LU half-session to do so, and (b) winning contention if both half-sessions attempt to begin a bracket simultaneously. Contrast with *bidder.* See also *bracket protocol.*

fixed header prefix (FHP). An optional control block that provides a place to keep message-related information needed by certain optional TCAM functions.

flush closedown. A closedown of TCAM during which incoming message traffic is suspended and queued outgoing messages are sent to their destinations ("flushed" from the message queues) before closedown is completed. Contrast with *quick closedown.*

FM. Function management.

FMD. Function management data.

FMD services layer. In SNA, the layer within a half-session that routes function management data (FMD) requests and responses to particular network addressable unit (NAU) services manager components and that provides session network services or session presentation services, depending on the type of session.

function management data (FMD) services. A general term for session network services and session presentation services, both of which process FMD requests and responses.

function management (FM) header. In SNA, one or more headers, optionally present in the leading request units (RU) of an RU chain, that allow one half-session in an LU-LU session to: (a) select a destination (for example, a program or a device) as the session partner and control the way that the end-user data it sends is handled at the destination, (b) change the destination or the characteristics of the data during the session, and (c) transmit between session partners status or user information about the destination (for example, a program or a device).

function management (FM) profile. In SNA, a specification of various data flow control protocols (such as request unit chains and data flow control requests) and function management data (FMD) options (such as use of FM headers, compression, and alternate codes) supported for a particular session. Each function management profile is identified by a number.

functional macro instruction. A TCAM macro instruction that performs the specific operations required for messages directed to the message handler. Contrast with *delimiter macro instruction.*

group entry. A terminal-table entry associated with a group of logical units (LUs).

group of logical units (LUs). In TCAM, a set of external LU definitions that are associated with the same group entry. See also *group entry.*

half-session. In SNA, a component that provides FMD services, data flow control, and transmission control for one of the sessions of a network addressable unit. See *primary half-session, secondary half-session.*

header. That portion of a message containing control information for the message; a header might contain one or more destination fields, the name of the originating station, an input sequence number, a character string indicating the type of message, and a priority level for the message. The message header is operated on by macro instructions in the inheader and outheader subgroups of the message handler. See *message header.*

header buffer. A buffer containing either all or the first part of a message header. Contrast with *text buffer.*

host computer. Synonym for *host processor.*

host logical unit (LU). An SNA logical unit (LU) located in a host processor, for example, a VTAM application program. See *TCAM host logical unit (LU).*

host node. In SNA, a subarea node that contains a system services control point (SSCP).

host processor. The controlling processor with its operating system, access methods, and application programs. A system services control point is located in a host processor. Synonymous with *host computer.*

IMH. Internodal message handler.

immediate-request mode. An operational mode in which the sender, after sending a definite-response request chain on a given flow, stops sending request units on the flow until the chain has been responded to. Contrast with *delayed-request mode.* See *immediate-response mode.*

immediate-response mode. An operational mode in which the receiver responds to request units on a given normal flow in the order it receives them; that is, in a first-in, first-out sequence. Contrast with *delayed-response mode.* See *immediate-request mode.*

inactive. In VTAM, pertaining to a resource that has never been activated or has been deactivated by a VTAM operator command. Contrast with *active.*

inblock subgroup. The part of a message handler (MH) incoming group that, if used, precedes the inheader subgroup and blocks several physical messages into a longer, logical message or unblocks a physical message into a shorter, logical message.

inbuffer subgroup. The part of a message handler (MH) incoming group that operates on each segment of an incoming message.

incident record. A checkpoint record that logs a change in external logical unit (LU) or application program status, and in the contents of an option field that occurred since the last environment record was taken. It is used to update the information contained in environment records at restart after a closedown or system failure. See *checkpoint request record, control record, environment record.*

incoming group. That portion of a message handler that is designed to handle incoming messages for the message control program (MCP). Contrast with *outgoing group.*

incoming message. A message sent from an external logical unit (LU) or application program to the message control program (MCP).

inheader subgroup. The part of a message handler (MH) incoming group that operates on all or part of an incoming message header.

initial chaining value (ICV). An eight-byte, pseudo-random number used to verify that both ends of a session with cryptography have the same session cryptography key. The initial chaining value is also used as input to the Data Encryption Standard (DES) algorithm to encipher or decipher data in a session with cryptography.

initiation. Synonym for *LU-LU session initiation.* See also *session-initiation request.*

initiator. The component of TCAM that is executed as the job-step task. The initiator starts, monitors, and restarts the message control program (MCP), TCAM system service programs, and user-supplied system service programs. It can also display status information at the system console.

inmessage subgroup. The part of a message handler (MH) incoming group that specifies actions to be taken after a complete message has arrived at the message control program (MCP).

input data set. A data set that contains all messages or records sent to an application program from a single process queue. Contrast with *output data set.*

inquiry/reply. A TCAM application in which a device message handler receives a message from an external logical unit (LU) and then routes it to an application program that processes the message and generates a reply. The reply is routed back to the inquiring external LU.

intercepted resource. An external logical unit (LU) to which no messages may be sent for a specified time interval or until an operator command or an application-program macro instruction is issued to release messages. An intercepted resource can enter messages, but messages destined for it are not sent.

intermediate function. In SNA, a path control capability within a subarea node that receives and routes path information units that neither originate in nor are destined for the network addressable units in that subarea node.

intermediate TCAM node. In TCAM extended networking, a TCAM node that processes messages flowing along the extended route but does not provide the queuing for the originating or destination resources for those messages. Processing by an intermediate TCAM node includes processing by the incoming group of the internodal message handler, queuing of each message on the internodal destination queue for the next TCAM node on its extended route, and processing by the outgoing group of the internodal message handler (IMH). See *TCAM node.*

internodal awareness. In TCAM extended networking, a function used by TCAM systems to share information about each other. This information includes the status of TCAM systems, the status of application programs in TCAM systems, and the contents of selected key-table entries. This function is provided by node path system service programs in the various TCAM systems that communicate with each other.

internodal destination queue. In TCAM extended networking, a destination queue for an external logical unit (LU) that is a partner in a utility session.

internodal message handler (IMH). In TCAM extended networking, a message handler that processes messages flowing on utility sessions.

internodal sequence number synchronization. In TCAM extended networking, the function of a particular system service program that operates in conjunction with the internodal message handler. Internodal sequence number synchronization is used to request retransmission from any TCAM node of sequence-numbered messages not received on that utility session and retransmit sequence-numbered messages flowing on a utility session when requested to do so by another TCAM node or an extended operator command.

internodal sequence prefix. In TCAM extended networking, a control block that is used to contain sequence-number information for messages flowing on utility sessions.

invariant routing. Message routing in which messages from the same source are always sent to the same destination. See *affinity-based routing, transaction-based routing.* See also *routing by destination, routing by key.*

key. A character string that matches a definition in the key table. This key identifies the destination of a message or special processing to be done on that message. See also *key table.*

key table. A main-storage table of keys and their definitions, which contain information on routing and special processing to be done on a message. See also *key.*

layer. In SNA, a grouping of related functions that are logically separate from the functions in other layers; the implementation of the functions in one layer can be changed without affecting other layers. See *data flow control layer, FMD services layer.*

LCB. Line control block.

level 1 data flow. In SNA, a data flow (within a single-domain network) in which each message's origin and destination logical units (LUs) reside in the same domain.

level 2 data flow. In TCAM extended networking, a data flow on an extended route in which each message enters the TCAM node that provides queuing for the originating resource, another TCAM node that provides queuing for the destination resource, and one or more intermediate TCAM nodes.

level 2+ data flow. In TCAM extended networking, a data flow on an extended route in which each message enters both the TCAM node that provides queuing for the originating resource and the TCAM node that provides queuing for the destination resource, but does not pass through any intermediate TCAM nodes.

level 3 data flow. In SNA, a data flow (within a multiple-domain network) in which each message's origin and destination logical units (LUs) reside in different domains.

line. Any physical medium, such as a wire or telephone circuit, that connects one or more stations to a communication control unit or connects one communications control unit with another. See also *link.*

line control block (LCB). A control block used for scheduling, sending, and receiving.

link. In SNA, the combination of the link connection and the link stations joining network nodes; a serial-by-bit connection under the control of SDLC. A link connection is a physical medium of transmission, such as a telephone wire or a microwave beam. A link includes a physical medium of transmission (a line), a protocol (SDLC), and associated communication devices and programming; it is both logical and physical.

load balancing. In TCAM extended networking, the technique for balancing the message flow between any pair of TCAM nodes by assigning different paths to different messages flowing between them.

lock mode. A mode in which an external logical unit (LU) entering an inquiry message for an application program is ensured that the next message it receives is a reply from the application program. See *conversational mode, extended lock mode, message lock mode.*

log. A collection of messages or message segments placed on a secondary-storage device for accounting or data collection purposes.

log data set. A data set consisting of the messages or message segments recorded on a secondary-storage medium by the TCAM logging facility.

logging service facility. A TCAM service facility that selectively causes incoming or outgoing messages or message segments to be copied onto tape or disk. The log produced by the logging service facility provides a record of message traffic through the message control program (MCP).

logical message. A user-defined message, consisting of one or more related units of data in a transmission, ending with an end-of-message code. Contrast with *physical message.*

logical unit (LU). In SNA, a port through which an end user gains access to the SNA network to communicate with another end user and through which the end user uses the functions provided by the SSCP. An LU can have at least two sessions--one with the SSCP, and one with another LU--and may be able to have many sessions with other LUs. Contrast with *physical unit.* See *host LU, auxiliary LU, primary logical unit, secondary logical unit.* See also *system services control point.*

logical unit (LU) services. In SNA, capabilities in a logical unit to: (a) receive requests from an end user and, in turn, issue requests to perform the requested functions, typically for session initiation; (b) receive requests from the SSCP to activate LU-LU sessions through Bind Session requests; and (c) provide session presentation and other services for LU-LU sessions.

logon mode. In VTAM, a set of session parameters specified in a logon mode table entry for communication with a logical unit. See also *session parameters.* Synonymous with *bind image.*

logon mode table. In VTAM, a set of entries for one or more logon modes. Each logon mode is identified by a logon mode name. Synonymous with *bind image table.*

logtype entry. A terminal-table entry associated with a queue on which complete messages reside while awaiting transfer by the logging service facility. A logtype entry is not needed if message segments only are to be logged.

LU. Logical unit.

LU-LU half-session. A half-session in which the session involved is an LU-LU session.

LU-LU session. In SNA, a session between two logical units (LUs) in an SNA network. It provides communication between two end users, or between an end user and an LU services component.

LU-LU session initiation. The process that begins with a session-initiation request from a logical unit (LU) to a system services control point and culminates in activation of an LU-LU session. See also *session activation.*

LU-LU session termination. The process that begins with either a session-termination request from a logical unit to a system services control point, or an Unbind request from one logical unit to another, and that culminates in deactivation of an LU-LU session. See also *session deactivation.*

LU-LU session type. The classification of an LU-LU session in terms of the specific subset of SNA protocols and options required by the logical units for that session. LU-LU session types 0, 1, 2, 3, 4, 6, and 7 are defined in SNA.

LU services. See *logical unit services.*

LU services manager. An SNA component that provides a logical unit (LU) with network services and end-user to end-user services. The LU services manager provides services for all half-sessions within the LU.

main storage unit. Synonym for *buffer unit.*

mandatory cryptographic session. A cryptographic session in which all outgoing data is enciphered and all incoming data is deciphered. Contrast with *selective cryptographic session.*

MCP. Message control program.

MCP definition. The collection of macro-language statements by which the network is defined to TCAM in the resource-definition section of the message control program (MCP).

message. In TCAM, a unit of data transmitted from one point to another. See *logical message, physical message.*

message control program (MCP). A general term referring to any specific implementation of TCAM, including initialization and termination routines, resource management routines, message handling routines, and service facilities.

message error record. Synonym for *error record.*

message handler (MH). A sequence of user-specified macro instructions and basic assembler language instructions that invoke routines that examine and process control information in message headers and perform functions necessary to prepare messages for forwarding to their destinations. See *application message handler, device message handler, internodal message handler.* See also *delimiter macro instruction, functional macro instruction.*

message header. The leading part of a message that contains information such as the source or destination code of the message, the message priority, and the type of message. See also *header.*

message lock mode. A type of lock mode in which an external logical unit (LU) is in lock mode for the duration of a single inquiry and reply. Contrast with *extended lock mode.* See also *station lock.*

message priority. The order in which messages in a destination queue are transmitted to a destination. Higher-priority messages are forwarded before lower-priority messages. See also *route transmission priority, station transmission priority.*

message queue data set. A TCAM data set that contains one or more destination queues. A message queue data set contains messages that have been processed by the incoming group of a message handler and are waiting for TCAM to dequeue them, route them through an outgoing group of a message handler, and send them to their destinations. Up to three message queue data sets (one in main storage, one on reusable disk, and one on nonreusable disk) may be specified for a TCAM message control program.

message routing. A message control program (MCP) function that determines the correct destination for each message received by the MCP and places the message on the appropriate destination queue. See *affinity-based routing, invariant routing, transaction-based routing.* See also *routing by destination, routing by key.*

message segment. The portion of a message that is contained in a single request unit (RU).

message text. Synonym for *text.*

message unit. In SNA, a general term for the unit of data processed by any layer; for example, a basic information unit, a path information unit, or a request/response unit (RU).

MH. Message handler.

mode name. The name of an entry in the logon mode table.

multiple-domain network. A network with more than one system services control point (SSCP). Contrast with *single-domain network.*

Multiple Virtual Storage (MVS). An IBM program product whose full name is the Operating System/Virtual Storage (OS/VS) with Multiple Virtual Storage/System Product for System/370. It is a software operating system controlling the execution of programs.

MVS. Multiple Virtual Storage operating system.

NAU. Network addressable unit.

NAU services. In SNA, the functions provided by the NAU services manager layer and the FMD services layer.

NCP. (1) Advanced Communications Function for the Network Control Program. An IBM program product that provides communication controller support for single-domain, multiple-domain, and interconnected network capability. (2) A general term for a program that is generated by the user from a library of IBM-supplied modules and controls the operation of a communication controller.

negative response. In SNA, a response indicating that a request did not arrive successfully or was not processed successfully by the receiver. Contrast with *positive response.* See *exception response.*

negotiable bind. In SNA, a function that allows two LU-LU half-sessions to negotiate the parameters of a session when the session is being activated.

network. In data processing, a user application network. See *public network, SNA network, user application network.*

network address. In SNA, an address, consisting of subarea and element fields, that identifies a link, a link station, or a network addressable unit (NAU). Subarea nodes use network addresses; peripheral nodes use local addresses. The boundary function in the subarea node to which a peripheral node is attached transforms local addresses to network addresses and vice versa. See *local address, TCAM network address.* See also *network name.*

network addressable unit. In SNA, a logical unit (LU), a physical unit (PU), or a system services control point (SSCP).

network control program. A program, generated by the user from a library of IBM-supplied modules, that controls the operation of a communication controller.

Network Control Program (NCP). An IBM program product that provides communication controller support for single-domain, multiple-domain, and interconnected network capability. Officially, the Advanced Communications Function for the Network Control Program.

network operator. In SNA, a person or program responsible for controlling the operation of all or part of a network.

network services procedure error (NSPE). A request unit that is sent by a system services control point (SSCP) to a logical unit (LU) when a procedure requested by that LU has failed.

Network Terminal Option (NTO). An IBM program product that allows certain non-SNA devices to participate in sessions with SNA application programs in the host processor. NTO converts non-SNA protocol to SNA protocol when data is sent to the host from a non-SNA device and reconverts SNA protocol to non-SNA protocol when data is sent back to the device.

networking. In a multiple-domain network, communication between domains. See *extended networking.*

NIB. Node initialization block.

no response. In SNA, a value in the form-of-response-requested field of the request header (RH) indicating that no response is to be returned to the request, whether or not the request is received and processed successfully. Contrast with *definite response, exception response.*

node. In SNA, a junction point in a network that contains a physical unit (PU). A node contains network addressable units, path control components, and may contain boundary function. See *TCAM node.*

node identifier. That portion of the TCAM network address of a resource that indicates which TCAM node provides the message queuing for that resource. See *resource identifier.*

node initialization block (NIB). In VTAM, a control block associated with a particular node or session that contains information used by the VTAM application program to identify the node or session and to indicate how communication requests on a session are to be handled by VTAM.

node table. For TCAM extended networking, a main-storage table that associates each node identifier with internodal destination queues.

non-SNA terminal. A terminal that supports non-SNA protocols; for example, channel-attached 3270 Information Display System or devices supported by Network Terminal Option (NTO) that use binary synchronous protocols. Contrast with *SNA terminal.*

normal flow. In SNA, a data flow that is used for most requests and responses. The expedited flow is independent of and used to control the normal flow. Requests and responses on a normal or expedited flow are processed sequentially within the path, but the expedited flow traffic may be moved ahead of the normal flow traffic within the path. Contrast with *expedited flow.*

NSPE. Network services procedure error.

NTO. Network Terminal Option.

OEF. Origin element field.

online retrieval. The function of a system service program that allows system operators to retrieve disk-queued messages based upon origin or destination, time of entry, or input or output sequence number.

OPCE. Operator control element.

open subroutine. A subroutine of which a replica must be inserted at each place in a computer program at which the subroutine is used. Contrast with *closed routine.*

operator command. Synonym for *operator control command.*

operator control. Synonym for *basic operator control, extended operator control.*

operator control command. A command entered from an operator control station to examine or alter the status of the TCAM system during execution of TCAM.

operator control element (OPCE). A unit assigned to each operator control command that is used by the operator control routines to process the command.

operator control station. Synonym for *basic operator control station, basic primary operator control station, basic secondary operator control station, extended operator control station, extended primary operator control station, extended secondary operator control station.*

option field. A storage area containing data relating to a particular external logical unit (LU) or application program. Certain message-handler routines that need origin- or destination-related data to perform their functions have access to data in an option field. User-written message-handler exit routines also have access to data in an option field.

option table. A table that contains option fields of user-provided information, using certain TCAM macro instructions, related to external logical units (LUs) or application programs.

origin. An external logical unit (LU) or application program from which a message or other data originates. See also *destination.*

outbuffer subgroup. The part of a message handler (MH) outgoing group that operates on each segment of an outgoing message.

outgoing group. That portion of the message handler that handles messages sent from the message control program (MCP) to any external logical units (LUs) or application programs. Contrast with *incoming group.*

outheader subgroup. The part of a message handler (MH) outgoing group that operates on all or part of an outgoing message header.

outmessage subgroup. The part of a message handler (MH) outgoing group that specifies actions to be taken after the entire message has been sent to an external logical unit (LU), or when special processing or error conditions are detected.

output data set. A data set that contains the messages or records returned from an application program to the message control program by a process entry in the terminal table. Contrast with *input data set.*

path switch. A field in the option table that determines whether a given subgroup is to be executed for a message.

PCB. Process control block.

physical message. The data entered on a link during a complete transmission sequence, from the first byte of data to the end of the transmission character. Contrast with *logical message.* In SNA, synonym for *RU chain.*

PLU. Primary logical unit.

POF restart. Point-of-failure restart.

point-of-failure (POF) restart. A type of warm restart of the message control program (MCP) that uses incident records to update an environment record when the system is restarted following closedown or system failure. Contrast with *point-of-last-environment restart.* See *cold restart, warm restart.*

point-of-last-environment (POLE) restart. A type of warm restart of the message control program (MCP) that ignores incident records when the system is restarted following closedown or system failure. Contrast with *point-of-failure restart.* See *cold restart, warm restart.*

POLE restart. Point-of-last-environment restart.

positive response. In SNA, a response unit that indicates that a request was successfully received and processed. Contrast with *exception response, negative response.*

prefix. Synonym for *buffer prefix.*

presentation services. Synonym for *session presentation services.*

primary end of a session. Deprecated term for *primary half-session.*

primary half-session. The half-session that sends the session-activation request. Contrast with *secondary half-session.* See also *primary logical unit.*

primary logical unit (PLU). In SNA, a logical unit that contains the primary half-session for a particular LU-LU session. A PLU issues a Bind Session command to establish an LU-LU session. Contrast with *secondary logical unit.* See also *logical unit.*

primary operator control station. See *basic primary operator control station, extended primary operator control station.*

priority. Synonym for *message priority, station transmission priority.*

process control block (PCB). A message control program (MCP) data area that is necessary for communication between the MCP and an application program.

process entry. A terminal-table entry that represents an application program. One entry must be defined for each queue to which an application program can issue a GET or READ macro instruction, and at least one entry must be defined for all PUT and WRITE macro instructions issued from the same application program.

process queue. A destination queue for an application program. See *destination queue.*

protocol. In SNA, the meanings of, and the sequencing rules for, requests and responses used for managing the network, transferring data, and synchronizing the states of network components. SDLC, BSC, and start-stop (SS) are link protocols.

public network. A network established and operated by communication common carriers or telecommunication Administrations for the specific purpose of providing circuit-switched, packet-switched, and leased-circuit services to the public. Contrast with *user application network.*

QCB. Queue control block.

queue. (1) A line or list formed by items in a system waiting for service; for example, tasks to be performed or messages to be transmitted in a message-routing system. (2) To arrange in, or form a queue. See also *queuing.*

queue control block (QCB). A control block that is used to regulate the sequential use of a programmer-defined facility among requesting tasks.

queuing. The programming technique used to handle messages that are awaiting delivery. See also *queue.*

quick closedown. A closedown in which message traffic is stopped as soon as any messages in the process of being sent or received at the time the request for closedown is received are transmitted. Contrast with *flush closedown.*

read-ahead queue. An area of main storage from which an application program obtains work units in advance of their being requested by the application program.

record. A collection of related data or words, treated as a unit; for example, in stock control, each invoice could constitute one record.

reply. (1) In TCAM, response to an inquiry. (2) In SNA, a request unit sent only in reaction to a received request unit. For example, Quiesce Complete is the reply sent after receipt of Quiesce at End of Chain.

request. In SNA, a message unit that signals initiation of a particular action or protocol. For example, Initiate Self is a request for activation of an LU-LU session.

request header (RH). In SNA, an RU header that precedes a request unit.

request/response header (RH). In SNA, control information preceding a request/response unit (RU) that specifies the type of RU (request unit or response unit) and contains control information associated with that RU. See also *request/response unit, connection point manager*.

request/response unit (RU). In SNA, a general term for a request unit or a response unit.

request parameter list (RPL). In VTAM, a control block that contains the parameters necessary for processing a request for data transfer, for establishing or terminating a session, or for some other operation.

request unit (RU). In SNA, a message unit that contains control information such as a request code or function management (FM) headers, end-user data, or both.

resource identifier. That portion of the TCAM network address of a resource that uniquely identifies the resource within the message control program (MCP) providing the message queuing for that resource. See also *node identifier*.

resource management block (RMB). A collection of control blocks all of which are associated with a particular external logical unit (LU). The RMB contains a line control block (LCB), a station control block (SCB), a savearea/workarea (SAU), and a request parameter list (RPL).

resource table. In TCAM extended networking, a main-storage table that associates each resource identifier with an external logical unit (LU) or application program.

response. In SNA, a message unit that acknowledges receipt of a request. A response consists of a response header, a response unit, or both.

response header (RH). A header, optionally followed by a response unit, that indicates whether the response is positive or negative and may contain a pacing response. See also *isolated pacing response, negative response, pacing response, positive response*.

response unit (RU). In SNA, a message unit that acknowledges a request unit; it may contain prefix information received in a request unit. If it is positive, the response unit may contain additional information (such as session parameters in response to a Bind Session request). If it is negative, the response unit may contain sense data that defines the exception condition.

RH. Request/response header.

RMB. Resource management block.

routing. Synonym for *message routing*.

routing affinity. A temporary relationship between a source and a destination.

routing by destination. Message routing based upon a destination name. Contrast with *routing by key*.

routing by key. Message routing based upon a key, which matches a definition in the key table. The key identifies the destination of a message or special processing to be done on that message. Contrast with *routing by destination*.

routing key. Synonym for *key*.

routing key table. Synonym for *key table*.

RPL. Request parameter list.

RU. Request/response unit.

RU chain. In SNA, a set of related request units (RU) that are consecutively transmitted on a particular normal or expedited data flow. The request unit (RU) chain is the unit of recovery. If one of the request units (RUs) in the chain cannot be processed, the entire chain is discarded.

save/restore message queues (SMQ). The function of a system service program that saves unsent messages on sequential storage devices and restores them to an altered message control program (MCP) following a cold restart. This program also assists in recovery when the message queue data set on nonreusable disk becomes full. The program may be used to obtain an online dump of unsent messages from one or more destination queues on disk.

scan pointer. A pointer that refers to the proper header field when the macro instruction that acts upon that field is given control. Some user-specified macro instructions use this pointer to locate the field on which they act and automatically move the pointer to the next field before passing control to the next macro instruction. The user must be aware of positioning of the scan pointer when designing the message handler.

SDLC. Synchronous data link control.

secondary end of a session. Synonym for *secondary half-session.*

secondary half-session. In SNA, the half-session that receives the session-activation request. Contrast with *primary half-session.* See also *secondary logical unit.*

secondary logical unit (SLU). In SNA, the logical unit that contains the secondary half-session for a particular LU-LU session. Contrast with *primary logical unit.*

secondary operator control station. Synonym for *basic secondary operator control station, extended secondary operator control station.*

segment. A portion of a message that can be contained in a buffer. See *BIU segment.*

selective cryptographic session. A cryptographic session in which an application program is allowed to specify the request units to be enciphered. Contrast with *mandatory cryptographic session.*

service facility. An auxiliary routine that runs under control of the message control program (MCP) and is invoked when needed by user code in the MCP. on an as-needed basis. Contrast with *system service program, utility.*

session. In SNA, a logical connection between two network addressable units that can be activated, tailored to provide various protocols, and deactivated, as requested. The session-activation request and response can determine options relating to the rate and currency of data exchange, the control of contention and error recovery, and the characteristics of the data stream. Sessions compete for network resources such as the links within the path control network. See *half-session, LU-LU session.* See also *LU-LU session type, PU-PU flow.*

session activation. In SNA, the process of exchanging a session activation request and a positive response between network addressable units. Contrast with *session deactivation.* See also *start.*

session-activation request. In SNA, a request that activates a session between two network addressable units and specifies session parameters that control various protocols during session activity. Contrast with *session deactivation request.*

session control. (1) One of the components of transmission control. Session control is used to purge data flowing in a session after an unrecoverable error occurs, to resynchronize the data flow after such an error, and to perform cryptographic verification. (2) A request/response unit (RU) category used for requests and responses exchanged between the session control components of a session and for session activation or deactivation requests or responses.

session count. (1) The number of currently active LU-LU sessions for a particular logical unit. (2) The number of currently active sessions for a particular virtual route.

session deactivation. The process of exchanging a session-deactivation request between two network addressable units. Contrast with *session activation.* See also *stop.*

session deactivation request. A request that deactivates a session between two network addressable units. Contrast with *session activation request.*

session end. Synonym for *half-session.*

session information block (SIB). A control block that contains information about a particular SNA session.

session initiation. Synonym for *LU-LU session initiation.* See also *LU-LU session termination.*

session-initiation request. An initiate or logon request from a logical unit (LU) to a systems services control point (SSCP) so that an LU-LU session can be activated.

session parameters. In SNA, the parameters that specify or constrain the protocols (such as bracket protocol) for a session between two network addressable units. See also *logon mode.*

session presentation services. A component of the function management data (FMD) services layer that provides, within LU-LU sessions, services for the application programmer or terminal operator such as formatting data to be displayed or printed.

session sequence number. In SNA, a sequentially incremented identifier that is assigned by data flow control to each request unit on a particular normal flow of a session, typically an LU-LU session, and is checked by transmission control. The identifier is carried in the transmission header of the path information unit and is returned in the transmission header of any associated response.

session termination. Synonym for *LU-LU session termination.*

SIB. Session information block.

single-domain network. A network with one system services control point (SSCP). Contrast with *multiple-domain network.*

single entry. A terminal-table entry associated with a single external logical unit (LU) or application program. Contrast with *distribution entry.*

SLU. Secondary logical unit.

SMQ. Save/restore message queues.

SNA. Systems Network Architecture.

SNA network. The part of a user-application network that conforms to the formats and protocols of Systems Network Architecture. It makes possible reliable transfer of data among end users and provides protocols for controlling the resources of various network configurations. An SNA network consists of network addressable units, boundary function components, and the path control network.

SNA node. A node that uses SNA protocols.

SNA session. A logical connection, established between two network addressable units (NAUs), to allow them to communicate. The session is uniquely identified by a pair of network addresses identifying the origin and destination NAUs of any transmissions exchanged during the session. See *LU-LU session, pseudo LU-LU session.*

SNA terminal. A terminal that supports SNA protocols. Contrast with *non-SNA terminal.*

SSCP. System services control point.

SSP. (1) In TCAM, a system service program. (2) Advanced Communications Function for the Systems Support Programs. An IBM product program made up of a collection of utilities and small programs. SSP is required for operation of the NCP.

start. For external logical units (LUs) in TCAM, the state in which an LU is able to enter an LU-LU session.

start-stop (SS) transmission. Asynchronous transmission in which a group of bits is preceded by a start bit that prepares the receiving mechanism for the reception and registration of a character and is followed by at least one stop bit that enables the receiving mechanism to come to an idle condition pending the reception of the next character. Contrast with *binary synchronous transmission, synchronous data link control.*

startup/restart message generation facility. A TCAM service facility that generates and sends tailored messages to external logical units (LUs) when the message control program (MCP) is started or restarted.

station. One or more terminals or devices at a particular location; for example, an external logical unit (LU).

station lock. A facility that maintains a connection between a station and an application program to ensure that the next message received by the station, after it enters an inquiry message, is a reply to that inquiry. See also *extended lock mode, lock mode, message lock mode.*

station transmission priority. The relative order of the host sending and receiving messages. Host sending has priority over host receiving. See *message priority.*

stop. In TCAM, the state in which a logical unit (LU) is not able to enter an LU-LU session. This state also terminates any existing LU-LU sessions involving that LU.

symbol. In assembler language, a character or character string that represents addresses or arbitrary values. A symbol must meet the following requirements: (a) A symbol may consist of no more than eight characters, the first character being a letter (A through Z, $, #, or @) and the other characters being either letters or digits. (b) No blanks or special characters are allowed in a symbol.

synchronous data link control (SDLC). A discipline conforming to subsets of the Advanced Data Communication Control Procedure (ADDCP) of the American National Standards Institute and High-Level Data Link Control (HDLC) of the International Standards Organization, for managing synchronous, code-transparent, serial-by-bit information transfer over a link connection. Transmission exchanges may be duplex or half-duplex over switched or nonswitched links. The configuration of the link connection may be point-to-point, multipoint, or loop. Contrast with *binary synchronous transmission, start-stop transmission.*

system service program (SSP). An IBM-supplied or user-supplied program that performs system-oriented auxiliary functions in support of the MCP. System service programs run under the control of the initiator as attached subtasks. Contrast with *service facility, utility.* See also *basic operator control, extended operator control, online retrieval, save/restore message queues, internodal awareness, internodal sequence number synchronization.*

Note: The abbreviation SSP has two references. See also SSP.

system services control point (SSCP). In SNA, a focal point within an SNA network for managing the configuration, coordinating network operator and problem determination requests, and providing directory support and other session services for end users of the network. Multiple SSCPs, cooperating as peers, can divide the network into domains of control, with each SSCP having a hierarchical control relationship to the physical units and logical units within its domain.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through and controlling the configuration and operation of networks.

TCAM. Advanced Communications Function for TCAM. An IBM program product that provides queued message handling. TCAM, Versions 1 and 2, are telecommunications access methods, but TCAM, Version 3, is a message handling subsystem.

TCAM application program. A program that is user written and interfaces with the message control program (MCP) using READ, WRITE, CHECK, GET, or PUT macro instructions.

TCAM destination address field (TDAF). A field in the fixed header prefix of a message that contains the TCAM network address of the destination of the message. Contrast with *TCAM origin address field.*

TCAM host logical unit (LU). A TCAM-generated logical unit (LU) that is the access method control block (ACB) interface to VTAM, for example, PROGID. External LUs must establish a session with a TCAM host LU in order to use TCAM services. See *host logical unit (LU).*

TCAM network address. A unique identifier for an application program or an external logical unit (LU) in an extended networking environment. A TCAM network address consists of a node identifier and a resource identifier. See also *node identifier, resource identifier.*

TCAM node. A message control program (MCP) to which there has been assigned a node identifier. See also *node identifier.*

TCAM origin address field (TOAF). A field in the fixed header prefix of a message that contains the TCAM network address of the originator of the message. Contrast with *TCAM destination address field.*

TCAM subtask table (TST). A table containing entries for programs eligible to run as initiator subtasks.

TCAM system. A subsystem controlled by a single message control program (MCP) that communicates with a collection of external logical units (LUs) and application programs.

TDAF. TCAM destination address field.

terminal table. In TCAM, an ordered collection of information about each origin or destination of messages in the network. See also *terminal-table entry.*

terminal-table entry (TTE). The information in the terminal table that identifies each origin or destination of messages in the network. See *cascade entry, logtype entry, process entry, single entry.*

termination. Synonym for *LU-LU session termination.*

text. That part of the message that is not the header or control information.

text buffer. A buffer containing any segment of a message other than the first segment, which is contained in a header buffer. Contrast with *header buffer.*

TOAF. TCAM origin address field.

transaction-based routing. Message routing in which messages are routed to their destinations individually, according to one or more destination names or routing keys entered in the message header by the originator. See *affinity-based routing, invariant routing.* See also *routing by destination, routing by key.*

transmission category. In TCAM extended networking, utility sessions. All messages in the same transmission category have similar characteristics and should be handled similarly. For example, messages flowing in an inquiry/reply application and messages flowing in a high-volume, low-priority data collection application are placed in different transmission categories. Different versions of the following TCAM techniques and capabilities may be applied to messages in different transmission categories: queuing medium, message priority, sequence checking, error handling, load balancing, and data staging.

transmission services profile. In SNA, a specification in a session-activation request of transmission control protocols (such as session-level pacing and the usage of session-control requests) to be supported by a particular session. Each defined transmission services profile is identified by a number.

TST. TCAM subtask table.

TTE. Terminal-table entry.

unit. Synonym for *buffer unit, work unit.*

user application network. A configuration of data processing products, such as processors, controllers, and stations, established and operated by users for the purpose of data processing or information exchange, which may use services offered by common carriers or telecommunications Administrations. Contrast with *public network.*

utility. In TCAM, an auxiliary routine designed to support the message control program (MCP), which runs under the control of the operating system. Contrast with *system service program, service facility.*

utility session. In TCAM extended networking, a pair of LU-LU sessions between TCAM nodes. One utility session is established between each pair of TCAM nodes for each transmission category defined for the pair. Data messages being routed from TCAM node to TCAM node flow on the utility session corresponding to their transmission category.

VTAM. (1) Advanced Communications Function for VTAM, an IBM program product. (2) Virtual Telecommunication Access Method.

VTAM application program. A program that has opened an access method control block (ACB) to identify itself to VTAM and can now issue VTAM macro instructions. See *TCAM application program.*

warm restart. Restart of TCAM following either a quick or a flush closedown. The TCAM checkpoint/restart service facility restores the TCAM environment as nearly as possible to its condition before closedown or failure. Contrast with *cold restart.* See *point-of-failure restart, point-of-last-environment restart.*

WATS. Wide Area Telephone Service, which provides a special line on which the subscriber may make unlimited calls to certain zones on a direct-distance-dialing basis for a flat monthly charge.

work area. An area of storage related to an application program that receives messages or records transferred to the application program from TCAM by GET or READ macro instructions, and from which messages or records are transferred to TCAM by PUT or WRITE macro instructions.

work unit. The amount of data transferred from TCAM to an application program by a single GET or READ macro instruction or transferred from an application program to TCAM by a single PUT or WRITE macro instruction. A work unit may be a message or a record.

zero-length buffer. A buffer that is sent to the message handler to indicate that there is an error on the link. If user code does not execute correctly for a zero-length buffer, the programmer must check for zero length and branch around the code that does not execute correctly.

Index

Advanced Communications Function for TCAM Version 3

READER'S COMMENT FORM

Installation, Resource Definition, and Customization Reference (MVS)

Order No. SC30-3236-1

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

____

____

____

____

What is your occupation?______________________________

Number of latest Newsletter associated with this publication:____________________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

SC30-3236-1

Reader's Comment Form

Fold and tape | **Please Do Not Staple** | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Dept. E03
P.O. Box 12195
Research Triangle Park, N.C. 27709-2195

Fold and tape | **Please Do Not Staple** | Fold and tape

IBM®

Advanced Communications Function for TCAM Version 3

READER'S COMMENT FORM

Installation, Resource Definition, and Customization Reference (MVS)

Order No. SC30-3236-1

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Possible topics for comment are:

Clarity   Accuracy   Completeness   Organization   Coding   Retrieval   Legibility

If you wish a reply, give your name, company, mailing address, and date:

____

____

____

____

What is your occupation?____________________________

Number of latest Newsletter associated with this publication:________________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Reader's Comment Form

Fold and tape **Please Do Not Staple** Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Dept. E03
P.O. Box 12195
Research Triangle Park, N.C. 27709-2195

Fold and tape **Please Do Not Staple** Fold and tape

IBM®