GC30-2025-0

Systems

OS TCAM User's Guide

IBM

GC30-2025-0

Systems

OS TCAM User's Guide

Program No. 360S–CQ–548

Release No. 20.1

IBM

First Edition (January 1972)
This edition applies to Release 20.1 and to all subsequent releases of OS TCAM until otherwise indicated in new editions or Technical Newsletters. Changes are periodically made to the information herein; before using this publication with IBM systems or equipment, refer to the latest SRL Newsletter for the editions that are applicable and current.

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

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

Preface

The TCAM User's Guide is for systems analysts and programmers who must design, write, and install a TCAM program. It is both a guide for diagnosis and a problem determination handbook. The INTRODUCTION to the TCAM User's Guide names and briefly describes the four chapters and their appendixes.

An acronym list and a list of illustrations which is organized by chapter and by appendixes follow the table of contents.

Before you read this book, you should be familiar with the *OS TCAM Programmer's Guide and Reference Manual,* GC30-2024, and the *OS TCAM Concepts and Facilities,* GC30-2022. You will also find the *TCAM PLM,* GY30-2029, helpful.

Use this publication in conjunction with the publications shown in the following chart. Abbreviated titles refer to other publications throughout this publication. The chart below shows both the abbreviated and the full titles.

Abbreviated Title	**Full Title**	**Order No.**
Principles of Operation	*IBM System/360 Operating System: Principles of Operation*	GA22-6821
Utilities	*OS Utilities*	GC28-6586
System Control Blocks	*OS System Control Blocks*	GC28-6628
Guide to Reading Dumps	*Guide to Reading OS System Dumps*	GC28-6670
OS Service Aids	*IBM System/360 Operating System: Service Aids*	GC28-6719
TCAM Concepts and Facilities	*IBM System/360 Operating System: Telecommunications Access Method (TCAM) Concepts and Facilities*	GC30-2022
TCAM Programmer's Guide	*IBM System/360 Operating System: TCAM Programmer's Guide and Reference Manual*	GC30-2024
I/O Supervisor PLM	*IBM System/360 Operating System: Supervisor Logic*	GY28-6616
TCAM PLM	*Telecommunications Access Method (TCAM) Program Logic Manual*	GY30-2029

Contents

Figures

Chapter 1. Overview

Chapter 2. TCAM Coding Aids

Chapter 3. TCAM Problem Determination Aids

Chapter 4. TCAM Diagnostic Aids

Appendix A. TCAM Control Block Relationships

Appendix B. TCAM Macro Operand Summary

Acronym List

ABEND	Abnormal End
ACSMETH	Access Method Work Area
ACK	Positive Acknowledgment Character
APAR	Authorized Program Analysis Report
AVT	Address Vector Table
BSAM	Basic Sequential Access Method
BSC	Binary Synchronous Communications
CC	Chain Command
CCW	Channel Command Word
CD	Chain Data
COMWRITE	Common Write Routine
CPB	Channel Program Block
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSW	Channel Status Word
CVT	Communications Vector Table
DASD	Direct Access Storage Device
DCB	Data Control Block
DCT	Device Characteristics Table
DD	Data Definition
DEB	Data Extent Block
DLE	Data Link Escape Character
DSCB	Data Set Control Block
ECB	Event Control Block
ENQ	Enquiry Character
EOA	End of Address Character
EOB	End of Block Character
EOD	End of Day
EOM	End of Message Character
EOT	End of Transmission Character
ERB	Element Request Block
ETB	End Transmission Block Character
ETX	End of Text Character
EXCP	Execute Channel Program
FE	Field Engineering
ID	Identification
I/O	Input/Output
IOB	Input/Output Block
IOS	Input/Output Supervisor
JCL	Job Control Language
LCB	Line Control Block
LMWA	Locate Mode Work Area
MCP	Message Control Program
MH	Message Handler
MS	Main Storage
OBR	Outboard Recorder
OS	Operating System
PCB	Process Control Block
PCI	Program Controlled Interruption
PEWA	Process Entry Work Area
PSW	Program Status Word

PTF	Program Trouble Fix
QCB	Queue Control Block
QSAM	Queued Sequential Access Method
RCB	Resource Control Block
SCB	Station Control Block
SCT	Special Characters Table
SDR	Statistical Data Recorder
SIO	Start Input/Output Operation
SLI	Suppress Length Indication
STCB	Subtask Control Block
STX	Start of Text Character
TCAM	Telecommunications Access Method
TCB	Task Control Block
TCU	Transmission Control Unit
TIC	Transfer In Channel
TIOT	Task Input/Output Table
TOTE	Telecommunications on Line Test Executive
TP	Teleprocessing
TSO	Time Sharing Option
TWX	Teletypewriter exchange
UCB	Unit Control Block
VCON	V Type Address Constant

Introduction

You can use the OS TCAM User's Guide in three ways:

1. As a source of hints for originally coding your TCAM message control program and application programs.
2. For diagnosing a TCAM problem when you first try to run TCAM.
3. For problem determination during the initial stages of trouble shooting in a system that uses equipment provided by more than one vendor.

Chapter 1, OVERVIEW, is an enhancement of TCAM Concepts and Facilities. After you have become familiar with the *TCAM Concepts and Facilities* and *TCAM Programmer's Guide* manuals, Chapter 1 will provide a transition to the remaining chapters of this guide.

Chapter 2, TCAM CODING AIDS, discusses pre-assembly aids to help you code your TCAM program so that it will be as error-free as possible. The first section shows the functions of a TCAM program in proper coding order. The second section describes macros that you can include in your program to detect and handle errors in messages and in the teleprocessing network.

Chapter 3, TCAM PROBLEM DETERMINATION AIDS, suggests where you can look in your code when you have an error. Each possible problem area is discussed. Lists of the more common errors that can be made are given. Use this chapter to review your code before you first run a TCAM program. Use it also, when you have a problem, to review possible problem areas. In addition to errors in your code, Chapter 3 also summarizes other sources of errors, such as hardware, software, and those that might be caused by system console operators and terminal users.

Chapter 4, TCAM DIAGNOSTIC AIDS, tells you what information TCAM provides for your use in diagnosing problems, and how you can get copies of the information. The first section, *Gathering and Interpreting Data From Dumps,* covers the TCAM program and all the data sets that you can dump and print. This first section also suggests the kinds of errors that you can find, where to look for them, and, in some cases, what normal operations look like. The second section, *Using Operator Commands,* summarizes operator commands that you can issue to determine and alter the status of your TCAM system while it is running. The last section, *Normal End-of-Day Closedown,* suggests what data you might want to copy during your normal end-of-day closedown.

APPENDIX A includes charts showing TCAM control block linkages.

APPENDIX B is a summary of TCAM macros and their operands.

APPENDIX C is a field-by-field description of the output from a formatted TCAM dump.

APPENDIX D includes charts showing device configurations supported by TCAM.

Following is a general overview of TCAM. Read this before coding, to familiarize yourself with the facilities provided by TCAM.

Overview

What is TCAM?

TCAM is:

- A general purpose TP *access method* that provides facilities to exchange data between a CPU and remote terminals.
- A *control program* that optimizes allocating and scheduling a computer's resources in a real-time teleprocessing environment.

Resources optimized:

1. CPU time
2. Main storage
3. I/O paths (lines and channels)

- A *high-level language* composed of macro instructions designed specifically to facilitate building a TP network control program.

How do you invoke the facilities of TCAM?

Code a message control program (MCP) containing sections in which you:

- define the TP hardware—terminals and lines—to TCAM;
- define data sets in which TCAM queues incoming messages until they are sent to their destinations;
- construct message handlers to 1)translate, edit, and verify the validity of the input data; 2)route incoming and outgoing messages to their destinations; 3) invoke certain system functions such as logging;
- define an interface to application programs for message processing;
- specify which of TCAM's service facilities, operator control, checkpoint/restart, logging, debugging aids, on-line test you want to be included; and
- include routines to activate and deactivate the TP network.

Network Definition

At system generation time, be sure your UCBs are correct. Know your network configuration and what you have (features).

Macro instructions involved:

- *Line Group DCB macro:* defines a group of lines with similar characteristics (for instance, you might define a group of lines for IBM 1050 terminals by a line group DCB macro). This macro specifies information applicable to terminals as a group, such as the translation table to be used to translate incoming and outgoing messages for the terminals, the buffer size for buffers servicing lines in the group, and the message handler to handle messages to and from terminals assigned to lines in this group. You do not have to define your similar terminals in a line group. Each terminal may be defined with a unique DCB. The decision to place your terminals in a line group rather than having individual DCBs is based on the planned usage of the terminals (are they output only?) and on main-storage conservation.
- *TERMINAL macro:* defines an individual remote terminal to TCAM. Gives the terminal a name, specifies the type of queuing to use for messages sent to

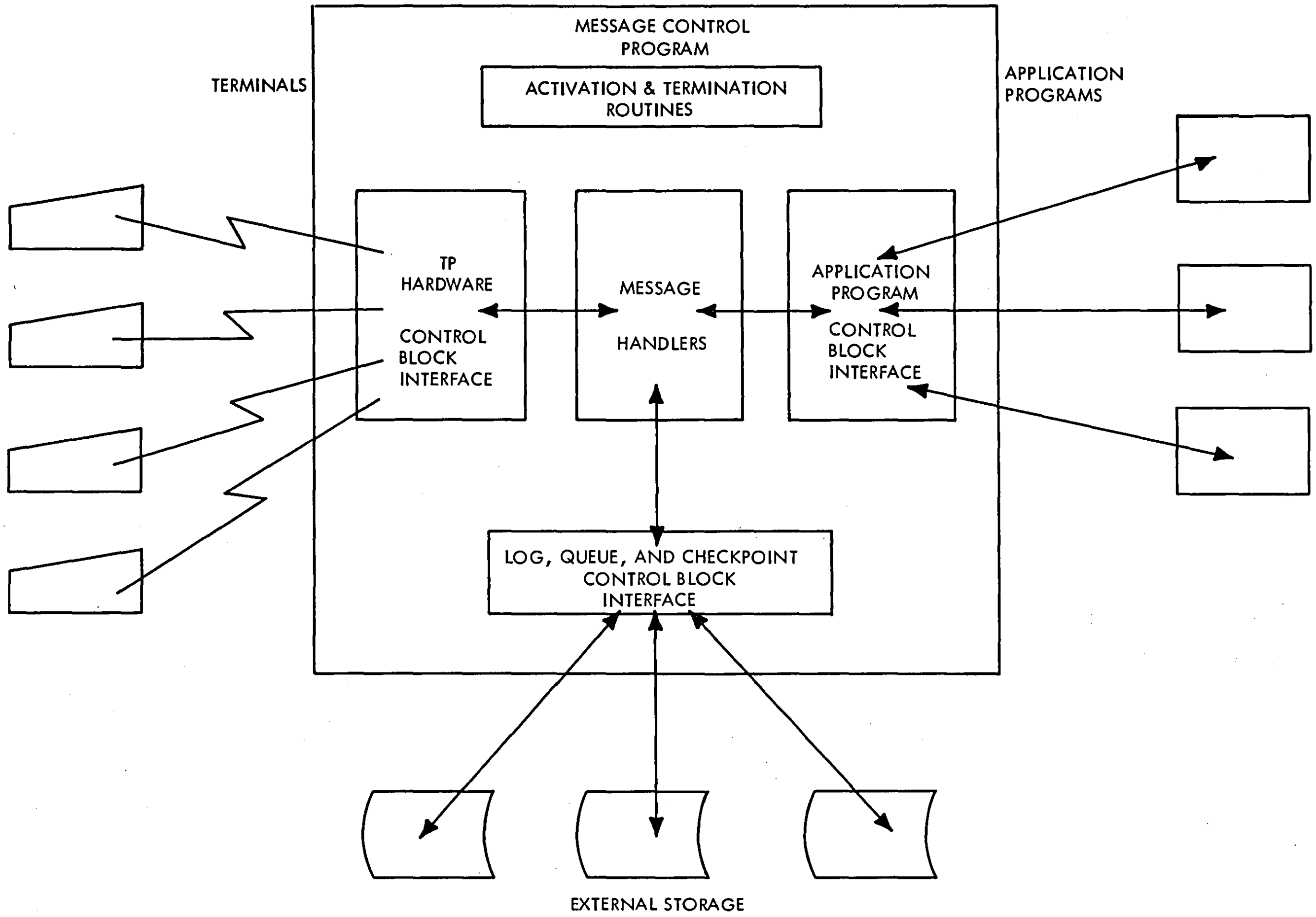


Figure 1. Overall Structure of the Message Control Program

this terminal, the addressing characters to use in addressing this terminal, this terminal's telephone number if it is on a dial line, etc.

- *INVLIST macro:* specifies the characters to invite (poll) each terminal on this line to enter data (one macro per line).

Tying it together:

The TERMINAL macro names the line group DCB macro for the line to which this terminal is assigned. The line group DCB macro names the INVLIST macros containing the invitation characters for each terminal on a line in the line group. The line group DCB macro also names a DD statement that specifies the hardware address of each line.

- Code one line group DCB macro for each group of lines to terminals with similar characteristics.
- Code one TERMINAL macro for each terminal in the network.
- Code one INVLIST macro for each line on which there are terminals that can enter data.

Starting TCAM

- *The TCAM MCP is just another problem program to OS.*
- Assembling, link-editing, and executing steps for a TCAM MCP are similar to those for any other problem program running under OS.

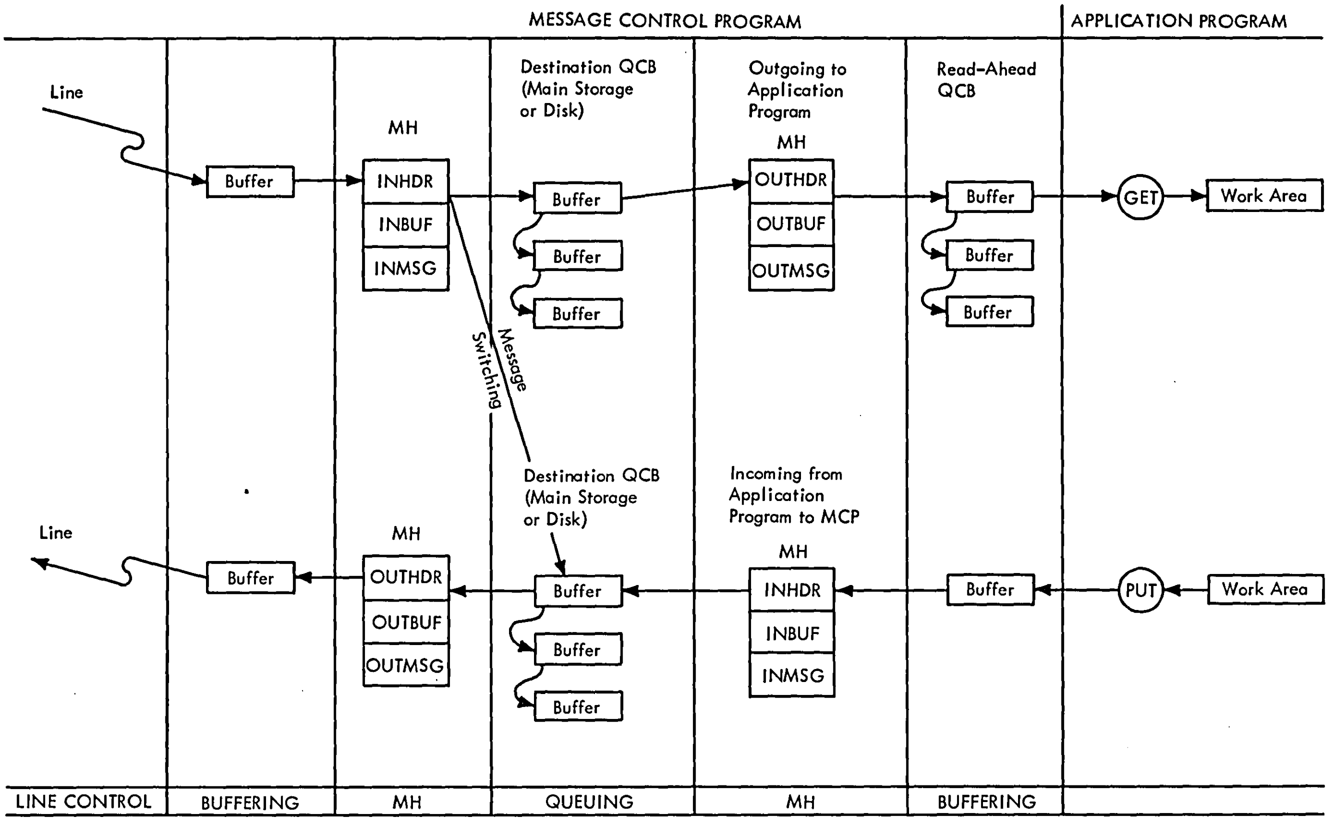


Figure 2. Overview of a TCAM Program

- A TCAM MCP normally executes as the highest-priority task in the highest-priority region or partition in the system (for performance reasons).
- You can issue any OS macro within the MCP but you must be aware of the system implications. That is, you significantly degrade MCP performance if you issue an OS WAIT as a result of an OS macro execution.
- You can start a TCAM MCP in three ways:

1. Place the appropriate execution JCL in the card reader and use the OS Reader/Interpreter to place the job in the system.
2. Catalog the MCP JCL in SYS1.PROCLIB and start the job from the system console with a START command. You can catalog different copies of the MCP and use the appropriate copy as your operational requirements vary.
3. Issue an ATTACH macro from another task.

Activate TCAM application programs any time after the MCP is activated; deactivate them independently of the MCP. If a message arrives in the MCP for an application program that is not currently active, TCAM places the message on the destination queue for that application program, and it remains there until the application program is activated and fetches the message with GET or READ macros.

TCAM application programs

- can be in a separate region or partition, or
- can be attached by including OS ATTACH macros after the OPEN macros but before the READY macro in the MCP activation and deactivation section;
- can also be attached with an ATTACH macro in line in the MCP message handler.

FUNCTIONAL GROUP	MACROS
Activation and Deactivation	INTRO OPEN READY CLOSE
Data Set Definition	DCB PCB
Terminal and Line Control	TTABLE OPTION LOGTYPE TPROCESS TERMINAL TLIST INVLIST
Message Handler Delimiter Macros	STARTMH INHDR INBUF INMSG INEND OUTHDR OUTBUF OUTMSG OUTEND

Figure 3. MCP Macro Instructions

- You must close down or detach TCAM application programs before closing the MCP.

More information on activating and deactivating the MCP and application programs and on the relationship between the MCP and application programs is contained in the *TCAM Programmer's Guide.*

Activating and Deactivating TCAM

Activate the TCAM program with INTRO, OPEN, and READY macros.

The INTRO macro

- performs the bulk of TCAM system initialization;
- establishes addressability for the MCP;
- has operands that specify various system-wide parameters dealing with buffering, type of start-up, queuing, operator control, checkpoint/restart, on-line test (TOTE), and diagnostic aids.
- Most operands can be specified or changed at execution time at the system console.

The OPEN macro:

- completes the initialization and activation of the MCP data sets;
- is required for each MCP data set represented by a DCB macro.

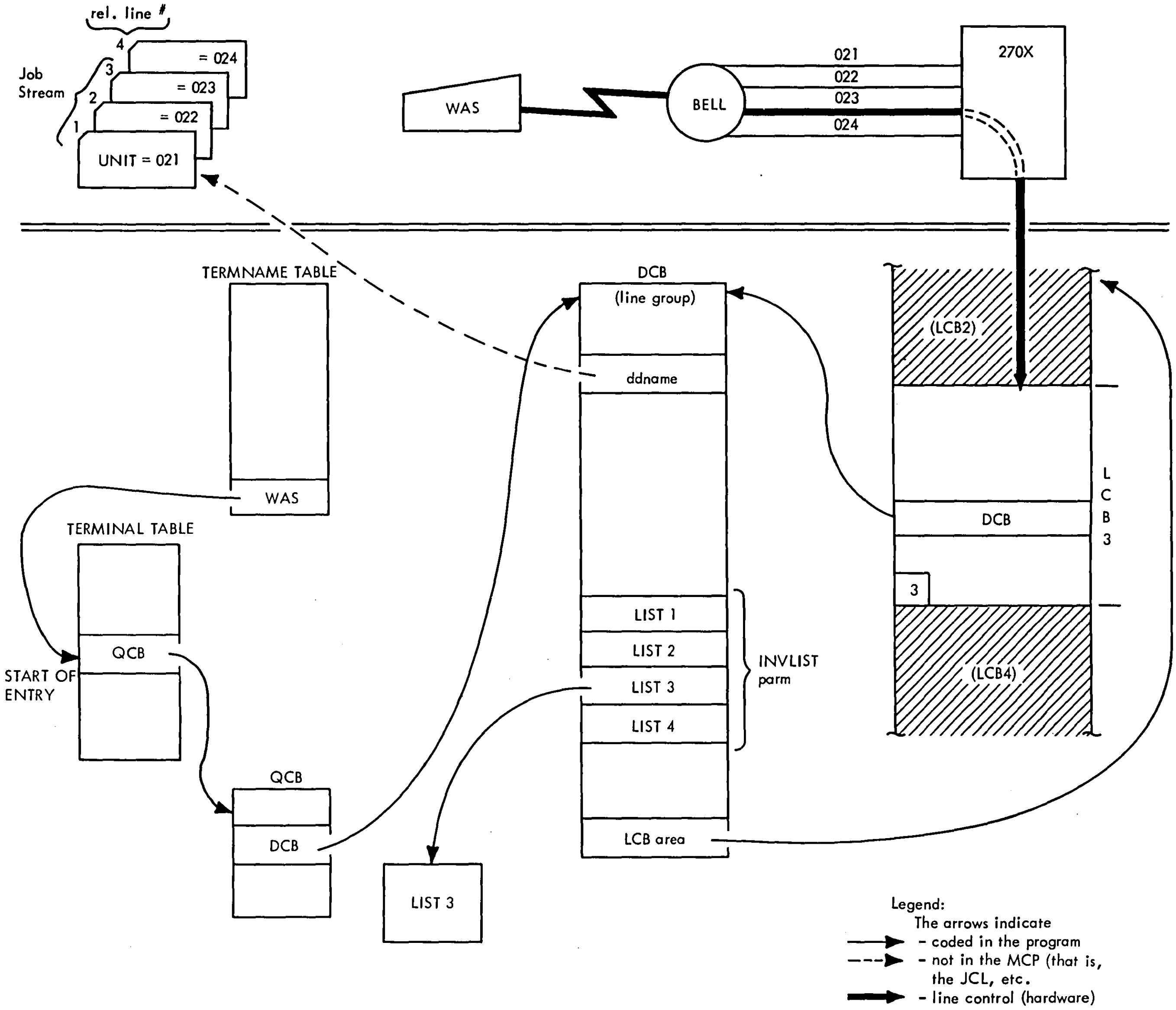


Figure 4. JCL, TCAM Network, and Control Block Relationship

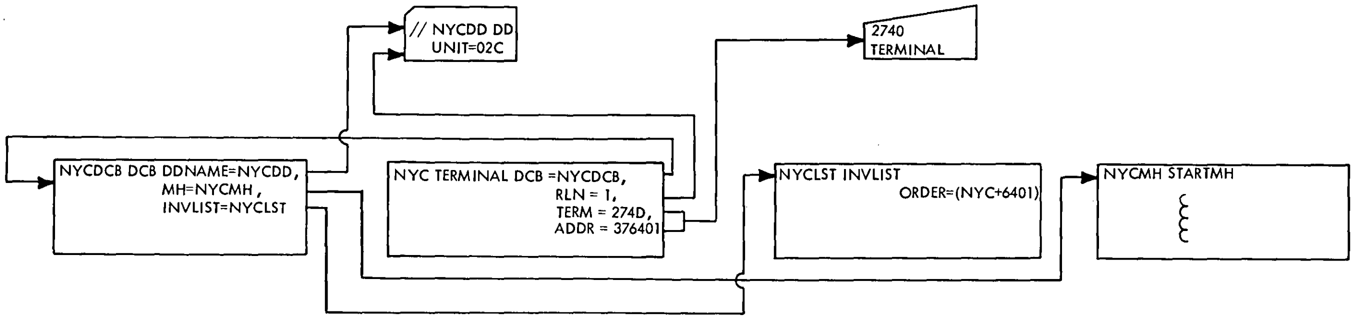


Figure 5. JCL, TCAM Network, and Macros Relationship

The READY macro:

- Completes the initialization and activation of the MCP; after READY executes, TCAM is ready to handle incoming messages.

Types of start-up (specified by an operand of INTRO):

- *Cold:* Start from scratch; ignore the previous environment.
- *Warm:* Use TCAM's checkpoint/restart facility to reconstruct the MCP environment as it existed before closedown, and start from that point.
- *Continuation:* Similar to warm, but restarts following a system failure rather than an orderly closedown, so that TCAM's checkpoint/restart facility is used in a somewhat different manner to achieve the same result—a reconstructed MCP environment without loss of completely received messages.

Deactivate with CLOSE macros, and with the MCPCLOSE macro or the SYSCLOSE operator command.

To close the MCP, deactivate your application programs, then issue an MCPCLOSE macro or a SYSCLOSE operator command specifying either a quick or a flush close.

Quick Close: TCAM stops message traffic on each line as soon as the current message is completely received or sent. When all traffic ceases, TCAM closes the MCP data sets and returns control to OS.

Flush Close: After the message currently being processed on each line is completely received or sent, TCAM sends all messages queued for terminals on that line to their destinations and closes the line. When all lines are closed, TCAM closes the MCP data sets and returns control to OS.

Message Flow

TCAM places a message coming into the MCP over a line into buffers that you have assigned to that line for input operations.

The message goes through the incoming group of the message handler for the line, and is then queued in a destination queue. If the destination queue is located on disk, the buffers are released; if the queue is in main storage, the buffers contain the message in the main-storage queue that you defined with the operand of the INTRO macro (MSUNITS=).

If the destination is an application program, TCAM reads the message from the disk or the main-storage queue into buffers, and sends it through the outgoing group of the message handler for the application program. It is then placed on a special main-storage read-ahead queue until it is moved into the application-program work area with a GET or READ macro.

Messages transferred from the application-program work area to the MCP with PUT or WRITE macros are put into buffers and sent through the incoming group of the message handler for the application program, after which they are placed on a destination queue on disk or in main storage.

If the destination of the message is a terminal, TCAM reads the message from the destination queue on disk or in main storage into buffers, and sends it through the outgoing group of the message handler for the line. It is then sent to the destination terminal. Once the message has been transmitted, the units making up the buffers that contained it are available for reallocation.

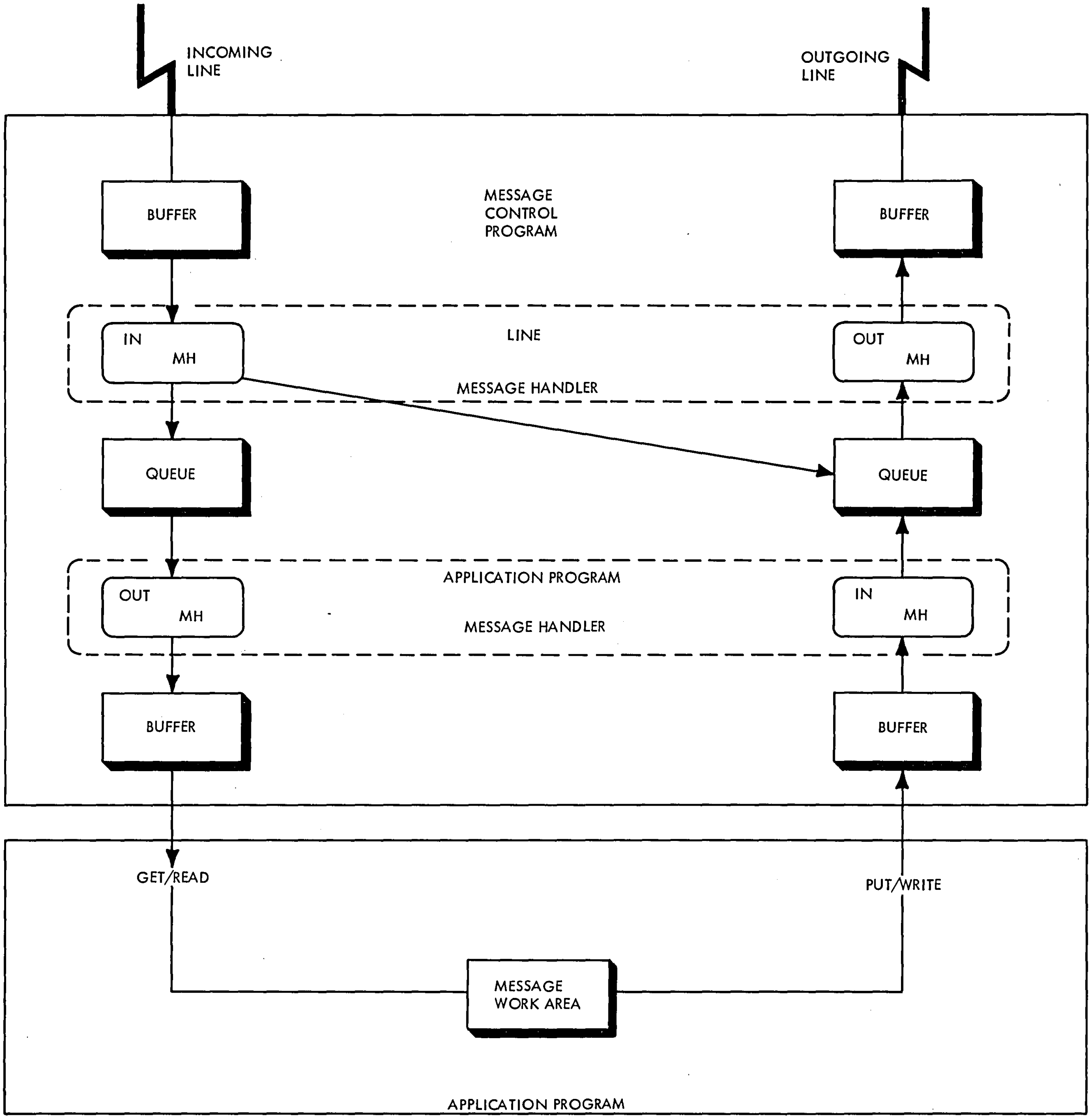


Figure 6. TCAM Message Flow

Buffering Scheme

Various size buffers are constructed from fixed-sized *units* that are taken from a *unit pool,* whose size you define:

- Each *unit* has a 12-byte prefix containing control information.
- In addition, each *buffer* has a prefix in which TCAM keeps message-related control information.

- The buffer holding the first piece of a message has a 30-byte prefix.
- Buffers holding subsequent pieces of the message have 23-byte prefixes.

You specify the size and number of buffers to handle I/O over the lines in a line group in the line group DCB macro. You specify the size and number of buffers to handle I/O between the TCAM message control program and an application program in the PCB macro.

Before starting an I/O operation for a line, TCAM constructs a user-specified number of buffers from units in the unit pool, and assigns them to the line. If enough units are not currently available to construct the required number of buffers, TCAM defers the I/O operation until units are available. The line group DCB macro allows you two options for allocating buffers:

1. You can specify that a relatively small number of buffers be allocated initially to handle an I/O operation, and that more buffers are to be allocated with PCI interrupts as they are needed (PCI=A,A).
2. You can specify that a fixed number of buffers, sufficient to hold the entire message being sent or received, is to be available before I/O begins (PCI=N,N).

Dynamic allocation (using PCI) improves performance by breaking work into small pieces over a period of time. With dynamic allocation (option #1), fewer buffers are tied up at any one time in an I/O operation than with static buffering (option #2), but CPU utilization is higher, and incoming data can be lost since TCAM may not be able to replace buffers as fast as they are filled (perhaps because traffic is heavy and no units are currently available to form buffers). You can minimize this possibility by assigning more buffers to your line, by making your buffers larger, or by increasing the number of units in your unit pool. All of these actions can be taken at INTRO execution time.

Queuing Scheme

In TCAM, messages entered by remote terminals or application programs are queued by destination.

Queuing by destination permits overlapping line usage in I/O operations; messages with a common destination may be received simultaneously from more than one source, even while the destination itself is busy sending or receiving a message. Queuing smooths out peaks and valleys in message traffic. Disk queuing permits a high volume of concurrent terminal operations to proceed without requiring excessive main storage for buffering.

You can locate destination queues either in main storage or on disk. You specify in your TERMINAL or TPROCESS macro (QUEUES=operand) whether you want disk or main-storage queuing for the terminal or application program.

A destination queue may be located

- in main storage
- on disk
- in main storage with disk backup.

Main-storage queuing gives the best performance, but

- it may require excessive main storage;
- it compromises recovery capability.
- it may cause a reliability problem and can lose messages if memory allocated by the MSUNITS= operand on INTRO fills up.

Disk queuing is slower than main-storage queuing and requires disk and channel resources, but you can checkpoint and restart the system after failure without losing data if you use disk queuing.

Disk backup for main-storage queuing is a compromise; it is faster than disk queuing but slower than main-storage-only. You can checkpoint and restart the system after failure without losing data when you use disk backup. Also, with disk backup, if the units specified by MSUNITS= are all used, you do not lose messages as you would with MS-only queuing.

- If you use disk queuing, you may elect to define reusable or non-reusable disk data sets.
- With reusable queuing, TCAM wraps around when it gets to the end of the unused space in the data set and reuses that part of the data set containing messages that have already been sent to their destinations. A revolving zone technique is employed internally.
- With nonreusable queuing, when TCAM gets to the end of unused space in the data set, it suspends invitation, sends out all queued messages, and closes itself down.
- Reusable disk permits perpetual operation, and makes the best use of disk space, but it costs CPU time and channel usage because the disk must be periodically reorganized.
- You can optimize disk performance by defining a data set on several volumes, assigning each volume to a different channel; TCAM optimizes I/O for multiple-arm support.

Message Handlers

Message handlers are sets of routines you code with TCAM macros and user code to process messages as they enter and leave the TCAM message control program. Message handlers examine and process control information in incoming and outgoing messages, and prepare these messages for forwarding to their destinations.

Structure

A message handler can have two *groups:*

1. an *incoming group* to handle messages coming into the TCAM MCP from stations or application programs;
2. an *outgoing group* to handle messages being sent from the MCP to a terminal or application program.

These groups have subgroups:

- the inheader and outheader subgroups, which handle only headers of incoming or outgoing messages (a message header contains control information for the message, such as the name of its destination, an input sequence number, its origin, etc.);
- the inbuffer and outbuffer subgroups, which handle all incoming and outgoing message segments;
- the inmessage and outmessage subgroups, which specify what is to be done after the entire message is received or sent (for instance, check for specified errors and send an error message to the source or destination).

You include message handler functions by coding macros; among these functions are:

- message editing
- validity checking
- message routing
- record keeping
- error handling
- system control

You can vary the path of a message through an incoming or outgoing group dynamically, based on the source or destination of the message, or on the presence or absence of certain character strings in the message header.

To supplement TCAM functions, you can code open or closed subroutines using assembler and OS macro instructions and include these subroutines in your message handlers.

Application Program Support

TCAM permits you to code one or more application programs and to interface them with the MCP. Application programmers are insulated from the TP environment; they issue OS GETs, PUTs, READs, and WRITEs to move data between the MCP and their application-program work areas.

TCAM application programs are SAM-compatible. You can debug them in a non-TP environment using BSAM or QSAM as the access method, and a tape, card reader, disk, card punch, printer, etc. as I/O devices. Once you have debugged them, you can run application programs with TCAM without reassembly by changing the DD statement. You can specify that either *messages* (OPTCD=U on the application program DCB macro) or user-defined *records* be transferred when you issue your GET/READs or PUT/WRITEs.

Interface Definition

In the MCP, you code two macros to define the application-program interface:

1. The PCB macro specifies the message handler for the application program, the size of the buffers to transfer data between the MCP and the application program, and the number of buffers to be assigned at one time to handle data transfer.
2. The TPROCESS macro establishes a destination queue for the application program, serves as part of the PUT/WRITE interface, and specifies the PCB for the application program.

In the application program, input and output DCB macros define incoming and outgoing data sets for the application program. These macros are extensions of OS DCB macros, and share many of the same parameters. Activate and deactivate these data sets with OPEN and CLOSE macros. The DCB macros specify the format and characteristics of the work units for the application program.

To transfer data between the application program and the MCP, issue a GET/READ or a WRITE/PUT. In the macro, name the input or output DCB macro. The DD statement named by the DCB specifies a TPROCESS macro in the MCP. The TPROCESS macro specifies a PCB macro that names the application-program message handler.

You can run the MCP independently of any application programs, collecting data for later processing or sending data previously written by the application program to its destination without having the application program resident. You can save a great deal of main storage when you operate in this mode.

You can also coordinate checkpoint and restart in the MCP and the application program.

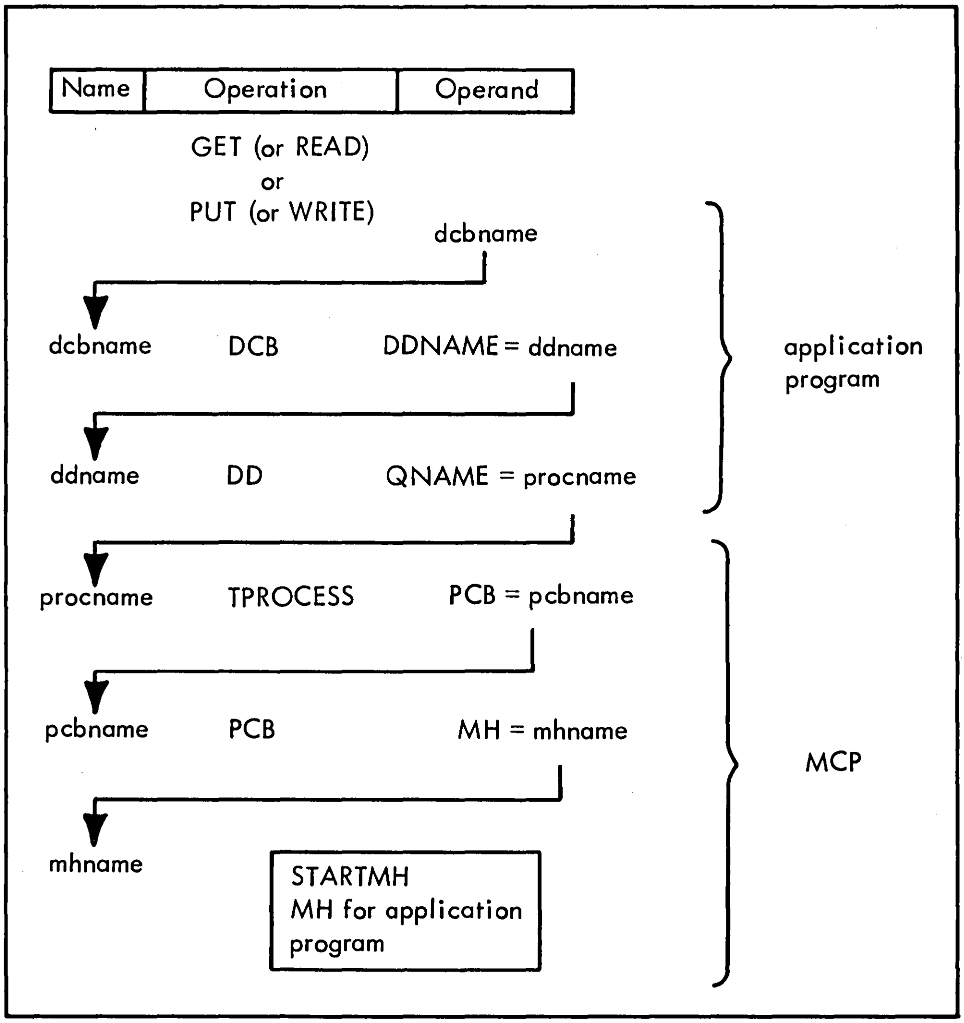


Figure 7. Interface Between the Application Program and the MCP

FUNCTIONAL GROUP	MACROS
Data Set Definition and Control	DCB OPEN CLOSE GET PUT READ WRITE CHECK
Network Control	ICHNG ICOPY MCPCLOSE MRELEASE POINT QCOPY TCHNG TCOPY
Checkpoint Control	QSTART CKREQ

Figure 8. Application Program Macro Instructions

Service Facilities

Operator Control

A set of commands allows you to determine the status of your TP system and alter, activate, or deactivate portions of that system by entering appropriate commands from the system console or a remote terminal.

Checkpoint/Restart

This facility allows the TCAM system to be restarted with minimum loss of message data following closedown or system failure, by periodically recording, in a special data set on disk, information on the status of each station, destination queue, terminal-table entry, and invitation list in the system. TCAM uses this information to restore the MCP environment to its condition before closedown or failure.

Logging

You can include code in your MCP to selectively copy incoming or outgoing messages or message segments on a tape or disk. This facility records message traffic through the MCP.

Diagnostic Aids (COMWRITE)

You can dump diagnostic information onto tape or disk. This information includes the subtask control block (STCB) trace, the line I/O interrupt trace and the buffer trace.

I/O Error Recording

You can use the extensive TCAM error-recording facilities (including OBR/SDR) if you have terminal-related I/O errors.

On-Line Test (TOTE)

Using the optional TCAM on-line test facility, you can test transmission control units and remote terminals without closing down the MCP. Use this function to:

- diagnose hardware errors;
- verify repairs;
- verify engineering changes;
- check devices periodically;
- check new stations brought on-line.

Other Internal Design Highlights

- Request-driven priority dispatching of TCAM subtasks.
- Use of ATTACH for operator control, checkpoint, TOTE and COMWRITE.
- Channel programs based on device characteristics rather than on device type.
- Multiple routing without complete multiple copies of messages.
- Disk queuing use of key and data fields to avoid extra disk activity.
- Channel program restart to initiate a new channel program for disk queuing.
- Line scheduling to provide send, equal, or receive priorities with unique handling for buffered terminals and switched connections.

TCAM Coding Aids

This chapter discusses preassembly aids to help you code your TCAM program so that it will be as error-free as possible. The first section shows the functions of a TCAM program in proper coding order. The second section describes macros that you can include in your program to detect and handle errors in messages and in the teleprocessing network.

Function Checklists

Seven checklists, in flowchart format, show the TCAM macros, their functions, and their proper coding order. Included are:

- how to arrange the message control program (MCP),
- how to define your buffer requirements,
- how to define message queues data sets,
- how to code checkpoint/restart needs,
- how to determine operator control requirements,
- how to include the TCAM diagnostic aids in your MCP, and
- how to arrange a TCAM application program.

Use these charts as you code; also use them to review your coded TCAM system before you assemble it.

MCP Arrangement Checklist

Figure 9 shows how to code a message control program. It includes all macros except the functional message handler (MH) macros. The five major sections of an MCP are shown in logical order. You must code the initialization, activation, and deactivation sections in the order shown. If you follow the order of the other sections as shown in the chart, your assembly listing will correspond to the order of related control blocks and routines in main storage. You will then find it easier to diagnose from a dump and your assembly listing.

Buffer Definition Checklist

Figure 10 shows all macros and operands that you must code to define the buffers you will use in your TCAM system.

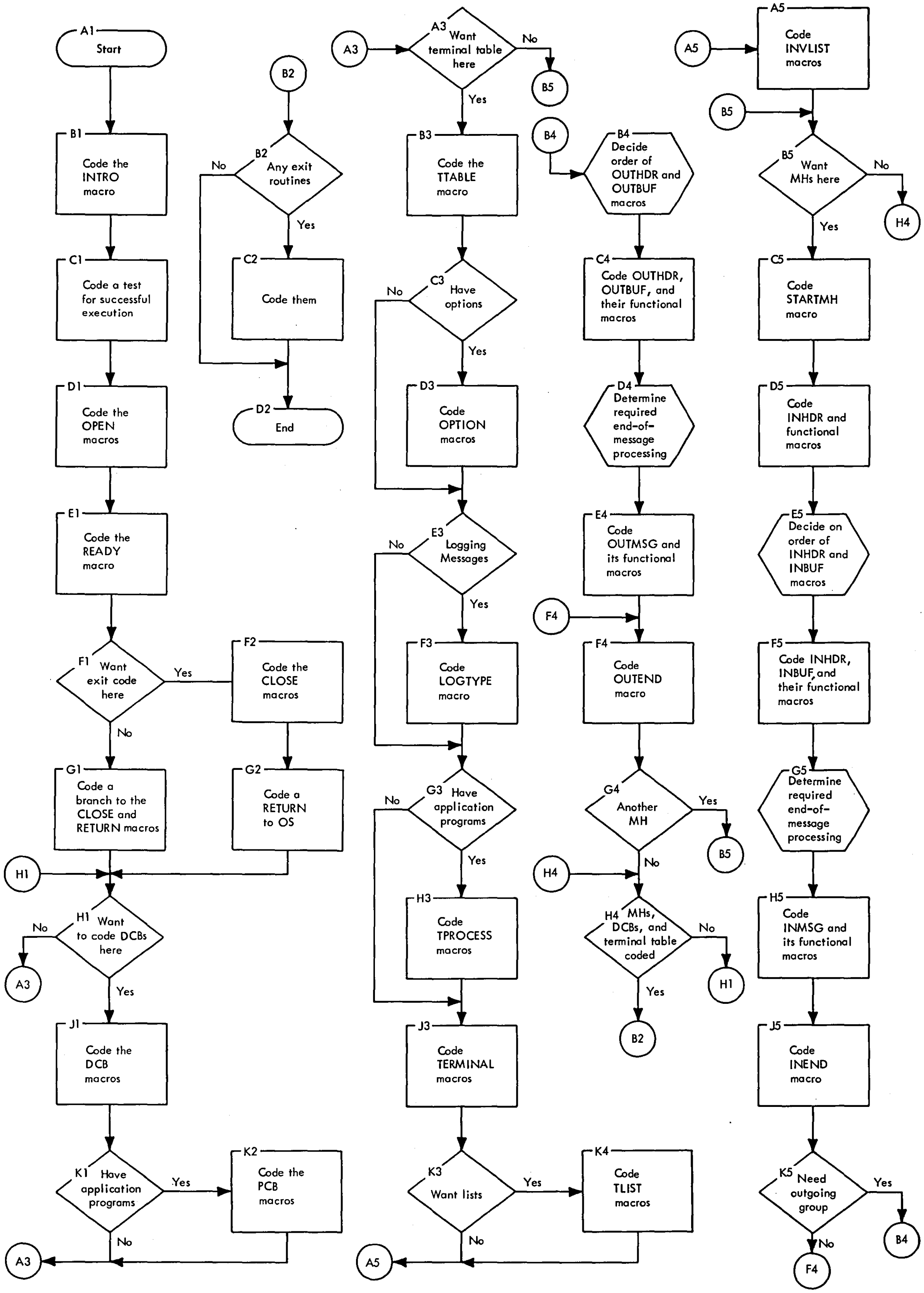


Figure 9. MCP Arrangement Checklist

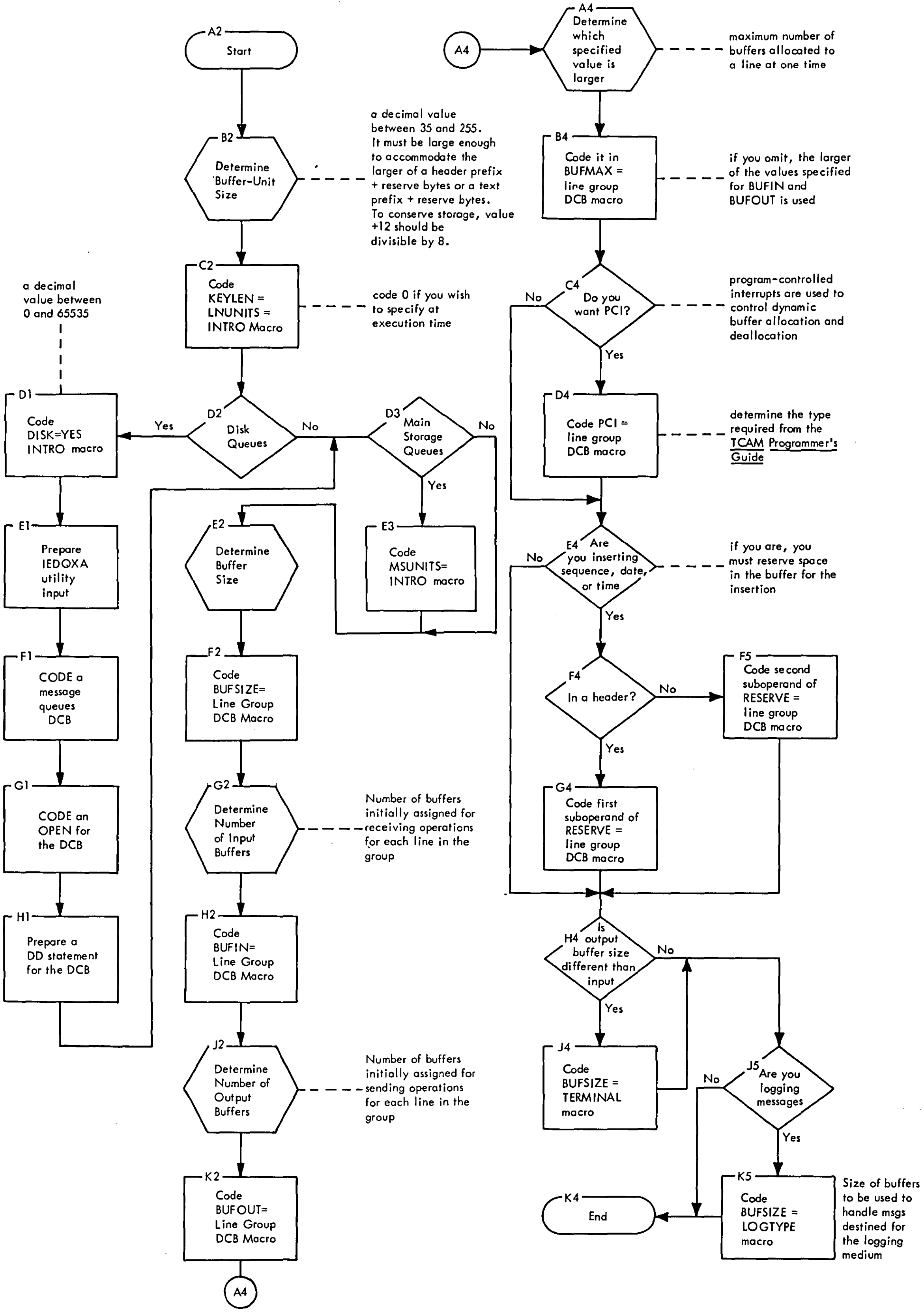


Figure 10. Buffer Definition Checklist

TCAM Unit Pool Analysis

The following forms may prove useful in specifying buffer unit and buffer size, and buffer pool requirements. They may also be useful in deciding preliminary requirements. Final requirements are application dependent and must be determined through operating experience.

TCAM UNIT POOL ANALYSIS

LINE TYPE____

Maximum Output Message =________ Bytes

Maximum Input Message =________ Bytes

Dynamic Buffering Required YES/NO PCI = _______

SELECTED BUFFER SIZE: BUFSIZE = _______ Bytes

Reason for Selected Buffer Size:

BUFFER REQUIREMENTS FOR 1 MESSAGE

Output Buffers = ______: Input Buffers = _____:

SELECTED BUFOUT = ______

SELECTED BUFIN = ______

If Dynamic Buffering, SELECTED BUFMAX = ______

NUMBER OF LINE UNITS REQUIRED

(BUFSIZE/UNITSIZE)*BUFMAX

= ______ LNUNITS

If Main-Storage Queuing:

NUMBER OF MAIN-STORAGE UNITS REQUIRED = ______ MSUNITS

Reason:

If Disk Queuing:

CALCULATE NUMBER OF CPBs REQUIRED

NUMBER OF CHARACTERS TO AND FROM DISK/sec = ______ Bytes

NUMBER OF UNITS TO AND FROM DISK/sec
(characters/sec/unitsize) = ______ Units

ALLOW 1 CPB/UNIT/sec

NUMBER OF CPBs REQUIRED = ______ CPBs

SUMMARY

UNIT REQUIREMENTS

LINE/APPLICATION PROGRAM	LNUNITS	MSUNITS	CPB UNITS
________________________	_______	_______	_________
________________________	_______	_______	_________
________________________	_______	_______	_________

________________________ ________ ________ _________

________________________ ________ ________ _________

________________________ ________ ________ _________

________________________ ________ ________ _________

________________________ ________ ________ _________

________________________ ________ ________ _________

________________________ ________ ________ _________

TOTALS ________ ________ _________

TOTAL UNITS = __________

CORE REQUIREMENTS

UNIT SPACE (UNITSIZE + 12 + wasted bytes) * TOTAL UNITS

= _________ Bytes

CPB SPACE = Number of CPBs * (72 + unitsize) = _________ Bytes

TOTAL MAIN STORAGE REQUIREMENT = _________

Message Queues Checklist

Figure 11 shows all macros and operands that you must code to use each of the five TCAM queuing types.

Checkpoint/Restart Checklist

Figure 12 shows all macros and operands that you must code to checkpoint and restart your TCAM system. It also shows the macros and operands that you must code in an application program when you want to coordinate TCAM checkpoints of the MCP with OS checkpoints of the application program.

Operator Control Checklist

Figure 13 shows all macros and operands that you must code if you want to use operator control from either the system console, remote terminals, or application program.

Diagnostic Aids Checklist

Figure 14 shows all the TCAM diagnostic aids, except operator control and checkpoint/restart, and all the macros and operands you must code to include each diagnostic aid in your MCP.

Application Program Checklist

Figure 15 shows how to code an application program to run with a TCAM MCP. All necessary macros, work areas, and special coding are shown.

Coding Hints to Alleviate Errors

This section discusses the TCAM macros that handle errors that occur while your TCAM system is running. Using these macros, you can test for and recover from both errors in messages and errors in hardware. You can also define logical errors for your system, and use TCAM macros to test for and recover from these errors. TCAM indicates errors in a message error record, which is defined for each message as it is being processed.

The Message Error Record

TCAM assigns a five-byte message error record to each message while it is being processed by the incoming or outgoing group of a message handler. Each of the 40 bits of the message error record, except reserved bits, indicates the presence (when 1) or the absence (when 0) of a specific error that has affected or may affect successful processing or transmission of the message.

Errors recorded in the message error record include transmission and equipment errors (lost data, bus-out check, etc.), mistakes in entering a message (wrong sequence number, invalid origin, etc.), and shortages of system resources (insufficient number of buffers, insufficient space in a main-storage-only message queues data set, etc.). The last byte of the message error record is the sense byte for the transmission control unit being used.

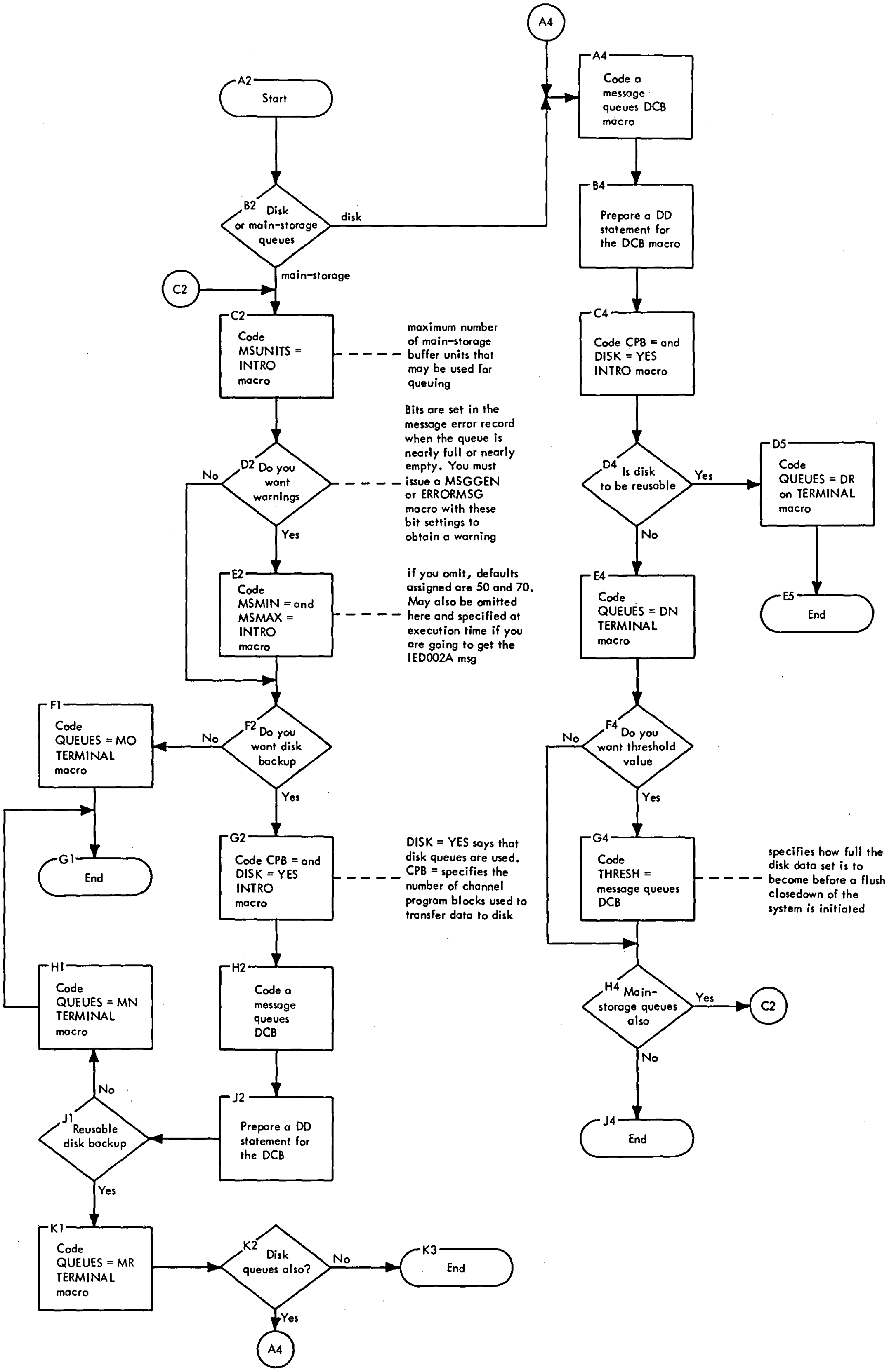


Figure 11. Message Queues Checklist

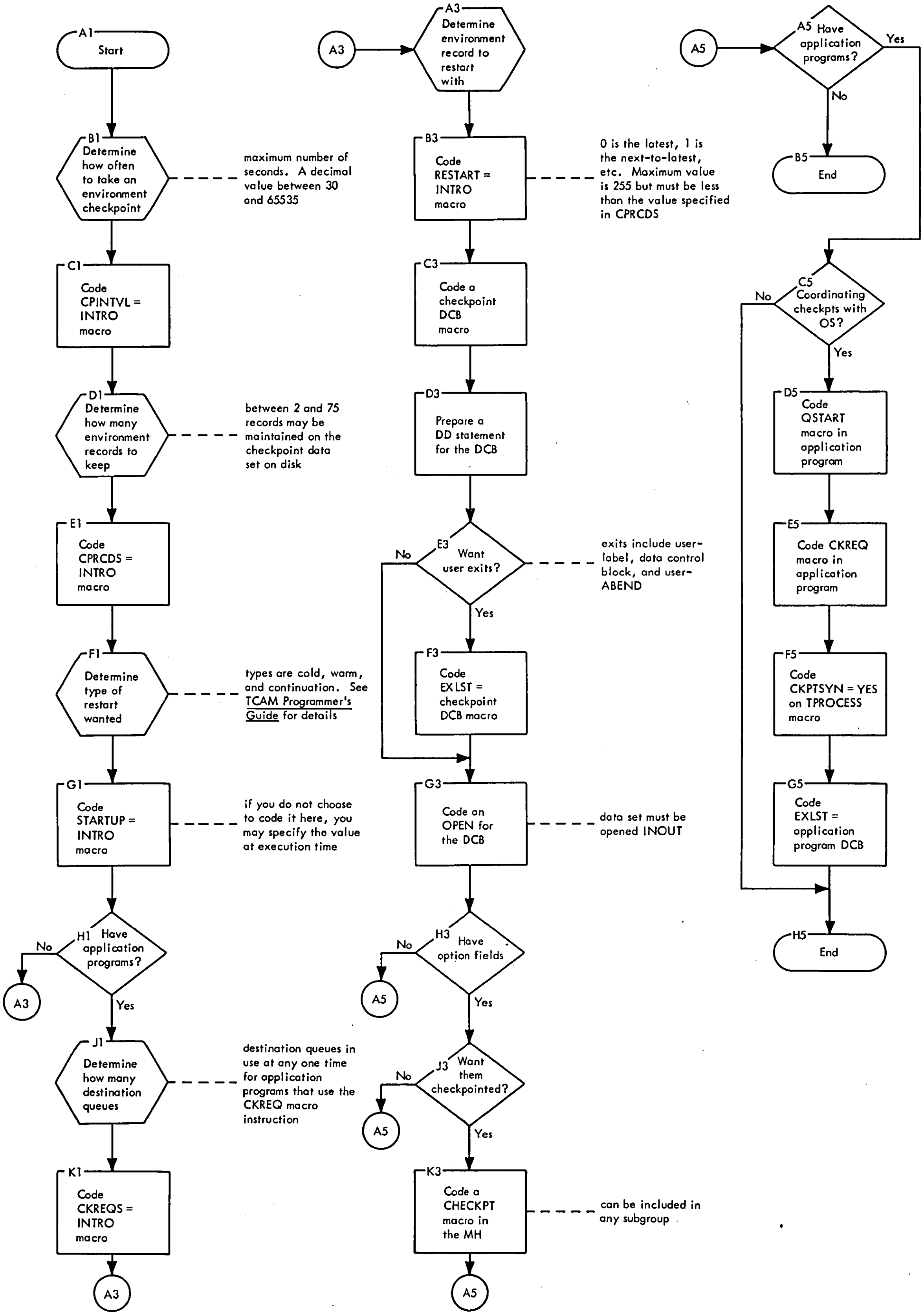


Figure 12. Checkpoint/Restart Checklist

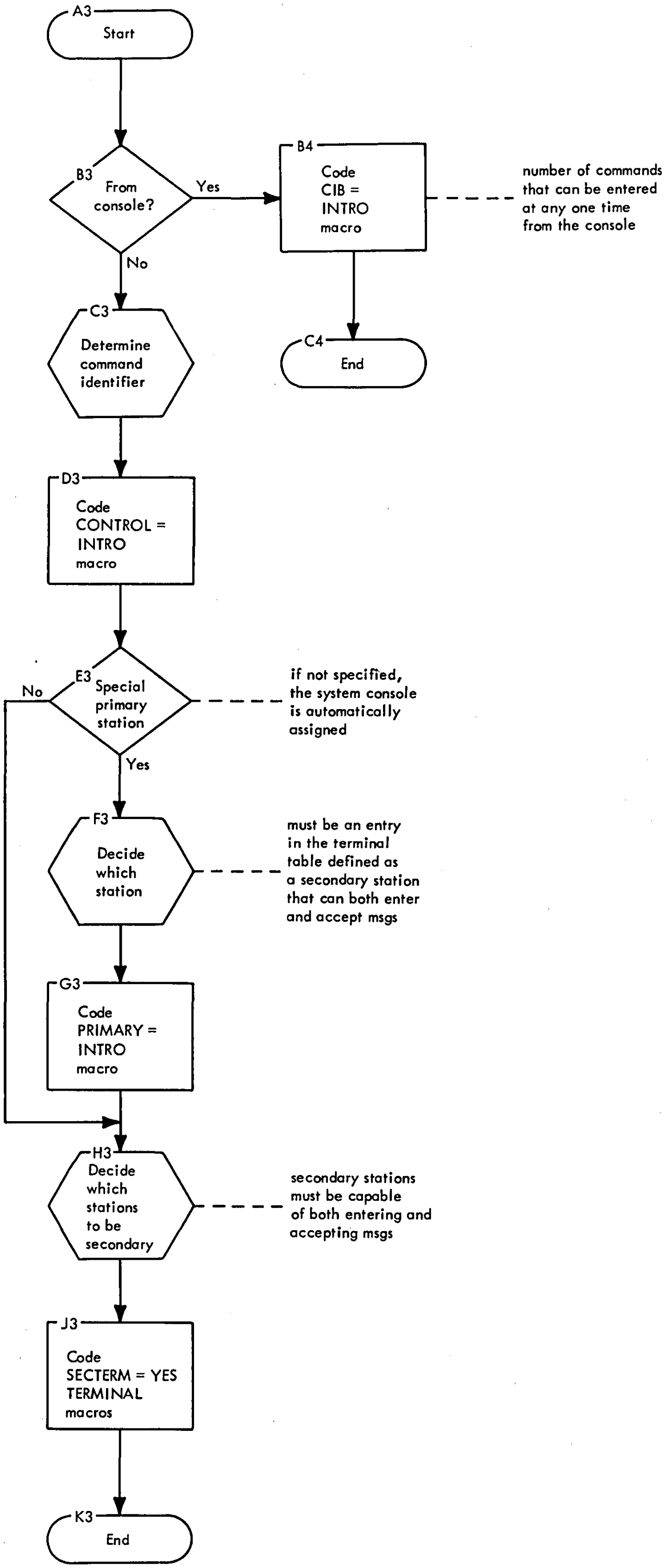


Figure 13. Operator Control Checklist

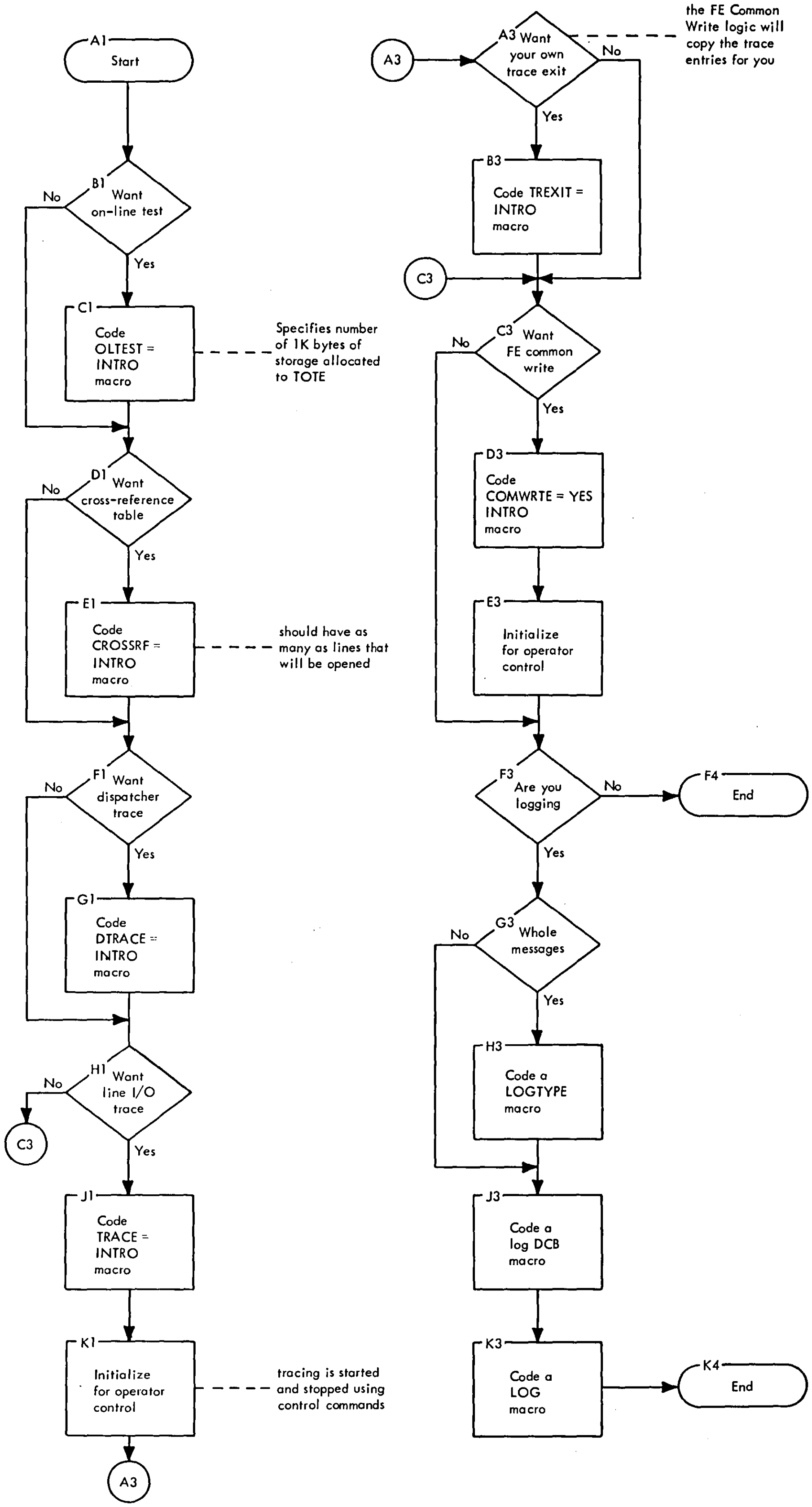


Figure 14. Diagnostic Aids Checklist

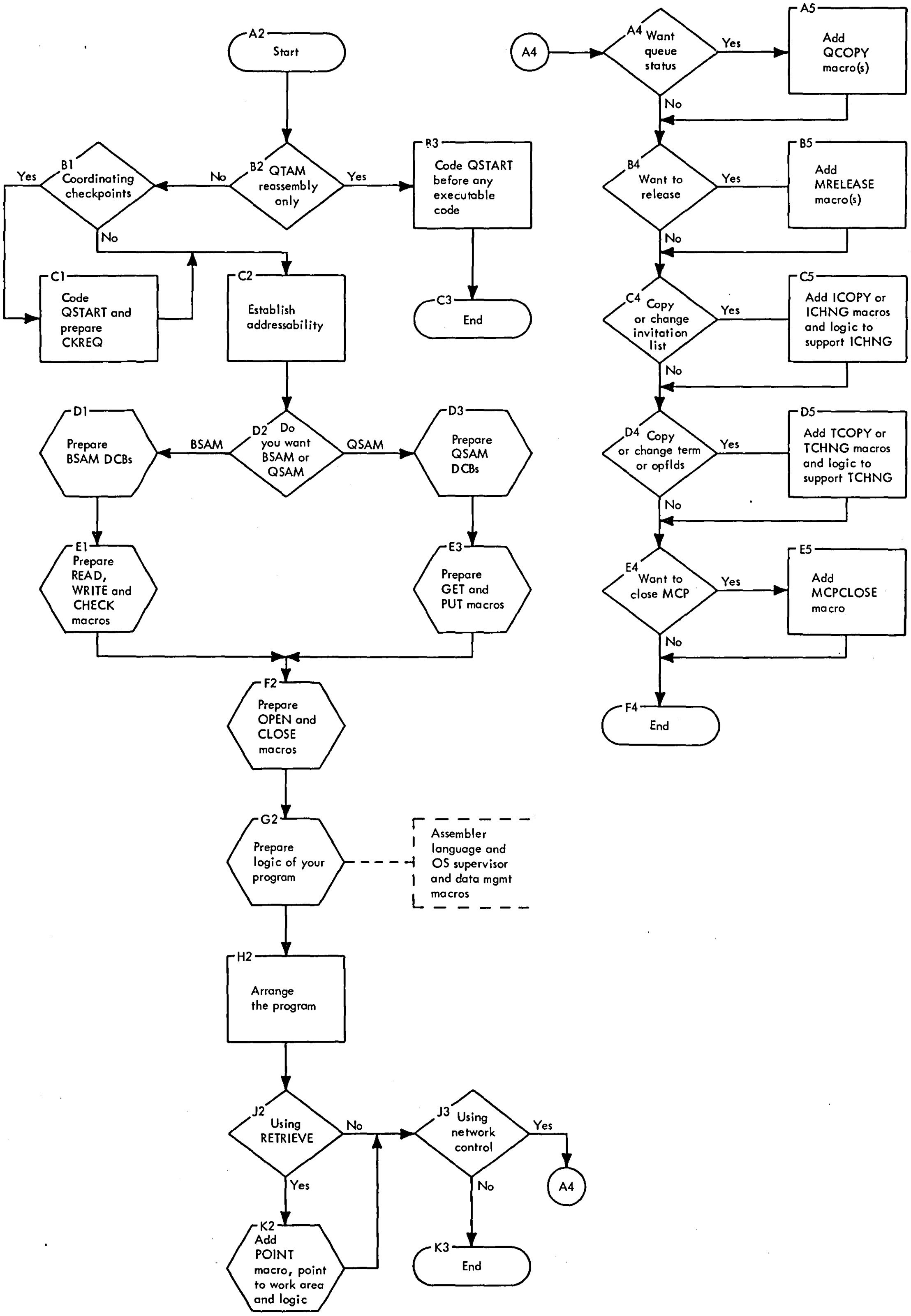


Figure 15. Application Program Checklist

The message error record indicates most user and hardware errors. You can minimize your problem determination time if you use this record and issue error messages for every error condition. Such use warns of impending trouble on the line or in the system. It can be used to indicate internal bugs and hardware conditions causing degradation. You may want to have an application program to collect data and give end-of-day tallies of errors to the system control programmer.

Figure 16 is a quick reference table of the message error record. See the *TCAM Programmer's Guide* for more information about the bit meanings.

Using the Message Error Record to Detect Message Errors

Several TCAM macros can help you find errors in messages. Each of the following macros sets a bit in the message error record for the message when an error in

BYTE	BIT	KEYWORD	VALUE	DESCRIPTION
First	0	Scan	X ' 80 '	Scan Pointer Has Passed Message End
	1	Origin	X ' 40 '	Invalid Origin Code
	2			(Reserved)
	3	Seq High	X ' 10 '	Sequence Number High or Not A Valid Decimal Number
	4	Seq Low	X ' 08 '	Sequence Number Low
	5			(Reserved)
	6	Buffers	X ' 02 '	Insufficient Buffers For Message
	7	Cutoff	X ' 01 '	Message Exceeds Cutoff Limit or RVI Error
Second	8	MSMIN	X ' 80 '	Main-Storage Queue is Below MSMIN
	9	MSMAX	X ' 40 '	Main-Storage Queue Exceeds MSMAX
	10			(Reserved)
	11			(Reserved)
	12	Tote	X ' 08 '	TOTE is Not In System
	13	BSC Abort	X ' 04 '	Abnormal Termination During Input/Output
	14	Dest	X ' 02 '	One or More Forward Destinations Invalid
	15			(Reserved)
Third	16	MS Full	X ' 80 '	Last Part of MSG Lost As Main-Storage Queue Full
	17	Bad Ident	X ' 40 '	Invalid Station ID From Terminal
	18	Dest Held	X ' 20 '	Destination Station Held (Intercepted)
	19			(Reserved)
	20	User Bit	X ' 08 '	As Required By User
	21	BSC Format	X ' 04 '	Invalid BSC Format (No Starting STX)
	22			(Reserved)
	23	Unit Excep	X ' 01 '	Unit Exception Set By Transmission Control Unit
Fourth	24	Selection	X ' 80 '	Error During Polling Or Addressing
	25	Text	X ' 40 '	Text Error During Transfer of Data
	26	Switching	X ' 20 '	Switching Error During Connection or Disconnection
	27	Station	X ' 10 '	Station Faulty
	28			(Reserved)
	29	Control	X ' 04 '	Control Unit Faulty
	30	Channel	X ' 02 '	Channel Faulty
	31	Unknown	X ' 01 '	Unknown Error (TCAM Cannot Determine Cause of Error)
Fifth (Sense)	32	Command	X ' 80 '	Invalid Command or Sequence
	33	Help	X ' 40 '	Operator Intervention Required
	34	Busout	X ' 20 '	Parity Error Between TCU and Channel
	35	Equipment	X ' 10 '	Transmission Control Unit Has Failed
	36	Data Check	X ' 08 '	Parity Error Bad Binary Chk Count on Received Data
	37	Overrun	X ' 04 '	Received Data Lost (MPX Channel Service Not In Time)
	38	Lost Data	X ' 02 '	MSG Too Long For Read Cmd or Data Read While No Read
	39	Timeout	X ' 01 '	Time Limit Termination of Any Receiving Command

Figure 16. TCAM Message Error Record Summary

the header is found. This validity checking improves the reliability of transmitted traffic. To use the macros most effectively, you should cancel any invalid input messages to be sure that only valid messages are transmitted. You should also issue an error message to the terminal operator who enters an invalid input message, so that he knows the message was not processed.

SEQUENCE Macro

When you code it in the inheader subgroup, the SEQUENCE macro verifies the input sequence number in the header by comparing it to an internal counter in the terminal entry. TCAM increments this input counter for each message that has a correct sequence number in the header. If the sequence number is not one greater than the sequence number of the last message received from the same station or application program, TCAM sets an error flag in bit 3 or bit 4 of the message error record for the message. The SEQUENCE macro sets bit 3 to 1 (on) in the message error record when the sequence number in the header is not a valid decimal integer or when it is higher than the expected number for the next message from the station. The SEQUENCE macro also sets bit 4 to 1 (on) when the sequence number is low.

TCAM also places one of the following return codes in register 15:

X'00'	good return
X'04'	sequence number in the message is high
X'08'	sequence number in the message is low
X'0C'	originating station is unknown

The message is processed normally, regardless of the sequence number, unless you cancel it.

When you code it in the outheader subgroup, the SEQUENCE macro inserts an output sequence number in the header of each outgoing message handled by the message handler (MH). The output sequence number is inserted when the message is actually sent to the destination. You must reserve five bytes in your message for the sequence number in the RESERVE= operand of the line group DCB macro or the application-program PCB macro. TCAM maintains an output sequence number counter in the terminal entry, and does not increment it until the message is actually sent to the destination. TCAM does not verify the output sequence number.

Although use of the SEQUENCE macro is optional, you should code it in both your inheader and outheader subgroups to check for lost messages and for bookkeeping. In the inheader subgroup, executing the SEQUENCE macro can warn you that the terminal has sent more than one message with the same sequence number or that numbers have been skipped. For outgoing messages, executing the SEQUENCE macro allows you to account for the messages received by a station. Both input and output sequence numbers should be sequential. If sequential order is not maintained in the input messages (that is, if a sequence number repeats), you know that a message was lost before it reached the MH. If sequential order is not maintained for outgoing messages, the terminal operator knows that a message was lost after the MH handled it. In either case, you can tell that your problem is caused by either trouble on the line or trouble in the station.

You should be aware, however, that sequential order in the sequence numbers does not guarantee that a message has not been lost. The incoming MH may handle a message and thereby update the input counter for the originating station, but may not forward the message correctly to the outgoing message handler.

Since the outgoing MH does not handle the message at all, TCAM does not update the output sequence number counter, and you have no indication that the message is lost.

Using the SEQUENCE macro, you can account for message traffic on the basis of numbers, rather than data. By examining the header it is much easier to verify that remote terminal B received input messages with sequence numbers 1, 4, 5, and 20 from terminal A than to compare the actual messages sent, especially when similar or identical messages are sent more than once to a station.

You should use the SEQUENCE macro for accounting and problem determination. You should use it to put sequence numbers in outgoing messages that you want to retrieve in an application program via the POINT macro (refer to the *TCAM Programmer's Guide*). The count is internally maintained and the sequence number in the outgoing subgroup lets you know which output message you can retrieve.

ORIGIN Macro

For nonswitched stations, the ORIGIN macro verifies that the origin field in the header contains the symbolic name of the station invited to send the message, by comparing the origin field with the name of the terminal-table entry for the station that was contacted. For switched stations, the ORIGIN macro both verifies the origin field in the header and identifies the calling station to TCAM. Unless the calling station is a BSC station that transmits a unique ID sequence when it successfully contacts the computer, TCAM does not know which station is on the line until you issue an ORIGIN macro in the inheader subgroup of the MH. If the origin field in the header does not match the name of a terminal entry, TCAM sets bit 1 on in the message error record for the message. TCAM also places one of the following return codes in register 15:

X'00'	good return
X'04'	invalid origin

Although use of the ORIGIN macro is optional except in message handlers for switched start-stop stations, you should code it in all your message handlers to improve the security of your system. You and you alone know the names assigned to your stations by the TERMINAL macros in the MCP. These names are the only valid sources for messages coming into your system. The ORIGIN macro simply verifies the source. You should cancel messages with invalid origins to be sure that messages from an "unknown" user are not transmitted.

An origin field in the header of your message readily identifies the station that entered the message. You should execute the ORIGIN macro and cancel any message with an invalid origin field in the header to eliminate any confusion that may develop at the receiving station about the source of the message. Canceling the message with an invalid origin is most important during inquiry processing, if you code OPTCD=W in the application program input DCB macro. TCAM automatically places the name of the originating terminal in the first eight bytes of the buffer. If the name is invalid, when an incoming subgroup for the application program handles the message with FORWARD DEST=PUT, it sends the message to the dead-letter queue, if provided, or loses it.

Also, it is easier to determine the source of each message in your end-of-day accounting of message traffic. Using the origin field, you can also calculate how much each terminal uses the system.

FORWARD Macro

When the FORWARD macro executes in the inheader subgroup, TCAM scans the destination field in the header of each incoming message and compares this field with the names of the terminal entries. If the destination code is valid (that is, if TCAM finds a matching entry in the terminal name table), the FORWARD macro queues the message for the specified destination. If the specified destination is invalid, TCAM sets bit 14 on in the message error record for the message. TCAM also places one of the following return codes in register 15:

X'00'	good return
X'04'	invalid destination

Besides checking the error bit or the return code, you can take three possible actions for an invalid message:

1. If you specify an exit routine in the EXIT= operand of the FORWARD macro, control passes to this routine. In the routine, you can correct the invalid destination, specify another destination, or indicate that the message is not to be processed. See the *TCAM Programmer's Guide* to learn how to code this exit.
2. If you do not specify an exit, or if you supply an invalid destination in the exit, TCAM queues the message for the station or application program that you specified as the dead-letter queue in the DLQ= operand of the INTRO macro.
3. If you specify neither an exit nor a dead-letter queue, the message is overlaid and lost.

You do not have to cancel a message with an invalid destination. Omitting both an exit routine and a dead-letter queue causes the incorrect message to be overlaid and lost. If, however, you wish to retain a copy of the messages directed to an invalid destination, use a dead-letter queue rather than an exit routine for two reasons. First, there are times you will write your own code and you might unknowingly supply erroneous information to TCAM when you return from the exit routine, and cause a program check in a TCAM module. The problem can seem to be in TCAM when, in reality, the information you supplied in your exit caused the trouble. Second, if you omit the EOA delimiter, at most two copies of the message are sent to the dead-letter queue; whereas, if you supply a valid destination in your exit routine, that destination will receive up to 255 copies of the message. When there is no EOA delimiter in the message, the FORWARD macro compares each maximum number of bytes in a terminal name (the value specified in the MAXLEN= operand of the TTABLE macro), and any number of bytes less than the maximum delimited by blanks, with the entry names in the terminal name table. Figure 17 illustrates the consequences experienced when a user-exit routine sent messages with an invalid destination to one specified terminal. The MAXLEN= value on the TTABLE macro was 8, and the EOA delimiter, a /, was missing. You can see from the example that using the dead-letter queue saves you computer processing time, line time, and terminal usage time.

TERRSET Macro

The TERRSET macro sets bit 20 on in the message error record for a message. Executing this macro is left entirely up to you. You define the conditions under which the bit is set. Usually, you would code it to flag as an error a message that is logically "wrong" for your message handler.

```
                        VALID  1   2 3  4   5    6          7        8       9      10      11
1  X NYC 1 09.49.11 NYC THIS IS A BUNCH OF SYMBOLS,|,,|||...|¬¬¬!!!$$|$""""###;|;;333:::|%%%'''''
                                                            8 BYTES
2  X NYC 1 09.49.11 NYC THIS IS A BUNCH OF SYMBOLS,,,|||...¬¬¬!!!$$$""""###;;;333:::%%%'''''
3  X NYC 1 09.49.11 NYC THIS IS A BUNCH OF SYMBOLS,,,|||...¬¬¬!!!$$$""""###;;;333:::%%%'''''
4  X NYC 1 09.49.11 NYC THIS IS A BUNCH OF SYMBOLS,,,|||...¬¬¬!!!$$$""""###;;;333:::%%%'''''
5  X NYC 1 09.49.11 NYC THIS IS A BUNCH OF SYMBOLS,,,|||...¬¬¬!!!$$$""""###;;;333:::%%%'''''
6  X NYC 1 09.49.11 NYC THIS IS A BUNCH OF SYMBOLS,,,|||...¬¬¬!!!$$$""""###;;;333:::%%%'''''
7  X NYC 1 09.49.11 NYC THIS IS A BUNCH OF SYMBOLS,,,|||...¬¬¬!!!$$$""""###;;;333:::%%%'''''
8  X NYC 1 09.49.11 NYC THIS IS A BUNCH OF SYMBOLS,,,|||...¬¬¬!!!$$$""""###;;;333:::%%%'''''
9  X NYC 1 09.49.11 NYC THIS IS A BUNCH OF SYMBOLS,,,|||...¬¬¬!!!$$$""""###;;;333:::%%%'''''
10 X NYC 1 09.49.11 NYC THIS IS A BUNCH OF SYMBOLS,,,|||...¬¬¬!!!$$$""""###;;;333:::%%%'''''
11 X NYC 1 09.49.11 NYC THIS IS A BUNCH OF SYMBOLS,,,|||...¬¬¬!!!$$$""""###;;;333:::%%%'''''
```

Figure 17. An Invalid Message with No Dead-Letter Queue

Using the Message Error Record to Detect Hardware Errors

Two message handler macros notify you of hardware errors by setting bits in the message error record. The first, STARTMH, is a delimiter macro that you must code. The second, CUTOFF, is an optional functional macro.

STARTMH Macro

Use the STARTMH macro, which you must code as the first macro in every MH, to determine transmission errors or errors that are logical errors for your system. If you specify either the STOP=, CONT=, CONV=, or LOGICAL= operand, end-of-block (EOB) checking is done. This checking determines, whenever an EOB, ETB, ETX, or EOT line control character is received, whether transmission or logical errors occurred. Through the STARTMH operands, you control what happens to messages in error.

For an incoming message, EOB checking is done before the message handler processes a buffer with an EOB. Terminals with or without error checking may be processed by the same MH even though EOB checking is done due to specification of one of the STARTMH operands. With multiple buffer blocks, preceding buffers could have been processed when an EOB error is detected in the message. If a hardware error is detected and retry is possible, the operation is retried. Retry is an error-recovery procedure in which the current block of data, from the last EOB or ETB, is re-sent a prescribed number of times (two retries for start-stop terminals and six retries for BSC terminals) or until it is accepted or entered correctly. If the retry count is exhausted, STARTMH either ignores the error and restarts the channel program to receive the next block (CONT= operand), or terminates transmission and sends the buffer through the MH as the last buffer of the message (STOP= operand). STARTMH branches to the user exit specified on the LOGICAL= operand on every EOB, so you can detect errors in the buffer containing the EOB. Use this exit to determine whether to stop or continue on the basis of the terminals or option fields.

For outgoing messages, EOB checking is done after each block is transmitted. You cannot check for logical errors on output messages. Transmission is successful when the receiving terminal acknowledges that it successfully received the block. Transmission errors detected by the terminal are retried. Once the retry count is exhausted, transmission of the message either terminates (STOP=) or continues (CONT=) with the next block.

If the STARTMH macro detects an error in the message, it sets bit 25 on in the message error record for the message. You should issue an error message (using

ERRORMSG or MSGGEN) to inform the terminal operator that the message was in error. He can determine the problem, since he knows if he entered an EOB or EOT at the end of his message. If he did, either the station or the line malfunctioned.

CUTOFF Macro

Use the CUTOFF macro to determine hardware errors. CUTOFF sets bit 7 on in the message error record for the message if a buffer is filled with identical characters or if an incoming message reaches the maximum allowable length. If the maximum is reached, TCAM stops receiving as soon as those buffers already assigned to the line are filled. The CUTOFF macro does not provide you with a precise limit on message size. If dynamic PCI is being employed, the timing may be such internally that the PCI requirement for more buffers is honored before the CUTOFF macro executes. After the CUTOFF macro executes, TCAM finishes filling up the buffers currently assigned to the station. If the operator at the station enters a very long message slowly, a request for more buffers may be honored before the CUTOFF macro executes, and the long message may be received. If, however, the operator enters his message quickly, he may have only the original allocation of buffers (no PCI before CUTOFF executes). You can sometimes receive a message much longer than one that supposedly was terminated after the predetermined length specified on the CUTOFF macro.

A good use for the CUTOFF macro is to issue it when message switching to a buffered terminal. In this way, you can inform the operator at the transmitting station that his message is longer than the hardware buffer length at the receiving station, and the receiving station did not get all of the message.

You should send an error message (using ERRORMSG or MSGGEN) to the operator who entered the message to notify him that the CUTOFF macro was executed, and that the rest of the message will not be received. He will be able to determine the problem, since he knows whether he entered a message that was too long. If the message length is below the maximum, then the station has malfunctioned.

Macros Dependent on the Message Error Record

The execution of several TCAM functional macros depends on the contents of the message error record. Each of these macros has a mask operand, which is compared to the message error record. The macro executes if any or all of the bits on in the mask operand are also on in the message error record. You can thus define what is to be done when the stated error occurs. You can unconditionally execute each of the following macros either by specifying a mask of all zeros or by omitting the mask.

HOLD Macro

The HOLD macro temporarily suspends outgoing message transmission to a station. You can suspend transmission either for a specified time interval or until you choose to resume traffic by issuing the RESMXMIT operator command or the MRELEASE application program macro.

Use HOLD to *intercept* a station; that is, to stop sending messages that should not be sent immediately because the destination station is failing or has failed. You cannot hold a station (via HOLD) that has main-storage queuing with no disk backup. You define the failures in the mask operand of the macro. If any or all of the bits in the mask are on in the message error record for the message, TCAM sends nothing to the station following that message. If you omit the HOLD macro, messages that cannot be transmitted because the station is out of order are treated as if they were transmitted; that is, the buffer units containing the messages are freed and become available for reuse, and the message is lost. Using the HOLD macro assures you that once the problem has been corrected, the station will receive all traffic directed to it. The message you issue HOLD for in the outmessage subgroup will be retransmitted when the HOLD is released.

You should code at least one HOLD macro in your MCP. If you do not, you will not be able to intercept a station with the SUSPXMIT operator command. You may make the mask operand an impossible combination of errors, so that HOLD never executes. This lets you issue operator commands, which you will need to do if a terminal unexpectedly fails and you do not want to lose any messages for the station.

CANCELMG Macro

The CANCELMG macro immediately cancels a message if any errors specified in the mask operand are also set in the message error record for the incoming message. A canceled message does not go to any destination, even if it is a multiply-routed message.

If you execute an INITIATE macro for an incoming message, do *not* execute a CANCELMG macro. CANCELMG is coded in the inmessage subgroup and therefore operates on the entire message. However, INITIATE sends each segment of a message as soon as possible after it is received at the destination queue. Therefore, one or more segments of the message may already have been sent before CANCELMG executes.

CANCELMG must be the first functional macro that you code in the inmessage subgroup, and you can execute only one CANCELMG macro for a message.

Use CANCELMG to be sure that only valid messages are processed. You should notify the operator who entered an invalid message (using MSGGEN or ERRORMSG) that the message was not processed and that he must reenter the message correctly.

REDIRECT Macro

The REDIRECT macro queues a message for a destination, in addition to the destinations specified by the FORWARD macro, when it finds that errors specified in the mask operand are present in the message error record for the message.

Use the REDIRECT macro when you want to return the incorrect message to the originating terminal. With REDIRECT, you do not have to code your MCP to find the origin field in the header and return the message. TCAM still sends the incorrect message to all destinations specified in the header, unless you cancel the message.

Using the REDIRECT macro, you can also send messages to an alternate destination when the original station is inoperative. If you have not coded a HOLD macro in your system, use REDIRECT to prevent any loss in message traffic.

ERRORMSG Macro

The ERRORMSG macro is one of TCAM's most useful macros for alerting you to errors in transmitted messages or to trouble in your TCAM system. The ERRORMSG macro sends an error message that you specify to a designated station when errors in the mask operand are detected in the message error record for the message. The error message includes the header of the message in error, followed by the text that you write. TCAM inserts your message beginning at the current location of the scan pointer in the first buffer. See the *TCAM Programmer's Guide* for considerations on overlaying header or data information.

The ERRORMSG macro places the error message on the destination queue for the station that you select to receive the message, and sends it through the outgoing group of the MH. Therefore, you must be sure that the format of the erroneous message header is compatible with the macros executed in the outgoing group that handles messages for the station receiving the error message. You can use alternate paths through the MH, by coding the MSGTYPE or PATH macros, so that, by distinguishing message types, you will not have to be concerned with the header format.

You should identify the originating station as the destination of the error message. You should also notify the operator who entered the message of what was wrong with the message and how TCAM is processing it. For example, if an invalid origin is detected, you can issue the message

```
INVALID ORIGIN - MESSAGE CANCELED - RESEND
```

Your message should be meaningful! The message can have a maximum of 255 characters or two buffer units (2 * KEYLEN= value), whichever is less. This count *must* include all necessary line-control characters. You should include all line-control characters (STX, ETX, EOB, ETB, etc.) in all messages or issue the MSGFORM macro in the outheader subgroup.

The ERRORMSG macro also has an EXIT operand that you can code to complete error message processing. For instance, you can use this exit to provide the terminal operator with the correct input sequence number if he enters an invalid number. Figure 18 shows how you can code the routine.

MSGGEN Macro

The MSGGEN macro generates a message that you define if the errors in the mask operand are detected in the message error record for the message. The generated error message bypasses all normal functions; there is no message handler processing, no queuing, no logging, and no buffer requesting. You must supply line-control characters. The error message refers to the last transmission since the line is never freed in between message transmission and execution of the MSGGEN macro. The MSGGEN macro informs you more rapidly than ERRORMSG that you have an error, but it does not return the header.

If you code MSGGEN in the incoming group, TCAM sends the error message to the origin. If you code it in the outgoing subgroup, TCAM sends the message to the destination. The maximum length of the error message is 24 bytes. This count includes all necessary line-control characters. Again, you should supply *all* line-control characters for *all* your MSGGEN messages.

ERRORMSG and MSGGEN

Both macros let you issue error messages. The following chart compares the two macros.

```
GETSEQ   CSECT
         USING     GETSEQ, 12
         USING     IEDQAVTD, 13
         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
         LTR       1, 1              TEST FOR ZERO
         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                       TERMINAL ENTRY DSECT

         END
```

Figure 18. An ERRORMSG Macro Exit Routine

ERRORMSG	*MSGGEN*
255 bytes or two units— maximum message length	24 bytes— maximum message length
header of message in error precedes error message text	no header
slow— message processed by MH	immediate response— no MH processing
exit for user-written routine	no exit
can specify destination of generated message	no choice of destination— incoming returned to origin— outgoing sent to destination

This chart shows that the major advantage of using MSGGEN is that it is faster, since you do not have to process the header through the message handler. However, you do not have an exit routine, the maximum length is small so it is difficult to send meaningful messages, you have no choice of destination, and you do not get the header, which can be a valuable tool to trace the message or terminal that created an error.

Logging

You can use logging in two ways: first, as an integral part of the system, recording messages for accounting; and second, as a programming aid, helping you to diagnose errors and to find the information you need to evaluate system performance.

You may want to record all messages for accounting, even though they were successfully sent to their destinations. The best way to obtain a meaningful accounting report is to either record the entire message (code the LOG macro in the inmessage or outmessage subgroup) or to record only the header segment (code the LOG macro in the inheader or outheader subgroup). You should record only the header segment if you have meaningful data in the header, such as the origin, time, date, and destination terminals. Some accounting uses of logging are:

1. Copying groups of messages sent over a long period of time to a variety of destinations.
2. Providing long-term backup for messages that are accepted by one or more destinations but later lost through human error.
3. Collecting exceptional cases.

The log is also a good programming aid. If you include a carefully designed message-logging facility in your message handler, you can trace the flow of messages through your MCP; thus, you can quickly find errors while you are diagnosing the MCP. By examining the log, you can see what message handler processing has been performed on the message, and locate the subgroup in which the message becomes incorrect.

In your initial stage of programming a TCAM MCP, the use of the OS/360 WTO macro interspersed at appropriate points is beneficial in tracing a message through message handler processing.

The log also helps you more efficiently allocate the resources of your telecommunications system. Do this by analyzing the flow patterns of the message traffic. When you first execute the MCP, include the log facility to record such information as time, origin, and destination for each message, or, in cases where traffic is heavy, for representative messages. You can then reallocate your resources for more efficient processing.

TCAM Problem Determination Aids

This chapter suggests where you can look in your code when you have an error. Each possible problem area is discussed. Lists of the more common errors that can be made are given. Use this chapter to review your code before you first run a TCAM program. Use it also when you have a problem to review possible problem areas.

In addition to errors in your code, this chapter also summarizes other sources of errors, such as hardware, software, and those that might be caused by system console operators, and terminal users.

Application Program Considerations

If you suspect a problem in one of your TCAM application programs, use this section to help you find it. The section includes suggestions for coding and examining your application programs and their interface with your TCAM message control program (MCP), a summary of message handler macro instructions that can affect your application program, and a checklist of common errors to help you isolate your problem. An application program is just another terminal as far as TCAM is concerned. It is a valid destination for messages, and must have a destination queue to which the GET is issued. The location of the queue is specified by the QUEUES= operand of the TPROCESS macro. The QNAME= parameter on the DD card specifies the name of the process entry with which the destination queue is related.

Examining and Coding an Application Program

When you begin to write an application program to run as part of a TCAM system, you should write your program and its MCP message handlers as simply as possible, and use only enough code to establish the TCAM interface and to test the transfer of messages or data between your program and the MCP. After you have successfully tested the interface, you can easily add more sophisticated code.

Before you code or diagnose application programs, you should be thoroughly familiar with *Writing TCAM-Compatible Application Programs* in the *TCAM Programmer's Guide.* Study carefully also, the discussion of the LOCK macro and how to code it, and how to code DCBs and PCBs, since severe errors can result from their misuse or non-use.

Define and open DCBs for the application program *in* the application program. Test for successful open for every data set (DCB) for which you issued an OPEN macro.

Define one PCB macro in the MCP, *not* in the application program, for each application program. Do *not* issue an OPEN for a PCB.

Define one TPROCESS macro in the MCP for each queue used by an application program—one for GET or READ, one for PUT or WRITE. More than one TPROCESS macro can name the same PCB. If two TPROCESS macros name the same PCB, the GET or READ TPROCESS macro *must* specify the QUEUES= operand, and the PUT or WRITE TPROCESS macro must *not* specify the QUEUES= operand.

If you will issue operator commands from an application program, you must code the ALTDEST= operand on the TPROCESS macro for the PUT or WRITE to name the terminal that is to receive replies. Otherwise, any reply to operator commands is sent to the dead-letter queue, or, if no dead-letter queue is specified, the reply is lost.

You can run application programs as separate tasks or as subtasks of the TCAM MCP, but, in either case, they *must* have a priority lower than that of the MCP. If the application program runs as a separate task, lower its priority with the OS CHAP macro. If the program runs as an attached subtask, lower its priority with the LPMOD= operand of the ATTACH macro.

All application programs must follow standard linkage conventions in saving and restoring the registers of the calling program, whether the program runs as a subtask or as a separate task.

You must close each application program, since TCAM does not close normally as long as there are any open data sets (DCBs) in the application programs. The SETEOF macro, used with the EODAD= operand of the application-program input DCB macro, is not intended to do this. You can use SETEOF this way, however, if you ensure that the DCBs are open when a GET or READ is issued, but closed when the closedown command (Z TP) is issued.

If the application program runs as a separate task, the system operator can close it with the CANCEL command. However, if it runs as a subtask, you must close it some other way, since the CANCEL command cannot locate the application program. One way to close an attached subtask is to have the application program test for a special closedown message sent to it by a terminal, and to branch to a closedown routine when it receives this message.

Remember that TCAM sometimes uses part of the work area you defined in your application program to pass data to you (see *Transferring Data Between an MCP and an Application Program* in the *TCAM Programmer's Guide.* This data can include the SAM prefix, the position field, and the name of the terminal that originated the message. You must not destroy or improperly update these fields.

For instance, if you specify OPTCD=W in the input DCB macro, TCAM places the name of the originating terminal in the first eight bytes of your work area. You can send a reply to that terminal by coding FORWARD DEST=PUT in the inheader subgroup of the application program message handler. If you code OPTCD=W, and the terminal is on a switched line with no ID characters, and if no ORIGIN macro identifies the terminal, TCAM has no way of knowing where the message was entered. This leaves the eight-byte field blank. Therefore, if you code FORWARD DEST=PUT, be sure that the work area is not blank. If you do not specify OPTCD= U on your output DCB, the work unit is assumed to be a record and TCAM will not transmit the work unit until you have indicated that it is an entire message. If you wish to transmit each work unit that you send be sure to specify OPTCD=U.

If you do not use the eight-byte prefix set up by TCAM, then code the destination terminal name in the message, just as you would for any terminal, and code a normal FORWARD macro in the message handler.

Any message sent by the application program should include a carrier return and an EOT. If you omit the EOT, the terminal will time out waiting for an EOT.

Use the MSGFORM macro to insert the EOT character automatically where needed. Use of the MSGFORM macro is restricted to the outheader subgroup of the message handler and should be the first macro after OUTHDR to assure its execution.

Either messages sent by an application program to a terminal must be coded in the line code for that terminal, or you must issue a CODE macro in the outgoing message handler for that terminal. If you use line code in your application program, then:

1. the types of terminals to which you can send messages is limited to those of a common line code, and
2. the chances of error are greatly increased.

If you use EBCDIC and translate messages to line code with the CODE macro, then:

1. you can send messages to any terminal in the system and
2. messages are error-free.

The amount of main storage and time you save by trying to use line code in the application program is usually not enough to offset the disadvantages.

To make your system more efficient, be sure that the work-unit size in the application program is compatible with the buffer size in the MCP. See *Application-Program Buffer Design Considerations* and *Transferring Data Between an MCP and an Application Program* in the *TCAM Programmer's Guide.* Note particularly the restrictions at the end of the latter section.

If you code any non-TCAM macros (for instance, STAE, SYNADAF, or SYNADRLS), or if you use the SYNAD= or any other exit, read the appropriate OS publications. The *TCAM Programmer's Guide* covers *only* what affects TCAM.

Message Handling for an Application Program

Six message handler macros affect or can affect an application program; seven macros cannot be coded in the message handler for an application program. The macros that you cannot code are CUTOFF, LOCK, MSGFORM, MSGGEN, MSGLIMIT, SCREEN, and UNLOCK. Following is a summary of the macros that can affect an application program.

Macro	*Function*
CODE	Code this macro if you want to issue operator commands in your application program.
COUNTER	Use this macro to statistically record message volumes processed in your program (such as the total messages in and out, data volume handled, types of messages).
FORWARD	Use this macro explicitly if your messages include the destination (DEST=**) or, if you define the destination in the PUT work-area prefix of your application program, use DEST=PUT.
MSGEDIT	Use this macro to deblock output messages going to your application program by inserting record delimiter characters (as specified in the RECDEL= operand of the TPROCESS macro). Use it also to delete insignificant data from input messages.

PRIORITY	The priority level specified places messages on the read-ahead queue in priority order. There is no further priority processing.
SETEOF	Use this macro to enter your EODAD routine. The application program enters the EODAD routine when it receives the message following the message for which SETEOF executes.

Typical Errors

Following is a list of common errors that can be made in coding an application program and its interface in the MCP. It is in the form of questions, with YES/NO answers, against which you can examine your code.

Question	*Right*	*Wrong*
1. Did you follow standard linkage conventions?	YES	NO
2. Did you code an OPEN for each DCB?	YES	NO
3. Did you check each OPEN for successful completion?	YES	NO
4. Did you issue an OPEN for a PCB?	NO	YES
5. Did you destroy or overlay your work-area prefix?	NO	YES
6. Did you code closedown procedures?	YES	NO
7. Is your work-area size compatible with TCAM buffer size?	YES	NO
8. Are your incoming and outgoing work units compatible?	YES	NO
9. Is your destination correct for a lock response?	YES	NO
10. Did you code the QUEUES= operand of the TPROCESS macro for GET or READ?	YES	NO
11. Did you include an EOT in every message from your application program or MSGFORM macro in your outheader subgroup of the terminal receiving the message?	YES	NO
12. Do your application programs have lower priority than your MCP?	YES	NO
13. Did you specify a terminal to receive replies from operator commands (in the ALTDEST= operand of the TPROCESS macro)?	YES	NO
14. Did you specify a record delimiter for fixed-length records or messages?	YES	NO
15. Did you specify enough buffer units?	YES	NO
16. Is your work-area size for copy functions large enough when using the TCOPY macro or when displaying the option fields by an operator control command?	YES	NO
17. Did you specify a work-unit size for PUT or WRITE?	YES	NO
18. Did you activate your application program before you started your MCP?	NO	YES
19. Did you omit any DD statements?	NO	YES
20. Did you specify initiate mode for a single-buffer message?	NO	YES
21. Did you omit the BLKSIZE= operand of the DCB macro for GET in locate mode?	NO	YES
22. If you are using initiate mode, is the *conchars* string entirely in the first buffer?	YES	NO
23. When you are using message processing, did you specify the OPTCD=U operand of the DCB macros?	YES	NO
24. Did you check all return codes provided by TCAM?	YES	NO
25. If you specified OPTCD=W on the INPUT DCB macro, did you make your work unit eight bytes larger than the buffer size defined on the DCB macro?	YES	NO

Message Control Program Considerations

As a system programmer writing a message control program (MCP), you have five basic tasks:

1. Defining TCAM terminal and line control areas.
2. Defining the buffers TCAM uses to handle, queue, and transfer message segments between communication lines and queuing devices.
3. Defining TCAM data sets.
4. Activating and deactivating TCAM and its data sets.
5. Defining the message handlers, the sets of routines that examine and process control information in message headers, prepare message segments for forwarding to the destination, and route messages to their proper destination. The following sections are lists of suggestions, considerations, and typical errors in each of these coding areas.

Defining TCAM Terminal and Line Control Areas

If you suspect a problem in your terminals or lines, review this section to help you find it. You should also be familiar with *Defining Terminal and Line Control Areas* and *Appendix G. Device-Dependent Considerations* in the *TCAM Programmer's Guide.*

General Hardware Considerations

You must know your hardware. Incorrect coding of polling and addressing characters is a common error. You can find these characters, along with end-to-end control sequences, in hardware publications.

All terminals connected to a given line must have the same characteristics.

Use transparent mode for BSC devices if you send messages containing binary data, fixed- or floating-point data, packed decimal digits, source programs, or object decks, because the binary structure of a character may be the same as that of a data-link control character.

TERMINAL Macro Instruction Considerations

Code a TERMINAL macro for a group entry that represents a group of terminals on a line that has the group addressing hardware feature and is for output only. Specifying a single set of unique addressing characters sends messages simultaneously to all terminals in the group. If you also want to address or poll a member of the group individually, you must code another TERMINAL macro for that entry.

Code a TERMINAL macro with the operand UTERM=YES for a line entry that defines a switched line for input or input/output operations. The stations on the line do not necessarily identify themselves when calling the computer.

Issue TERMINAL macros for stations on the same line together. Do not code two TERMINAL macros with different names for the same buffered station, since message segments may become intermixed during receiving or sending, and a text segment may be treated as a header.

Specify ALTDEST= in the TERMINAL macro for terminals on reusable disk queues. When a reusable disk is cleaned up, TCAM requeues any unsent messages in the queue for the terminal specified. If you omit this operand, unsent messages on the queue are marked serviced and may be overwritten and lost with no error indicated. It is preferable not to specify the alternate destination with the same name as this TERMINAL macro. If you do, and if there is hardware trouble on the line, your messages are not lost, but they consume both space on the queue and processing time to move them on each cleanup.

Option Field Considerations

The OPTION macro specifies the name and type of the option field. It does not initialize or allocate storage. The OPDATA= operand of the TERMINAL macro initializes the option field for the particular terminal entry.

You can assign option fields having identical names and attributes but different contents to different stations, components, lines, or application programs.

Example:

```
COUNT   OPTION H
MSGLMT  OPTION CL1
REDRECT OPTION CL3
ERRMSG  OPTION CL4
```

The OPTION macros define a 10-byte option area for entries in the terminal table. If the OPDATA= operand of terminal A (a 1050) was coded OPDATA= (0, 0, NYC, PITT) a 10-byte storage area would be set aside in the option table for use by MH macros in handling messages to and from terminal A. The COUNT and MSGLMT field would initially contain 0, REDRECT would contain NYC, and ERRMSG would contain PITT. If the OPDATA= operand for terminal B (a 2740) was coded OPDATA= (,,ALA,CHI), a 7-byte storage area would be set aside in the option table for use by MH macros in handling messages from terminal B. REDRECT would contain ALA and ERRMSG would contain CHI.

The order in which you code OPTION macros determines the order in which you must code the initial data in the OPDATA= operand of the TERMINAL macros.

Do not waste space in your option table. For example, if you code

```
AA OPTION FL1
AB OPTION CL4
AC OPTION H
```

you waste a byte of storage, since AC must be on a halfword boundary.

Other Considerations

Do not use main-storage-only queuing in a LOGTYPE macro. If the log DCB is not open, messages build up in main storage and exhaust your buffer units.

Typical Errors

Following is a list of common errors that can be made in coding terminal and line control areas. It is in the form of questions, with YES/NO answers, against which you can examine your code.

Question	*Right*	*Wrong*
1. Does the UCBTYPE field in the UCB for the line in the nucleus specify the correct characteristics for the terminal or terminals on the line?	YES	NO
2. Did you consider device dependencies? (See Appendix G of *TCAM Programmer's Guide.)*	YES	NO

3. Are your polling and addressing characters correct?	YES	NO
4. Do all terminals connected to a given line have the same characteristics?	YES	NO
5. Did you issue TERMINAL macros in a line group together and in ascending relative line sequence?	YES	NO
6. Did you immediately follow the TERMINAL macro for a station with the TERMINAL macros for the individual components of that station?	YES	NO
7. Did you specify BFDELAY= in the TERMINAL macro for a terminal other than a 2740 Model 2 or a multipoint 2770?	NO	YES
8. Did you define a TPROCESS macro for each queue to which an application program can issue a GET or READ?	YES	NO
9. Did you define at least one TPROCESS macro for all PUTs and WRITEs from the same application program?	YES	NO
10. Did you code a name on each OPTION macro?	YES	NO
11. Do your OPTION macros immediately follow the TTABLE macro?	YES	NO
12. If you have OPDATA= defined in the TERMINAL macro, did you replace option fields not defined for the particular entry with a comma (except trailing commas)?	YES	NO
13. Is the BUFSIZE= operand of the LOGTYPE macro a multiple of the value specified in the KEYLEN= operand of the INTRO macro?	YES	NO
14. Does the NCP= operand of the LOG DCB have a value which is at least the number of units in the buffer you are going to be logging? (NCP= is the number of writes before a check.)	YES	NO

Defining TCAM Buffers

If you suspect a problem in your buffers, review this section to help you find it. You should also be familiar with *Defining Buffers* in the *TCAM Programmer's Guide.*

Remember that a buffer is made up of one or more buffer units. A buffer unit can be between 35 and 255 bytes, and a buffer can be between 35 and 65535 bytes.

Use larger buffers (more units per buffer) because:

1. Fewer buffers are required for a message. Therefore, TCAM requires less overhead to manipulate buffers.
2. When you use dynamic buffer allocation (PCI), the possibility of losing data because of a delayed PCI is decreased.
3. The number of PCIs required, if PCI is specified, is decreased.
4. You make better use of the TCAM disk accessing method (multiple-arm support), because there is a larger number of contiguous records than there would otherwise be.
5. There are fewer queuing operations per quantity of data; this saves time.

Use smaller buffers (fewer units per buffer) because:

1. Units in smaller buffers return to the available-unit queue more rapidly than units in larger buffers, since it takes less time to empty and fill a smaller buffer. Therefore, you can have a smaller unit pool since allocation of resources occurs more frequently.
2. TCAM's work load is broken into smaller pieces, resulting in a more equitable allocation of processing time among message segments in main storage.

Use more units in the system because:

1. You are less likely to lose message data coming in over a line.
2. You are less likely to delay outgoing messages due to waiting for a buffer.

Use fewer units in the system because:

1. Main storage is used more efficiently. Since the number of units in the free pool is not excessive, you save main storage.

Use larger units because:

1. Disk space is used more efficiently, since there are fewer interrecord gaps.
2. The area available for text compared to the area containing management information is relatively large.
3. Since more data is transmitted per CCW on lines and disk, the channel activity is relatively light; this saves channel fetch and CPU time.
4. You need fewer channel program blocks (CPBs) to transfer the same amount of data to and from disk; this saves storage space and time, since there is less CPB queuing.

Use smaller units because:

1. Duplicate headers, used for multiple routing of messages, take up relatively little room.
2. You can specify a relatively large range of buffer sizes without wasting space in main storage and on disk.
3. You can reallocate buffers more frequently with smaller units, since they pass through the system more rapidly than larger units.

Use dynamic buffer allocation because:

1. When you code PCI=A, fewer buffers are assigned initially to a line, since dynamic allocation brings the number of buffers assigned up to the value specified by BUFMAX= and maintains this number if possible.
2. When you code PCI=A and a negative response to invitation occurs, only the number of buffers assigned initially, rather than the maximum number assigned to the line, have been fruitlessly allocated.
3. When you code PCI= as A or R, buffers are continuously deallocated. The free-unit pool is therefore continuously being replenished, and a smaller unit pool is required.
4. When you code PCI= as A or R, a message moves one buffer at a time; therefore, fewer CPBs are required to achieve the same performance.

Use static buffer allocation because:

1. Dynamic allocation and deallocation of buffers takes processing time.
2. When you use reusable disk queues, records written to disk by the PCI interrupt are not serviced until the entire message is queued. If the length of time required to enter a message is excessive, or if reusability servicing is very frequent, records may be overlaid. If this occurs, TCAM terminates abnormally with a system code of 045 and a return code of 02 or 03 in register 15.

For start-stop lines using dynamic allocation, if you specify BUFIN=2, BUFMAX=2, dynamic allocation may be inefficient.

The number of buffers you assign initially to each line (BUFIN= and BUFOUT= operands) depends on:

- terminal type,
- terminal speed,
- line speed,
- whether dynamic allocation of buffers is specified.

The faster the data is transmitted, the higher the initial assignment should be.

For high-speed BSC lines, dynamic allocation may not be totally effective; that is, there may not be a one-to-one correspondence of replacement buffers to replaced buffers.

Remember that a line does not have both BUFIN= and BUFOUT= assigned at the same time. In deciding how many units to define, you need be concerned only with the initial requirements for send or receive operations. A formula to approximate how many units you need in your system is:

1. Determine for each line the maximum, average, and minimum message length.
2. Select the optimum buffer size for each line group for input and output.
3. Based on all line group buffer sizes, select an optimum unit size for the message control program.
4. Based on optimum unit size, re-specify buffer sizes for each line group to more efficiently utilize the units.
5. To Determine the maximum line units required for all lines, take the sum of the product of the maximum number of buffers for each line multiplied by the quotient of buffer length divided by unit length.

$$\text{LNUNITS} = \Sigma \left(\begin{bmatrix} \text{Maximum} \\ \text{Number} \\ \text{of Buffers} \\ \text{Per Line} \end{bmatrix} * \begin{bmatrix} \dfrac{\text{Buffer Length}}{\text{Unit Length}} \end{bmatrix} \right)$$

If you use disk queuing, try to make the buffer size specified by the source of a message equal to the buffer size specified by the destination. When the buffer sizes specified for the origin and destination are different, data movement occurs because TCAM must add or delete prefixes when it places the message in the buffers for the destination. *Moving data takes time.*

Remember that BUFIN= or BUFOUT= is satisfied when a line is opened active. When you start an operation and have dynamic buffering, BUFMAX= is satisfied. Do not be frugal with your unit-pool size. If you are, you degrade your system, since TCAM does not have enough buffer units to perform adequately.

Operator commands from stations and application programs must be contained in a single line buffer; if the buffer is too small, the command is truncated and an attempt is made to process it.

You can spot unused buffer units in the buffer-unit pool because they have only the link field filled in the prefix. The remainder of the buffer prefix and unit are zeros.

Typical Errors

Following is a list of common errors that can be made in defining buffers. It is in the form of questions, with YES/NO answers, against which you can examine your code.

Question	*Right*	*Wrong*
1. To save main storage, is 12+KEYLEN=evenly divisible by eight?	YES	NO
2. Is the BUFSIZE= operand on the DCB macro evenly divisible by the unit size specified in the KEYLEN= operand of the INTRO macro?	YES	NO
3. Did you allow room for the buffer prefix (30 bytes for a header buffer and 23 bytes for a text buffer)?	YES	NO
4. Is each buffer unit at least 35 bytes and no longer than 255 bytes (not counting the 12-byte control area that TCAM adds)?	YES	NO
5. Is each buffer at least 35 bytes and no longer than 65535 bytes?	YES	NO
6. For BSC lines using dynamic allocation, did you code the BUFMAX= operand at least two greater than the larger of BUFIN= or BUFOUT=?	YES	NO
7. Is BUFMAX≥BUFIN/BUFOUT?	YES	NO
8. Are your buffers long enough to hold an operator control command?	YES	NO

Defining TCAM Data Sets

If you suspect a problem in your data sets, review this section to help you find it. You should also be familiar with *Defining the MCP Data Sets* in the *TCAM Programmer's Guide.*

Line Group

A line group may consist of from one to 255 lines. The size of a line group is limited by the fact that the INVLIST= operand of the DCB macro can be no longer than 255 characters, including commas; thus you cannot have 255 invitation lists for a line group.

All lines in a group must have the following common characteristics:

1. All must be switched or all must be nonswitched.
2. All use start-stop or all use binary synchronous transmission.
3. All lines are associated with stations having the same device characteristics.
4. All use the same invitation delay.
5. All use the same message handler.
6. No line in the group is a member of another group.
7. All are preassigned the same number of buffers to handle initial segments of incoming messages.

Be aware of the A/B suboperands on the INVLIST= operand if you use a 2701 Transmission Control Unit.

Any number of output-only lines may refer to the same invitation list name.

The RESERVE= operand reserves space in incoming header units and text units, although data may be inserted in either the incoming or outgoing message handler.

If there is not enough space (if BUFSIZE= is too small), the macros that insert data (DATETIME, SEQUENCE) do not execute.

Be sure you use the right translation table, and know its characteristics. For instance, a folded table recognizes both uppercase and lowercase letters as valid.

You must code the SCT= operand if you specify your own table in the TRANS= operand. The SCT= operand must be a valid TRANS= entry. You cannot specify your own special characters table.

When you concatenate DD statements for a line group, their arrangement determines the relative line numbers of the lines. The relative line number is a number assigned by you to a communications line of a line group at system generation time or MCP execution time. If a line group is defined at system generation time by the UNITNAME macro, the lines in the group are assigned relative line numbers according to the order in which their hardware addresses are specified in the UNIT= operand; the line whose address is specified first is relative line one, the address specified second is relative line number 2, etc. If a line group is defined at MCP execution time by concatenated DD statements, the arrangement of the DD statements determines the relative line numbers for the lines.

Example:

```
//GROUPONEDD  UNIT =  015
//        DD  UNIT =  016
//        DD  UNIT =  017
```

Line 015 is RLN=1, line 016 is RLN=2, and line 017 is RLN=3. Since RLN= operand is assembled in your MCP in the TERMINAL macro, the order of the DD cards cannot be disturbed. If one is removed, a dummy must replace it.

Do not have more DD cards than INVLIST = operands in the DCB.

Message Queues

Remember that one channel program block (CPB) is involved whenever the contents of a buffer unit are written to disk or read from disk. The number you need depends on the amount of message traffic during the peak period of activity for the TCAM system.

Too few CPBs cause poor disk performance. Messages are delayed while TCAM waits for CPBs to become available to place the messages on or remove them from disk.

Too many CPBs waste main storage.

To investigate CPB availability, AVT+X'46C' points to the first entry in the CPB free pool. The thirteenth word points to the next lower CPB entry on the queue. In a dump, if the first few words of the CPB are zero, then that CPB and all that follow are unused. If no CPBs are zero, then you probably need more CPBs.

When you preformat your disk data set, using utility IEDQXA, be sure that the KEYLEN= operand is the same as that specified on the INTRO macro when you attempt to open the data set.

You increase disk efficiency if you space disk message queues data sets over several volumes.

On the DCB macro for a message queues data set, be sure that the OPTCD= operand has the correct specification:

OPTCD=L for nonreusable disk data sets
OPTCD=R for reusable disk data sets.

Checkpoint and Log

In the DD statement for the checkpoint data set, if you specify DISP=NEW, you will always get a cold restart.

If you log both messages and message segments, define two separate data sets. You can have only one LOGTYPE macro per DCB.

Typical Errors

Following is a list of common errors that can be made in coding DCBs for TCAM data sets. It is in the form of questions, with YES/NO answers, against which you can examine your code.

Question	*Right*	*Wrong*
1. Did you specify one line group DCB macro for each line group in the system? (Does each line have a DCB associated with it?)	YES	NO
2. Do you have more than 255 lines in a line group?	NO	YES
3. Are the BUFIN=, BUFOUT=, and BUFMAX= operands of the DCB all specified from the same source?	YES	NO
4. Are the listnames in the INVLIST= operand specified according to ascending relative line numbers of the lines in the group?	YES	NO
5. Is there one invitation list name in the sublist for each line in the line group?	YES	NO
6. Did you include framing parentheses in the PCI= operand (for instance, PCI=(A,A))?	YES	NO
7. If you specify CPRI=R and you want to send output messages to the terminal, did you code a polling interval delay in the INTVL= operand?	YES	NO
8. Did you include at least one DD statement for each line group data set?	YES	NO
9. Did you specify at least two CPBs for reusable disk queuing?	YES	NO
10. Did you specify at least one CPB for nonreusable disk queuing?	YES	NO
11. Do you have at least as many CPBs as the maximum number of buffer units per buffer in the system (so that an entire buffer can be dispatched with a minimum number of operations)?	YES	NO
12. Is the KEYLEN= operand on the log data set DCB macro the same as the KEYLEN= operand on the INTRO macro?	YES	NO

Activating and Deactivating TCAM

If you suspect a problem in activating or deactivating TCAM, review this section to help you find it. You should also be familiar with *Activating and Deactivating the Message Control Program* in the *TCAM Programmer's Guide,* GC30-2024.

INTRO Macro

Do not code the INTRO macro until you have read the sections in the *TCAM Programmer's Guide* for the functions to which the operands refer.

You should allow for a dynamic INTRO macro by omitting one of the following operands when you assemble:

```
STARTUP=
KEYLEN=
LNUNITS=
if DISK=YES,CPB=.
```

In response to the message

```
IED002A SPECIFY TCAM PARAMETERS
```

each response can be a maximum of 41 characters. You keep getting the same message until you specify 'U', which indicates that you have no more operands to enter.

If you still omit one of the four required operands, TCAM tells you the specific operand missing.

An error in a keyword for an operand in the reply prevents interpretation of any keywords in the same response to the right of the keyword in error.

MSMIN= must be less than MSMAX= or the INTRO macro does not execute. If you change these values at execution time, the value is compared to the current values, if specified, to see that the rule is not broken. If you specify at assembly time

```
MSMAX=90,MSMIN=85
```

and at execution time

```
MSMIN=95,MSMAX=99
```

INTRO will not execute because 95 is greater than 90.

You should specify the operands that provide the trace tables:

```
CROSSRF=
TRACE=
DTRACE=
```

You should provide a dead-letter queue (DLQ=) for your network.

Be sure to check the return code after the INTRO macro executes. If it is anything other than zero, the MCP is unlikely to work satisfactorily, and you should deliberately ABEND in the MCP.

OPEN Macro

Opening a line group data set causes all lines in the line group to be prepared for operation. You can defer activation until later by opening the line idle and later issuing the STARTLINE operator command.

Open your data sets in the correct order:

First: message queues data sets
Next: checkpoint data set
Last: line group and log data sets

If you open a large number of data sets, you must be conscious of the base register. You can use the following procedure before your first OPEN macro.

```
   BASE       DC            A(DCBSTART)
              L             2,BASE
              USING         DCBSTART,2
              OPEN
              .
              .
              .
              DROP          2
   DCBSTART   EQU           *
              DCB Macros
```

Check each OPEN to see if it was successful (test DCBOFLAGS at DCB+X'48' with a mask of X'10'), and inform the system console of the result of the test. This provides immediate information about the status of your network. You will know if all your data sets were opened, and eliminate useless diagnosing of an error caused by an unopened data set. A recommended procedure is

```
      OPEN    (DISK,(INOUT))
      TM      DISK+48,X'10'
      BO      NEXT
      WTO     'REUSABLE DISK NOT OPEN'
NEXT  OPEN    (DCB1050,(INOUT))
      TM      DCB1050+48,X'10'
      BO      NEXT1
      WTO     '1050 DIAL LINE NOT OPEN'
NEXT1....
```

READY Macro

Remember that "good morning" and "restart in progress" messages pass through the outgoing message handler, and need an appropriate header.

After READY executes, TCAM is ready for message processing.

CLOSE Macro

The CLOSE macros must follow the READY macro or be branched to from instructions immediately following READY.

Be sure you close your data sets in the correct order:

First: line group and log data sets
Next: checkpoint data set
Last: message queues data sets

Typical Errors

Following is a list of common errors that can be made in coding the activation and deactivation section of an MCP. It is in the form of questions, with YES/NO answers, against which you can examine your code.

Question	*Right*	*Wrong*
1. Do the INTRO, OPEN, and READY macros precede the message handler sections of the MCP?	YES	NO
2. Do any instructions that you coded before the INTRO macro contain any TCAM macros? (INTRO expects to be first. It gets control from BALR 14, 15 with the save area set. It expects to get control from OS.)	NO	YES
3. Did you specify both the KEYLEN= and the UNITSZ= operand for the buffer-unit size?	NO	YES
4. Is MSMIN= less than MSMAX=?	YES	NO
5. Did you code FEATURE=(,,TIMER) if you use any of the following functions: checkpoint, any interval, dial-out options, main-storage queuing, reusable disk queuing?	YES	NO
6. Did you check the return code after the INTRO macro executes?	YES	NO
7. Are your OPEN macros in the correct order (disk data sets, then checkpoint data set, then line group and log data sets)?	YES	NO
8. Did you check each OPEN to see if it was successful?	YES	NO
9. Before you closed TCAM, were all data sets for application programs closed (for instance, with a special message)?	YES	NO
10. Does the deactivation section of your MCP end with a RETURN macro?	YES	NO
11. Did you prepare for the return by loading register 13 with the save area address?	YES	NO
12. Are your CLOSE macros in the correct order (line groups and log data sets, then checkpoint data set, then disk data sets)?	YES	NO

Queuing

If you suspect a problem in queuing, review this section to help you find it. You should also be familiar with *Defining the MCP Data Sets* in the *TCAM Programmer's Guide.*

Main-Storage Queues

Main-storage-only queuing is the fastest method in response time, but it uses more main storage than any other method.

For main-storage-only queuing, when you use a distribution list, the multiple routing and redirect routines place another copy of the header buffer in main storage for each station in the list.

Avoid main-storage queuing for a log data set if at all possible, as it can use up your main-storage buffer units (MSUNITS) very quickly, especially if the log data set is not open or going directly to the printer.

Nonreusable Disk Queues

Nonreusable disk queuing requires more space on the disk than reusable queuing.

Nonreusable disk queuing may require periodic system closedown to clean up the disk queues. If the nonreusable disk queue fills up and the closedown fails because the message TCAM was receiving was too big to fit in the remaining space on the disk, TCAM terminates abnormally with a system code of 045.

Reusable Disk Queues

Reusable disk queuing requires periodic reorganization. Response time during reorganization may be longer.

Reusable disk queuing can often handle the same amount of message traffic as nonreusable queuing, while occupying less disk space.

Messages that are unsent and have no alternate destination are lost when the reusable disk data set is reorganized.

Message queues on reusable disk never run out of space under normal conditions.

You can compromise by specifying main-storage queuing with backup on reusable disk. This preserves most of the advantages of disk queuing, while achieving a faster response time than with disk queuing alone.

You limit TCAM's capability to retrieve messages that have already been sent when you use reusable disk queuing, because the original copy of a transmitted message is eventually overlaid by another message.

Queuing by Line

If you queue by line, you can send messages by priority on a line basis to stations on a multipoint nonswitched line. All messages of a given priority on the queue are transmitted before any message of a lower priority, whether or not the higher-priority messages are destined for two different stations on the line.

If you queue by line, you need less storage than if you queue by terminal. If you queue by line rather than by terminal, you save at least 65 bytes for each station after the first on a line, plus about 28 bytes per station after the first for each priority level specified beyond one.

If you queue by line, you will switch between stations on the line rather than maintain connection with a station.

Queuing by Terminal

You must specify *queuing by terminal* for switched stations and for buffered terminals. If you queue switched stations by line, a station that calls in receives not only its messages, but those for all other stations in the line group as well.

If you queue by terminal, you can send messages by priority on a station-by-station basis. All messages in a given queue for a station on a line are transmitted before any messages in other queues for the remaining stations on the line are transmitted, whether or not the other queues contain messages with priorities higher than those for the messages being transmitted.

If you queue by terminal, you need more storage than if you queue by line.

Other Considerations

You use more main storage by mixing queue types than by specifying only one queue type for all terminals.

A segment for which the INITIATE macro has been executed is treated as if it were a completed message having the highest priority on the queue, and is sent before any other message on the queue is sent. In addition, no message on the queue is sent until all segments of the message for which INITIATE was executed have arrived at the queue and been sent to their destination.

Disk queuing ties up disk space and disk channels that could otherwise be used by other jobs.

Typical Errors

Following is a list of common errors that can be made in defining queues. It is in the form of questions, with YES/NO answers, against which you can examine your code.

Question	*Right*	*Wrong*
1. Have you specified the type of disk queuing you want (the OPTCD= operand on the DCB macro specifies the type; L is nonreusable and R is reusable)?	YES	NO
2. Did you try to use the HOLD macro with main-storage-only queues?	NO	YES
3. Did you try to retrieve messages from main-storage-only queues?	NO	YES
4. Did you try to take checkpoints of main-storage-only queues?	NO	YES
5. Did you specify queuing by terminal for switched stations or for buffered terminals?	YES	NO
6. If you are using main-storage queuing with disk backup, did you define at least two message queues data sets, one residing in main storage and the other on reusable or nonreusable disk?	YES	NO

Defining the Message Handlers

If you suspect a problem in a message handler, review this section to help you find it. You should also be familiar with *Designing the Message Handler* in the *TCAM Programmer's Guide.*

Delimiter Macros

The STARTMH macro identifies the beginning of a message handler (MH).

You may omit either the incoming or the outgoing group of the message handler.

Remember:

1. INHDR and OUTHDR handle only those message segments that include all or part of a message header.
2. INBUF and OUTBUF handle *all* message segments.
3. INMSG and OUTMSG execute after the complete message has arrived at the CPU or been sent.

You can code *one and only one* INEND and OUTEND macro in an MH.

Message Format

Depending on the application, messages may consist of a header only, text only, or header and text. You determine what is header and what is text.

You should design your message format so that each message starts with a specific character (any character will do). Otherwise, carrier returns, spaces, etc., entered before the actual message, make it virtually impossible to find the start of data. You will find that most terminal operators return the carriage several times to assure themselves that the terminal is turned on and working. If your message format starts with an X, a sample sequence might be:

```
INHDR
CODE
SETSCAN C'X'
```

You have now passed over any miscellaneous characters that may have been put on the line before your message, and you know exactly where valid data starts.

You should include in your message format an end-of-address (EOA) character to allow you to route messages to multiple destinations. (This EOA is not to be confused with the hardware-generated line-control character). This also gives you another landmark that you can use to separate the actual text of the message from its header. You must have an end-of-address for multiple routing. Your message format might be

```
X origin dest1 dest2 dest3 / ... text ... EOT
```

where / is the EOA character.

Scan Pointer

The scan pointer maintained by TCAM points to the current field in the message header. Since some macros use the pointer to locate the field on which they act, you must be aware of the scan pointer position when designing your message handlers. Macro instructions in a message handler should be placed in the same order within a subgroup as the fields of the header on which they act. The scan pointer controls access to these fields, processing across the header from left to right as the various macro instructions are executed.

Note 1: *If you code LC=IN in the STARTMH macro and plan to issue operator commands from remote terminals, you must move the scan pointer to the first data byte before issuing the CODE macro. Example 1 shows incorrect code.*

Example 1: The MH is coded

```
STARTMH      LC=IN
INHDR
CODE
```

CONTROL=OPID is coded in the INTRO macro.

The operator command

```
OPID V   01C,ONTP
```

is issued from a 1050 terminal.

The buffer has the following format:

Unit Control Area	Buffer Prefix	Ⓓ	OPID V 01C,ONTP Ⓑ Ⓒ

↑
position of scan pointer

The CODE macro determines if the message entered is an operator command by matching the characters specified on the CONTROL= operand with the character string following the scan pointer. A valid match will not be detected since the CODE macro compares ⒹOPI with OPID. To be correct, the MH should be coded as shown in example 2.

Example 2:

```
STARTMH    LC=IN
INHDR
SETSCAN    1
CODE
```

The buffer will now have the following format.

Unit.Control Area	Buffer Prefix	Ⓓ	OPID V 01C,ONTP Ⓑ Ⓒ

↑
position of scan pointer

Note 2: *When a message segment is received for processing in an incoming group of a message handler, the space reserved for expansion by the RESERVE= operand of the line group DCB or PCB macro is moved to the front of the segment and the scan pointer is positioned to the last reserved byte.*

Example 3: RESERVE=23 specified in the line group DCB for the line that sent the message.

12	30	23	
Unit Control Area	Buffer Prefix	Reserve Bytes	Data

↑
position of scan pointer

If no reserve bytes are specified, the scan pointer points to the last byte of the buffer prefix.

Example 4: No reserve bytes

12	30	
Unit Control Area	Buffer Prefix	Data

↑
position of scan pointer

When a message segment is received for outgoing processing, the scan pointer is positioned to the last remaining reserve byte, if there are unused bytes (example 3).
If there are no more unused reserve bytes or none originally specified, the scan pointer points to the last byte of the buffer prefix (example 4).

The position of the scan pointer after execution of the STARTMH macro depends on the coding of the LC= operand.
If LC=IN is coded, the scan pointer is positioned to the first line-control character.
If LC=OUT is coded, the scan pointer points to the first data text byte.

The following is a list of macros that use the scan pointer.
The scan pointer location is the starting position for macro execution.
Specific uses are indicated for some macros.

The position of the scan pointer after macro execution is also shown.

Macro	*Specific Use*	*Position of Scan Pointer After Macro Execution*
CODE	For operator control checking on the first incoming buffer	Unchanged
DATETIME		Unchanged (points to last character of inserted data)
ERRORMSG	To insert error text after header	End of error text
FORWARD	If DEST=** or DEST=(number) is coded	Last character of destination or EOA character if multiple destinations, or last character of character string
INITIATE	If control characters are used	See Note 3.
LOCK	If control characters are used	See Note 3.
MSGEDIT	If AT=SCAN If TO=SCAN	See Note 4.
MSGTYPE		See Note 3.
ORIGIN		Last character in character string
PATH	If control characters are used	See Note 3.
PRIORITY	If the priority is in the header or if control characters are used	1. Last character of priority if priority is in the header with no character string or matched character string 2. Last character of control characters if priority is in the macro
SCREEN	If control characters are used	See Note 3.
SEQUENCE		1. Unchanged if output 2. Last character in sequence number if input
SETEOF	If control characters are used	See Note 3.

SETSCAN		1. Unchanged if MOVE=RETURN 2. *n* characters forward or backward 3. Last character in character string
UNLOCK	If control characters are used	See Note 3.

Note 3:

1. *The position of the scan pointer is unchanged if an invalid or no condition is given.*
2. *The scan pointer points to the last character in a character string if a valid condition is given.*

Note 4:

1. *MSGEDIT functions performed on the buffer contents to the left of the scan pointer position before macro entry:*
   a) possible physical scan pointer movement
   b) no logical scan pointer movement

   Scan pointer before MSGEDIT

   |*AAABBBCCCCCX-----------X*|
   ↑
   position of scan pointer

   MSGEDIT replaces AAA with ZZZZ and removes BBB

   Scan pointer after MSGEDIT

   |*ZZZZCCCCCX-----------X*|
   ↑
   position of scan pointer

2. *When MSGEDIT functions are performed on buffer contents to the right of the scan pointer position before macro entry, there is no physical/logical scan pointer movement.*

Be aware of multiple-buffer header processing across buffers (see Figure 19). Try to limit your header to one buffer.

You can vary the path of a message through an MH dynamically using the PATH or MSGTYPE macro. The PATH macro controls the routing of a message among subgroups. The MSGTYPE macro controls the path of a message within a subgroup.

When you use a character string to control macro execution, do not have partially identical strings, such as:

```
MSGTYPE ABC
MSGTYPE AB
```

You should cancel all messages that are in error due to inheader processing and validity checking.

User Code

You can include either open or closed subroutines in your message handler.

Avoid system macros that issue an SVC, unless you are fully aware of the implications of using such macros in a TCAM system. This is especially true if the macro has an implied WAIT state in its execution.

Inheader and Outheader Macros	Does Not Effect Buffer Contents	Will Cross Buffers	Will Not Cross Buffers	Conditional (Note 1)
CHECKPT	X			
CODE	(Note 2)			
COUNTER	X			
DATETIME	X			
FORWARD		(Note 3)	DEST in message	
INITIATE				X
LOCK				X
LOCOPT	X			
LOG	X			
MSGEDIT			X	
MSGFORM	X			
MSGLIMIT	X			
MSGTYPE				X
ORIGIN		(Note 4)		
PATH				X
PRIORITY		(Note 5)		
SCREEN				X
SEQUENCE	output only		input only	
SETEOF				X
SETSCAN		chars	integer POINT=BACK chars, RETURN=	
TERRSET	X			
UNLOCK				X

Note 1: Will cross if conchars is not specified, or if entire character string is in a subsequent buffer.

Note 2: Except that an operator command must be complete in a single buffer.

Note 3: Will cross if destination is in the macro or an option field and the macro is executed for the first buffer.

Note 4: Will cross but origin may not be known on dial lines for first buffer.

Note 5: Will cross if conchars not specified and priority level is in macro.

Figure 19. Multiple-Buffer Header Processing Across Buffers

You can include TCAM macros in an open subroutine; you cannot include them in a closed subroutine.

When your MH handles messages with multiple-buffer headers, any code within the inheader and outheader subgroup should test register 15 for a negative return code before executing any open subroutines or before branching to a closed subroutine if the routine to be executed depends on certain data being in the buffer or on the location of the scan pointer.

Typical Errors

Following is a list of common user code errors. It is in the form of questions, with YES/NO answers, against which you can examine your code.

Question	*Right*	*Wrong*
1. If you included a subroutine is it serially reusable?	YES	NO
2. Did you include executable code with an inmessage or outmessage subgroup or between such subgroups?	NO	YES
3. Did you do anything that relinquishes control in a subroutine?	NO	YES
4. Did you include TCAM macros in a closed subroutine?	NO	YES
5. Did you supply your own linkages and save and restore registers in a closed subroutine?	YES	NO
6. Did you branch from one MH to another?	NO	YES
7. If you have code in an inheader or an outheader subgroup that may handle multiple-buffer headers, did you code USEREG= operand in the INTRO macro?	YES	NO
8. If register 13 is used in an open subroutine, did you save and restore its original contents?	YES	NO
9. In an open subroutine, did you alter the base register?	NO	YES

If you plan to test the return codes from TCAM macros, see Figure 20. A bad test, such as testing the return code in register 15 when it is in another register, can cause incorrect processing of a message.

Functional Macros

You must include the CODE macro if you plan to enter operator commands from terminals or application programs.

If you code LC=OUT on the STARTMH macro, issue CODE as the first functional macro in the inheader subgroup for a line on which operator commands may be entered.

Your error message should contain some indication as to whether the error occurred in the incoming or outgoing group.

Remember that the maximum length of an error message created by MSGGEN is 24 bytes.

When you generate messages for BSC and 2260 Local terminals, be sure to include the STX in the error message.

Use the ERRORMSG and MSGGEN macros to keep the terminal operator aware of the status of his messages (were they canceled? rerouted? why?).

Make your error messages meaningful.

The table below lists those TCAM macros whose return codes may be checked by user code in a Message Handler. The return code occupies the low-order byte in the register indicated; the rest of the register normally contains all zeros. Return codes of X'FC' are negative return codes; the high-order three bytes of the register contain binary ones. Some macros also return an address in a register; the locations and nature of such addresses are also indicated in the following table of MH macro return codes.

Macro	Register	Return Code	Meaning
COUNTER	15	X'00'	Good return
	15	X'FF'	Option field not found
DATETIME	15	X'00'	Good return
	15	X'04'	Insufficient reserve characters
FORWARD	15	X'00'	Good return
	15	X'04'	Invalid destination
LOCK	15	X'00'	Good return
	15	X'04'	Destination not specified
	15	X'08'	Destination not a process entry
LOCOPT a) if return requested in R15	15	Address of option field.	Good return
	15	X'00'	Option field not found
b) if return requested in user-specified register (USEREG)	15 USEREG	X'00' Address of option field.	Good return
	15 USEREG	X'04' Unchanged	Option field not found
LOG	15	X'00'	Good return
	15	X'04'	DCB or LOGTYPE entry named in macro not found
MSGEDIT	15	X'00'	Good return
	15	X'04'	No units available
MSGLIMIT	15	X'00'	Good return
	15	X'04'	Option field not found
ORIGIN	15	X'00'	Good return
	15	X'04'	Invalid origin
SCREEN	15	X'00'	Function not done
	15	Function byte	Good return

Figure 20. MH Return Codes (Part 1 of 2)

SEQUENCE a) macro issued in inheader subgroup	15	X'00'	Good return
	15	X'04'	Sequence number in message high
	15	X'08'	Sequence number in message low
	15	X'0C'	Originating station unknown
b) macro issued in outheader subgroup	15	X'00'	Good return
	15	X'04'	Insufficient reserve characters
SETSCAN a 1) locate specified character string and return address in R15	15	Address of last character in string	Good return
	15	X'00'	Specified character string not found in this buffer
	15	X'FC'	Scan pointer beyond end of buffer
a 2) locate specified character string and return address in user-specified register (USEREG)	15 USEREG	X'00' Address of last character in string	Good return
	15 USEREG	X'04' Unchanged	Specified character string not found in this buffer
	15 USEREG	X'FC' Unchanged	Scan pointer beyond end of buffer
b 1) skip n characters and return address in R15	15	Address or character skipped to	Good return
	15	X'00'	n greater than the number of characters remaining in this buffer
b 2) skip n characters and return address in user-specified register (USEREG)	15 USEREG	X'00' Address of character skipped to	Good return
	15 USEREG	X'04' Unchanged	n greater than the number of characters remaining in this buffer
c) skip n characters backward	15	X'00'	Good return
	15	X'04'	n greater than the number of characters preceding the scan pointer in this buffer
d) Locate scan pointer address	15	Address of scan pointer	Good return
	15	X'FC'	Scan pointer beyond end of buffer

Figure 20. MH Return Codes (Part 2 of 2)

If some hardware problem causes an error (bits are on in the last byte of the message error record), send the error message to some other terminal, not to the terminal in error.

Be careful when coding the FORWARD macro, since it is valid with no operands. If you code FORWARD PUT, no error is flagged in the assembly, since there is no keyword. TCAM sees PUT as a comment, and you get the default of DEST=**.

If you execute the HOLD macro in the outmessage subgroup for a LOCK response, the LOCK is not broken, the terminal is not held, and the message is retransmitted immediately. This can cause an infinite loop if the condition for the HOLD is permanent and the line or terminal is inoperative.

LOCK does not execute if the station that entered the message being handled is a buffered station.

Use MSGEDIT to insert idle characters at the end of messages and new lines for terminals such as the 2740 and 1050 that can write while the type element is returning to a new line.

When you code multiple groups of operands, rather than multiple MSGEDIT macros each with a single group of operands, data inserted in one operation is not considered to be part of the message segment when another operation is performed. The following summary of the MSGEDIT macro and examples are included in this section of the *OS TCAM User's Guide* as an additional aid for helping you to understand the MSGEDIT macro. In order to use the MSGEDIT macro in your program it will be necessary to refer to the *OS TCAM Programmer's Guide and Reference Manual.*

Summary:

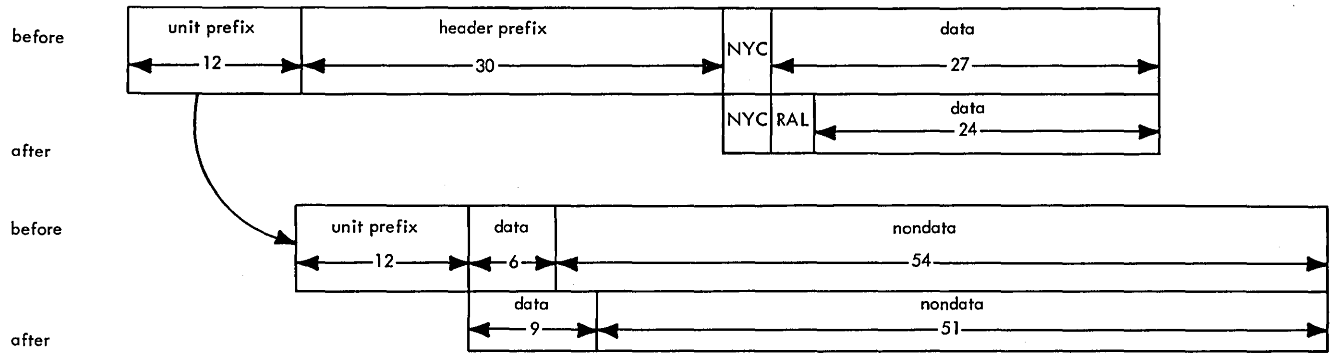
1. The MSGEDIT macro allows a maximum of 31 groups of functions.
2. Multiple MSGEDIT macros versus multiple function groups in one MSGEDIT macro is a tradeoff between speed and flexibility.
3. Group execution is independent of coding position of the MSGEDIT macro but is dependent on buffer contents.
4. All groups work on the original message contents, which means that inserted characters from one function cannot be the search argument for another group.
5. However, inserted characters from MSGEDIT macro can be the search argument for a subsequent MSGEDIT macro.
6. Data moved is all that data from the first AT= position to buffer end.
7. MSGEDIT does not require reserved space, but allocates units automatically, if additional space is needed.
8. MSGEDIT reallocates units automatically if remove functions empty units.
9. Execution of MSGEDIT in the inheader/outheader subgroups acts only on one header-buffer.
10. Execution of the MSGEDIT in the INBUF/OUTBUF acts on each buffer.
11. Execution of MSGEDIT across buffers is not possible.
12. The maximum length of a contiguous character string is eight which is the size of the AVT work area.

The following MSGEDIT examples use a keylength of 60 and a buffer size of 120.

MSGEDIT-examples

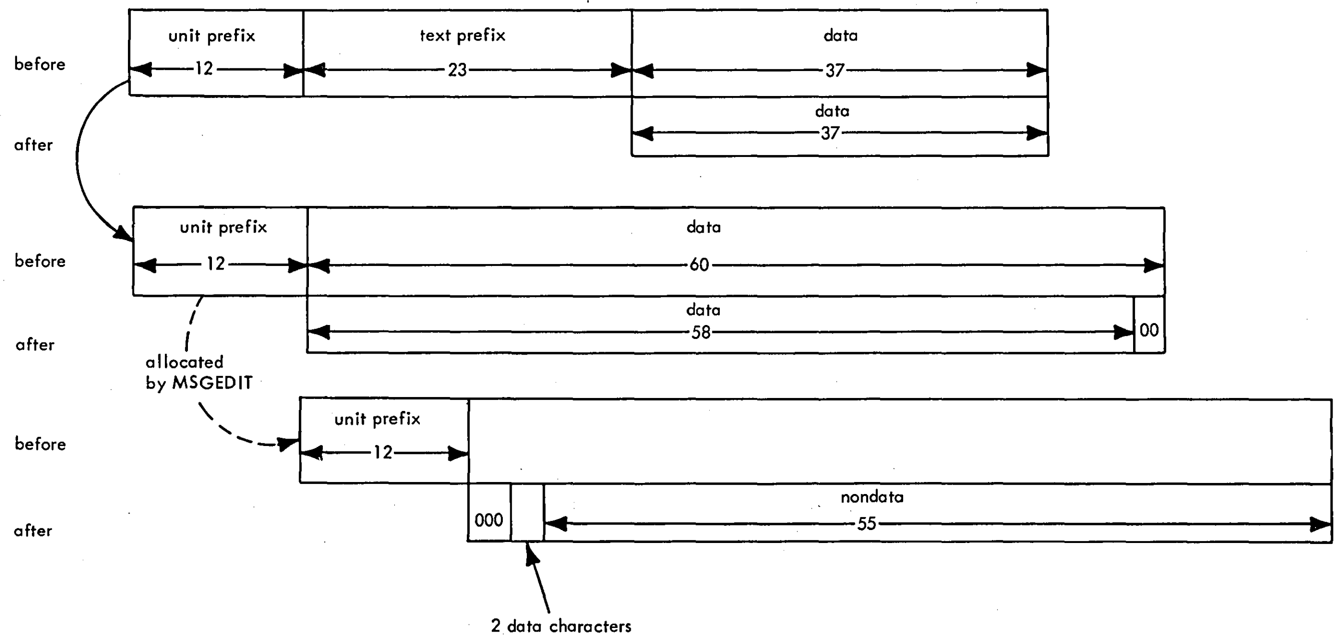
Example 1: insert char. after special char. string

```
INHDR
MSGEDIT ((I,CL3'RAL',CL3'NYC'))
```



Example 2: insert chars. after offset

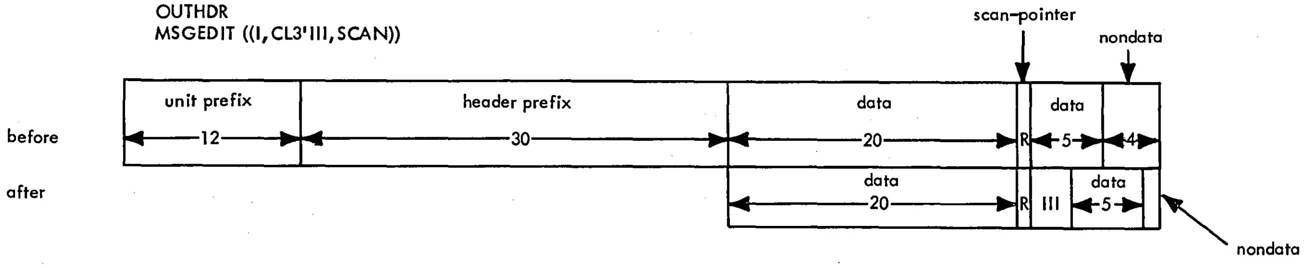
```
OUTBUF
MSGEDIT ((I,(C'0',5),95))
```



MSGEDIT-examples

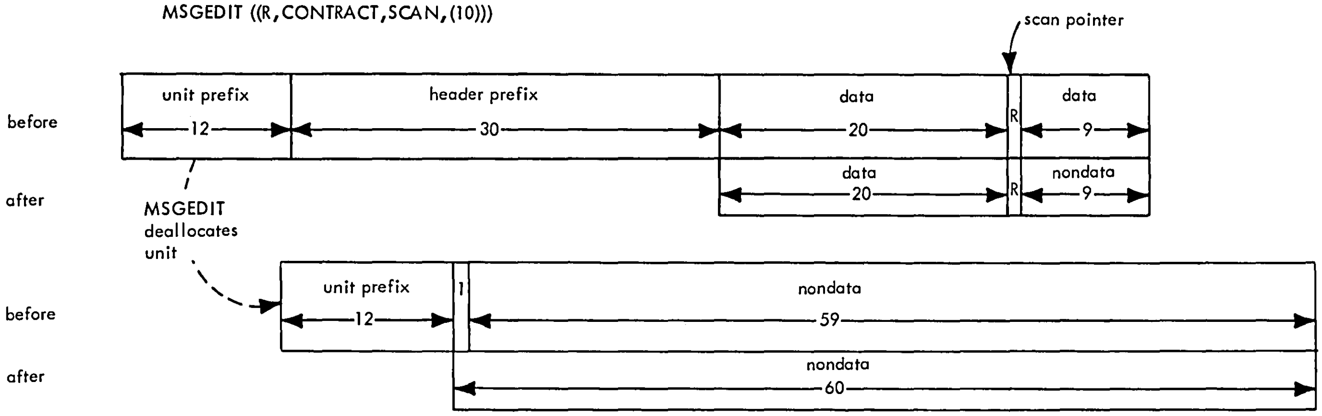
Example 3: insert after scan-pointer position

```
OUTHDR
MSGEDIT ((I,CL3'III,SCAN))
```



Example 4: remove characters between AT- and TO character strings

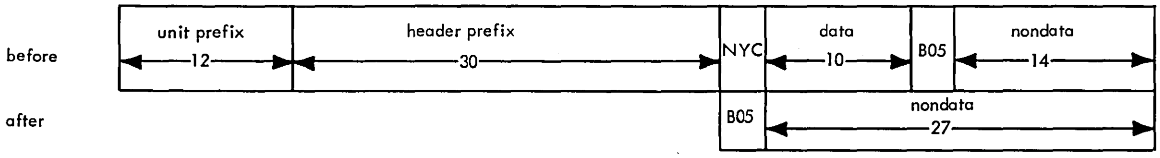
```
INHDR
MSGEDIT ((R,CONTRACT,SCAN,(10)))
```



MSGEDIT-examples

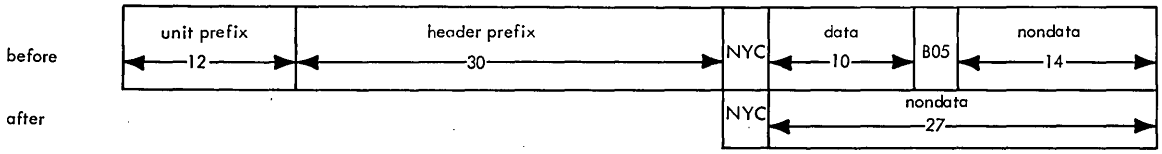
Example 5: Remove characters, including AT string

```
INHDR
MSGEDIT ((RA, CONTRACT, CL3'NYC',CL3'B05))
```



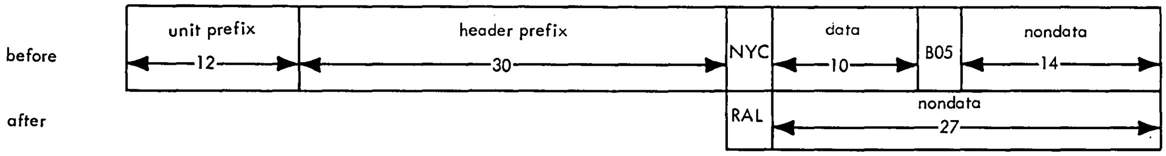
Example 6: Remove characters, including TO string

```
INHDR
MSGEDIT ((RT,CONTRACT,CL3'NYC',CL3'B05'))
```



Example 7: Replace character string including AT/TO strings

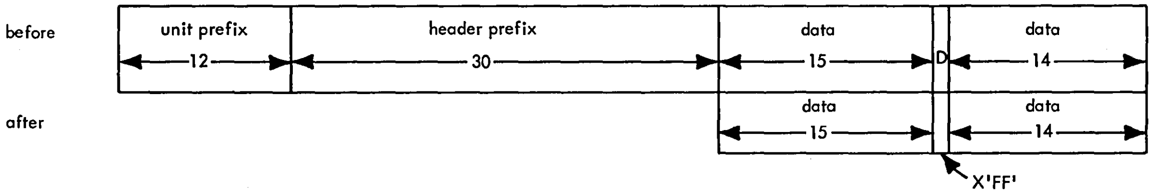
```
INHDR
MSGEDIT ((RAT,CL3'RAL',CL3'NYC',CL3'B05'))
```



MSGEDIT-examples

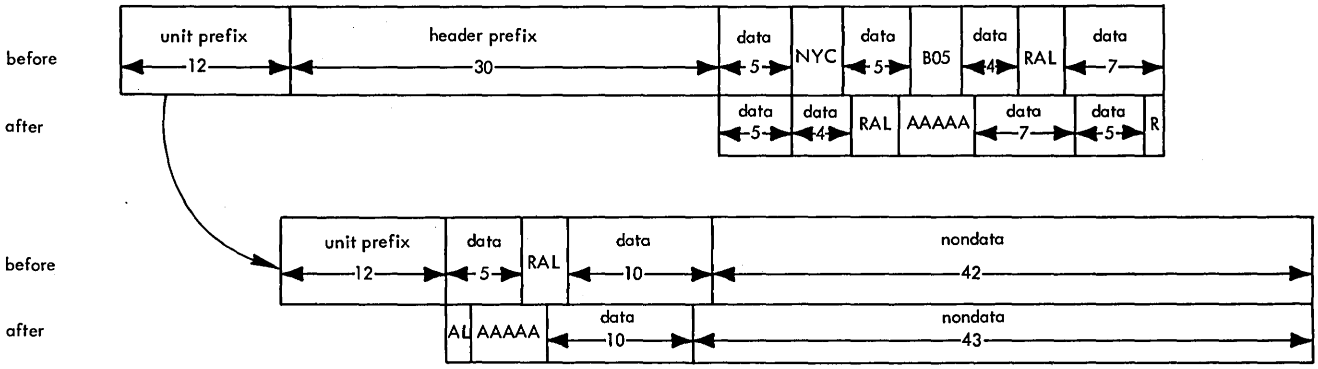
Example 8: insert record delimiter characters

```
OUTBUF
MSGEDIT ((RA,DELIMIT,CL1'D')) (PROCESS....,RECDEL=X'FF')
```



Example 9: multifunctions

```
INHDR
MSGEDIT ((I,(A,5),RAL),(RAT,CONTRACT,CL3'NYC',CL3'B05'))
```



For security in your system, use an ORIGIN macro as early as possible in your inheader subgroup to verify that only authorized users enter data in your system.

When you code a macro that has the operand BLANK= and specify BLANK=YES to say that blanks are to be ignored, be sure that the field you are trying to match does not have a blank defined in it. For instance, if you define a character string AB as CL3'AB', the field is automatically padded to the right with a blank, and a matching field can never be found if you code BLANK=YES.

Begin with a simple message handler, and add functions one step at a time.

Be sure to code macros in the right subgroup.

Typical Errors

Following is a list of common errors that can be made in coding message handlers. It is in the form of questions, with YES/NO answers, against which you can examine your code. Following this checklist is Figure 21, MH Functional Macros by Subgroup.

Question	*Right*	*Wrong*
1. If the MH is to handle incoming messages, did you code the INHDR, INEND, OUTEND delimiter macros?	YES	NO
2. If MH is to handle outgoing messages, did you code OUTEND?	YES	NO
3. Does the incoming group precede the outgoing group if you have included both in your MH?	YES	NO
4. If you coded an incoming group, did you include an INHDR macro first?	YES	NO
5. Are inmessage and outmessage the last subgroups in the incoming and outgoing groups, respectively?	YES	NO
6. If you provide a field or work area, such as for MSGGEN, ERRORMSG, or MSGEDIT, is the field addressable by the MH (is it after the OUTEND macro)?	YES	NO
7. If you have more than one MH and your code includes literals, did you code a LTORG instruction after the last delimiter (OUTEND) of each MH?	YES	NO
8. Is STARTMH the first instruction in every MH?	YES	NO
9. Did you try to execute more than one inmessage or outmessage subgroup for any message?	NO	YES
10. Is CANCELMG the first macro after INMSG in the inmessage subgroup if you use this macro?	YES	NO
11. Is the CANCELMG macro in the inmessage subgroup only?	YES	NO
12. Did you try to execute more than one CANCELMG macro for a message?	NO	YES
13. If you plan to process headers, is the CODE macro the first macro in the inheader subgroup?	YES	NO
14. Did you process a macro like DATETIME on an input segment before you issued the CODE macro or on an output segment after you issued CODE?	NO	YES
15. If you have CODE in the incoming group of the message handler for an application program, did you	YES	NO

specify the operand NONE so that there is no translation?		
16. Did you issue the CODE macro more than once in an incoming or outgoing group for a segment?	NO	YES
17. If you coded LC=IN in the STARTMH macro and plan to issue operator commands from remote terminals, did you move the scan pointer to the first date byte before issuing the CODE macro?	YES	NO
18. Did you issue the CUTOFF macro more than once in the inbuffer subgroup?	NO	YES
19. Did you provide line-control characters at the end of error messages you created with the ERRORMSG and MSGGEN macros?	YES	NO
20. Did you use the MSGEDIT macro to remove the EOT from messages destined for a 2741 terminal?	YES	NO
21. If you omitted the BLOCK= operand on the MSGFORM macro, did you code NTBLKSZ= or TBLKSZ= on the TERMINAL macro for the destination station?	YES	NO
22. Did you issue ORIGIN in the MH for a switched station to identify the station to TCAM?	YES	NO
23. Did you issue ORIGIN after the CODE macro?	YES	NO
24. Did you try to redirect to a distribution list?	NO	YES

Operating and Procedural Considerations

This section suggests operation and procedural (JCL) techniques for running a TCAM system.

If your MCP is large, you should, when assembling it, either specify a SPACE= parameter on your SYSPRINT DD statement, since the SYSGEN default for SYSOUT=A may not be large enough, or, preferably, directly allocate the output devices, as in

```
//SYSPRINT DD UNIT=00E
//SYSPUNCH DD UNIT=00D
```

Otherwise, you may lose your assembly due to lack of space (abnormal completion code of B37).

Obtain an object deck from the assembly run and link it into a JOB library. You can then bring up your TCAM system with a procedure that executes your MCP. This is more efficient than running a link-and-go procedure.

Dump SYS1.LOGREC every day as part of your regular cleanup procedure. Otherwise, if it gets full, it can impact your system so that you might not be able to execute your MCP. Also, if you let it fill up, dumping it is very time-consuming. If you do not want the output (because you have had no trouble with any of your lines or terminals), code DUMMY in the EREPPT DD statement of the IFCEREP0 utility program.

To save main storage after your application programs are fully tested, attach them (with ATTACH macros) in the MCP. Run them as separate jobs until tested, however, because if you do not, you will not get a dump of the programs.

SUBGROUP	MACROS	
INHEADER (INHDR)	CHECKPT CODE COUNTER DATETIME FORWARD INITIATE LOCK LOCOPT LOG	MSGEDIT MSGLIMIT MSGTYPE ORIGIN PATH PRIORITY SEQUENCE SETSCAN TERRSET UNLOCK
INBUFFER (INBUF)	CHECKPT CODE COUNTER CUTOFF	LOCOPT LOG MSGEDIT PATH TERRSET
INMESSAGE (INMSG)	CANCELMG CHECKPT ERRORMSG	HOLD LOG MSGGEN REDIRECT
OUTHEADER (OUTHDR)	CHECKPT CODE COUNTER DATETIME LOCOPT LOG MSGEDIT MSGFORM	MSGLIMIT MSGTYPE PATH SCREEN SEQUENCE SETEOF SETSCAN TERRSET
OUTBUFFER (OUTBUF)	CHECKPT CODE COUNTER LOCOPT	LOG MSGEDIT PATH TERRSET
OUTMESSAGE (OUTMSG)	CHECKPT ERRORMSG HOLD	LOG MSGGEN REDIRECT

Figure 21. MH Functional Macros by Subgroup

As a precautionary measure, scratch all "scratch" data sets before you bring up a system.

If you routinely send all output to SYSOUT=A, use a SPACE= parameter to be sure that the task providing a diagnostic aid or a dump does not run out of space.

To stop a line if you have not yet responded to the message SPECIFY TCAM PARAMETERS, use the OS command

```
V lineaddress,OFFLINE
```

You may list multiple line addresses by enclosing them in parentheses. After you have responded to the message, TCAM is in the system and you should use the TCAM operator command

```
V lineaddress,OFFTP,[C or I]
```

You must issue this command for each line you wish to stop.

If, when you open a line, you get the message IED079I ENDING STATUS NOT RECEIVED addr-LINE UNAVAILABLE, do not issue a STOPLINE operator command on the line until you have issued a STARTLINE command.

Be careful when closing TCAM. If you issue the command

```
Z TP,FLUSH
```

TCAM tries to send any outgoing messages that are queued for a station, unless the station is intercepted. If a terminal in your network is not intercepted, but is down because of hardware trouble, you will never close down because TCAM continually tries to flush the message queue of all messages. FLUSH is the default if you specify only

```
Z TP
```

Typical Errors

Question	*Right*	*Wrong*
1. Does the time period for CPU processing that you defined at system generation cover the entire time you plan to run continuously (coding TIME=1440 on the EXEC statement gives you unlimited time)?	YES	NO
2. Since the DEBUG, GOTRACE, and NOTRACE operator commands do not allow multiple operands, did you issue a command for each diagnostic aid you want loaded and for each line for which you want the line I/O interrupt trace either started or stopped?	YES	NO
3. For a line with hardware trouble, did you issue a STOPLINE operator command? (If operator control cannot successfully stop the line you lose operator control. If the command was entered from a remote terminal, you also lose its line, since it waits for the response to the command. Operator control is not reentrant, therefore, no other operator control commands will be accepted until this request is honored. Because of the hardware trouble, operator control cannot successfully stop the line.)	NO	YES

Terminal User Errors

When you are assessing the situation after you discover an error, consider the areas where a terminal user is involved as possible sources of error. Three of these areas are

1. bad input,
2. wasted processing time, and
3. impacting the TCAM system.

In a simple message-switching environment, the terminal user is not responsible for errors. You should detect any bad data entered by the terminal in your MCP and cancel the message. MCP validity checking and canceling prevents bad messages from cluttering up your system and impacting processing time. Invalid data in the text is evident at the receiving terminal; its invalidity does not impact the TCAM system. However, in a data-collection environment, bad text data entered by the terminal user can cause bad output to be produced, as in reports and monthly statements.

A careless terminal user can make TCAM perform unnecessary functions, thereby wasting processing time. Every message that a terminal enters, valid or invalid, passes through the message handler. To reduce processing time by reducing the number of invalid messages, the terminal user should know:

- the expected format of an input message;
- the correct spelling of all valid destination terminal names; and
- the type of translation table used by his terminal—is the table "folded" to allow both uppercase and lowercase letters to be accepted as valid?
- The terminal user should be cautioned about inserting an EOB (depressing EOB key) within the header since these EOBs will impact header processing. EOBs are not removed when LC=OUT is specified on the STARTMH macro.

Furthermore, a terminal user at a secondary operator control terminal may unknowingly create a great deal of trouble, since he can reconfigure parts of the TCAM network without anyone knowing. If your system suddenly fails to operate properly for no apparent reason, first look at the terminal sheets from all secondary terminals to see if someone issued an operator command that impacted system operation.

The major terminal user error in entering an operator command from a secondary terminal is failure to follow the command with a blank. Each command *must* be followed by one or more blanks before the EOT; otherwise, the command is invalid. A command from SYSCON may be no longer than 126 characters.

A terminal user should do certain things to keep the system clean and running. First, he should keep track of his input sequence numbers to avoid wasting processing time if they are invalid. The input sequence number is at most four bytes long and *must* be followed with a blank. Leading zeros can be omitted. Second, he should examine the output sequence numbers to be sure no number is skipped, indicating a lost message. If a number is skipped, he should realize that there may be trouble on his line or terminal. Third, he should end all his messages with a carrier return, so that the carrier of the receiving terminal is positioned at the beginning of the next line for the next message. Otherwise, the carrier of the receiving terminal may be at the end of the line when it starts receiving the next message, and the message contents are lost since the message is overtyped. Another technique is to have a carrier return as the first character in a message.

Typical Errors

Question	*Right*	*Wrong*
1. Is the input message format correct?	YES	NO
2. Did the operator enter the message in the wrong shift?	NO	YES
3. Did the operator follow an operator command with a blank?	YES	NO
4. Are EOBs embedded in the header data?	NO	YES

Other Possible Areas of Error

You should consider several possible sources of error that are outside TCAM. Following is a list of the more common sources.

1. Control program error. You could have an error in your operating system or in the interface between TCAM and the system.
2. Application program error. For example, an error in the COBOL compiler.
3. Central hardware error. A failure, for instance, in a 2314.
4. Remote hardware error. A failure, for instance, in a 1050 terminal in another city.
5. Communications line error.

Once you pinpoint the general area of error, you are well on your way to determining the actual problem and how it can be corrected.

TCAM Diagnostic Aids

This chapter tells you what information TCAM provides for your use in diagnosing problems, and how you can get copies of the information. The first section, *Gathering and Interpreting Data from Dumps,* covers the TCAM program and all the data sets that you can dump and print. This first section also suggests the kinds of errors that you can find, where to look for them, and, in some cases, what normal operations look like. The second section, *Using Operator Commands,* summarizes operator commands that you can issue to determine and alter the status of your TCAM system while it is running. The last section, *Normal End-of-Day Closedown,* suggests what data you might want to copy after your normal end-of-day closedown.

Gathering and Interpreting Data from Dumps

This section tells you how to obtain main-storage and secondary-storage dumps. Included with each category of dumps is a detailed explanation of how to interpret the information. The last part of this section refers to console and terminal listings that you may need when you debug.

Main-Storage Dumps

Several types of dumps are available to you. The first, and perhaps the most common, is the formatted dump provided by the system ABEND routines. All you need to get this dump is the proper JCL as discussed below.

However, you must be aware that eventually you will probably need a stand-alone dump, and you must plan for this need. For instance, when a program check occurs, you may lose data when the ABEND routines gain control of the system. Ways to obtain stand-alone dumps are also discussed.

TCAM Formatted Dump

The job you use to bring up TCAM should include a SYSUDUMP or SYSABEND DD statement in the JCL for the step that executes the message control program. Examples are

```
//SYSUDUMP DD SYSOUT=A and
//SYSABEND DD SYSOUT=A
```

This ensures that you have a data set for a dump if TCAM abnormally ends or if you cancel TCAM with a dump for debugging. The SYSUDUMP is of the program region only; the SYSABEND dump includes the OS nucleus.

A SYSABEND dump may be too large for the default space of SYSOUT=A, thus you may need a SPACE= parameter.

If you include either of the DD statements described, you can terminate TCAM normally (Z TP) and get no dump, or terminate it abnormally (C tcamjob,DUMP) and get a dump identical to the one the OS ABEND routine would produce.

Because TCAM occupies a large region and dumping main storage may take a long time, you may wish to put your dump on tape and print it at a later time or on another machine, so that TCAM can restart as soon as possible. See the publication, *Service Aids,* to learn how to transfer SYS1.DUMP, the ABEND data set, to tape.

Example:

```
//TRNSDUMP    JOB  MSGLEVEL=(1,1)
//* TO PUT SYS1.DUMP ON TO TAPE
//             EXEC PGM=IMDPRDMP
//SYSPRINT    DD   SYSOUT=A
//PRINTER     DD   SYSOUT=A
//TAPE        DD   DSNAME=SYS1.DUMP,DISP=OLD
//SYSUT2      DD   UNIT=2400,VOL=SER=DUMP,LABEL=(,NL),*
//                 DISP=NEW
//SYSIN       DD   *
        END
/*
```

A TCAM dump contains a great deal of information, only part of which you can use for any particular problem. The information you use varies according to the nature of the problem, and may be different for each problem. You must decide what to look for. Therefore, you should examine the coding of your message control program (MCP) and determine as much of the problem as possible *before* looking at the dump. You must know what events led up to the problem and the general nature of the problem; if you do not, you will be hopelessly lost in the dump.

This section shows some of the major items in a TCAM formatted dump, as well as possible starting points in TCAM debugging. For a complete description of the fields in a TCAM formatted dump, see *Appendix C. Formatted TCAM Dump.* For a description of the fields in an OS formatted dump, see the *Guide to Reading Dumps.*

Reading the Dump: A formatted TCAM dump is produced as part of the OS formatted dump when TCAM resides in the system. The TCAM part of an MFT dump starts after the trace-table entries; the TCAM part of an MVT dump starts after the save area trace. The following 14 parts of Figure 22 illustrate the items of a TCAM dump.

The first item printed in the TCAM dump is

```
TCAM ADDRESS VECTOR TABLE hhhhhh
```

where *hhhhhh* is the starting address, in hexadecimal format, of the TCAM address vector table (AVT). The AVT is the major control block of the TCAM system, and begins eight bytes beyond the start of the INTRO macro.

Next are five save areas, with their offsets from the start of the AVT printed in the left-most column of the dump. The save areas are shown in Figure 22 Part 1.

Table Pointers: The table pointers (Figure 22, Part 2), with their offsets from the start of the AVT printed in the left-most column of the dump, include:

1st word (at offset X'148') —last three bytes is a pointer to (address of) the device characteristics table

11th word (at offset X'170') —last three bytes is pointer to the TCAM MCPtask control block (TCB)

12th word (at offset X'174') —last three bytes is a pointer to the TCAM line I/O interrupt trace table if you included this table in your system.

```
TCAM ACDRESS VECTOR TABLE  039970

SAVE AREA 1 USER REGISTERS

0000      0C0C0C0C C00777B0 0C0399B8 5003A6C4    00068BC0 00000000 000399B8 40039EA4
C020      5C0198C8 C00158C8 C00157A0 0001962C    00015860 000198B0 C0019B88 000198D8
0040      0C0C0CCC 4007EC8A

SAVE AREA 2 DISPATCHER

0048                        BC04C38A 0000016C    B003F2CC 00068BC0 0004233A 000399B8
0060      80C68C8A 00042902 0C06D818 02C69280    C006B8B0 E406E660 1203B35C 00C00C04
0080      8CCC0CCC 0003A9B8 00068BC0 0003B370

SAVE AREA 3 1ST SUBROUTINE CALLED BY DISPATCHER

0090                                             4003DA88 00067BCA 8303AD58 FF047622
00A0      00C00000 40046620 0C07DA70 00039AC8    0006D818 03069280 0006DB40 E406E660
00C0      12039910 C0000004 0C039970 00000001    C000006A 00047288

SAVE AREA 4 2ND SUBROUTINE CALLED BY DISPATCHER

00D8                                                               00000001 00000000
00E0      0C03A9B8 600684CC 0C03E8B8 00000000    00039AC8 000692A0 C00692A0 E0069280
0100      0C000027 0206DBE0 1903D2EC 0003AEEC    0C000025 0103A9B8 00000054 00068428

DISABLED SAVE AREA USED BY SVC 102 AND APPENDAGES

0120      0000C17C 00002BDC 0C06A0F8 1B016B14    EF03A98C 00001000 00000018 000014A0
0140      0C0C0938 00000000
```

Figure 22. A Formatted ABEND Dump Printout (Part 1 of 14)

```
                                  DEVICE
                                  CHARACTERISTICS
TABLE POINTERS                    TABLE ADDRESS

0148                              0003E248 5000084C    80065BB4 0003B1FB 02A800A0 0006AB20
0160      D6D7C9C4 40404040 4C404040 40404040    00019D18 00069370
                                             ADDRESS OF ADDRESS OF
                                             MCP'S TCB  TCAM LINE I/O
                                                        INTERRUPT
                                                        TRACE TABLE
```

Figure 22. A Formatted ABEND Dump Printout (Part 2 of 14)

Dispatcher Ready Queues: Following the table pointers are the dispatcher ready queues. These are shown with their offsets in the left-most column of Figure 22, part 3. They include:

1st word (at offset X'178')	—pointer to the enabled ready queue (first element to be dispatched)
12th word (at offset X'1A4')	—pointer to the TCAM dispatcher subtask trace table if you included this table in your system.
13th word (at offset X'1A8')	—pointer to the terminal name table.

```
                                                                    ADDRESS OF
                                                                    ENABLED
DISPATCHER REACY QUEUES                                             READY QUEUE

0178                                                               00039C2C 00000000
0180      0CC6ACC8 00077348 00048A60 D2273054    C01CD227 D01C3054 2800282C 00039970
01A0      02039AC8 C006B8B0 0003CDC0 0003A698    00066CAA 00039D88 00039CC4 00039CC4
01C0      000777F8 140000C8 0C04839C
                           |        |
              ADDRESS OF            ADDRESS OF
              TCAM DISPATCHER       TERMNAME
              SUBTASK TRACE         TABLE
              TABLE
```

Figure 22. A Formatted ABEND Dump Printout (Part 3 of 14)

TCB Pointers: Next are the TCB pointers to four TCAM TCBs, with their offsets in the left-most column. See Figure 22, Part 4.

ECBs: Figure 22, Part 5 contains the addresses of some internal routines and event control blocks (ECBs); their offsets from the start of the AVT are printed in the left-most column of the dump.

The first four words are ECB pointers; the remainder are pointers to TCAM internal routines and subtasks. The tenth word (at offset X'200') is a pointer to the cross-reference table if you included this table in your system.

Special Elements: Figure 22, Part 6 contains, at an offset of X'2D0' (5th word of the last line), the address of the current buffer. At an offset of X'2D8' is the address of the VCON table. Offsets from the start of the AVT are printed in the left-most column.

QCB Pointers: A list of queue control block (QCB) pointers, with offsets in the left-most column is shown in Figure 22, Part 7. At an offset of X'384' is the address of the start of the buffer-unit pool. The third word on the last line (at offset X'388') contains the number of buffer units being used by main-storage queues.

Interface: Figure 22, Part 8 contains parameter lists, interface work areas, and constants; offsets are printed in the left-most column of the dump. The third word of the last line (at offset X'408') contains, in the first two bytes, the key length of the message queues, and, in the last two bytes, the number of lines opened. At an offset of X'410', the first two bytes are the number of free units in the buffer-unit pool.

```
                                                        OPERATOR
                                                        CONTROL    ON-LINE   FE COMMON
                                          CHECKPOINT    TCB        TEST TCB  WRITE TCB
TCB POINTERS                              TCB ADDRESS   ADDRESS    ADDRESS   ADDRESS

01CC                                       00016650     00017098  00019218  000190C8
```

Figure 22. A Formatted ABEND Dump Printout (Part 4 of 14)

```
          ADDRESS OF
          CROSS-REFERENCE
          TABLE
ECBS              |

01DC              |                                                                      00000C00
01E0      CC00000C| 00000000 00000000 80019B50     0003A7E8 0003CFCC 00042A0E 00000000
0200      00[077688] 00019828 000422B8 000422A8     0A041FFE 00041D08 00042612 00047C78

0220      00041C38 00068BC0 00068E3A 00067B5A     00063F32 C004D1E8 00077868 0003E91A
0240      00065490 00000000 0C000000
```

Figure 22. A Formatted ABEND Dump Printout (Part 5 of 14)

```
SPECIAL ELEMENTS

024C                                    00000000     00000000 00000000 00000000 00000C00
0260      000000C0 00000000 00000510 47F01266     00000000 00000000 00000000 00000C00
0280      0C068E60 000000C0 0C042048 20039CD0     F0000000 00039C0C 00039C94 70039C5C
02A0      EC039C2C 7C039C94 8C02012C 62550800     00000802 00041F84 08039C5C 00039AE0
02C0      0C039C2C 40039CA0 EC039C2C 0000FFFF     E4[06E66C] 00000000 80[03E280]

                                                  ADDRESS OF         ADDRESS OF
                                                  CURRENT            VCON TABLE
                                                  BUFFER
```

Figure 22. A Formatted ABEND Dump Printout (Part 6 of 14)

Core Queue: Figure 22, Part 9 shows this section, with offsets from the start of the AVT printed in the left-most column. See *Appendix C* for a description of each field.

Disk: This section contains the queue and control information for the disk message queues. Figure 22, Part 10 shows this information, with the offsets into the AVT printed in the left-most column.

At an offset of X'46C' is the address of the CPB free pool. The first word of the fourth line (at offset X'480') is a pointer to the reusable disk data extent block (DEB). The seventh word of the fourth line (at offset X'498') is a pointer to the nonreusable disk DEB.

Termname Table: TNT 03CDC0 is the address, in hexadecimal format, of the TCAM terminal name table, which contains the names and addresses of all the terminal-table entries. Figure 22, Part 11 illustrates this section. See *Appendix C* for a description of each entry.

```
QCB PCINTERS

02DC                                                                                        0C000C00
02E0      00068418 C0000000 0C0445F8 00039C5C     F0039C2C 00041D68 000074E6 E1720100
0300      0C039C5C C0039C5C 0C039C0C 0806DA00     00000000 00042338 02039C2C 00000000
0320      000424F6 C2039C2C 8C016C20 00039C94     02039C2C 80019658 00039CA0 02039C2C
0340      80019E30 00039CAC 02039C2C 00000000     00042900 02039CC4 C0000000 000434D8
0360      02039CC0 C0000000 00041F98 02039C2C     00000000 0003FFB0 02039CE8 00039C2C
0380      0CC40526 C006DA00 0C000007 00000000
                   ADDRESS OF   NUMBER OF BUFFER
                   START OF     UNITS USED BY
                   BUFFER-UNIT  MAIN-STORAGE
                   POOL         QUEUES
```

Figure 22. A Formatted ABEND Dump Printout (Part 7 of 14)

```
INTERFACE

0390                                                  0004233C 0004DB88 80019AF8 808C0C0C
03A0      0C04EF58 E0019AF8 FFC4D9E4 FF04886E         FF04E654 0000C000 00000000 00000C00
03C0      CC0000CC 000000C0 0C000000 00C0CC00         C00C000C 0000000C 00000000 00C00C0C
03E0      0CCCCCCC CC0000C0 0C00C000 00000000         00000000 0000013C 00039AC8 20C00C02
04C0      00030CC4 C0070010 CC54C0C6 00000000         C17D0006 C0000019 07FB291C 0B000C00

                SIZE OF        NUMBER OF              NUMBER OF FREE
                BUFFER UNIT    LINES OPENED           UNITS IN BUFFER-
                (KEYLEN=VALUE)                        UNIT POOL
```

Figure 22. A Formatted ABEND Dump Printout (Part 8 of 14)

```
CORE CUEUE

0420      0C03FE54 CC000064 0C0CCC8C 000000C8     040C00CC 0006A124
```

Figure 22. A Formatted ABEND Dump Printout (Part 9 of 14)

```
DISK

0438                                                                    00049038 00C4A00A
0440      0004A9A8 2800000C 4C06A5E0 28000000     0006A5E0 2800000C 00000000 28C00C00
0460      0006A5E0 28000000 0006A5E0 0006A5E0     0006A54C 00077478 00077510 0000013B
0480      00019710 C00000C1 0C000015 0000000A     00000348 00000015 000197A0 00000001
04A0      0C000C15 C000000A 00000348 00000015     00000338 0000001B 00000065 00039CA0
04C0      DC000C00 310C00FF 0C1E

           ADDRESS OF                ADDRESS OF                       ADDRESS OF
           REUSABLE                  THE CPB                          NONREUSABLE
           DISK DCB                  FREE POOL                        DISK DCB
```

Figure 22. A Formatted ABEND Dump Printout (Part 10 of 14)

Terminal Table: Following the terminal name table are the terminal-table entries. They are listed in alphabetical order with one entry for each terminal. The format of an entry is shown in Figure 22, Part 12.

```
NAME AA
TRM 03AFD4
```

where *AA* is the name of the terminal and 03AFD4 is the hexadecimal address of the terminal-table entry.

The last three bytes of the field STATE/DESTQ are a pointer to the QCB for this terminal. The field IN/OUTSEQ contains the next expected sequence numbers, input and output, for this terminal.

A sequence number of 0001 means that the first message is the next message expected.

TCAM Destination QCBs: Following this heading are entries for all destination QCBs for the terminal-table entries. There is one set of entries for each QCB. Depending on the type of queuing used, one QCB may service several terminals. See Figure 22, Part 13 for the format of the QCBs.

The last three bytes of the field RELLN/DCBAD contain a pointer to the data control block (DCB). The first byte is the relative line number for this QCB. The last two bytes of the field INTVL/MSGCT contains a count of the number of messages on this queue.

TCAM DCBs: This section includes the three different types of TCAM DCB—the line group DCBs with their related line control block (LCB), the checkpoint DCB, and the message queues DCBs. The DCBs are illustrated in Figure 22, Part 14.

```
ADDRESS OF
TERMNAME TABLE

TNT  03CDCC  CCCE  18018910       00031A10       1A001A10       1B004301       F04F8900
                   00084301       F0508900       00084301       F0511810       07FE
             SRCHX 0010     ENLEN 08       MIDEN 03CF67     LEN 0027
             CCCDE 18898990       00031A98       1A881A98       1B884389       704F8980
                   00084389       70508980       00084389       705107F6
```

Figure 22. A Formatted ABEND Dump Printout (Part 11 of 14)

```
                                 ADDRESS
                                 OF QCB               IN   OUT
NAME AA
TRM   03AFD4   STATE/DESTQ 18030550    IN/OUTSEQ 00010001    ALTD/DEVFL 0000640C    STAT 00000000    CHCIN 08

               CIAL CIGITS     03020006          NEXT EXPECTED
               ACCR CHAR       02276126          SEQUENCE
               TRANS BLOCK     0090              NUMBER
```

Figure 22. A Formatted ABEND Dump Printout (Part 12 of 14)

```
TCAM DESTINATICN QCB'S

QCB   03D00C   CSFLG/ELCHN 420C0000    PRI/LINK      00000000    STVTO/STCHN 1003D008    STPRI/SLINK 5003EBA8
               ECLDT/STAT  00000000    SCBOF/INSRC 0003D000    INTVL/MSGCT 000000C0    PRLVL/LKRRN 0003ADF0
               RELLN/DCBAD 0003A840    FLAG/QBACK    02000000
                                                                           MESSAGE
                            DCB                                            COUNT
                            ADDRESS                                        ON THIS
                                                                           QUEUE
```

Figure 22. A Formatted ABEND Dump Printout (Part 13 of 14)

DCB 03A9B8 (LINE GROUP) is the starting address, in hexadecimal format, of this line group DCB.

On the line D/S INTERFACE, the last three bytes of the first word contain a pointer to the message handler. The last three bytes of the third word are a pointer to the input/output block (IOB) base. The last three bytes of the fourth word are a pointer to the translate table. The first byte of the fifth word is the IOB index.

On the line FOUNDATION, the first two bytes of the first word are the offset of the DD entry from the beginning of the task I/O trace (TIOT) table. The last three bytes of the second word are a pointer to the DEB.

LCB 069280 is the starting address, in hexadecimal format, of the LCB.

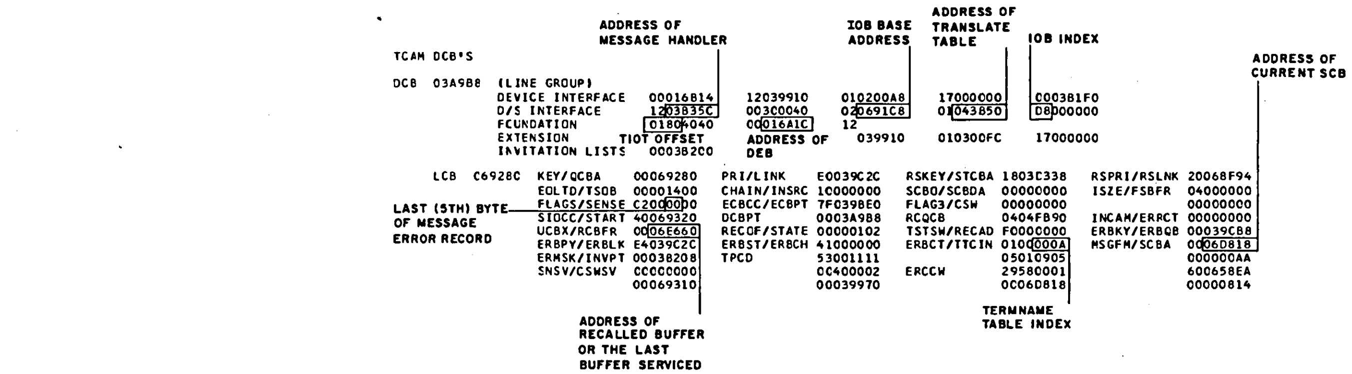
The third byte of the field FLAGS/SENSE is the last (fifth) byte of the message error record. The last three bytes of the field UCBX/RCBFR are a pointer to either the recalled buffer or the last buffer serviced by the buffer allocation (PCI) routines. The last two bytes of the field ERBCT/TTCIN are the index into the terminal name table of the terminal currently connected to this LCB. The last three bytes of the field MSGFM/SCBA are a pointer to the current station control block (SCB).

See *Appendix C* for a complete description of the three types of TCAM DCBs and the line group LCB.

Using the Dump: If you cannot find your problem by examining your code, one of the first things that you should do with the dump is identify the TCAM QCBs, DCBs, and LCBs by terminal. This can save you a lot of page turning.

First locate the terminal table in your formatted dump (see Figure 22, Part 12). The last three bytes of the first entry (STATE/DESTQ xxhhhhhh) are the address of the QCB for this terminal. Go through the terminal table; find and mark the QCB for each terminal with the name of the terminal.

Then go to the first QCB entry under TCAM destination QCBs. The last three bytes of the field RELLN/DCBAD are the address of the DCB. For each QCB, find the DCB pointed to by this address and mark the DCB with the name of the terminal. The field ERBCT/TTCIN in the LCB for the DCB may be useful if you have more than one terminal on a line. The last two bytes of the field are the offset into the terminal name table of the terminal currently connected on this



```
TCAM DCB'S

DCB  03A9B8  (LINE GROUP)
             DEVICE INTERFACE   00016B14    12039910    010200A8    17000000    0003B1F0
             D/S INTERFACE      1203B35C    003C0040    020691C8    01043B50    D8000000
             FCUNDATION         01804040    00016A1C    12
             EXTENSION                                   039910     010300FC    17000000
             INVITATION LISTS   0003B2C0

     LCB  C6928C  KEY/QCBA     00069280   PRI/LINK     E0039C2C   RSKEY/STCBA  1803C338   RSPRI/RSLNK  20068F94
                  EOLTD/TSOB   00001400   CHAIN/INSRC  10000000   SCBO/SCBDA   00000000   ISZE/FSBFR   04000000
                  FLAGS/SENSE  C2000000   ECBCC/ECBPT  7F039BE0   FLAG3/CSW    00000000                00000000
                  SIOCC/START  40069320   DCBPT        0003A9B8   RCQCB        0404FB90   INCAM/ERRCT  00000000
                  UCBX/RCBFR   0006E660   RECOF/STATE  00000102   TSTSW/RECAD  F0000000   ERBKY/ERBQB  00039CB8
                  ERBPY/ERBLK  E4039C2C   ERBST/ERBCH  41000000   ERBCT/TTCIN  0100000A   MSGFM/SCBA   0006D818
                  ERMSK/INVPT  00038208   TPCD         53001111                05010905                000000AA
                  SNSV/CSWSV   CCC00000                00400002   ERCCW        29580001                600658EA
                               00069310                00039970                0C06D818                00000814
```

Figure 22. A Formatted ABEND Dump Printout (Part 14 of 14)

line. The field may be zero, indicating a dial line with no terminal connected at this time. Otherwise, the field gives the offset into the terminal name table of the single terminal connected to this line.

At this time, you can find the associated DEB and DD entry for each DCB, in case you should need it for further debugging.

In the DCB, the second word of the line FOUNDATION is a pointer to the DEB. The first two bytes of the first word of this line are the TIOT offset.

To find the DD entry from the TIOT offset in the DCB, use the following steps (see Figure 23):

1. Convert the TIOT offset from hexadecimal to decimal.
2. Subtract the total length of jobname+stepname+procname. This length is always 24 bytes.

   404
    24
   380

3. From the formatted portion of the OS dump, under the TIOT, the line labeled DD is a DD entry in the TIOT. The first byte on this line is the length of the DD entry. Divide the total from step 2 by this length.

   X'14' = Decimal 20

   380/20=19

   Therefore, the DCB is associated with the 19th DD entry (starting with 0).

Now that you have found and identified all your QCBs, DCBs, LCBs, DEBs, and DD entries, you can begin to look for your problem.

Find the current buffer. AVT+X'2D0' is a pointer to the current buffer.

The current buffer, at an offset of X'0C', gives the address of the LCB for this message, at an offset of X'10', the terminal-name table offset of the source terminal, and, at an offset of X'28' (if this is the first unit), the terminal-name table offset of the destination terminal. You now know which message was being processed when the problem occurred. You also know where the LCB is, which terminal was sending, and which terminal was receiving. The source and destination offset fields may not be filled in. These fields are updated upon execution of certain message handler macros. However, the LCB is correct and always present in the prefix.

Find the LCB pointed to by the buffer prefix, and, in the LCB, find the SCB address at an offset of X'5C'. The SCB address is the last three bytes of the entry MSGFM/SCBA.

```
DCB   03A908   (LINE GROUP)
               DEVICE INTERFACE   000177E4    12039910    010200A8    17000000    C003B1AC
               D/S INTERFACE      1203B35C    00300040    0206D860    01043740    E0000000
               FCLNCATION         01944040    0C016F94    12
               EXTENSION                                    039910    010200A8    17000000
               INVITATION LISTS   0003B1C0

                                 TIOT OFFSET
                                 X'0194'=404 DECIMAL
```

Figure 23. Finding a DD Entry from the DCB (Part 1 of 2)

Locate the SCB in the unformatted section of the dump. The fifth word (at offset X'10') in the SCB is the first four bytes of the message error record. The last byte of the message error record is in the LCB at the third byte of the entry FLAGS/SENSE. Examine the message error record thoroughly, as it contains all the status information about the current (or last) message.

The last message, and thus the message error record, for each line can be examined in the same manner as the current message. The address of the last buffer processed by each line is in the LCB at the entry UCBX/RCBFR.

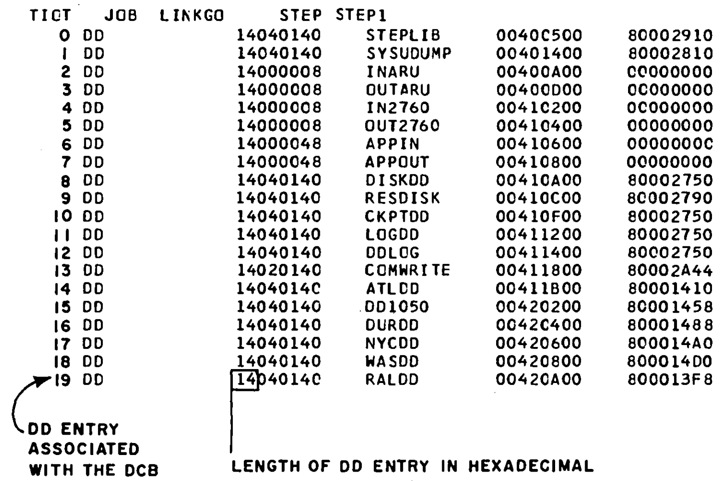
If your system fails when transmitting a message, find the message-handler macro routine that is the next, current, or last macro executed on the current buffer. This helps you determine what the system was doing when it failed.

First, find the current buffer at AVT+X'2D0'. Add X'0C' to the address found at this location to obtain the LCB address. Add X'5C' to the LCB address to obtain the SCB address.

The second word of the SCB points to the parameter list of the message handler macro that is the next, current, or last macro executed on the buffer. The macro is usually an inmessage or outmessage macro. However, if you have a multiple-buffer header, the parameter list is for an inheader or outheader macro. Go to the parameter list. The first byte is an index into the VCON table for the message handler macro routine involved. A pointer to the VCON table is found at AVT+X'2D8'.

If the first two bytes of the parameter list contain X'0100', inmessage or outmessage processing is either complete or has yet to begin, or you had a single-buffer header when performing inheader or outheader processing.

From your code, from the information gathered at the system console, from terminal listings, from the operator's description of what happened, from the message error record, from the buffer prefix, and from the various trace tables if



```
TIOT    JOB  LINKGO       STEP  STEP1
   0 DD           14040140     STEPLIB      0040C500     80002910
   1 DD           14040140     SYSUDUMP     00401400     80002810
   2 DD           14000008     INARU        00400A00     C0000000
   3 DD           14000008     OUTARU       00400D00     0C000000
   4 DD           14000008     IN2760       0041C200     0C000000
   5 DD           14000008     OUT2760      00410400     00000000
   6 DD           14000048     APPIN        00410600     0000000C
   7 DD           14000048     APPOUT       00410800     00000000
   8 DD           14040140     DISKDD       00410A00     80002750
   9 DD           14040140     RESDISK      00410C00     8C002790
  10 DD           14040140     CKPTDD       00410F00     80002750
  11 DD           14040140     LOGDD        00411200     80002750
  12 DD           14040140     DDLOG        00411400     80002750
  13 DD           14020140     COMWRITE     00411800     80002A44
  14 DD           1404014C     ATLDD        00411B00     80001410
  15 DD           14040140     DD1050       00420200     8C001458
  16 DD           14040140     DURDD        0042C400     80001488
  17 DD           14040140     NYCDD        00420600     800014A0
  18 DD           14040140     WASDD        00420800     800014D0
  19 DD           1404014C     RALDD        00420A00     800013F8
```

Figure 23. Finding a DD Entry from the DCB (Part 2 of 2)

you specified them, you must decide what the problem is–or what to look for next. From this point on, only experience can tell you what to do.

Stand-Alone Dump

To prevent losing data in the OS ABEND routine, and to see the complete status of the system at the time of a program check, you may set the wait bit on in the program new PSW and recreate the condition that caused the program check. When the system enters a wait state, the program check has occurred, and a stand-alone dump revealing the entire system status at the moment of the failure can be obtained. For non-program check problems (system ABEND), you can obtain a stand-alone dump revealing the entire system status at the moment of failure by placing an address compare stop at the entry to ABTERM.

Use the OS service aid programs to create a stand-alone dump. You should assemble and link-edit the dump (IMDSADMP) service aid before problems are encountered. The time you spend to assemble one of these programs and to add it to your system is time well spent. OS service aid dumps are:

1. Low-speed dump. This dumps main storage directly to the printer. It is an unformatted dump, printed in hexadecimal, 4 bytes to the word, 8 words to the line. The main-storage address is printed at the left-most column of the dump, and the character equivalent of the 8 words at the right-most column. The first two lines on the first page of the dump are the general-purpose register contents. Figure 24 illustrates the start of a printout from a low-speed stand-alone dump.
2. High-speed dump. This dumps main storage to a magnetic tape that can be printed and formatted at a later time using another service aid program. This printout can be made on the same or some other system.

Several options are provided by the service aid programs. These are discussed in *IBM System/360 Operating System: Service Aids,* GC28-6719, in the section *IMDPRDMP.*

You can use the following JCL to assemble the low-speed dump.

```
//SLOWDUMP     JOB    MSGLEVEL=(1,1)
//STEP1        EXEC   ASMFC
//ASM.SYSLIB   DD     DSN=SYS1.MACLIB,DISP=SHR
//ASM.SYSIN    DD     *
   IMDSADMP    IPL=191,CPU=360,PROTECT=NO,TYPE=LO,OUTPUT=P00E
               END
/*
```

```
R  0-7    00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *................................*
R  8-15   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *................................*

000000    00000191 00001C00 00000000 60000028   08000080 40000001 FF060080 80000000  *.................... ...........*
000020    FFF50001 50072D1A 00000000 00000000   0000FF00 00000000 FF060018 80000000  *.5..............................*
000040    F0064348 0D400001 F0064330 00007708   00F32800 0000B0DC 00040000 000091E8  *0.... ..0........3.............Y*
000060    00040000 000097D8 00040000 000092DE   00000000 00012CC0 00040000 0000926A  *.......Q........................*
000080    310000A6 40000005 08000080 40000001   05001C00 40000440 06001000 00000440  *.... ....... ....... .. ........*
0000A0    00000000 00000000 00000400 00000000   00000000 00000000 00000000 00000000  *................................*
0000C0    00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *................................*

000160    00000000 00000000 00000000 82000170   00040000 000389FC 00000000 00000000  *.......................0........*
000180    FF060018 80000000 0000018A 018A018A   FF000190 00000190 00000001 00034B64  *................................*
0001A0    00067000 00034C34 00034C34 00066880   0006A260 00034AE8 00000000 12034938  *.......................Y........*
0001C0    00000000 00072B88 0003CFAA 000349C0   80072D0C 0003D572 00000000 00000000  *......................N.........*
0001E0    00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000  *................................*
```

Figure 24. Start of a Printout from a Low-Speed Stand-Alone Dump

You can use the following JCL to assemble the high-speed dump.

```
//FASTDUMP     JOB  MSGLEVEL=(1,1)
//STEP1        EXEC ASMFC
//ASM.SYSLIB   DD   DSN=SYS1.MACLIB,DISP=SHR
//ASM.SYSIN    DD   *
      IMDSADMP      IPL=190,CPU=360,PROTECT=NO,TYPE=HI,
                    OUTPUT=T283
               END
/*
```

The following JCL prints the dump tape.

```
//DUMPPRT      JOB  MSGLEVEL=(1,1)
//             EXEC PGM=IMDPRDMP
//SYSPRINT     DD   SYSOUT=A
//PRINTER      DD   SYSOUT=A,SPACE=(121,(8000,100))
//TAPE         DD   UNIT=2400,VOL=SER=DPTAPE,LABEL=(,NL),*
//                  DISP=OLD
//SYSUT1       DD   UNIT=SYSDA,DISP=(NEW,DELETE),*
//                  SPACE=(2052,(257,10))
//SYSIN        DD   *
      GO
      END
/*
```

Note: *The SPACE= parameter shown in the sample JCL may not reserve enough space if you select all the options.*

Once you know how to locate the address vector table (AVT) and a line control block (LCB) in this type of dump, using a stand-alone dump is similar to using a formatted dump. This section shows how to locate these and other important control blocks, and gives some examples using a stand-alone dump.

Finding the AVT: To find the AVT in a stand-alone dump, get the address of the communications vector table (CVT) from absolute location X'10'. X'10' may not contain this address, because the stand-alone dump program may destroy it. If this is the case, find the CVT address at absolute location X'4C'.

Add X'F0' to the address of the CVT. The address at this location is a pointer to the address of the AVT, which starts eight bytes beyond the beginning of the INTRO macro.

Example:

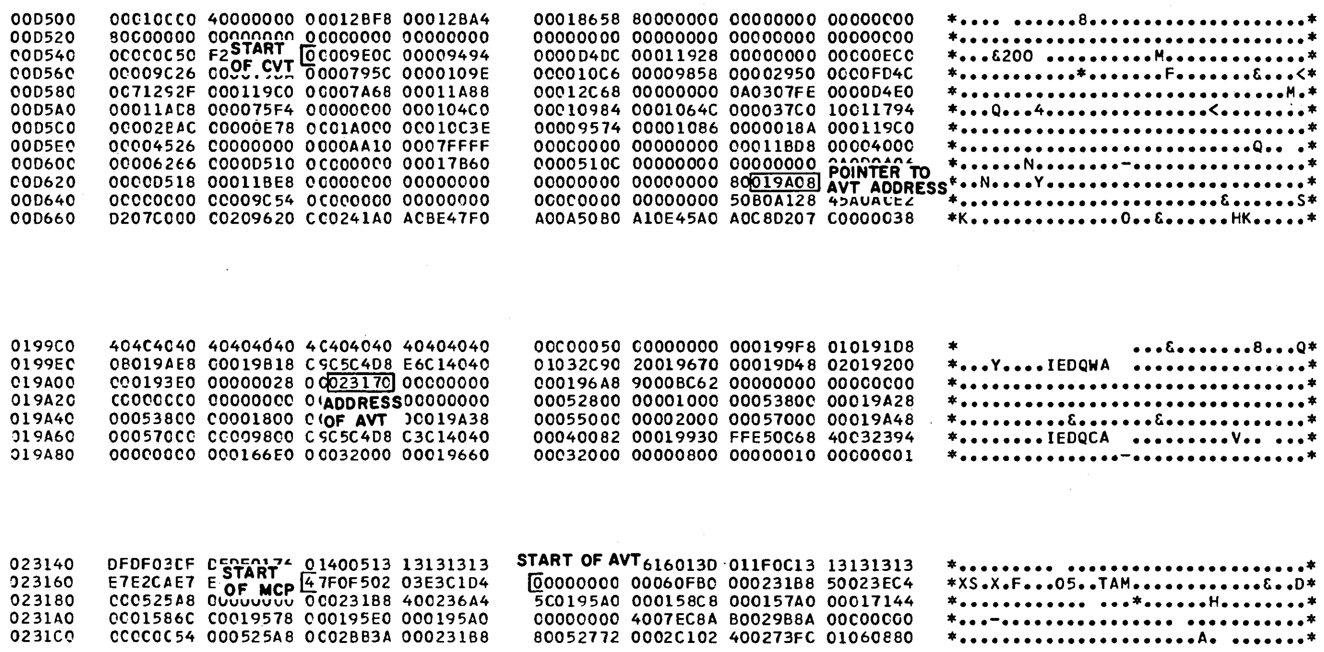
The contents of the word at location X'4C' is X'D548'.

D548	CVT address
F0	Offset into the CVT
D638	Pointer to the AVT address

The last three bytes at location X'D638' is 019A08, which is the pointer to the AVT address. The last three bytes at location 019A08 is 023170, the address of the AVT.

Finding the Current Buffer: AVT+X'2D0' contains a fullword pointer to the current buffer.

Finding the Line I/O Interrupt Trace Table: AVT+X'174' contains a fullword pointer to the control block for the line I/O interrupt trace table. The control block contains:



```
00D500   00C10CC0 40000000 00012BF8 00012BA4   00018658 80000000 00000000 00000C00   *.... ......8......................*
00D520   80C00000 00000000 00000000 00000000   00000000 00000000 00000000 00000C00   *..................................*
00D540   0CC0C050 F2 START  0C009E0C 00009494   0000D4DC 00011928 00000000 00C00ECC   *...&200 ..........M...............*
00D560   0C009C26 00 OF CVT 0000795C 0000109E   000010C6 00009858 00002950 0CC0FD4C   *............*.......F........&...<*
00D580   0C71292F 000119C0 0C007A68 00011A88   00012C68 00000000 0A0307FE 0000D4E0   *................................M.*
00D5A0   00011AC8 000075F4 00000C00 000104C0   00C10984 0001064C 000037C0 10C11794   *...Q...4.................<........*
00D5C0   0C002EAC C0000E78 0C01A000 0C010C3E   00009574 00001086 0000018A 000119C0   *..................................*
00D5E0   0C004526 C0000000 0000AA10 0007FFFF   000C0000 00000000 00011BD8 00004000   *..............................Q.. .*
00D60C   00006266 C000D510 0CC000C0 00017B60   0000510C 00000000 00000000 POINTER TO *......N.........-.................*
C0D620   0000D518 00011BE8 0C0000C0 00000000   00000000 00000000 80019A08 AVT ADDRESS*..N....Y......................*
00D640   0CC0C000 CC009C54 0C000000 00000000   0C0C0000 00000000 50B0A128 45AUAUCE2  *.........................&......S*
00D660   D207C000 C0209620 CC0241A0 ACBE47F0   A00A5080 A10E45A0 A0C8D207 C0000038   *K...............O..&......HK.....*


0199C0   404C4040 40404040 4C404040 40404040   00C00050 C0000000 000199F8 010191D8   *                 ...&.......8...Q*
0199EC   0B019AE8 C0019B18 C9C5C4D8 E6C14040   01032C90 20019670 00019D48 02019200   *...Y....IEDQWA  .................*
C19A00   CC0193E0 00000028 0C023170 00000000   000196A8 9000BC62 00000000 00000000   *..................................*
019A20   CC000CC0 00000000 0(ADDRESS00000000   00052800 00001000 0005380C 00019A28   *..................................*
019A40   000538C0 C0001800 C(OF AVT )C019A38   00055000 00002000 00057000 00019A48   *..........&.......&...............*
019A60   000570CC CC009800 C9C5C4D8 C3C14040   00040082 00019930 FFE50C68 40C32394   *.........IEDQCA  ..........V.. ...*
019A80   00000000 000166E0 0C032000 00019660   00C32000 00000800 00000010 00000001   *...............-..................*


023140   DFDF03CF C START   01400513 13131313  START OF AVT616013D 011F0C13 13131313   *.......... .......................*
023160   E7E2CAE7 E OF MCP  47F0F502 03E3C1D4   00000000 00060FB0 000231B8 50023EC4   *XS.X.F...05..TAM............&..D*
023180   CC0525A8 00000000 0C0231B8 400236A4   5C0195A0 000158C8 000157A0 00017144   *............ ...*......H.........*
0231A0   0C01586C C0019578 C00195E0 000195A0   00000000 4007EC8A B0029B8A 00C000C0   *...-................ ...........*
0231C0   CCCC0C54 000525A8 0C02BB3A 000231B8   80052772 0002C102 400273FC 01060880   *.........................A. .......*
```

Figure 25. Finding the AVT in a Stand-Alone Dump

offset +0	address of the current entry
offset +4	address of the first entry
offset +8	address of the last entry
offset +12	address of the middle entry

Finding the Subtask Trace Table: AVT+X'1A4' contains a fullword pointer to the control block for the subtask trace table. The control block contains:

offset +0	address of the next entry
offset +4	address of the first entry
offset +8	address of the last entry
offset +12	size of the table

Finding the Cross-Reference Table: AVT+X'200' contains a fullword pointer to the control block for the cross-reference table. The control block contains:

offset +0	address of the first available entry
offset +4	address of the last entry

Finding the QCB for a Terminal: AVT+X'1A8' is a pointer to the terminal name table. At the terminal name table + X'52' the entries for each terminal start. Each entry includes the name of the terminal (a maximum of eight bytes) followed by the address of the terminal entry (always three bytes). The first word of each terminal entry contains a pointer to its QCB.

Finding the DCB: QCB+X'20' contains a three-byte pointer to the DCB. The first byte is the relative line number.

Finding the LCB: There are four different ways to find an LCB. You can find the LCB for an open line in the third word of the cross-reference table entry. Each of the other three ways is discussed below.

From the DCB: If the DCB is opened, DCB+X‘1C’ contains the three-byte IOB base address. Find the one-byte IOB index (DCB + X‘24’), multiply this value by the relative line number of the line you are interested in (QCB+X‘20’), and add the result to the IOB base address, then subtract X‘20’.

The following example uses a fictitious base and index just for the purpose of illustration.

IOB base		45848	
IOB index	+	D0	RLN=1
X‘20’	-	20	

LCB address at 458F8

LCB address= [IOB base + (RLN * IOB index)] – 20

From the Buffer: The buffer prefix contains the three-byte LCB address at the offset X‘0D’.

From the Terminal Entry: First, locate the terminal name table. Its address is in the AVT at an offset of X‘1A8’. The actual table entries begin at X‘52’ into the table. See Figure 26.

The format of a terminal-name table entry is

NAME FIELD	TERMINAL-TABLE ENTRY ADDRESS

where the name field is a maximum of eight bytes (the length is the value specified in the MAXLEN= operand of the TTABLE macro), and the terminal-table entry address is a three-byte field.

Decide which terminal you are interested in, and locate its terminal-name table entry and the address of the terminal entry.

The last three bytes of the first word in the terminal entry are the address of the QCB. At X‘20’ beyond the start of the QCB is the relative line number of the line (first byte) and the address of the DCB (last 3 bytes).

Add X‘1C’ to the address of the DCB (address present only if DCB is open). The word at this location is the base address of the IOB. Add the base address of the IOB to the product of the contents of the byte at DCB+X‘24’ and the relative line number. Subtract X‘20’ from this sum. The result is the LCB address for the terminal you are interested in.

Example:

023170	starting address of the AVT
1A8	offset to address of terminal name table
023318	address of the terminal name table

The last three bytes of the word at location 023318 contain the address of the terminal name table. Go to the terminal name table at location 0265C0.

```
023260   000232C8 00057148 00057148 E0057128   00000027 020573E0 19026AEC 000246EC   *...H............................*
023280   00000C25 C10241B8 0C000054 00051C28   00000191 00002750 00000938 40C0084C   *.......................&.... ..<*
0232AC   EF02418C 00001000 00000018 0C0014A0   0C000938 0000000C 00027A48 50C0084C   *...........................&..<*
0232C0   8C04F3B4 C2000037 02000000 00057F20   D6D7C9C4 4040404C 40404040 40404C40   *..3.............OPID           *
0232E0   CC0196A8 00052870 0C02342C 0000C000   000234E8 00000000 00032260 D2273C54   *.....................Y.......-K...*
02330C   DC1CD227 D01C3054 0C0150C4 00023170   20026579 000550B0 000265C0 C0023E98   *..K.......&D...........&..........*
023320   CC0504AA C0023588 C00234C4 000234C4   00060FF8 140000C8 ADDRESS OF0C00000   *...........D...D...8...H.........*
023340   000166E0 00016A78 C0000000 40000000   40000000 0000000C TERMNAME 0019B48   *........... ... ..............*
023360   00023FE8 000267C0 0002C20E 00000C00   00060EB8 00019188 TABLE     002BAA8   *...Y......B.....................*
```

Figure 26. Finding the LCB in a Stand-Alone Dump (Part 1 of 6)

0265C0	address of the terminal name table
52	length of the control area
26612	beginning of terminal-name table entries

Assume you want to find the LCB for a terminal named NYC. You can find its entry in either of two ways. The simplest is to glance down the converted portion of the dump printed in the right-most column and find the characters NYC.

The other method uses the control area of the terminal name table. At an offset of X'28' into the table, a one-byte field gives the length of the name field. Add three to this length to find the length of an entry. Multiply this sum by the number of entries that alphabetically precede the terminal you are interested in. Be sure to count the names of the TPROCESS, TLIST, and LOGTYPE macros as entries.

0265C0	terminal-name table address
28	offset to the length of the name field
0265E8	address of the byte containing the length of the name
8 + 3 = 11	length of a terminal-name table entry.

Alphabetically, NYC is the 25th entry. Therefore,

11 x 24 = 264 = X'108'

This is the offset to NYC from the start of the table entries.

026612	start of terminal-name table entries
108	offset to NYC entry
02671A	address of the terminal-name table entry for NYC

From this entry, you can see that the terminal entry for NYC starts at location 0246D4.

```
026560   60029224 C00264F0 0C020000 00C00000   00000000 0000002F 20E2C5D8 4B40D5E4   *-......0.................SEQ. NU*
026580   D44B40C8 C9C7C86B 40E2C8D6 E4D3C440   C2C540F0 F0F0F24B 372040E2 C5D84B40   *M. HIGH, SHOULD BE 0002... SEQ. *
0265AC START OF TABLED3D(LENGTH OF NAME5E4D3C4   40C2C540 F0F0F0F2 4B370000 000265C0   *NUM. LOW, SHOULD BE 0002........*
0265C0   18C18910 00031AFIELD IN BYTES 3004301   F04F8900 00084301 F0508900 00C84301   *................0|......0&......*
0265E0   FC511810 C7FE0010 08026767 00271889   899C0003 1A981A88 1A981B88 43897C4F   *0..............................|*
026600   898000C8 43897050 89800008 43897051   07F6C1C1 40404040 40400247 D4C1C1C1   *........&..........6AA      ..MAAA*
026620   40404C40 4002471C C1D3C140 40404040   0.START OF 3D34040 40404002 46C0C2C2   *       ...ALA      ...ATL      ...BB*
026640   40404040 40400247 F4C2C2C2 40404040   4 ENTRIES  2D6D540 40404040 02478CC2   *        ..4BBB      .. BON      ...B*
026660   D6E24040 40404002 4824C3C3 C3404040   40400247 64C3C8C1 D9404040 40024704   *OS      ...CCC      ...CHAR      ...*
026680   C4E4D940 40404040 0246ACC5 C5C54040   40404C02 4778C5E5 C5D9E840 40400249   *DUR      ...EEE      ...EVERY      ..*
0266A0   78C7C1D9 E4D84040 4C024640 C7C5E3D8   40404040 0245F0C7 C5E3F2F7 F6F04C02   *.GARUQ    .. GETQ      ..OGET2760 .*
0266C0   4618C8C8 C8404040 40400247 A4C8E4E8   C3D2404C 40024668 TERMNAME TABLE ,0   *..HHH      ...HUYCK      ...III      *
0266EC   0247BCC3 C9D5C5C1 F1404002 48ECD3C9   D5C5C1F2 40400249 ENTRY FOR 'NYC' ,0   *...LINEA1  ...LINEA2  ...LOCAL1 *
026700   40024854 D3D6C7C5 D5E3D9E8 02493CD4   C1D9E840 40404002 46TCD5E8 C3404040   * ...LOGENTRY...MARY      ...NYC   *
026720   40400246 C4D7C1D9 E4D84040 40024654   D7E4E3D8 40404040 02460 NAME FIELD 7   *  ..MPARUQ    ...PUTQ      ...PUT27*
026740   TERMINAL-TABLE C1 D3404040 40400246   9CD9C5D4 D6E3C5F1 4002486C D9C5D4D6   *60 ...RAL      ...REMOTE1 ..%REMO*
026760   ENTRY ADDRESS :D9 C5D4D6E3 C5F34002   48ACD9C5 D4D6E3C5 F4400248 CCE3C1F1   *TE2 ...REMOTE3 ...REMOTE4 ...TA1*
02678C   40404040 400249C0 E3C1F240 40404040   024928E3 C5E74040 40404002 483CE6C1   *       ...TA2      ...TEX      ...WA*
0267A0   E2404C40 40400246 ECE6E3E3 C1404040   4002495C E6E3E3C2 40404040 02496400   *S       ...WTTA      ..&WTTB      ....*
0267C0   000267CA 000267E0 F0F0F1F0 F0F1F1F1   F0F1F0F1 F0F1F0F1 F0F10003 00C000C0   *.........00100111010101010l......*
0267E0   0C00D6D7 C6C9C5D3 C4400000 E2C3D9C5   C5D5E2E6 0000C3D6 D5E5E2E6 4040FF00   *..OPFIELD ..SCREENSW..CONVSW   ..*
026800   42000C00 C0000000 CE0283A8 500283A8   0000000C 00026800 00000000 000245F0   *............&...................0*
```

Figure 26. Finding the LCB in a Stand-Alone Dump (Part 2 of 6)

Go to the terminal-table entry for NYC at location 0246D4. The last three bytes of the first word are the address of the QCB.

Go to the QCB at location 26AA8. X‘20’ into the QCB is the address of the DCB.

26AA8	address of the QCB
20	offset into QCB of pointer to DCB
26AC8	pointer to address of the DCB.

At this location, the first byte is the relative line number for the line.

Go to the DCB at location 02418C. The last three bytes are the address of the DCB. The last three bytes of the ninth word in the DCB contain the base address of the IOB.

2418C	address of the DCB
1C	offset into DCB of IOB base address
241A8	pointer to the IOB base address

Multiply the value found in the byte located at an offset of X‘24’ into the DCB by the relative line number. Add this result to the IOB base address.

2418C	DCB starting address
24	offset to bytes containing IOB size
241B0	address of the byte containing IOB size

C8 X 1 = C8

607D8	base address of IOB
+C8	size of the IOB
608A0	address of the IOB for this LCB

Subtract X‘20’ from this address. This is the LCB address.

608A0	IOB address
-20	
60880	LCB address for terminal NYC

```
024680     000100C4 C00020C0 0C060000 01026413 START OF       00010002 00006000 00030C00    *........................-.....*
0246A0     01040508 C8000262 132A0630 18026A20 TERMINAL       00000000 00080000 023040B8    *............................ .*
0246C0     18026A64 CC010004 0C000000 00080000 TABLE ENTRY [19|026AA8| 00020004 00C02000    *..............................*
0246E0     C0180C0C 01033764 01044780 19026AEC FOR 'NYC'       0 QCB  ADDRESS 10000 03C40500    *....................... ........*
024700     03C04720 19026B30 0C010004 00004000         00080000 04040500 0000D202 1B026B74    *.......... ..........K....*
024720     00010001 0000A200 00000000 0103C008         00010202 00780337 E2010200 05B08B10    *.......................S.......*
```

Figure 26. Finding the LCB in a Stand-Alone Dump (Part 3 of 6)

```
026A80     0C0CC000 (START OF C8C00000 0000000C     0C00000C 00000000 0000000C 0000057C    *.......-......................*
026AA0     8C000C00 (QCB     |62000000 00000000     0E0283A8 400521DC 00000000 00026AA8    *................. ..........*
026AC0     0CC0CCC4 C0000000 |01|02418C| 08000019     000C1B00 00010000 01000000 00000C00    *..............................*
026AE0     0C120000 19057620 |DCB ADDRESS 100000     0C0CC000 0E0283A8 60057120 00000000    *......................-.......*
026B00     00026AEC 000000C3 |CC00C000 010241B8     08000000 00000000 00000000 00000000    *..............................*
026B20     CC0000C5 7440057D |4C05744C 00057D40     42000C00 0000000C 0E026B38 600283A8    *..... .. .. ... ...........-...*
                              RELATIVE LINE
                              NUMBER
```

Figure 26. Finding the LCB in a Stand-Alone Dump (Part 4 of 6)

```
024140     17000C00 000249C0 12024B5C 00300040     C2060968 0102D350 C8000000 01084040    *...........*... ......L&H.....  *
024160     00017634 12023110 01 START   17000000     000249D0 12024B5C 00300040 02C608A0    *......................*... ....*
024180     0102D350 C8C00000 0( OF DCB |00017594      12023110 010200A8 17000000 000249E0    *..L&H.....  ..................*
0241A0     12024B5C C03 IOB BASE |060708| 0202D350    |C8|000000 011C404C 00016FA4 12023110    *...*... ...Q..L&H.....   ..?.....*
0241C0     01020CA8 170 ADDRESS 0249F0 12024B5C   LCB SIZE 140 02057070 0102D350 D8C00000    *...........0...*... ......L&Q...*
0241E0     013C4C40 00016EC4 12023110 010300FC      170C000C 00024A00 2202580C 00300C40    *..   ..>D.................... *
024200     04023170 C102D350 8CC00000 C4C4F2F7     F6F04040 02004040 01170054 00000000    *......L&....DD2760   ..  ........*
```

Figure 26. Finding the LCB in a Stand-Alone Dump (Part 5 of 6)

```
06 START  0C0C0CC0 CC0000C0 0C00C000 00C0G000    0000C000 C0000000 00C000C0 58DDCC04   *................................*
06 OF LCB EC06C880 EC02342C CC060888 40026AB0    00001400 00000000 000273FC 18C00CC0   *............ ...................*
06        C20C00C0 7F0233E0 0C000000 0000C000    40060920 C002418C 000305E8 00000C00   *B............... ..........Y....*
0608C0    0CC57F20 CC000200 CC000000 000234B8    E402342C 01057F20 00000019 40C273FC   *.................U.......... ...*
0608E0    CC0249F8 51110103 C4090000 00C0C000    00FF0000 00000000 005800C2 00C0CC1D   *...8............................*
060900    0C054AA0 C0023170 400273FC 00001014    C20608F1 B000C0C1 08060910 00C57F20   *......... ..........1...........*
060920    0102312D 60000003 C10249F8 60000C02    02057F61 80000002 08057F20 00C00CC0   *....-......8-....../............*
060940    0806C910 C00000C0 0C060948 E002342C    0C060950 200521DC 00001400 10000C00   *..................&.............*
```

Figure 26. Finding the LCB in a Stand-Alone Dump (Part 6 of 6)

Finding the SCB: LCB+X'5C' is a three-byte pointer to the SCB.

Finding the Message Error Record: The first four bytes of the message error record are found at SCB+X'10'. The last byte is found at LCB+X'22'.

Secondary-Storage Dumps

This section discusses TCAM tables and data sets, maintained on secondary-storage devices, that you can dump to tape for later printing. Most of these tables and data sets are optional in TCAM, and are included or excluded when you code your TCAM program. A description of the data in the table or data set and a description of what you must do to obtain and print a secondary-storage dump of the data follows.

Disk Message Queues Dump

TCAM handles message traffic using queues. If your queues are in main storage only, you cannot dump them with a utility. More commonly, they will be on a direct access device or in main storage with disk backup, and you can print them. This can help you diagnose, because TCAM records all messages transmitted in the system destined for stations with disk queuing. Dump the message queues data sets to tape at the end of the day if you wish an up-to-date log of your message traffic. Dump them also when you have a disk queuing problem, or when you have any problem in which queuing cannot positively be ruled out.

A TCAM utility program, IEDQXC, prints a formatted dump of all traffic directed to stations with disk queuing. You can dump the message queues data set sequentially either by record number or by queue. You obtain the most useful output by dumping the data set sequentially by queue. See Figures 29 and 30.

The PARM= parameter on the EXEC statement for IEDQXC determines the contents of the dump. The format of the EXEC statement is

```
    //stepname EXEC PGM=IEDQXC,PARM='Q=options'
```

Options include

DMP — Prints all messages sequentially by record number.
xxx,DMP — Prints all messages sequentially by record number. Replace *xxx* with the 3-digit decimal total number of queues (see Note 1).
xxx,ALL — Prints all messages sequentially by queue. Replace *xxx* with the 3-digit decimal total number of queues.
xxx,n1 n1 n1,n2 n2 n2,...Prints all messages for queues *n1 n1 n1* through *n5 n5 n5* (5 is the maximum number of queues that can be selectively dumped). *xxx* is the total number of queues, and *nnn* is a 3-digit decimal number corresponding to the queue whose contents are to appear in the dump.

The first two options are equivalent to one another and equivalent to omitting the PARM= parameter entirely.

Note 1: *The total number of queues is the number of stations that use that particular type of queuing being dumped (reusable or nonreusable). Find the total number of disk queues in your MCP assembly listing. In the expansion of the macro for the terminal-table entry named by TTABLE LAST=name, you will find the instruction*

```
ORG IEDNADDR
```

followed by the instruction

DC A(n*4+1),A(r*4+3)

where

n is the total number of queues on the nonreusable data set, and
r is the total number of queues on the reusable data set.

The n or r variable is the maximum value that you can assign to Q=xxx in the PARM=parameter.

Example:

```
TTABLE LAST=EVERY,MAXLEN=8
```

Figure 27 shows the total number of queues.

Note 2: *Define each extent of the disk data set with a DISKQnn DD statement, where* ***nn*** *is the extent number (DISKQ01 is the first extent, DISKQ02 is the second, etc.). For single-extent cataloged data sets, DSN= and DISP= are the only required parameters. For multi-extent (multivolume)*

```
     TERMINAL MACROS                                                                                    PAGE   76

  LOC  OBJECT CODE      ADDR1 ADDR2  STMT    SOURCE STATEMENT                                        F15OCT70   7/14/71

                                     3331 EVERY     TLIST TYPE=D,LIST=(NYC,AAA,BBB,REMOTE3,REMOTE4,REMOTE1,        *
                                                          REMOTE2,AA,BB,BOS,TEX,ALA,HUYCK,MARY,RAL,DUR,ATL,        *
                                                          LOCAL1,WAS,CHAR)


003458                               3334+IEDQTNT   CSECT
00364C C5E5C5D9E8404040              3335+EVERY     DC    CL8'EVERY' .    ENTRY NAME
003654 001810                        3336+          DC    AL3(IED39A)

000000                               3338+PDTCAM    CSECT
001810                               3339+IED39A    DS    0F
001810 48                            3340+          DC    BL1'01001000'
001811 000000                        3341+          DC    VL3(IEDQBC)
001814 0014                          3342+          DC    AL2(20) .               COUNT OF TLIST ENTRIES
001816 000C                          3343+          DC    AL2((NYC-IEDQTNT-71)/11)
001818 000F                          3344+          DC    AL2((AAA-IEDQTNT-71)/11)
00181A 0010                          3345+          DC    AL2((BBB-IEDQTNT-71)/11)
00181C 001E                          3346+          DC    AL2((REMOTE3-IEDQTNT-71)/11)
00181E 001F                          3347+          DC    AL2((REMOTE4-IEDQTNT-71)/11)
001820 001C                          3348+          DC    AL2((REMOTE1-IEDQTNT-71)/11)
001822 001D                          3349+          DC    AL2((REMOTE2-IEDQTNT-71)/11)
001824 0016                          3350+          DC    AL2((AA-IEDQTNT-71)/11)
001826 0017                          3351+          DC    AL2((BB-IEDQTNT-71)/11)
001828 0019                          3352+          DC    AL2((BOS-IEDQTNT-71)/11)
00182A 001A                          3353+          DC    AL2((TEX-IEDQTNT-71)/11)
00182C 0018                          3354+          DC    AL2((ALA-IEDQTNT-71)/11)
00182E 0007                          3355+          DC    AL2((HUYCK-IEDQTNT-71)/11)
00183C 0008                          3356+          DC    AL2((MARY-IEDQTNT-71)/11)
001832 0009                          3357+          DC    AL2((RAL-IEDQTNT-71)/11)
001834 000A                          3358+          DC    AL2((DUR-IEDQTNT-71)/11)
001836 000B                          3359+          DC    AL2((ATL-IEDQTNT-71)/11)
001838 001B                          3360+          DC    AL2((LOCAL1-IEDQTNT-71)/11)
00183A 000D                          3361+          DC    AL2((WAS-IEDQTNT-71)/11)
00183C 000E                          3362+          DC    AL2((CHAR-IEDQTNT-71)/11)

0004BC                               3364+          ORG   IEDNADDR
0004BC 000000BC0000000B              3365+          DC    A(10*4+1,A(10*4+1)
00183E                               3366+          ORG
003658                               3367+IEDQOPT   CSECT
003672 0003                          3368+IEDQOPTN  DC    AL2(3)
000000                               3369+PDTCAM    CSECT
```

10 STATIONS HAVE REUSABLE DISK QUEUING

10 STATIONS HAVE NONREUSABLE DISK QUEUING

Figure 27. Finding the Number of Queues in a TCAM System

data sets, the catalog information ***cannot*** *be used. Each DD statement must define the volume identification in the* ***same*** *order as the volume identification listed in the IEDQDATA DD statement for the disk initialization program IEDQXA.*

IEDQXC issues an OPEN for the maximum number of extents permitted (16). Therefore, on your JCL printout, ignore the message

```
IEC103I DISKQnn DD STATEMENT MISSING
```

where nn is a number from 2 to 16. The program runs successfully without these DD statements if they are not applicable.

Example: The following JCL lists all queues from a multivolume data set:

```
//jobname      JOB
//stepname     EXEC PGM=IEDQXC,PARM='Q=xxx,ALL'
//DISKQ01      DD   DSN=dsname,DISP=OLD,UNIT=23xx,*
//                  VOL=SER=xxxxxx
//DISKQ02      DD   DSN=dsname,DISP=OLD,UNIT=23xx,*
//                  VOL=SER=xxxxxx
//SYSPRINT     DD   SYSOUT=A
/*
```

where dsname is the same name you assigned the disk data set when you executed IEDQXA to preformat the disk.

The queues may be long. Therefore, you can select specific queues to dump, up to a maximum of 5. To determine the queue number for the station whose message queue you want to dump, refer to your assembly listing. The queue numbers are assigned in the order in which you coded the TERMINAL macros. Therefore, the first TERMINAL macro having nonreusable queuing is queue number 001 on the nonreusable disk data set. The same is true for reusable queuing.

You must keep two things in mind. First, if you are queuing by line (QBY=L), TCAM assigns only one queue to all terminals on the line. Second, if you have priority levels (LEVEL=), then TCAM assigns each priority level a queue, and all terminals with priority levels have an additional level of zero; therefore, there is one more queue than the number of levels you list in the operand. For instance, LEVEL=(241,242) generates three queues on the data set for that terminal.

Example:

```
NYC TERMINAL QUEUES=MN,QBY=T,LEVEL=(240,247)...
KAN TERMINAL QUEUES=MO,QBY=T,...
KIX TERMINAL QUEUES=DR,QBY=L,LEVEL=(241,243)...
BOS TERMINAL QUEUES=MR,QBY=L,LEVEL=(241,243)...
ALA TERMINAL QUEUES=DN,QBY=T...
FLA TERMINAL QUEUES=MO,QBY=T...
GAL TERMINAL QUEUES=DN,QBY=T,LEVEL=(242,244)...
RAL TERMINAL QUEUES=DN,QBY=T...
BON TERMINAL QUEUES=DR,QBY=T...
WAS TERMINAL QUEUES=MR,QBY=T...
ATL TERMINAL QUEUES=MN,QBY=T...
```

On the reusable disk data set, the stations have the following queue numbers:

```
KIX          LEVEL 243 QUEUE 001
             LEVEL 241 QUEUE 002
             LEVEL   0 QUEUE 003
BON                    QUEUE 004
WAS                    QUEUE 005
```

Note: *BOS has the same queue numbers as KIX, since they are queued by line.*

On the nonreusable disk data set, the following queue numbers are assigned:

```
NYC     LEVEL 247 QUEUE 001
        LEVEL 240 QUEUE 002
        LEVEL   0 QUEUE 003
ALA               QUEUE 004
GAL     LEVEL 244 QUEUE 005
        LEVEL 242 QUEUE 006
        LEVEL   0 QUEUE 007
RAL               QUEUE 008
ATL               QUEUE 009
```

Therefore, if you are interested only in the traffic directed to NYC, use the following EXEC statement:

```
//stepname EXEC PGM=IEDQXC,PARM='Q=009,001,002,003'
```

009 is the total number of queues on the nonreusable disk data set.

Run IEDQXC separately to dump the reusable and nonreusable disk queues, since the DSN= parameter on the DISKQnn DD statement must be the name you assigned when you executed IEDQXA to preformat the disk, and you must define and preformat two data sets, one reusable and one nonreusable.

If you cannot dump your message queues data set because it is too large, either

1. code a SPACE= parameter on your SYSPRINT DD statement if you are using a SYSOUT queue (the SPACE= default on your system may not be large enough to contain the entire dump), or
2. allocate SYSPRINT directly to the unit on which you wish to dump the queues (printer, magnetic tape, or disk).

Message Queues Data Set

The output from the IEDQXC utility program contains a great deal of information about the message, including its source terminal, destination terminal, and how TCAM processed it. The column headings on the printed output correspond to the 30-byte buffer prefix for each message. See Figure 28.

Heading	*Explanation*
NT	The number of units in the buffer. TCAM determines this using the KEYLEN= operand (the number of bytes in one unit) on the INTRO macro and the BUFSIZE= operand (the number of bytes in one buffer) on the TERMINAL macro, if specified, or the DCB macro for the station.
LCB	The LCB (line control block) address for the source terminal.
SRCE	The terminal-name table offset for the source of the message. The number is the position of the source terminal alphabetically in a list of all terminals.
SIZE	The number of bytes of data in this buffer.
ST	The status byte, which is the state of the buffer when it was written on the data set.

Value	*Meaning*
X'80'	message has been canceled
X'40'	this buffer contains an error message
X'20'	not used
X'10'	this is a TSO buffer
X'08'	this is a duplicate-header buffer
X'04'	SETEOF was executed
X'02'	this is not the last buffer of the message
X'01'	this is not the first buffer of the message
X'00'	only one buffer is in the message

NXTREC	Pointer to the next unit in the buffer.
SCAN	Hexadecimal offset from the beginning of the buffer prefix to the location of the scan pointer. The offset is in the first byte.
NXTTXT	Pointer to the next buffer in the message if this is not the last buffer, or the message queue-back chain if it is the last buffer.
FSTREC	Pointer to the first unit of the current header buffer.
NXTHDR	Pointer to the first buffer of the next message (the next-header segment).
QBACK	Queue-back chain of the first buffers of messages (the chain of header segments).
SEQO	The input sequence number.
DEST	The terminal-name table offset for the destination of the message (the position of the destination terminal alphabetically in the list of all terminals).

In addition to the information in the prefix, information about the message is in the last six bytes of every record, which is the data field of the disk record.

If byte 0 is X'80', then the record has no prefix; it is an extra record. The next three bytes contain the disk address of this record. The fifth byte contains the number of bytes of valid data in the record. The last byte is unused. See Figure 28.

If byte 0 is not X'80', then the record does have a prefix.

If byte 0 is X'00' and the ST field of the prefix is X'01', then the record is all text. The remaining five bytes of the data field are unused. See Figure 28.

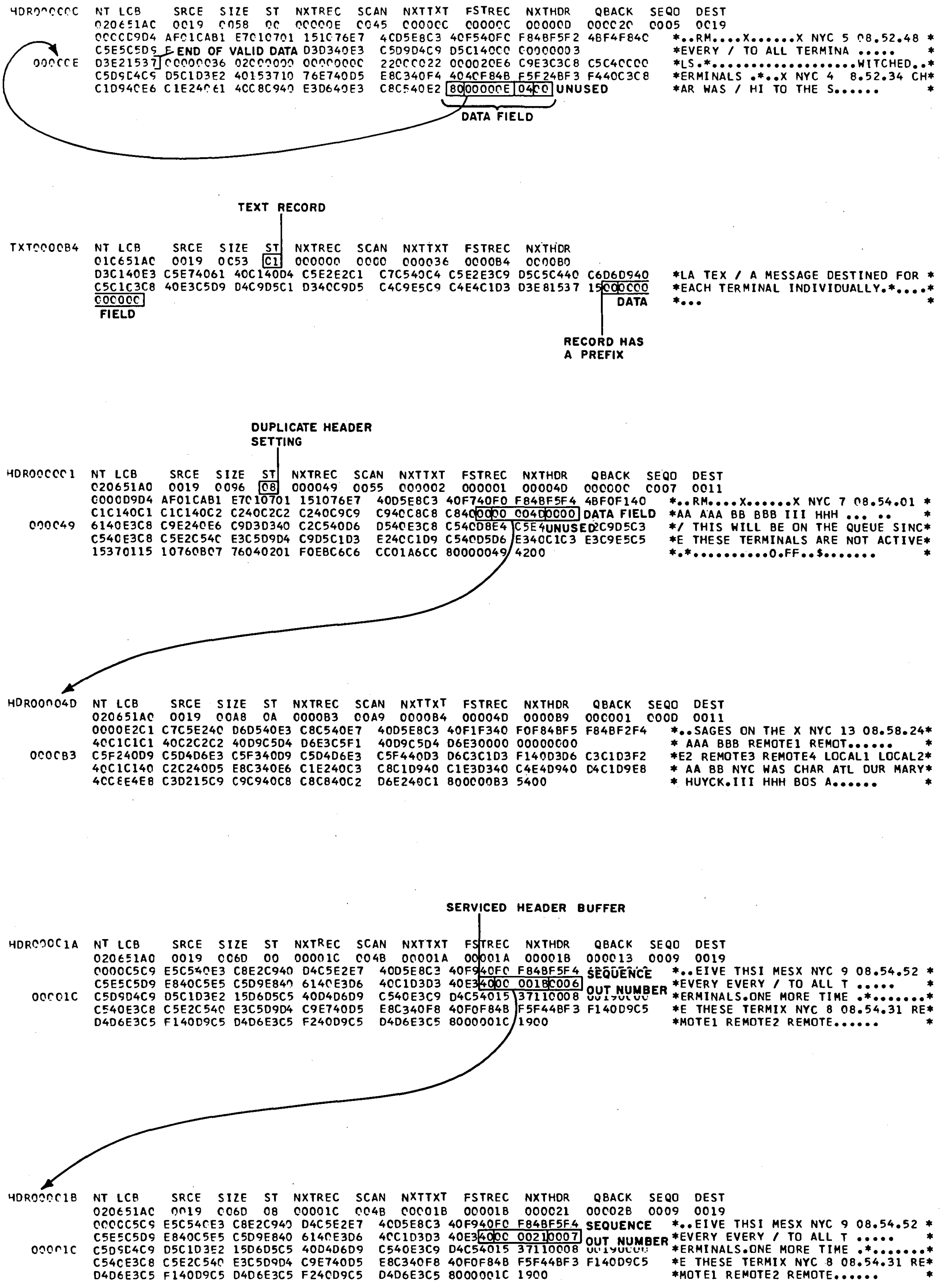
If byte 0 of the data field is X'00' and the ST field is not X'01', then the record is a header record that has not been serviced (the message has not yet been sent successfully). The last two bytes are unused. See Figure 28.

If byte 0 is X'40', then the record is a header record for a message that has been serviced. The next three bytes contain a pointer to the next FEFO header record, and the last two bytes contain the output sequence number (a sequential count of the records on the queue). See Figure 28.

If the first byte of the data field is X‘20’, then the record is a header record with a prefix, and it has been canceled (not transmitted).

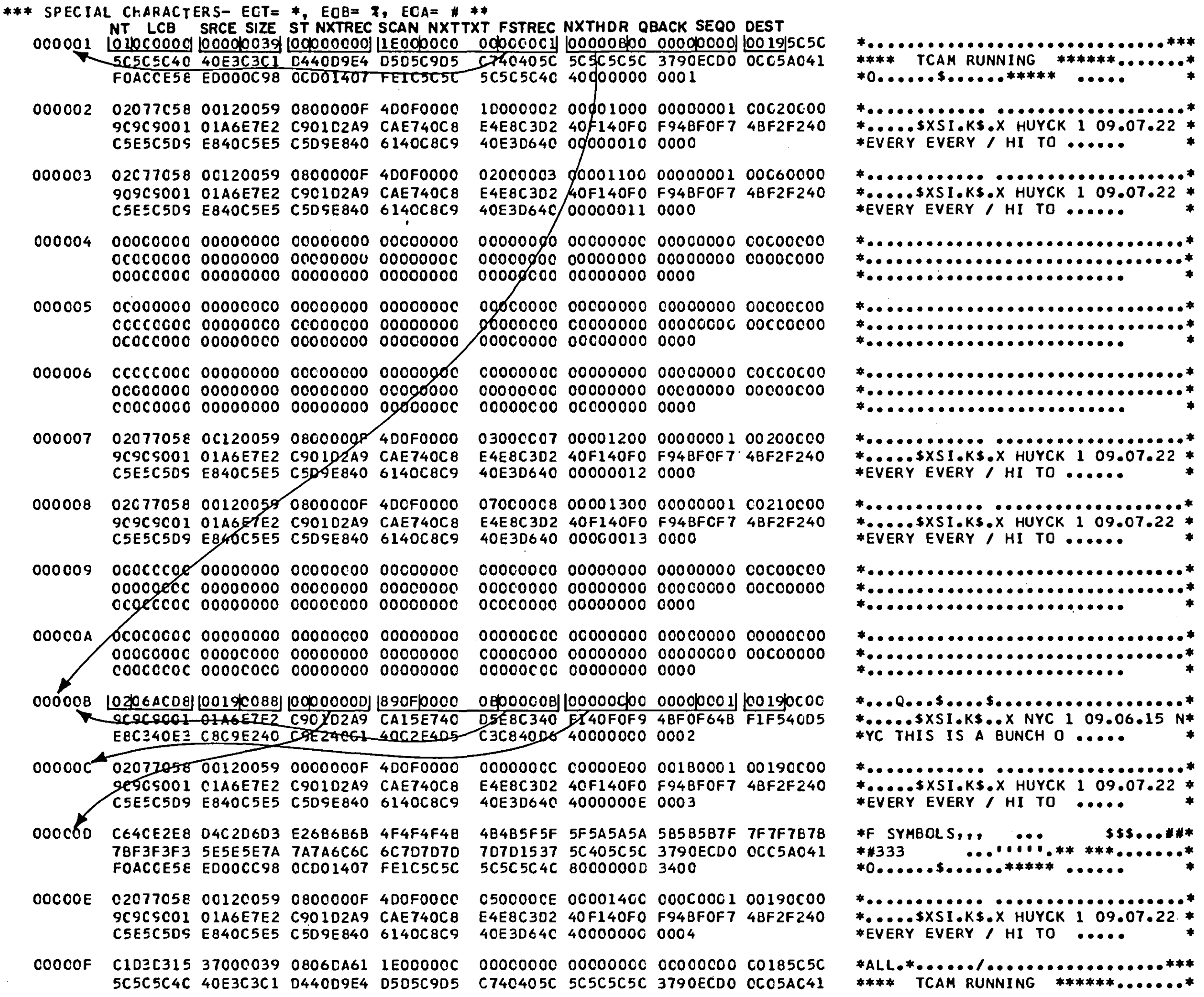
It will help you diagnose queuing problems by having all your terminals on some type of disk queue, either disk queuing or main-storage queuing with disk backup because it gives you a permanent record of message traffic and processing.

Figure 29 shows a sequential-by-record dump and Figure 30 shows a sequential-by-queue dump.



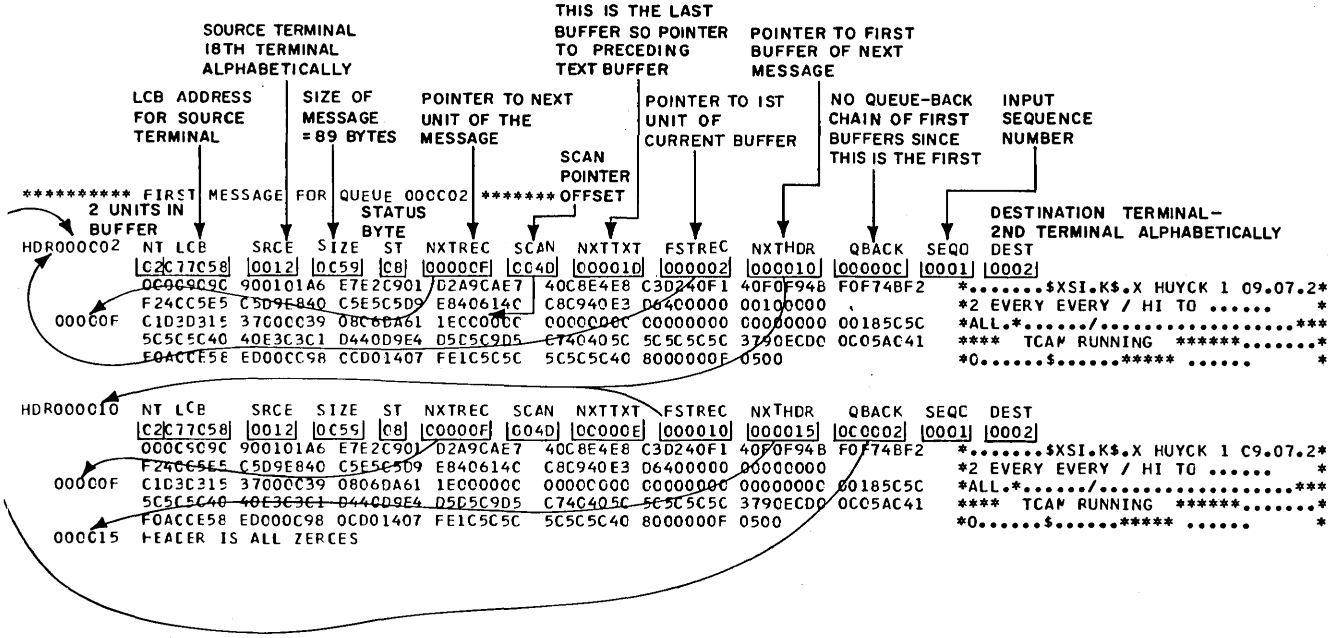
```
HDR00000C   NT LCB    SRCE  SIZE  ST  NXTREC  SCAN  NXTTXT  FSTREC  NXTHDR   QBACK  SEQ0  DEST
            020651AC  0019  0058  00  00000E  0045  00000C  00000C  00000D  000020  0005  0019
            0000D9D4 AF01CAB1 E7010701 151076E7  40D5E8C3 40F540F0 F84BF5F2 4BF4F840   *..RM....X......X NYC 5 08.52.48 *
            C5E5C5D9 E END OF VALID DATA D3D340E3  C5D9D4C9 D5C14000 00000003              *EVERY / TO ALL TERMINA .....    *
   00000E   D3E21537 00000036 02000000 0000000C  2200C022 000020E6 C9E3C3C8 C5C40000   *LS.*.................WITCHED..*
            C5D9D4C9 D5C1D3E2 40153710 76E740D5  E8C340F4 4040F84B F5F24BF3 F440C3C8   *ERMINALS .*..X NYC 4  8.52.34 CH*
            C1D940E6 C1E24061 40C8C940 E3D640E3  C8C540E2 80 00000E 04 00 UNUSED       *AR WAS / HI TO THE S......      *
                                                            DATA FIELD

                                  TEXT RECORD

TXT0000B4   NT LCB    SRCE  SIZE  ST  NXTREC  SCAN  NXTTXT  FSTREC  NXTHDR
            01C651AC  0019  0C53  C1  000000  0000  000036  0000B4  0000B0
            D3C140E3 C5E74061 40C140D4 C5E2E2C1  C7C540C4 C5E2E3C9 D5C5C440 C6D6D940   *LA TEX / A MESSAGE DESTINED FOR *
            C5C1C3C8 40E3C5D9 D4C9D5C1 D340C9D5  C4C9E5C9 C4E4C1D3 D3E81537 15 00 0C00  *EACH TERMINAL INDIVIDUALLY.*....*
            00000C                                                              DATA   *...                             *
            FIELD
                                                                        RECORD HAS
                                                                        A PREFIX

                                  DUPLICATE HEADER
                                  SETTING

HDR000001   NT LCB    SRCE  SIZE  ST  NXTREC  SCAN  NXTTXT  FSTREC  NXTHDR   QBACK  SEQ0  DEST
            020651A0  0019  0096  08  000049  0055  000002  000001  00004D  00000C  C007  0011
            0000D9D4 AF01CAB1 E7010701 151076E7  40D5E8C3 40F740F0 F84BF5F4 4BF0F140   *..RM....X......X NYC 7 08.54.01 *
            C1C140C1 C1C140C2 C240C2C2 C240C9C9  C940C8C8 C840 00 00 004D 0000 DATA FIELD  *AA AAA BB BBB III HHH ... ..    *
   000049   6140E3C8 C9E240E6 C9D3D340 C2C540D6  D540E3C8 C540D8E4 C5E4 UNUSED 2C9D5C3     */ THIS WILL BE ON THE QUEUE SINC*
            C540E3C8 C5E2C540 E3C5D9D4 C9D5C1D3  E240C1D9 C540D5D6 E340C1C3 E3C9E5C5   *E THESE TERMINALS ARE NOT ACTIVE*
            15370115 10760B07 76040201 F0E8C6C6  CC01A6CC 80000049 4200                *.*...........0.FF..$.......     *

HDR00004D   NT LCB    SRCE  SIZE  ST  NXTREC  SCAN  NXTTXT  FSTREC  NXTHDR   QBACK  SEQ0  DEST
            020651AC  0019  00A8  0A  0000B3  00A9  0000B4  00004D  0000B9  00C001  C00D  0011
            0000E2C1 C7C5E240 D6D540E3 C8C540E7  40D5E8C3 40F1F340 F0F84BF5 F84BF2F4   *..SAGES ON THE X NYC 13 08.58.24*
            40C1C1C1 40C2C2C2 40D9C5D4 D6E3C5F1  40D9C5D4 D6E30000 00000000            * AAA BBB REMOTE1 REMOT......    *
   0000B3   C5F240D9 C5D4D6E3 C5F340D9 C5D4D6E3  C5F440D3 D6C3C1D3 F140D3D6 C3C1D3F2   *E2 REMOTE3 REMOTE4 LOCAL1 LOCAL2*
            40C1C140 C2C240D5 E8C340E6 C1E240C3  C8C1D940 C1E3D340 C4E4D940 D4C1D9E8   * AA BB NYC WAS CHAR ATL DUR MARY*
            40C8E4E8 C3D215C9 C9C940C8 C8C840C2  D6E240C1 800000B3 5400                * HUYCK.III HHH BOS A......      *

                                                  SERVICED HEADER BUFFER

HDR00001A   NT LCB    SRCE  SIZE  ST  NXTREC  SCAN  NXTTXT  FSTREC  NXTHDR   QBACK  SEQ0  DEST
            020651A0  0019  006D  00  00001C  004B  00001A  00001A  00001B  000013  0009  0019
            0000C5C9 E5C540E3 C8E2C940 D4C5E2E7  40D5E8C3 40F940F0 F84BF5F4 SEQUENCE    *..EIVE THSI MESX NYC 9 08.54.52 *
            C5E5C5D9 E840C5E5 C5D9E840 6140E3D6  40C1D3D3 40E3 40 00 001B C006 OUT NUMBER  *EVERY EVERY / TO ALL T .....    *
   00001C   C5D9D4C9 D5C1D3E2 15D6D5C5 40D4D6D9  C540E3C9 D4C54015 37110008            *ERMINALS.ONE MORE TIME .*.......*
            C540E3C8 C5E2C540 E3C5D9D4 C9E740D5  E8C340F8 40F0F84B F5F44BF3 F140D9C5   *E THESE TERMIX NYC 8 08.54.31 RE*
            D4D6E3C5 F140D9C5 D4D6E3C5 F240D9C5  D4D6E3C5 8000001C 1900                *MOTE1 REMOTE2 REMOTE......      *

HDR00001B   NT LCB    SRCE  SIZE  ST  NXTREC  SCAN  NXTTXT  FSTREC  NXTHDR   QBACK  SEQ0  DEST
            020651AC  0019  006D  08  00001C  004B  00001B  00001B  000021  00002B  0009  0019
            0000C5C9 E5C540E3 C8E2C940 D4C5E2E7  40D5E8C3 40F940F0 F84BF5F4 SEQUENCE    *..EIVE THSI MESX NYC 9 08.54.52 *
            C5E5C5D9 E840C5E5 C5D9E840 6140E3D6  40C1D3D3 40E3 40 00 0021 0007 OUT NUMBER  *EVERY EVERY / TO ALL T .....    *
   00001C   C5D9D4C9 D5C1D3E2 15D6D5C5 40D4D6D9  C540E3C9 D4C54015 37110008            *ERMINALS.ONE MORE TIME .*.......*
            C540E3C8 C5E2C540 E3C5D9D4 C9E740D5  E8C340F8 40F0F84B F5F44BF3 F140D9C5   *E THESE TERMIX NYC 8 08.54.31 RE*
            D4D6E3C5 F140D9C5 D4D6E3C5 F240D9C5  D4D6E3C5 8000001C 1900                *MOTE1 REMOTE2 REMOTE......      *
```

Figure 28. Message Queues Data Set Printout



```
*** SPECIAL CHARACTERS- EOT= *, EOB= %, EOA= # **
         NT  LCB   SRCE SIZE  ST NXTREC SCAN NXTTXT FSTREC NXTHDR QBACK SEQO DEST
000001   010C0000 00000039 00000000 1E0000C0  0000C0C1 00000B00 00000000 00195C5C   *..............................***
         5C5C5C40 40E3C3C1 D440D9E4 D5D5C9D5  C740405C 5C5C5C5C 3790ECD0 0CC5A041   ****  TCAM RUNNING  ******.......*
         F0ACCE58 ED000C98 0CD01407 FE1C5C5C  5C5C5C40 40000000 0001                *0......$......*****  .....      *

000002   02077C58 00120059 0800000F 4D0F0000  1D000002 00001000 00000001 C0C20C00   *............ ...................*
         9C9C9001 01A6E7E2 C901D2A9 CAE740C8  E4E8C3D2 40F140F0 F94BF0F7 4BF2F240   *.....$XSI.K$.X HUYCK 1 09.07.22 *
         C5E5C5D9 E840C5E5 C5D9E840 6140C8C9  40E3D640 00000010 0000                *EVERY EVERY / HI TO ......      *

000003   02C77058 00120059 0800000F 4D0F0000  02000003 00001100 00000001 00C60000   *............ ...................*
         909C9001 01A6E7E2 C901D2A9 CAE740C8  E4E8C3D2 40F140F0 F94BF0F7 4BF2F240   *.....$XSI.K$.X HUYCK 1 09.07.22 *
         C5E5C5D9 E840C5E5 C5D9E840 6140C8C9  40E3D64C 00000011 0000                *EVERY EVERY / HI TO ......      *

000004   000C0000 00000000 00000000 00C00000  00000000 0000000C 00000000 C0C00C00   *................................*
         0C0C0000 00000000 00C00000 0000000C  00000000 00000000 00000000 0000C000   *................................*
         000C000C 00000000 00000000 00000000  00000C00 00000000 0000                *..........................      *

000005   0C000000 00000C00 00000000 0000000C  000C0000 00C00000 C0000000 00C0CC00   *................................*
         CCCC000C 00000C00 CC000C00 00000000  00000CC0 C0000000 0000000C 00CC0000   *................................*
         0C0CC000 00000C00 00000000 00000000  000C0000 00C00000 0000                *..........................      *

000006   CCCCC00C 00000000 00C00000 0000000C  C000000C 00000000 00000000 C0CC0C00   *................................*
         0C000000 00000000 00000000 00000000  0000000C 00000000 00000000 00000C00   *................................*
         C00C0000 00000000 00000000 0000000C  00000C00 0C000000 0000                *..........................      *

000007   02077058 0C120059 0800000F 4D0F0000  0300CC07 00001200 00000001 00200C00   *............ ...................*
         9C9C9001 01A6E7E2 C901D2A9 CAE740C8  E4E8C3D2 40F140F0 F94BF0F7 4BF2F240   *.....$XSI.K$.X HUYCK 1 09.07.22 *
         C5E5C5D9 E840C5E5 C5D9E840 6140C8C9  40E3D640 00000012 0000                *EVERY EVERY / HI TO ......      *

000008   02C77058 00120059 0800000F 4D0F0000  070C00C8 00001300 00000001 C0210C00   *............ ...................*
         9C9C9C01 01A6E7E2 C901D2A9 CAE740C8  E4E8C3D2 40F140F0 F94BFCF7 4BF2F240   *.....$XSI.K$.X HUYCK 1 09.07.22 *
         C5E5C5D9 E840C5E5 C5D9E840 6140C8C9  40E3D640 000C0013 0000                *EVERY EVERY / HI TO ......      *

000009   0C0CCC0C 00000000 00000C00 00C00000  00000CC0 00000000 00000000 C0C00CC0   *................................*
         00000CCC 00000000 00000000 00000000  000C0000 00000000 000C0000 00C00000   *................................*
         0C0CCC0C 00000000 00000000 00000000  0C0C0000 00000000 0000                *..........................      *

00000A   0C0C0C0C 00000000 00000000 00000000  00000CCC 0C000000 000C0000 00000C00   *................................*
         000C000C 0000C000 00000000 00000000  C000C000 00000000 00000000 00C00000   *................................*
         C00CCC0C 0000C0C0 00000000 00000000  00000CCC 00000000 0000                *..........................      *

00000B   0206ACD8 0019C088 0000000D 890F0000  0B00000B 00000C00 00000001 00190C0C   *...Q...$.....$..................*
         9C9C9C01 01A6E7E2 C901D2A9 CA15E740  D5E8C340 F140F0F9 4BF0F64B F1F540D5   *.....$XSI.K$..X NYC 1 09.06.15 N*
         E8C340E3 C8C9E240 C9E240C1 40C2E4D5  C3C840D6 40000000 0002                *YC THIS IS A BUNCH O .....      *

00000C   02077058 00120059 0000000F 4D0F0000  000000CC C0000E00 00180001 00190C00   *............ ...................*
         9C9C9001 C1A6E7E2 C901D2A9 CAE740C8  E4E8C3D2 40F140F0 F94BF0F7 4BF2F240   *.....$XSI.K$.X HUYCK 1 09.07.22 *
         C5E5C5D9 E840C5E5 C5D9E840 6140C8C9  40E3D640 4000000E 0003                *EVERY EVERY / HI TO  .....      *

00000D   C64CE2E8 D4C2D6D3 E26B6B6B 4F4F4F4B  4B4B5F5F 5F5A5A5A 5B5B5B7F 7F7F7B7B   *F SYMBOLS,,,   ...      $$$...##*
         7BF3F3F3 5E5E5E7A 7A7A6C6C 6C7D7D7D  7D7D1537 5C405C5C 3790ECD0 0CC5A041   *#333      ...''''''.** ***.......*
         F0ACCE58 ED00CC98 0CD01407 FE1C5C5C  5C5C5C4C 8000000D 3400                *0......$......***** ......      *

00C00E   02077058 00120059 0800000F 4D0F000C  C50000CE 0000140C 000C0001 00190C00   *............ ...................*
         9C9C9C01 01A6E7E2 C901D2A9 CAE740C8  E4E8C3D2 40F140F0 F94BF0F7 4BF2F240   *.....$XSI.K$.X HUYCK 1 09.07.22 *
         C5E5C5D9 E840C5E5 C5D9E840 6140C8C9  40E3D64C 4000000C 0004                *EVERY EVERY / HI TO  .....      *

00000F   C1D3D315 37000039 0806DA61 1E00000C  00000000 0000000C 0C0000C0 C0185C5C   *ALL.*....../..................***
         5C5C5C4C 40E3C3C1 D440D9E4 D5D5C9D5  C740405C 5C5C5C5C 3790ECD0 0C05AC41   ****  TCAM RUNNING  ******.......*
```

Figure 29. A Sequential-by-Record Dump



```
********** FIRST MESSAGE FOR QUEUE 00CC02 *******
HDR000C02  NT LCB     SRCE  SIZE  ST  NXTREC  SCAN  NXTTXT  FSTREC  NXTHDR  QBACK   SEQO  DEST
           C2C77C58   0012  0C59  C8  0000CF  CC4D  00001D  000002  000010  00000C  0001  0002
           0C0C9C9C 900101A6 E7E2C901 D2A9CAE7  40C8E4E8 C3D240F1 40F0F94B F0F74BF2    *........$XSI.K$.X HUYCK 1 09.07.2*
           F24CC5E5 C5D9E840 C5E5C5D9 E840614C  C8C940E3 D6400000 00100000  ,          *2 EVERY EVERY / HI TO ......    *
000C0F     C1D3D315 37C00C39 08C6DA61 1ECC00CC  0000C00C C0000000 00000000 00185C5C    *ALL.*....../....................***
           5C5C5C40 40E3C3C1 D440D9E4 D5C5C9D5  C740405C 5C5C5C5C 3790ECDC 0C05AC41    ****  TCAM RUNNING  ******.......*
           F0ACCE58 ED00CC98 CCD01407 FE1C5C5C  5C5C5C40 8000000F 0500                 *0.......$.......***** ......    *

HDR000010  NT LCB     SRCE  SIZE  ST  NXTREC  SCAN  NXTTXT  FSTREC  NXTHDR  QBACK   SEQC  DEST
           C2C77C58   0012  0C59  08  C0000F  G04D  0C000E  000010  000015  0C0002  0001  0002
           000C9C9C 900101A6 E7E2C901 D2A9CAE7  40C8E4E8 C3D240F1 40F0F94B F0F74BF2    *........$XSI.K$.X HUYCK 1 C9.07.2*
           F24CC5E5 C5D9E840 C5E5C5D9 E840614C  C8C940E3 D6400000 00000000             *2 EVERY EVERY / HI TO ......    *
000C0F     C1DC3C315 37000C39 0806DA61 1E00000C  0000C000 C000000C 0000000C 00185C5C   *ALL.*....../....................***
           5C5C5C40 40E3C3C1 D440D9E4 D5D5C9D5  C740405C 5C5C5C5C 3790ECDC 0C05AC41    ****  TCAM RUNNING  ******.......*
           F0ACCE58 ED000C98 0CD01407 FE1C5C5C  5C5C5C40 8000000F 0500                 *0.......$.......***** ......    *
000C15     HEADER IS ALL ZEROES
```

Figure 30. A Sequential-by-Queue Dump

Checkpoint/Restart Dump

The optional TCAM checkpoint/restart facility restarts the TCAM system with a minimum loss of message data following system failure. To do this, TCAM periodically records, in a special data set on disk, the status of each station, destination queue, terminal-table entry, and invitation list in the system. When the system starts up after closedown or failure, TCAM uses this information to restore the MCP environment to its condition before closedown or failure.

No TCAM utility dumps and formats the checkpoint data set. The best way to dump it is to use the OS service aid IMASPZAP (see *Service Aids,* GC28-6719, for details). You can use the following sample JCL to dump the checkpoint data set:

```
//DUMPCHK      JOB  MSGLEVEL=1
//STEP         EXEC PGM=IMASPZAP
//SYSLIB       DD   DSNAME=CHECKPT,DISP=SHR,UNIT=23xx,*
//                  VOL=SER=xxxxxx
//SYSPRINT     DD   SYSOUT=A
//SYSIN        DD   *
   ABSDUMPT ALL
/*
```

This JCL dumps the entire data set named on the SYSLIB statement in hexadecimal, with the EBCDIC translation and the mnemonic equivalent of the data.

Dump the checkpoint data set if you have any trouble restarting a system. As a precautionary measure, also dump at the end of the day if you plan to start the next day with a warm or continuation restart. If you cannot restart the system, immediately compare the dump from the preceding day with a dump of the current checkpoint data set to see if the data set was inadvertently scratched. If the dumps are identical, there may be a problem in the restart facility.

Log Data Set Dump

TCAM's message-logging facility records, on a sequential data set, the message traffic handled by an MCP. The LOG macro instruction records either a message or a message segment on a log data set while the message is being processed by an

MH subgroup. The LOG macro operand and the subgroup in which you code the LOG macro determine which is to be logged—message segments or complete messages. Anticipate the need for diagnostic aids in designing your MCP by including logging. Once your program is error-free, you can easily remove the log without rewriting the MCP. You should be aware that the LOG macro has an implied WAIT in its execution. Logging a segment impacts the system performance more than logging a message. If you are logging both segments and messages, define a separate data set for each. Once the log data set is filled, normal processing continues but logging is suspended. See the *TCAM Programmer's Guide* for details on how to code for segment and message logging.

Examine the log segment or log message output for a quick diagnosis of errors while debugging the MCP. Dump the log data sets when you have any problems in your MCP. By examining them you can see what message handler processing has been performed on each message, and you can see in which subgroup the message becomes incorrect. Dump the log data sets periodically and analyze your message traffic to be sure you are using your resources efficiently. You can also dump the log data sets as an accounting record, since they show all messages processed by your MCP.

Dumping the Log Segment Data Set: To dump the log segment data set use the TCAM utility IEDQXB. This utility prints a hexadecimal dump of the segment (a *segment* is the number of bytes in the KEYLEN= operand of the INTRO macro), with an English translation on the right. Thus, you can easily find your messages, and you also have the prefix of the header buffer for debugging.

The following sample JCL prints the log segments from the data set LOGMSG located on disk. The data set was created at MCP execution time.

```
//PRINTSEG     JOB  MSGLEVEL=1
//EDIT         EXEC PGM=IEDQXB
//SYSPRINT     DD   SYSOUT=A
//SYSUT1       DD   DSN=LOGMSG,UNIT=2311,DISP=OLD,*
//                  VOL=SER=111111
/*
```

Using the Log Segment Dump: The log segment facility records each segment processed by the message handler. TCAM places segments on the log data set in the sequence in which they are handled. Therefore, the segments of one message are likely to be intermixed with the segments of other messages on the data set. Figure 31 shows the log segment output produced by the utility IEDQXB. The LOG macro was included in the inheader subgroup before any processing, and in the outbuffer subgroup after all processing. The log segment entries for the message are on the log data set sequentially, although segments of other messages may be intermixed. Each time the LOG macro is executed, an entry is made into the data set. Therefore, there should be a one-to-one correspondence between the number of entries for a message and the number of LOG macros in the message handler. If you do not have the same number of log segment entries as you have LOG macros, then you know when, in message handler processing, you lost the message.

In Figure 31 the message is directed to two terminals; therefore the buffer segment passes through the outgoing message handler twice. By examining the buffer prefix (the destination offset or the LCB if you have a dump of main storage), you can tell which terminal received the message first. Also, the time in the output message shows the response time of the system.

By examining the log segment, you can see how the buffer is processed, which can be helpful when you are debugging your message handler.

Dumping the Log Message Data Set: No TCAM utility prints the output of the message log data set. The best way to get the dump is to use the OS utility IEBPTPCH. The prefix of a buffer on the log message data set is of no value. Use of the log message function (LOGTYPE macro) causes any useful information in the logged message prefix to be overlaid. Therefore, you may want to get only an English translation of the data set contents. The following sample JCL dumps a log message data set from an unlabeled tape. The FIELD= and LRECL= values are the same as the value specified in the KEYLEN= operand of the INTRO macro.

```
//PRINTMSG     JOB  MSGLEVEL=1
//DUMP         EXEC PGM=IEBPTPCH
//SYSPRINT     DD   SYSOUT=A
//SYSUT1       DD   DSN=MSGLOG,UNIT=2400,LABEL=(,NL),*
//                  VOL=SER=LOGTYP,DISP=OLD,*
//                  DCB=(RECFM=FU,LRECL=84,BLKSIZE=168)
//SYSUT2       DD   SYSOUT=A
//SYSIN        DD   *
   PRINT       MAXFLDS=1
   RECORD      FIELD=(84)
/*
```

```
**UNKNOWN TRACE ENTRY TYPE**            LENGTH- 0054

          020651A0 00000036 02000000 00000000  1A00001A 0000174C F1F0F5F0 7DE20C00   *....................... 1050.S..*
          37D4C9D5 C1D3E215 375E5E5E 5EE740D5  E8C340F2 4040F84B F5F14BF5 F340D4C1   *.MINALS......X NYC 2  8.51.53 MA*
          D9E840C8 E4E8C3D2 406140C8 C940E3D6  40E3C8C5 00404040 40404040 40404040   *RY HUYCK . HI TO THE.          *
**UNKNOWN TRACE ENTRY TYPE**            LENGTH- 0054

          020651AC 00190061 000690C0 00430000  0C0690CC 0690C000 00150003 00190000   *................................*
          D9D4E740 D5E8C340 F340F0F8 4BF5F24B  F0F740D5 E8C34061 40F240F0 F84BF5F2   *RMX NYC 3 C8.52.07 NYC . 2 08.52*
          4BF1FC40 C8C5D3D3 D640E3D6 40D4C515  17171717 00404040 40404040 40404040   *.10 HELLO TO ME......          *
**UNKNOWN TRACE ENTRY TYPE**            LENGTH- 0054

          17171717 17171717 17171717 37000000  1F00001F 00001D40 F1F0F5F0 7DE24B15   *....................... 1050.S..*
          37D4C9D5 C1D3E215 375E5E5E 5EE740D5  E8C340F2 40F0F84B F5F14BF5 F340D4C1   *.MINALS......X NYC 2 08.51.53 MA*
          D9E840C8 E4E8C3D2 406140C8 C940E3D6  37E3C8C5 D7404040 40404040 40404C40   *RY HUYCK . HI TO.THEP          *
**UNKNOWN TRACE ENTRY TYPE**            LENGTH- 0054                    BUFFER
  INHDR                                                                 PREFIX
  LOG     020651AC 00190068 00000000 003D0000  0000000B 0690C000 00150004 00190000   *................................*
  ENTRY   D9D4AF01 CAB1E701 07C11510 760B0476  150E01CA B116E740 D5E8C340 F440C3C8   *RM....X...............X NYC 4 CH*
          C1D940E6 C1E24061 40C8C940 E3D640E3  C8C540E2 47404040 40404040 40404C40   *AR WAS . HI TO THE S.          *
**UNKNOWN TRACE ENTRY TYPE**            LENGTH- 0054

          E6C9E3C3 C8C5C440 E3C5D9D4 C9D5C1D3  E2401537 00000000 00150003 00190C0C   *WITCHED TERMINALS ..............*
          D9D4C9D5 C1D3E215 375E5E5E 5EE740D5  E8C340F3 40F0F84B F5F24BF0 F74CD5E8   *RMINALS......X NYC 3 08.52.07 NY*
          C3406140 C8C5D3D3 D640E3D6 40D4C515  37E3C8C5 00404040 40404040 40404C40   *C . HELLO TO ME..THE.          *
**UNKNOWN TRACE ENTRY TYPE**            LENGTH- 0054                    BUFFER
  OUTBUF                                                                PREFIX
  LOG     030650C8 00190078 0CC68A00 00480000  00068A0C 068A0000 00150004 00CA0C00   *...H............................*
  ENTRY   D9D4E740 D5E8C340 F440F0F8 4BF5F24B  F3F440C3 C8C1D940 E6C1E240 6140F240   *RMX NYC 4 C8.52.34 CHAR WAS . 2 *
          F0F84BF5 F24BF3F4 40C8C940 E3D640E3  C8C540E2 47404040 40404040 40404040   *08.52.34 HI TO THE S.          *
**UNKNOWN TRACE ENTRY TYPE**            LENGTH- 0C54

          E6C9E3C3 C8C5C440 E3C5D9D4 C9D5C1D3  E2401517 17171717 17171717 17171717   *WITCHED TERMINALS ..............*
          17171737 40E3C3C1 D440D9E4 D5D5C9D5  C740405C 5C5C5C5C 3790ECD0 0CC5AC41   *.... TCAM RUNNING  .............*
          F0ACCE58 ED000C98 0CD01407 FE1C5C5C  5C5C5C4C 91404040 40404040 40404040   *0................. .            *
**UNKNOWN TRACE ENTRY TYPE**            LENGTH- 0054

          030650C8 0000C024 02000000 0000000C  C0068C37 00000000 0000C000 0C255C5C   *...H............................*
          5C5C5C40 40E3C3C1 D440D9E4 D5D5C9D5  C740405C 5C5C5C5C 3790ECDC 0CC5AC41   *...  TCAM RUNNING  .............*
          F0AC0E58 ED000C98 0CD01407 FE1C5C5C  5C5C5C4C 00404040 40404040 40404C40   *0................. .            *
**UNKNOWN TRACE ENTRY TYPE**            LENGTH- 0054                    BUFFER
  OUTBUF                                                                PREFIX
  LOG     03C650C8 00190078 08065620 00480000  C0065620 06562000 00150004 00250000   *...H............................*
  ENTRY   D9D4E740 D5E8C340 F440F0F8 4BF5F24B  F3F440C3 C8C1D940 E6C1E240 6140F240   *RMX NYC 4 C8.52.34 CHAR WAS . 2 *
          F0F84BF5 F34BF2F3 40C8C940 E3D640E3  C8C540E2 47404040 40404040 40404040   *08.53.23 HI TO THE S.          *
**UNKNOWN TRACE ENTRY TYPE**            LENGTH- 0054

          E6C9E3C3 C8C5C440 E3C5D9D4 C9D5C1D3  E2401517 17171717 17171717 17171717   *WITCHED TERMINALS ..............*
          17171737 CAB1E7C1 07011510 76E740D5  E8C340F5 40F0F84B F5F24BF4 F840C5E5   *......X......X NYC 5 08.52.48 EV*
          C5D9E840 6140E3D6 40C1D3D3 40E3C5D9  D4C9D5C1 00404040 40404040 40404040   *ERY . TO ALL TERMINA.          *
```

Figure 31. Log Segment Output

Using the Log Message Dump: Logging messages gives you an excellent data collection facility. Use it to provide a long-term backup for messages transmitted in your network for accounting. In Figure 32, the LOG macro is included in both the inmessage and outmessage subgroups. Since there are two destinations, there are three log entries for the message. It is difficult to tell the input message from the outgoing message, since the log entry is made before outgoing processing. However, the entries are sequential. You know that the first entry found for a message is the input message. TCAM makes an entry each time the message passes through the message handler. As with logged segments, entries are made in the order of processing, so there may be intermixed messages.

Note: *The unreadable data appearing in the message is the translation of the buffer prefix.*

```
<                          RM     X     ①X NYC 4 08.52.34 CHAR WAS / HI TO THE S
<                  WITCHED TERMINALS    ②X NYC 4 08.52.34 CHAR WAS / HI TO THE S
<                          RM     X      X NYC 5 08.52.48 EVERY / TO ALL TERMINA
<                  LS   HED  ERMINALS   ③X NYC 4  8.52.34 CHAR WAS / HI TO THE S
<                          RM     X      X NYC 5 08.52.48 EVERY / TO ALL TERMINA
```

Figure 32. Log Message Output

OBR/SDR File Dump

A TCAM I/O error-recording facility creates records on disk when terminal-related I/O errors occur. This recording, an extension of the OS Outboard Recorder (OBR) and Statistical Data Recorder (SDR) error-recording programs, can be used to diagnose line and terminal problems and thus increase line availability and efficiency.

TCAM ordinarily keeps a certain amount of information about line and terminal behavior. If you suspect that a specific line or terminal is malfunctioning, you can increase the amount of information kept about the suspected terminal with intensive-mode recording. The operator command ERRECORD creates temporary error (intensive mode) records for recoverable I/O errors occurring on a specified line or station. The format of the command is:

control characters	*operation*	*operand*
control chars	{MODIFY} {F}	{[procname.] id}, {jobname} INTENSE= {LINE,{grpname,rln / address},sense, / TERM,statname} {count / 15}

where

grpname is the name of the line group containing the line for which incident records are desired.

rln is the relative line number of the line within the group.

address is the machine address of the line.

statname is the name of the station for which incident records are desired.

sense is the type of intensive recording desired. You can select

BO	bus-out check
CR	command reject
DC	data check
EC	equipment check
IM	general intensive mode
IR	intervention required
LD	lost data
M2	leading graphics for 2740 Model 2 terminal
OR	overrun
TO	time-out
UE	unit exception

count is the decimal number of records for the incident type. The maximum and the default are 15.

If you do not use intensive mode, recoverable errors for a station or line are not recorded, but an internal counter is incremented by one. The command

```
OPID F JOBNAME,INTENSE=TERM,NYC,TO,12
```

entered from a secondary terminal specifies that you want an error recording on disk for the station named NYC in the job JOBNAME whenever there is a time-out, up to a maximum of 12 records.

The OS utility, IFCEREP0, retrieves the error recordings from disk, dumps them, and formats them. The recordings are maintained in the SYS1.LOGREC data set. The sample JCL below formats the data set, prints it (both individual and summary records for each terminal), and scratches the OBR/SDR file.

```
//OBRSDR JOB  MSGLEVEL=1
//STEP   EXEC PGM=IFCEREP0,PARM=(MCOS,PS)
//SERLOG DD   DSNAME=SYS1.LOGREC,DISP=OLD,VOL=SER=SYSRES,*
//            UNIT=2314
//EREPPT DD   UNIT=00E
/*
```

The TERMN= option in the PARM= parameter allows you to dump your SYS1.LOGREC data by terminal name. See the *OS Utilities* publication for a complete description of the PARM= parameter and the control statements.

You should consider several things when running IFCEREP0. First, since this data set is not reusable and does not wrap around, it will have to be reinitialized when it fills up or on some periodic schedule. IFCEREP0 reinitializes the data set as it dumps. Second, you should code the parameter PS to ensure that all records and summaries are written. PS is the default. You will generally be more interested in the summary records than in the individual records. Finally, if you use SYSOUT=A rather than allocating directly to the output device, code a SPACE= parameter, since the OBR/SDR file is fairly large.

Dump the OBR/SDR file whenever you have a problem that seems to be caused by a malfunctioning line or station, such as lost data or lost line. You should also dump the file at the end of the day, to keep yourself aware of the hardware status of your network. In addition, once the SYS1.LOGREC data set is full, it is very time-consuming to dump.

OBR/SDR Table: TCAM I/O error recording writes certain terminal-related I/O errors on disk. This recording, an extension of the OS Outboard Recorder (OBR) and Statistical Data Recorder (SDR) programs, reduces the time that the TCAM system is inoperative by providing useful information for diagnosing line and terminal problems.

Four types of I/O error records are written in the data set:

1. *Permanent error record.* Written for each permanent I/O error. A permanent I/O error is either an unrecoverable error (an undefined, unanticipated I/O error for which TCAM provides no error-recovery procedure), or an I/O error for which TCAM provides an error-recovery procedure, but which TCAM has tried several times to correct and failed each time.
2. *Temporary error record.* Written whenever an error occurs for a particular line or station specified by an ERRECORD operator command, if TCAM successfully recovers from the error.
3. *Counter overflow record.* Written when either the SIO counter (the number of start I/O commands issued) or the temporary error counter for a particular terminal-table entry is about to overflow.
4. *End-of-day record.* Written for each station and line in the line group that has a terminal entry and a nonzero temporary error counter.

The OS utility program IFCEREP0 prints a formatted dump of these error records from the data set SYS1.LOGREC. The following sections discuss the output that you can use to determine problems.

I/O Device (Outboard) Records: TCAM produces and stores these records for permanent device errors. TCAM terminal statistics and errors are outboard records. Two types of recording mode are possible for an outboard record:

1. *Unrecoverable.* A record of a permanent I/O error. See the description of a permanent error above.
2. *Intensified.* A temporary error record which is described above.

Figures 33 and 34 are examples of an unrecoverable and an intensified error record, respectively. Each has the same formatted data. The meaning of each field follows. The command issued to get the intensified I/O error record was:

```
f linkgo,INTENSE=TERM,NYC,TO,15
```

```
PROGRAM SECTION:

TCAM OUTBOARD DATA EDITING AND PRINTING SECTION

MODEL-UNIVERSAL

--- RECORD ENTRY SOURCE - OBR ---        TYPE - OUTBOARD

CHANNEL/UNIT ADDRESS 0018                DEVICE TYPE 2702

                   COMMUNICATION ADAPTER TYPE IBM TERM 1
                   TERMINAL TYPE IBM 2740

PROGRAM IDENTITY LINKGO

       DAY   YEAR                                HH    MM    SS TH
DATE - 210    71                        TIME  -  00    16    09.35

                 COMMAND        DATA
                   CODE (CC)   ADDRESS (DA)  FLAGS (FL)              COUNT(CT)
FIRST CCW           01          03492D         60          00          0003
FAILING CCW         01          03492D         60          00          0003

                               COMMAND       UNIT       CHANNEL
                    KEY (K)   ADDRESS (CA)  STATUS (US)  STATUS (CS)  COUNT (CT)
CSW                  00        072118         0E           00          0003

   UNIT STATUS                              CHANNEL STATUS
     ATTENTION             0                  PRGM-CTLD IRPT      0
     STATUS MODIFIER       0                  INCORRECT LENGTH    0
     CONTROL UNIT END      0                  PROGRAM CHECK       0
     BUSY                  0                  PROTECTION CHECK    0
     CHANNEL END           1                  CHAN DATA CHECK     0
     DEVICE END            1                  CHAN CTL CHECK      0
     UNIT CHECK            1                  I/F CTL CHECK       0
     UNIT EXCEPTION        0                  CHAINING CHECK      0

SENSE BYTE DATA
 INITIAL FAILURE                              FINAL RETRY
   BYTE 0         01000000                  BYTE 0      01000000

     CMND REJ     0                           CMND REJ     0
     INTV REQD    1                           INTV REQD    1
     BUS O CHK    0                           BUS O CHK    0
     EQUIP CHK    0                           EQUIP CHK    0
     DATA CHK     0                           DATA CHK     0
     OVERRUN      0                           OVERRUN      0
     LOST DATA    0                           LOST DATA    0
     TIME-OUT     0                           TIME-OUT     0

TERMINAL NAME NYC                           RECORDING MODE *UNRECOVERABLE*

SIO CNTR 00006                              TEMPORARY ERR CNTR 000

MASK 00000000                               INITIAL SELECTION 0
```

Figure 33. An Unrecoverable I/O Error Record

```
PROGRAM SECTION:

TCAM OUTBOARD DATA EDITING AND PRINTING SECTION

MODEL-UNIVERSAL

--- RECORD ENTRY SOURCE - OBR ---         TYPE - OUTBOARD

CHANNEL/UNIT ADDRESS 002C                 DEVICE TYPE 2703

                COMMUNICATION ADAPTER TYPE IBM TERM 1
                TERMINAL TYPE IBM 2740

PROGRAM IDENTITY LINKGO

       DAY   YEAR                                 HH    MM    SS TH
DATE - 211    71                          TIME - 07    17    55.13

                 COMMAND      DATA
                   CODE      ADDRESS     FLAGS                COUNT
FIRST CCW           01       03492D       60        00         0003
FAILING CCW         02       065C01       80        00         0002

                            COMMAND      UNIT     CHANNEL
                    KEY     ADDRESS     STATUS     STATUS      COUNT
CSW                 00      065190        0E         40         0001

  UNIT STATUS                            CHANNEL STATUS
    ATTENTION              0               PRGM-CTLD IRPT       0
    STATUS MODIFIER        0               INCORRECT LENGTH     1
    CONTROL UNIT END       0               PROGRAM CHECK        0
    BUSY                   0               PROTECTION CHECK     0
    CHANNEL END            1               CHAN DATA CHECK      0
    DEVICE END             1               CHAN CTL CHECK       0
    UNIT CHECK             1               I/F CTL CHECK        0
    UNIT EXCEPTION         0               CHAINING CHECK       0

SENSE BYTE DATA
 INITIAL FAILURE                           FINAL RETRY
   BYTE 0       00000001                  BYTE 0      00000000

    CMND REJ     0                         CMND REJ     0
    INTV REQD    0                         INTV REQD    0
    BUS O CHK    0                         BUS O CHK    0
    EQUIP CHK    0                         EQUIP CHK    0
    DATA CHK     0                         DATA CHK     0
    OVERRUN      0                         OVERRUN      0
    LOST DATA    0                         LOST DATA    0
    TIME-OUT     1                         TIME-OUT     0

TERMINAL NAME NYC                         RECORDING MODE *INTENSIFIED*

SIO CNTR 00222                            TEMPORARY ERR CNTR 001

MASK 00011110                             INITIAL SELECTION 0
```

Figure 34. An Intensified I/O Error Record

Program section: The program section that is generating the printout.

Model: The IBM System/360 model for which the printout is applicable. In this example, UNIVERSAL indicates that the printout is applicable to models 40, 50, 65, 67, 75, 85, 91, and 195.

Record entry source: The error environment or recovery management program that generated the record in the SYS1.LOGREC data set.

Type: The type of printout.

Channel/Unit address: The hardware address of the line on which the error occurred or on which the terminal in error is located.

Device: The transmission control unit being used.

Program identity: The name of the job that was active when the error occurred.

Date time: The date and time at which the error occurred. The day is the Julian calendar date and the hour is in continental (24-hour) time.

First CCW: The first executed channel command word (CCW) in the channel program.

Failing CCW: The channel command word that failed to execute.

CSW: The channel status word when the failure occurred.

Sense byte data: For a description of the contents of the sense byte, see the component description publication for the transmission control unit that you are using.

Terminal name: The name of the terminal on which the error occurred. This is the name you assigned in the TERMINAL macro.

Recording mode: The type of error recording. It is either unrecoverable or intensified.

SIO cntr: The start I/O counter, a count of the number of EXCPs issued for the line before the error occurred. It is reset to zero each time an entry is made in the SYS1.LOGREC data set.

Temporary err cntr: A count of the number of temporary I/O errors that occurred for the terminal since the last record was made. It is reset to zero each time an entry is made in the SYS1.LOGREC data set.

Mask: An eight-bit field that is used if you issue the ERRECORD operator command. The first four bits indicate the type of error for which the terminal was intensified.

Bits	*Meaning*
0001	time-out
0010	lost data
0011	overrun

0100	data check
0101	equipment check
0110	bus-out check
0111	intervention required
1000	command reject
1001	unit exception
1010	leading graphics for 2740 Model 2 terminals

The last four bits indicate the number of error recordings yet to take place. The original value is specified in the command and decremented by one for each recording made.

Initial selection: Set to 1 if an error occurs on the first attempt to contact the control unit.

Examine this error record after you suspect that you have trouble on a line or terminal. You can determine the reliability of your line by comparing the number of start I/Os to the number of temporary errors. Once you know there is trouble on the line, the CCW helps you determine which activity was failing to execute. This can lead you to the hardware feature that is failing.

Summary of Outboard Records: You can find valuable information in the summary of TCAM I/O outboard records for each line in your network. By examining this output, you can determine the reliability of the line and of terminals on the line. If you see that a line is continually failing with permanent errors, then look at the individual outboard record for the terminal to pinpoint the failure. Examine the summary output for each of your lines daily, to remain aware of the status of your network. Take a summary listing of each line as soon as your TCAM system is operating to your satisfaction, and use it as a base for examining the line reliability. By comparing each day's summary to this base summary, you can see if your line and terminal reliability has decreased.

Figure 35 shows a summary output for the line address 011 on a 2702 control unit. The ratio of unrecoverable errors to start I/Os (30 to 165) is relatively low, indicating that the line is reasonably reliable. However, you should keep a watchful eye on this line, since the majority of the permanent errors occurred on the same operation, time-out on read next text.

If you have more than one terminal on a line, the summary output gives information about each terminal, with a total summary of the line. Figure 36 shows the output for line address 022 on a 2703 control unit with two stations located on the line. There is no question about the reliability of this line or the terminals on the line, since the unrecoverable error to start I/O ratio is extremely low (2 to 128 for the line, 1 to 44 for the terminal CHAR, and 1 to 84 for the terminal WAS).

Figure 37 shows an entry for a terminal that was placed in intensified mode by the ERRECORD operator command. There were two unrecoverable errors of 380 start I/Os before intensification began. After the command was issued, 265 start I/Os occurred and 2 temporary errors for which the terminal was intensified occurred. The error-to-start I/O ratio is extremely low in both cases, indicating a very reliable line and terminal.

End-of-Day Recording: You get an end-of-day recording for each station that was active and had errors (nonzero temporary error counter). It is *not* a summary of the station activity; it simply indicates what occurred on the station since the last error record entry was made in SYS1.LOGREC. Figure 38 shows an end-of-day

DAY YEAR DAY YEAR
OUT BOARD DATE RANGE-204 71 TO 204 71
SUMMARY OF TCAM I/O OUTBOARD RECORDS FOR CUA/LINE 0011 DEVICE TYPE 2702
TOTAL NUMBER OF RECORDS 0030

TOTAL NUMBER OF SIO'S 0165
TOTAL NUMBER OF TEMPORARY FAILURES 0002
TOTAL NUMBER OF UNRECOVERABLE (UNREC) ERRORS 0030
TOTAL NUMBER OF INTENSIVE MODE (INTEN) ERRORS 0000

ERROR TYPES		TOTL SIOS	TOTL TEMP ERR	TOTL PERM ERR	LOST DATA	COMD REJ	COMD REJ	UNIT XCPT	TIME OUT	TIME OUT	TIME OUT	TIME OUT	INTR REQ	OVER RUN	BUSO CHK	BUSO CHK	BUSO CHK	DATA CHK	DATA CHK	DATA CHK	EQPT CHK
CONDITION					ALL	INIT SEL	OTHR	ALL	PRE PARE	READ NTXT	DIAL	OTHR	ALL	ALL	WRIT	DIAL	OTHR	WRIT	READ	OTHER	ALL
2740-II GRAPHIC RESP					TERM ELEC ERR		TERM I/O ERR		REC PARITY ERR		TRANS PARITY ERR										
TERMINAL NAME	RECORD MODE																				
RAL	UNREC	0165	0002	0030	0001					0028			0001								
	INTEN																				

Note: In this and the following two examples, the solid lines have been added for clarity.
They are not part of the output from IFCEREPO.

Figure 35. A Summary Outboard Record

DAY YEAR DAY YEAR
OUT BOARD DATE RANGE-211 71 TO 211 71
SUMMARY OF TCAM I/O OUTBOARD RECORDS FOR CUA/LINE 0022 DEVICE TYPE 2703
TOTAL NUMBER OF RECORDS 0002

TOTAL NUMBER OF SIO'S 0128
TOTAL NUMBER OF TEMPORARY FAILURES 0002
TOTAL NUMBER OF UNRECOVERABLE (UNREC) ERRORS 0002
TOTAL NUMBER OF INTENSIVE MODE (INTEN) ERRORS 0000

ERROR TYPES		TOTL SIOS	TOTL TEMP ERR	TOTL PERM ERR	LOST DATA	COMD REJ	COMD REJ	UNIT XCPT	TIME OUT	TIME OUT	TIME OUT	TIME OUT	INTR REQ	OVER RUN	BUSO CHK	BUSO CHK	BUSO CHK	DATA CHK	DATA CHK	DATA CHK	EQPT CHK
CONDITION					ALL	INIT SEL	OTHR	ALL	PRE PARE	READ NTXT	DIAL	OTHR	ALL	ALL	WRIT	DIAL	OTHR	WRIT	READ	OTHR	ALL
2740-11 GRAPHIC RESP					TERM ELEC ERR		TERM I/O ERR		REC PARITY ERR		TRANS PARITY ERR										
TERMINAL NAME	RECORD MODE																				
CHAR	UNREC	0044	0001	0001						0001											
	INTEN																				
WAS	UNREC	0084	0001	0001						0001											
	INTEN																				

Figure 36. A Summary Outboard Record for an Unrecoverable I/O Error

entry. The fields have the same meaning as those in the individual outboard records. However, the program identity is not available, since you have removed TCAM from your system.

The outboard records (OBR) can be a valuable tool to determine problems, because it keeps you aware of line and terminal status. The statistical data records

```
                    DAY YEAR    DAY YEAR
OUT BOARD DATE RANGE-211  71  TO  211  71
SUMMARY OF TCAM I/O OUTBOARD RECORDS FOR CUA/LINE  0020          DEVICE TYPE 2703
TOTAL NUMBER OF RECORDS  0004

    TOTAL NUMBER OF SIO'S                   0645                 TOTAL NUMBER OF UNRECOVERABLE (UNREC) ERRORS  0002
    TOTAL NUMBER OF TEMPORARY FAILURES  0002                     TOTAL NUMBER OF INTENSIVE MODE (INTEN) ERRORS  0002
```

ERROR TYPES		TOTL SIOS	TOTL TEMP ERR	TOTL PERM ERR	LOST DATA	COMD REJ	COMD REJ	UNIT XCPT	TIME OUT	TIME OUT	TIME OUT	TIME OUT	INTR REQ	OVER RUN	BUSO CHK	BUSO CHK	BUSO CHK	DATA CHK	DATA CHK	DATA CHK	EQPT CHK
CONDITION					ALL	INIT SEL	OTHR	ALL	PRE PARE	READ NTXT	DIAL	OTHR	ALL	ALL	WRIT	DIAL	OTHR	WRIT	READ	OTHER	ALL
2740-II GRAPHIC RESP					TERM ELEC ERR		TERM I/O ERR		REC PARITY ERR		TRANS PARITY ERR										
TERMINAL NAME	RECORD MODE																				
NYC	UNREC	0380		0002						0002											
	INTEN	0265	0002																		

Figure 37. A Summary Outboard Record for an Intensified I/O Error

```
TCAM OUTBOARD DATA EDITING AND PRINTING SECTION

MODEL-UNIVERSAL

--- RECORD ENTRY SOURCE - OBR ---            TYPE - OUTBOARD

CHANNEL/UNIT ADDRESS 001E                    DEVICE TYPE 2701

                COMMUNICATION ADAPTER TYPE IBM TERM 1
                TERMINAL TYPE IBM 2740

PROGRAM IDENTITY NOT AVAILABLE

        DAY    YEAR                                  HH    MM    SS TH
DATE  - 196     71                           TIME  - 00    13    53.13

TERMINAL NAME BBB                            RECORDING MODE  *END OF DAY*

SIO CNTR 00025                               TEMPORARY ERR CNTR 001

MASK 00000000                                INITIAL SELECTION 0
```

Figure 38. An End of Day Record

(SDR) are not as valuable to you in a TCAM environment, since they contain temporary error records for devices other than lines or terminals, such as tapes and disks (not discussed in this manual).

TCAM Libraries Dump

You should always have a listing of your TCAM and system base and PTF level available. You *must* have this listing for all problems that require IBM assistance. An OS service aid program, IMAPTFLS, prints this listing. The following sample JCL lists all members in the named libraries. Applied PTFs and local fixes are listed with the associated module.

```
//JOBLIST     JOB   MSGLEVEL=1
//STEP        EXEC  PGM=IMAPTFLS
//LISTREST    DD    DUMMY
```

```
//SVCDD       DD   DSNAME=SYS1.SVCLIB,DISP=SHR
//MACDD       DD   DSNAME=SYS1.MACLIB,DISP=SHR
//LINKDD      DD   DSNAME=SYS1.LINKLIB,DISP=SHR
//TCAMDD      DD   DSNAME=SYS1.TELCMLIB,DISP=SHR
//SYSPRINT    DD   SYSOUT=A
/*
```

All names on the DD statements are optional except LISTREST. See *OS Service Aids* for a complete description of the IMAPTFLS programs.

Service Aids

You can use the four optional TCAM service aids for diagnosing problems. These are the line I/O interrupt trace table, the subtask trace table, the buffer trace, and the cross-reference table. These trace facilities record valuable data about TCAM, and in the case of a malfunction, they can be very useful during the testing and diagnostic stage. You should include them in your system. See the following sections, and the *TCAM Programmer's Guide,* to learn how to include and use these facilities in your system.

Dumping TCAM Trace Tables

A TCAM routine, named COMWRITE, writes the I/O interrupt trace, the subtask trace, and the buffer trace tables onto a sequential data set named COMWRITE. To use this routine, you must include either at assembly time or at INTRO execution time the INTRO operand COMWRTE=YES. COMWRITE is required, in order to see a total picture of the system activity before and after TP failures, since it provides a complete history of system activity.

For the I/O and subtask traces, if you did not specify COMWRTE=YES on the INTRO macro, the table in main storage wraps around after it is filled. So, with COMWRTE=YES, you can have a smaller trace table, requiring less main storage, with little fear of losing entries.

Each trace is most commonly written to tape. There are three reasons for using a tape as the trace data set. First, you can dump the trace selectively by time. Second, you can have a large trace table. If your data set is on a direct access device, you must be sure that 1/2 n(16)+16, where *n* is the total number of entries in your trace table, is less than the byte capacity of one track. A tape supports large records; therefore, there is little worry about the trace-table size. Third, once your data set on disk is full, the data set wraps around and you are apt to lose trace-table entries as they are overlaid. Since you must have a very large data set to avoid wrapping, it is more economical to have your data set on tape.

Printing Trace Table Dumps

If the COMWRITE routine has been used to dump the trace tables to secondary storage, the utility program to format and print these trace tables is COMEDIT (IEDQXB). An example of the JCL to print the trace from an unlabeled tape follows; the data set named COMWRITE on the SYSUT1 statement is the name of the DD statement in the MCP execution deck that created the COMWRITE data set.

```
//PRINT        JOB  MSGLEVEL=1
//STEP         EXEC PGM=IEDQXB,PARM='xxxx'
//SYSPRINT     DD   SYSOUT=A
//SYSUT1       DD   DSN=COMWRITE,UNIT=2400,DISP=OLD,*
//                  LABEL=(,NL),VOL=SER=DUMMY
/*
```

Replace *xxxx* in the EXEC statement with IOTR to print the I/O trace, with STCB to print the subtask trace, and with BUFF to print the buffer trace. If you omit the PARM= parameter, all three traces are printed.

Another parameter on the EXEC statement prints the trace-table entries starting at a specified time. For example, if a problem occurs after 3:00 on a certain day, you can print the trace from 3:00 on. The parameter is

```
BLOCK=hhmmddd
```

where *hh* represents the hours in continental (24-hour) time, *mm* is the minutes (60 minutes to an hour), and *ddd* is the Julian date (January 1 is 001, etc.). The following EXEC statement prints the subtask-trace table entries that occur after 2:15 p.m. on January 8:

```
//EXEC PGM=IEDQXB,PARM='STCB,BLOCK=1415008'
```

Always include a small trace table (relative to the number of lines in your network) in your MCP.

Line I/O Interrupt Trace Table

This TCAM service aid sequentially records the I/O interrupts that occur on a specified line. When an I/O interrupt occurs on a line for which you requested line I/O trace, TCAM stores information about the interrupt, including the channel status word (CSW) and channel command word (CCW), as an entry in the line I/O interrupt trace table. However, TCAM does not record interrupts resulting from retries by error-recovery procedures.

Activating the Trace: Whether this trace is available in main storage depends on how you design your MCP. To include it, code on the INTRO macro instruction the operand TRACE=n, where *n* is an integer from 1 to 65535 that specifies the number of entries in the table. The default, TRACE=0, excludes the table. You can include the operand at assembly time, or at INTRO execution time in response to the message

```
IED002A SPECIFY TCAM PARAMETERS
```

that you receive only if you omit one of the following INTRO operands at assembly time:

```
STARTUP=, LNUNITS=, KEYLEN=, and, if DISK=YES, CPB=
```

The response keyword is T=n or TRACE=n.

Start and stop the I/O interrupt trace for a line with the GOTRACE and NOTRACE operator commands, respectively. Their formats are:

GOTRACE:

control characters	*operation*	*operand*
control chars	{MODIFY F}	{[procname.]id jobname},TRACE={grpname,rln address},ON

NOTRACE:

control characters	*operation*	*operand*
control chars	{MODIFY F}	{[procname.]id jobname},TRACE={grpname,rln address},OFF

where

grpname is the name of the line group and is identical to the DDNAME= operand of the DCB macro instruction for the line group for which you enter the command.

rln is the relative line number of the line within the line group.

address is the hardware address of the line and is identical to the UNIT= operand of the DD statement for the line for which you enter the command.

Example: F GOTCAM,TRACE=022,ON is the command from the system console to start the I/O trace on the line whose address is 022 in the job named GOTCAM. The command F GOTCAM,TRACE=022,OFF stops the trace on line 022.

Start the line traces one at a time. You cannot enter multiple addresses in the trace parameter.

The trace table resides in main storage allocated to the MCP and, therefore, to get a copy of the table, you must dump your MCP. See COMEDIT in the *TCAM Programmer's Guide.*

If you wish to dump the I/O trace table to a sequential data set to provide a history of I/O activity, you must activate the COMWRITE routine for the I/O trace table dump by issuing the DEBUG operator command.

control characters	*operation*	*operand*
control chars	{MODIFY {F	{[procname.]id},DEBUG=L,IEDQFE20 {jobname}

Note: *COMWRTE=YES must have been specified either at assembly time or INTRO execution time.*

This loads (L) the dump routine for I/O trace. If you want to deactivate the routine, replace the L with D; otherwise, the command is the same. The routine prints half of the table at a time to the sequential data set while the other half in main storage is being filled. Therefore, your most current entries in the trace table are still in main storage.

Example: Printing the line I/O trace table when COMWRITE is used.

```
//PRINT       JOB  MSGLEVEL=1
//STEP        EXEC PGM=IEDQXB,PARM='IOTR'
//SYSPRINT    DD   SYSOUT=A
//SYSUT1      DD   DSN=COMWRITE,UNIT=2400,DISP=OLD,*
//                 LABEL=( ,NL),VOL=SER=DUMMY
/*
```

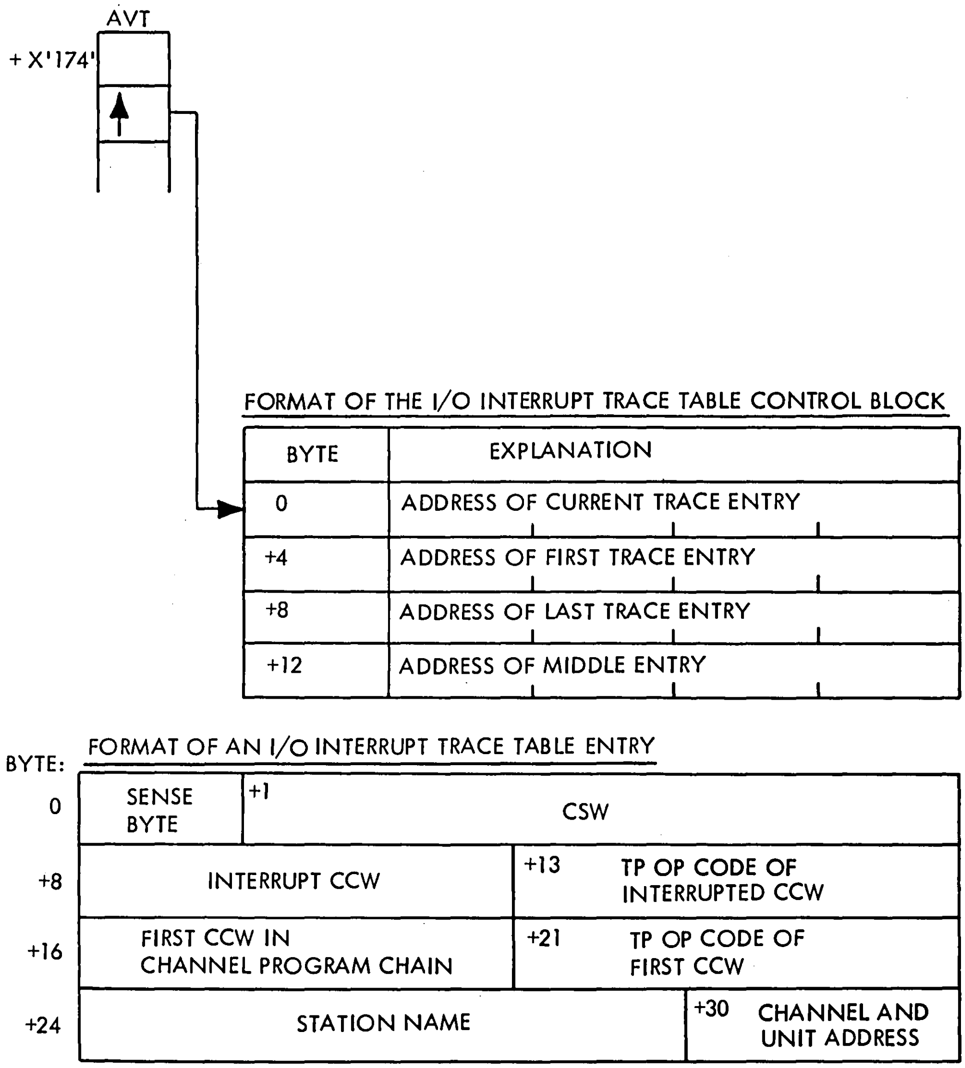
The I/O trace table is also printed if no PARM= parameter is specified on the EXEC statement.

The line I/O trace is most important for all line-oriented problems. Start the trace for a particular line as soon as you detect trouble on the line (lost or bad data or lost line), and then dump it.

Using the Line I/O Interrupt Trace: The TCAM line I/O interrupt trace table records I/O interrupts occurring on specified lines. Interrupts that result from retries by TCAM error-recovery procedures are not recorded.

Use this table to determine problems at a station. By examining the first channel command word (CCW), the interrupt CCW, and the channel status word (CSW), you can determine which channel program was executing on the line, and possibly determine whether the station or data set is in error.

The Table in Main Storage: If you specify a nonzero value for TRACE= in the INTRO macro, the trace table is located in main storage; its address is at AVT+X'174'. Figure 39 shows the line I/O interrupt trace table format.



BYTE	EXPLANATION
0	ADDRESS OF CURRENT TRACE ENTRY
+4	ADDRESS OF FIRST TRACE ENTRY
+8	ADDRESS OF LAST TRACE ENTRY
+12	ADDRESS OF MIDDLE ENTRY

Figure 39. Line I/O Interrupt Trace Table Format

The meaning of each field in the entry follows.

Sense byte: For a description of this byte, see the component description publication for the transmission control unit that you are using.

CSW: The channel status word. The entry contains the last seven bytes of the CSW, which has the following format.

COMMAND ADDRESS	STATUS	COUNT

8 32 48 63

Command address: Bits 8 to 31 form an address that is eight bytes higher than the address of the last CCW used.

Status: Bits 32 to 47 identify the reasons why the CSW was stored.

Bits 32 to 39 are obtained over the I/O interface and are set by the device or the control unit.

Bits 40 to 47 are set by the channel for conditions in the subchannel.

Each of the 16 bits indicates a condition.

Bit	*Condition*	*Bit*	*Condition*
32	attention	40	program-controlled interruption
33	status modifier	41	incorrect length
34	control unit end	42	program check
35	busy	43	protection check
36	channel end	44	channel data check
37	device end	45	channel control check
38	unit check	46	interface control check
39	unit exception	47	chaining check

Count: Bits 48 to 63 form the residual count of the last CCW used.

CCW: The channel command word. It is 64 bits (8 bytes) and has the following format.

COMMAND CODE	DATA ADDRESS	FLAGS	COUNT

0 8 32 37 48 63

Command code: Bits 0 to 7 specify what is to be done. An X indicates that the bit position is ignored; an M is a modifier bit.

Bits	*Meaning*
XXXX0000	invalid
MMMM0100	sense
XXXX1000	transfer in channel (TIC)
MMMM1100	read backward
MMMMMM01	write
MMMMMM10	read
MMMMMM11	control

Data address: Bits 8 to 31 specify the address of a byte in main storage; it is the first location referred to in the area designated by the CCW.

Flags:

Bit 32: Chain data (CD) flag. When on, specifies data chaining, causing the storage area designated in the next CCW to be used with the current command.

Bit 33: Chain command (CC) flag. When on and the CD flag is off, it specifies chaining of commands, so that the command specified in the next CCW is initiated when the current operation completes normally.

Bit 34: Suppress length indication (SLI) flag. It specifies whether an incorrect length is indicated to the program. When this bit is on and the CD flag is off in the last CCW used, the incorrect length indication is suppressed. When both the CC and SLI flags are on, command chaining takes place regardless of the presence of an incorrect length condition.

Bit 35: Skip (SKIP) flag. It specifies that the transfer of information to storage during a read, read backward, or sense operation is to be suppressed.

Bit 36: Program controlled interruption (PCI) flag. If on, the channel generates an interrupt when the CCW takes control of the channel.

Count: Bits 48 to 63 specify the number of byte locations in the storage area designated by the CCW.

See *Principles of Operation,* GA22-6821, for a detailed discussion of the CCW and CSW.

Teleprocessing Operation (TP OP) Code: A TP OP code with an even-numbered value represents a text or nontext CCW for which an interrupt is anticipated. A TP OP code with an odd-numbered value represents a CCW for which no interrupt is anticipated. TP OP codes are shown in Figure 40.

Name	Value	Description
TPWREOT	X'01'	Write EOT for selection
TPOPEN	X'02'	Open
TPWRPOLL	X'03'	Write Polling Characters
TPRDRESP	X'04'	Read Response to Polling
TPWRPAD	X'05'	Write pad characters
TPENABLE	X'06'	Enable on Dial Line
TPWRAD	X'07'	Write Addressing Sequence
TPRDRSPD	X'08'	Read Response to Addressing
TPWREOA	X'09'	Write EOA Sequence
TPRDRPEB	X'0A'	Read Response to EOB/ETB
TPWRCPU	X'0B'	Write CPU ID
TPRDENQ	X'0C'	Read ENQ
TPWRENQ	X'0D'	Write ENQ
TPRSPENQ	X'0E'	Read Response to ENQ
TPWRDLET	X'0F'	Write DLE EOT
TPRDID	X'10'	Read ID (TSO)
TPNULL	X'11'	Non-Read Write CCWs for which no Interrupt is anticipated
TPBREAK	X'12'	Write BREAK (TSO)
TPENQAD	X'13'	Write ENQ after Selection Response
TPRDLC	X'14'	Read LCOUT
TPWRACK	X'15'	Write Response Before Text
TPWRAKNK	X'16'	Write Response
TPWRTONE	X'17'	Write Tone (WTTA BSC)
TPRDIDNQ	X'18'	BSC Read ID ENQ
TPRDIDAK	X'1A'	BSC Read ID ACK
TPRESET	X'1C'	Abort for Send/Receive
TPTWXID	X'1E'	Read TWX ID
TPBUFEOT	X'20'	Buffered Terminal Reset after Block
TPCLOSE	X'22'	Close SDR Recording
TPRSPAD	X'24'	Write Reset after Selection
TPRDSKIP	X'51'	Read Skip Loop
TPWRIDLE	X'53'	Write Idles Loop
TPDLESTX	X'57'	Write DLE STX
TPDLEETX	X'59'	Write DLE ETB (ETX)
TPENQRSP	X'5B'	Write ENQ in Response to Text
TPTEXT	X'FF'	Text CCW

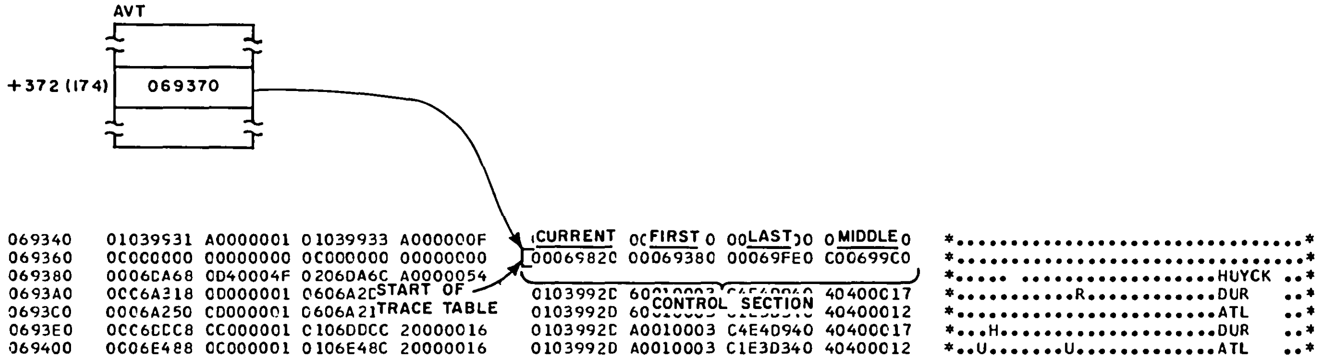
Figure 40. TCAM TP OP Codes

Station name: The name of the terminal on which the interrupt occurs.

Channel and unit address: The channel and unit address of the connected station when the interrupt occurs.

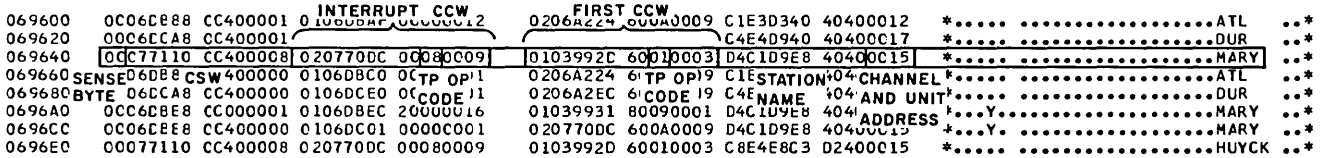
Part 1 of Figure 41 illustrates the four-word control section for the trace table generated when TRACE=100 was coded in the INTRO macro.

Part 2 of Figure 41 shows an interrupt on a 2740 Model 2 station named MARY.



```
069340   01039931 A0000001 01039933 A000000F                                        *..............................*
069360   0C000000 00000000 0C000000 00000000   00069820 00069380 00069FE0 C00699C0  *..............................*
069380   0006DA68 0D40004F 0206DA6C A0000054                                        *..... ..................HUYCK ..*
0693A0   0CC6A318 0D000001 0606A2D             0103992D 60                40400C17  *..........R............DUR   ..*
0693C0   0006A250 CD000001 0606A21             0103992D 60                40400012  *........................ATL   ..*
0693E0   0CC6DCC8 CC000001 C106DDCC 20000016   0103992D A0010003 C4E4D940 40400C17  *...H....................DUR   ..*
069400   0C06E488 0C000001 0106E48C 20000016   0103992D A0010003 C1E3D340 40400012  *..U.......U.............ATL   ..*
```

Figure 41. The Line I/O Interrupt Trace Table in Main Storage (Part 1 of 2)



```
069600   0C06C888 CC400001 0                 2   0206                0009 C1E3D340 40400012  *..... ..................ATL   ..*
069620   0006CCA8 0C400001                                                C4E4D940 40400C17  *..... ..................DUR   ..*
069640   0CC77110 CC400008 020770DC 00080009     0103992C 60010003 D4C1D9E8 40400C15  *..... ..................MARY  ..*
069660        06DB8        400000 0106DBC0 0C         1   0206A224 6         9 C1E         04  *..... ..................ATL   ..*
069680        06CCA8 CC400000 0106DCE0 0C         1   0206A2EC 6         9 C4E         04  *..... ..................DUR   ..*
0696A0   0CC6DBE8 CC000001 0106DBEC 20000016   01039931 80090001 D4C1D9E8 404        *...Y....................MARY  ..*
0696CC   0C06CEE8 CC400000 0106DC01 0000C001   020770DC 600A0009 D4C1D9E8 404        *...Y. ..................MARY  ..*
0696E0   00077110 CC400008 020770DC 00080009   0103992D 60010003 C8E4E8C3 D2400C15  *..... ..................HUYCK ..*
```

Figure 41. The Line I/O Interrupt Trace Table in Main Storage (Part 2 of 2)

1. There is no sense information for the 2702 control unit.
2. The CSW
   - command address is 77110. Therefore, the last CCW used storage at 77108.
   - status is 0C40. This is channel end, device end, and incorrect length.
   - count is 8. The residual count is 8 for the last CCW.
3. The interrupt CCW
   - command code is 02. This is a read.
   - data address is 0770DC.
   - has no flags set.
   - TP OP code is 08. This is a read response.
   - count is 9.

   Therefore, the channel program was interrupted when reading the 9-byte response found at storage location 0770DC.
4. The first CCW
   - command code is 01. This is a write.
   - data address is 03992D.
   - flags specify chain command (CC) and suppress length indication (SLI).
   - TP OP code is 01. This is a write EOT for selection.
   - count is 3.

   Therefore, the first channel command word in the channel program was to write a three-byte EOT sequence for selection from storage location 03992D. This is a write initial channel program. The *TCAM PLM*, shows channel programs for terminal operations.
5. The station name is MARY. This is a six-byte field padded with blanks.
6. The terminal is on line 015.

The Formatted Table: If you specify COMWRTE=YES in the INTRO macro, you can get a formatted listing of the trace table. Only half of the table is written at a time on the COMWRITE data set. Use the IEDQXB utility to print the formatted trace table. If you use the utility, remember that the most current entries in the trace table are still in main storage.

Figure 42 is an example of the formatted output. SEQUENCE is a sequential count of the number of I/O trace tables printed. If a number is skipped, records have been lost. Prevent this loss by enlarging the size of your trace table.

Each table shows the time and date it is placed on the data set. Use the time to trace the activity on a line just before it fails, since you know when you lost the line.

Subtask Trace Table

A TCAM service aid, the subtask trace, records the flow of all dispatched elements. It shows where elements go in the TCAM system and which subtasks work on them.

To use the subtask trace table, you must first understand the data flow as controlled by the TCAM dispatcher, and know how to use IEDQFE10, the subtask trace dump module. You must also understand the Method of Operation charts in the TCAM PLM.

Activating the Trace: Whether this trace is available in main storage depends on how you design your MCP. To include it, code on the INTRO macro instruction the operand DTRACE=n, where *n* is an integer from 1 to 65535 that specifies the number of entries in the table. To format and print the table with the COMWRITE routine and the IEDQXB utility, *n* must be between 4 and 65535. The default, DTRACE=0, specifies that there is to be no subtask trace. Include this operand at assembly time, or at INTRO execution time in response to the message

```
**LINE I/O TRACE**  SEQUENCE- 00000037                                                  DATE- 71.211   TIME- 07.32.29
 SENSE     CSW           INTERRUPT                FIRST          TERMINAL     LINE  STATUS  INTRPT TP   FIRST TP
                            CCW                    CCW             NAME       ADDR           OP CODE    OP CODE

00 0900070C400001   0206904180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 0640000C400008   0206408C00800009   010348E560010003   C2D6E2404040   0069           08          01          BOS
00 05F7F00C400001   0206910180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 0643180C400000   0106885000000013   0103492DA0010003   C3C8C1D94040   0022           00          01          CHAR
00 05F7F00C400001   0206904180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 05F7F00C400001   0206910180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 05F7F00C400001   0206904180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 05F7F00C400001   0206910180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 05F7F00C400001   0206904180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 05F7F00C400001   0206910180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 05F7F00C400001   0206904180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 068C480C000030   01068C4C20000053   0103492DA0010003   C3C8C1D94040   0022           00          01          CHAR
00 05F7F00C400001   0206910180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 05F7F00C400001   0206904180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 05F7F00C400001   0206910180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 068C480C400001   01068C6F00000030   02064304600A0009   C3C8C1D94040   0022           00          0A          CHAR
00 068C480C400000   01068C9E00000001   02064304600A0009   C3C8C1D94040   0022           00          0A          CHAR
00 05F7F00C400001   0206904180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 05F7F00C400001   0206910180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 0697E80C000000   010697EC20000007   010348E980090004   C2D6E2404040   0069           00          09          BOS
00 05F7F00C400001   0206904180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
01 0643380E400002   02068BC180040002   0103492D60010003   C3C8C1D94040   4022   UC      04          01          CHAR
00 0640000C400001   02069AC180040002   010348E560010003   C2D6E2404040   0069           04          01          BOS
00 05F7F00C400001   0206910180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 0640000C400001   0206982180040002   010348E560010003   C2D6E2404040   0069           04          01          BOS
00 05F7F00C400001   0206904180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 0640000C400001   02069A0180040002   010348E560010003   C2D6E2404040   0069           04          01          BOS
00 05F7F00C400001   0206910180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 0640000C400001   0206982180040002   010348E560010003   C2D6E2404040   0069           04          01          BOS
00 05F7F00C400001   0206904180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 05F7F00C400001   0206910180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 0640000C400001   02069A0180040002   010348E560010003   C2D6E2404040   0069           04          01          BOS
00 05F7F00C400001   0206904180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
00 0640000C400001   0206982180040002   010348E560010003   C2D6E2404040   0069           04          01          BOS
01 0643380E400002   02068BC180040002   0103492D60010003   C3C8C1D94040   4022   UC      04          01          CHAR
00 05F7F00C400001   02069AC180040002   010348E560010003   C1D3C1404040   0068           04          01          ALA
```

Figure 42. Formatted Line I/O Interrupt Trace Table

```
IED002A SPECIFY TCAM PARAMETERS
```

that is generated only if you omit one of the following INTRO operands at assembly time:

```
STARTUP=, LNUNITS=, KEYLEN=, and, if DISK=YES, CPB=.
```

The response keyword is A=n or DTRACE=n.

The trace table resides in the main storage allocated to the MCP and, therefore, to get a copy of the table, you must dump your MCP region. If you wish to dump the subtask trace table to a sequential data set to provide a history of TCAM activities, you must activate the COMWRITE routine for the subtask trace table dump by issuing the DEBUG operator command.

control characters	*operation*	*operand*
control chars	MODIFY F	[procname.]id,DEBUG=L,IEDQFE10 jobname

Note: *COMWRTE=YES must have been specified either at assembly or INTRO execution time.*

This loads (L) the dump routine for subtask trace. If you want to deactivate the routine, replace the L with D; otherwise, the command is the same. The routine prints half the table at a time to the sequential data set while the other half is being filled. Therefore, your most current entries are still in main storage.

Example: Printing the subtask trace table when COMWRITE is used.

```
//PRINT        JOB  MSGLEVEL=1
//STEP         EXEC PGM=IEDQXB,PARM='STCB'
//SYSPRINT     DD   SYSOUT=A
//SYSUT1       DD   DSN=COMWRITE,UNIT=2400,DISP=OLD,*
//                  LABEL=(,NL),VOL=SER=DUMMY
/*
```

The subtask trace table is also printed if no PARM= parameter is specified.

Use the subtask trace to determine what TCAM was doing when it failed. A subtask trace table must be included with APAR documentation. You should print the subtask trace for any problem that occurs. To fully utilize the subtask trace, you should also dump main storage to get the remaining entries in the table.

Using the Subtask Trace: Use the subtask trace, a logged history of internal resource and data movement in TCAM, to help you diagnose TCAM problems. It records the flow of all dispatched elements, showing where they go in the TCAM system and what subtasks work on them.

While not all TCAM functions are logged, you can get a good picture of the flow. Although you do not need the trace to find a program check address, it can tell you the data movement preceding the check and which module passed control to the module that failed.

Also, you can analyze hard waits and loops more closely when you have an excessive throughput reduction. You can trace loops among modules passing through the dispatcher and spot unnecessary WAIT conditions caused by poor resource allocation.

To begin with, know how to find the trace in main storage and how to define its parameters. The formatted trace printed by the IEDQXB utility serves as a good history, but your current problem is probably still in the main-storage table. This is usually true in the case of a program check, and in the case of most WAIT conditions.

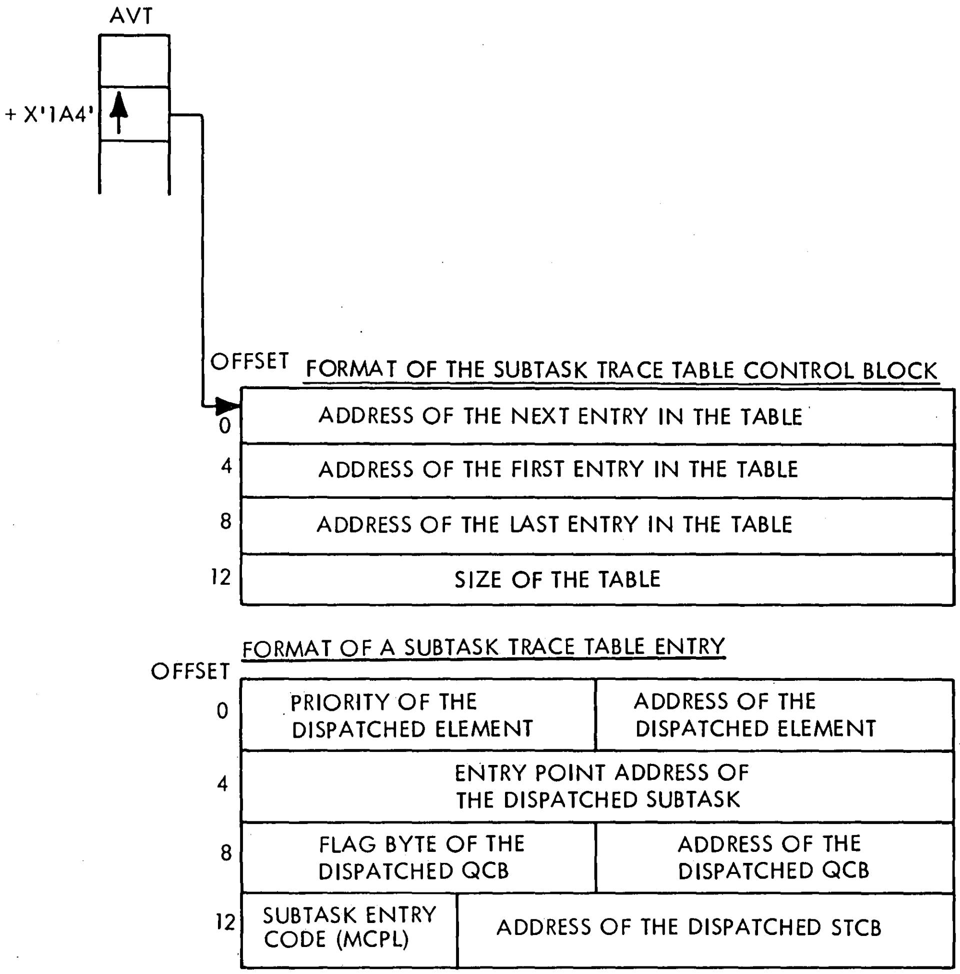
The Table in Main Storage: AVT+X'1A4' points to the 16-byte header of the table. If the dump program IEDQFE10 is not in the system, the table follows the header, which has the format shown in Figure 43.

IEDQFE10 modifies the header and adds prefixes to each half of the table when it splits the table into halves for its own use. After IEDQFE10 splits the table, each half has the format shown in Figure 43.

When you look at the trace table in a dump, if you see the C'STCB', you know that IEDQFE10 is in the system. If not, you see the header as it appears before calling IEDQFE10.

An example of a formatted subtask-trace table prefix is shown in Figure 44. The first entry points to either the first or second half of the table.

In Figure 45 the second half of the table is being used, as indicated by the header pointers.



FORMAT OF THE SUBTASK TRACE TABLE CONTROL BLOCK

OFFSET	
0	ADDRESS OF THE NEXT ENTRY IN THE TABLE
4	ADDRESS OF THE FIRST ENTRY IN THE TABLE
8	ADDRESS OF THE LAST ENTRY IN THE TABLE
12	SIZE OF THE TABLE

FORMAT OF A SUBTASK TRACE TABLE ENTRY

OFFSET		
0	PRIORITY OF THE DISPATCHED ELEMENT	ADDRESS OF THE DISPATCHED ELEMENT
4	ENTRY POINT ADDRESS OF THE DISPATCHED SUBTASK	
8	FLAG BYTE OF THE DISPATCHED QCB	ADDRESS OF THE DISPATCHED QCB
12	SUBTASK ENTRY CODE (MCPL)	ADDRESS OF THE DISPATCHED STCB

Figure 43. Subtask Trace Table Format

Note: *The address of the current entry is a pointer to the location where the next entry will be placed. Therefore, the latest entry in the table is located 16 bytes before the address contained in the first word of the header.*

Contents of an Entry: Each entry in the trace table is 16 bytes of information in the following format.

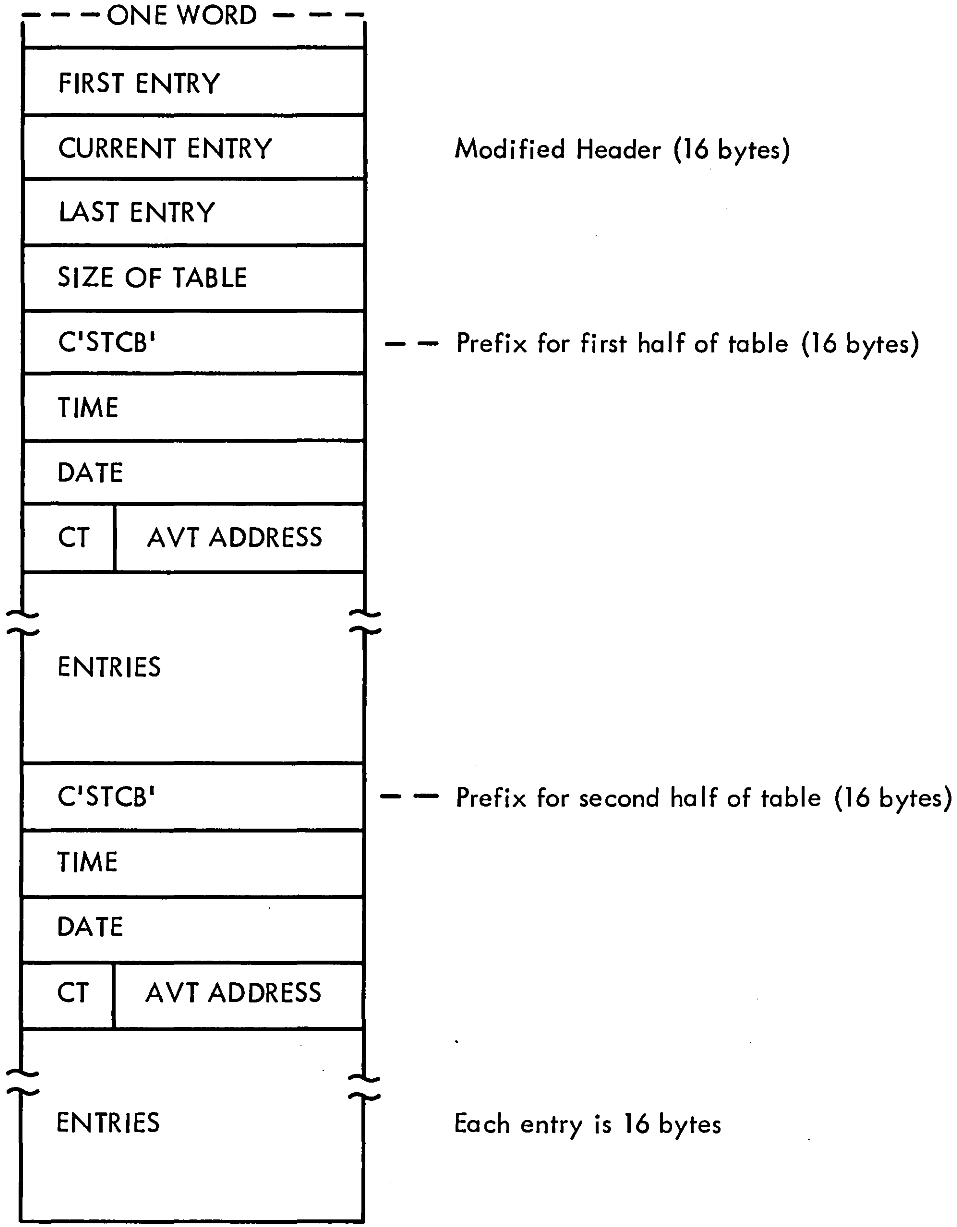


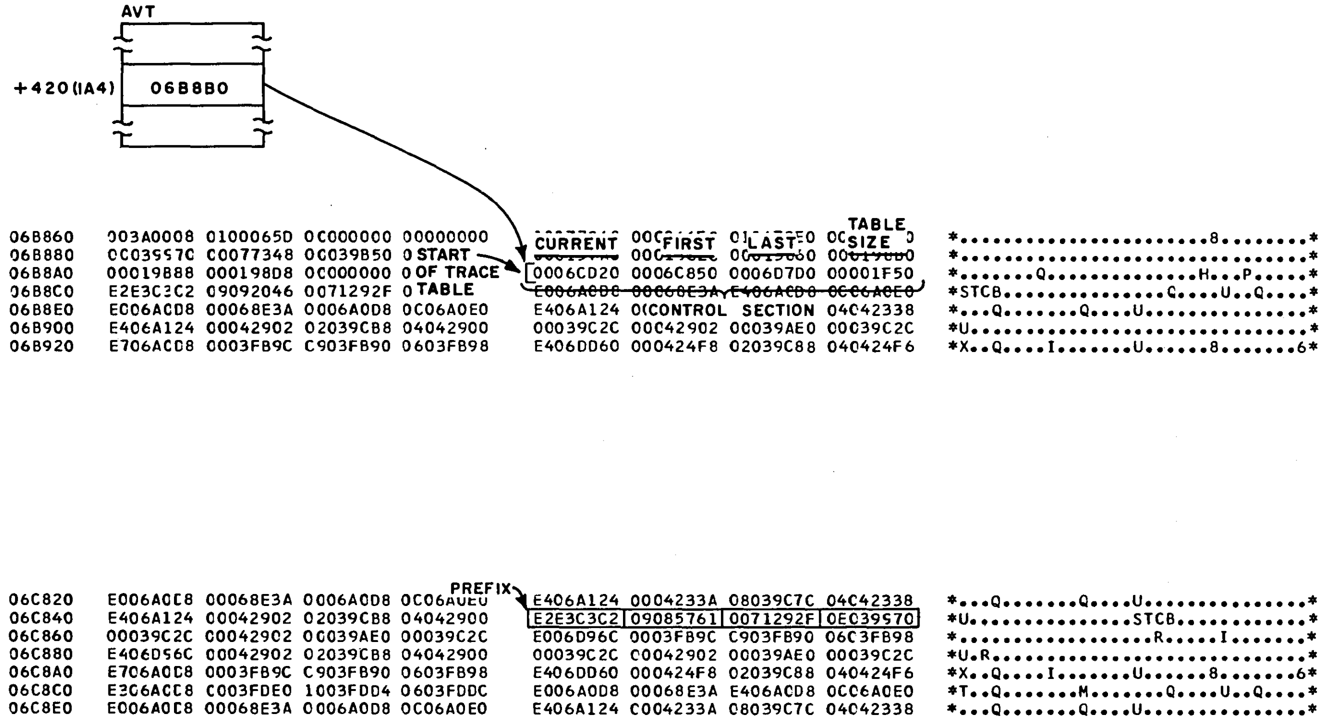
Figure 44. Formatted Subtask Trace Table Prefix

+0	priority of the dispatched element	address of the dispatched element
+4	address of the entry point of the dispatched subtask	
+8	flag byte of the dispatched QCB	address of the dispatched QCB
+12	subtask entry code (MCPL)	address of the dispatched STCB

Priority of the dispatched element: The one-byte relative priority of the element used by the TCAM dispatcher. Figure 46 is a list of relative priorities. The actual meaning of the priority field depends on the type of element.

Address of the dispatched element: The main-storage address of the resource control block (RCB) associated with the dispatched element. The RCB is a two-word prefix to an element that allows the TCAM dispatcher to determine the QCB to which an element will be posted. Each element in the TCAM system is represented by an RCB. An element is an individual part of a system resource (for instance, a buffer, an LCB, etc.). To determine what type of element is being dispatched, examine the element. First, check the formatted section of your dump to see if it is an LCB. If it is not, it is a buffer if it is located in the buffer pool area (AVT+X'384' points to the start of the buffer pool). If it is neither an LCB nor a buffer, it could be an ERB (element request block) used to request buffers for transmitting data. The ERB is X'4C' beyond the beginning of the LCB.

Address of the entry point of the dispatched subtask: The entry-point address of the module that will act on the element.



```
06B860    003A0008 0100065D 0C000000 00000000                                          *.....................8..........*
06B880    0C03S57C C0077348 0C039B50 0                                                 *................................*
06B8A0    00019B88 000198D8 0C000000 0         0006CD20 0006C850 0006D7D0 00001F50    *......Q.............H...P.......*
06B8C0    E2E3C3C2 09092046 0071292F 0         E006ACD8 00068E3A E406ACD8 0C06A0E0    *STCB..............Q....U..Q.....*
06B8E0    EC06ACD8 00068E3A 0006A0D8 0C06A0E0  E406A124 0(CONTROL  SECTION 04C42338    *...Q.......Q....U...............*
06B900    E406A124 00042902 02039CB8 04042900  00039C2C 00C42902 0C039AE0 00039C2C    *U...............................*
06B920    E706ACD8 0003FB9C C903FB90 0603FB98  E406DD60 000424F8 02039C88 040424F6    *X..Q....I.......U......8.......6*
```

```
06C820    E006A0C8 00068E3A 0006A0D8 0C06A0E0  E406A124 0004233A 08039C7C 04C42338    *...Q.......Q....U...............*
06C840    E406A124 00042902 02039CB8 04042900  E2E3C3C2 09085761 0071292F 0E039570    *U.............STCB..............*
06C860    00039C2C C0042902 0C039AE0 00039C2C  E006D96C 0003FB9C C903FB90 06C3FB98    *................R.....I.........*
06C880    E406D56C 00042902 02039CB8 04042900  00039C2C C0042902 00039AE0 00039C2C    *U.R.............................*
06C8A0    E706A0C8 0003FB9C C903FB90 0603FB98  E406DD60 000424F8 02039C88 040424F6    *X..Q....I.......U......8.......6*
06C8C0    E3C6ACC8 C003FDE0 1003FDD4 0603FDDC  E006A0D8 00068E3A E406ACD8 0CC6A0E0    *T..Q.......M.......Q....U..Q....*
06C8E0    E006A0C8 00068E3A 0006A0D8 0C06A0E0  E406A124 C004233A C8039C7C 04C42338    *...Q.......Q....U...............*
```

Figure 45. Second Half of the Subtask Trace Table

Name	Value	Use in an ERB	Routines Using
PRINTRQ	E4	to request full buffers from Disk I/O	Send Scheduler Receive Scheduler Get Scheduler Put Scheduler Create an error message routine
PRIFSPCI	E8	to request empty buffers from buffer request QCB; to request full buffers from Disk I/O	PCI Appendage-on first PCI only Multiple Routing subtask
PRISBPCI	E0	to request empty buffers from buffer request QCB; to request full buffers from Disk I/O	PCI Appendage-all PCIs except the first
PRIDSKRQ	EC	to request an empty unit by chaining the ERB on the buffer return QCB	CPB Cleanup
PRIACTIV	E4	in tposting ERB to the activate QCB to request building an initial contact program and EXCP for the line	CPB Cleanup Buffer Request Buffer Return
PRIDKEOB	E0	to enable EOB to recall; to tpost to EOB Handling after an EOB error; must be lower priority than PRIMHBFR	CPB Cleanup CPB Initialization
PRIRECAL	E0	to request from Disk I/O a copy of the header	All routines requesting recalled headers Multiple Routing subtask
PRIRCQCB	E0	to return the ERB to any routine specified in LCBRCQCB	CPB Cleanup-after recall Create an error message routine
PRIAPERB	D0	to request full buffers	Application program
PRIEDISP	E0	to tpost ERB to itself on send operations when an error occurs before EOM; must be lower than PRIMHBFR	Buffer Disposition
PRIMHBFR	E4	to have a buffer processed by MH	PCI Appendage CPB Cleanup Line End Appendage-receive, last buffer only
PRIUREQ	E8	to request an empty unit for insert function in MH; must be higher than PRIMHBFR	Unit Request
PRIAPBFR	DC	to tpost a buffer to an application program	Incoming/Outgoing Message Delimiter routine
PRILNEND	E4	to have Buffer Disposition finish processing macros and clean up the line	Line End Appendage-send, last buffer only
PRIRCBFR	E0	to return a duplicate header to a specified routine	CPB Cleanup Destination Scheduler
RIBFRTB	E4	to return a buffer or unit to the buffer-unit pool	Incoming/Outgoing Message Delimiter routine PCI Appendage CPB Cleanup Destination Scheduler Multiple Routing subtask

gure 46. TCAM Relative Priorities (Part 1 of 3)

Name	Value	Use in an ERB	Routines Using
PRIDSKBF	EC	to give a unit to CPB Cleanup	Buffer Return
PRICOPY	E0	to have a message copied to a different data set	Destination Scheduler
PRIDESTQ	E4	to put a buffer on a message queue	Incoming/Outgoing Message Delimiter routine Multiple Routing subtask Create an error message routine
PRIDKWRT	E4	to have a full buffer written on disk	Destination Scheduler
PRIDKSRV	EC	to have a message flagged serviced	Buffer Cleanup
PRIDKCNC	E0	to have a message canceled in the message queue	Cancel Message
PRIDKINT	E0	to have a message intercepted	Hold/Release Terminal routine
PRICKPLN	EC	to tpost the LCB to Checkpoint requesting a checkpoint	Buffer Disposition
PRIMULTR	E0	to tpost the LCB to the Multiple Router routine to continue	Buffer Disposition TLIST
PRIOPCTL	DC	to tpost an operator control buffer	Message Handling routine Operator Control Interface routine
PRIDSPLB	E4	to tpost last buffer of message to buffer disposition QCB; must be lower than any PCI tpost of an ERB	Incoming/Outgoing Message Delimiter routine
PRIONLT	DC	to request on-line test	STARTMH subtask
PRILAEND	E4	to start error processing	Line End Appendage
PRIMHUNT	E8	to tpost a unit to MH; must be greater than PRIMHBFR	Unit Request
PRIRELSE	E0	to release a subtask from Time Delay or Operator Control	Operator Control Hold/Release Terminal
PRICPBCL	E8	to post CPB Cleanup complete	Disk End Appendage
PRICKPT	DC	to request a complete checkpoint	Reusability-Copy subtask MCPCLOSE Time Delay subtask
PRILNFRE	E8	to free a line; must get to Destination Scheduler before line is free	Buffer Disposition Put Scheduler Send Scheduler
PRICLSDN	10	to request closedown; must be lowest priority	
PRIAPCKP	DC	to request an incident checkpoint	Application Program
PRIOPCKP	DC	to request an incident checkpoint	Operator Control

Figure 46. TCAM Relative Priorities (Part 2 of 3)

Name	Value	Use in an ERB	Routines Using
PRILNCL	EC	to clean up buffers and to free a line; to tpost a line to Buffer Disposition	Line End Appendage INEND OUTEND
PRILOGLB	E0	to tpost the Log LCB to itself	LOG Scheduler
PRISSOLT	DC	tposted to Operator Control to request STARTLINE/STOPLINE to return an element from the time delay queue	On-Line Test Time Delay
PRIATTN	DC	to tpost the attention element for local devices	Attention Handler
PRISYSDL	DC	to initiate system delay	Operator Control
PRISYSDT	D8	to tpost the system delay QCB to Time Delay	System Delay
PRILCBDL	20	to indicate to Environment Checkpoint that an LCB is on the System Delay	System Delay subtask Environment Checkpoint

NOTE: All EOM (end of message) buffers have DF in the priority field of the RCB.

Figure 46. TCAM Relative Priorities (Part 3 of 3)

Flag byte of the dispatched QCB: The first byte of the QCB. Sometimes its contents are meaningless. If the flag byte is C9, the buffer disposition routine is to be tposted. If, however, this is a destination QCB, these flags indicate which destination QCB the dispatcher is to use, and which message queues data set is to receive messages for the destination.

Bit definitions are:

Bits	*Value*	*Meaning*
1,2	X'60'	main-storage queues with backup on nonreusable disk
1,3	X'50'	main-storage queues with backup on reusable disk
1	X'40'	main-storage-only queues
2	X'20'	nonreusable disk queues
3	X'10'	reusable disk queues
6	X'02'	this is a QCB
7	X'01'	stop sending while reusability clears this queue

Address of the dispatched QCB: The address of the queue control block (QCB) whose first STCB will be activated. A QCB regulates the sequential use of elements among requesting tasks. Every queue or item waiting for service in the system has a QCB.

Subtask entry code (MCPL): Using this field, the TCAM dispatcher calculates the subtask entry point.

Code	*Meaning*
X'04'	the subtask entry point immediately follows a 2-byte STCB (subtask control block)
X'06'	the subtask entry point immediately follows a 4-byte STCB
X'08'	the subtask entry point immediately follows a 6-byte STCB
X'0A'	the subtask entry point immediately follows an 8-byte STCB

If the MCPL value is greater than X'0A', the TCAM dispatcher activates a subtask by using the MCPL field as an index into the AVT subtask branch table at AVTDISP (AVT+X'228'). The following values of MCPL cause the dispatcher to activate the associated subtask.

Code	*Subtask*
X'0C'	leased receive scheduler
X'0E'	send scheduler
X'10'	GET scheduler
X'12'	PUT scheduler
X'14'	GET FIFO scheduler
X'16'	log scheduler
X'18'	dial receive scheduler
X'1A'	buffered terminal scheduler
X'1C'	retrieve scheduler
X'1E'	local receive scheduler

If the MCPL field is X'00', no real elements are currently tposted to the ready queue. A subtask residing in the dispatcher issues an OS WAIT command.

If the MCPL field is X'02', the element has been tposted to a QCB that represents an attached TCAM task (operator control, checkpoint, or on-line test). The dispatcher activates a subtask residing in the dispatcher that enqueues the element. OS posts the subtask.

Address of the dispatched STCB: The address of the dispatched subtask control block for the module that will be activated (the module whose entry point is in the second word of the trace entry).

The TCAM dispatcher places an entry in the subtask trace table immediately before branching to the routine. Therefore, the entry indicates what is going to be done to the element. The function has not yet been performed.

The following example of a main-storage subtask trace table points out the normal flow of a received message, a sent message, and a negative response to polling. Once you become familiar with this flow, you will find it easier to identify additional TCAM functions in a table. Checkpoint, application programs, logging, and multiple routing change the number of entries, but you should always be able to pick out the basic message flow.

The subtask trace can be used more effectively with a storage dump to allow you to verify the exact module or element involved in the activity.

Figure 47 (Parts 1-3) shows how to read an entry. The numbers under each field correspond to the numbers in the following discussion.

```
06CAC0   EC06D96C C003FB9C C903FB90 0603FB98   E306D920 0003FDE0 1003FDD4 0603FDDC   *..R.....I.......T.R........M....*
06CAE0   E006D920 00065490 EC06D920 1806D928   E406D96C 0004233A 08039C7C 04042338   *..R.......R...R.U.R.............*
06CB00  |E4|06D96C|00042902|02|039CB8|04|042900|  00039C2C 00042902 00039AE0 00C39C2C   *U.R.............................*
06CB20   ①06②C8 000③B9C ④03F⑤0 ⑥03⑦98   E406AB20 000424F8 02039C88 04C424F6   *X..Q....I.......U......8.......6*
06CB40   E306A D8 000  DE0 1 03F  4  003  DC   E006A0D8 00068E3A E406A0D8 0C06ACE0   *T..Q.......M.......Q....U..Q....*
06CB60   EC06ACC8 00068E3A 0C06A0D8 0C06A0E0   E406A124 0004233A C8039C7C 04042338   *...Q.......Q....U...............*
```

Figure 47. Reading a Subtask Trace Entry (Part 1 of 3)

1. E4 is the relative priority. You need more information from the trace before you know its actual relevance.
2. 06D96C is the address of the RCB (the dispatched element). Go to this address in the storage dump.

```
06D900   CC0C0CC0 C0000000 0C000000 00000000   00000000 19498916 0006D918 0006D918   *.........................R...R.*
06D920   E006D920 E0039C2C 1806D928 20068F94   00001400 10000000 00000000 18C00C00   *..R.......R....................*
06D940   C2000C0C 7F039BE0 0 RCB 00 00000000   4006D9C0 0003A908 0404FB90 0C000C00   *B............. .R..............*
06D960   0006E7E0 C0000202 00000000 |00|039CB8|  |E4|039C2C| 0106E7E0 00000000 4006D898   *..X............U.....X..... .Q.*
06D980   0003B1C8 51111106 01030408 09 QCB      00 ADDRESS OF 001 01580001 600658EA   *...H................ ..........*
06D9A0   0C06D9BC 00039970 0C06D898 0C ADDRESS  02 NEXT RCB  001 0806D9B0 0006E7E0   *..R.......Q.......R.......R...X.*
```

Figure 47. Reading a Subtask Trace Entry (Part 2 of 3)

The RCB is a two-word prefix to an element with the following fields.

+0	key field	address of QCB to which this RCB is tposted
+4	priority	address of next RCB in the chain in which this TCB is located

To determine what type of element is being dispatched, examine the element. First, check the formatted section of your dump to see if it is an LCB. If it is not, it is a buffer if it is located in the buffer-pool area (AVT+X'384' points to the start of the buffer pool). If it is neither an LCB nor a buffer, as in this case, it could be an ERB (element request block) used to request buffers to transmit data. The ERB is X'4C' beyond the beginning of the LCB. To verify that it is an ERB, subtract X'4C' from the address in this field. Now check the formatted section of your dump for this LCB address. In this example,

```
6D96C
  -4C
-----
6D920
```

6D920 is an LCB address indicating an ERB as the element being dispatched.

3. The entry-point address of the dispatched subtask is 042902. At this address in storage, you can see which module is about to be activated. Looking in the right-most column, you see the module identification IEDQKA, the Activate-I/O Generator subtask.

```
          ENTRY POINT
0428CC    CC061B2B 42230007 9202300B 92083008    1299072E 92203004 109907FE 1B009101   *..............................*
0428E0    4C4 47E0 C5B89101 6C140711 4300401C    07F14300 A03907F1 0004233A 08000C01   * ...E......... ..1.....1........*
042900   |0400|90BF C00C47F0 F01CC9C5 C4D8C2C1    10561821 91081008 4780F022 5821CC0C   *.......00.IEDQKA..........0.....*
042920  STCB 0FC26 4B20FBBE 58402C34 91402C13    478CF03E 5860D29C 5860601C 45360C18   *.00...... ... ....0...K.........*
042940    41C0F1CC C55C98BF DCCCD2C1 205A203E    9101202C 4780F06E 58704014 58102C40   *..1.......K...........0... .... *
042960    92E41004 94FD1014 5C71000C 47F0B04E    41102020 0A0098BF D00C9140 20474770   *.U...........0............. ....*
```

Figure 47. Reading a Subtask Trace Entry (Part 3 of 3)

4. The flag byte of the QCB is meaningless in this example.
5. The address of the dispatched QCB is 039CB8. This is the QCB to which the element is tposted. As you can see, it is the same as the first word in the RCB.
6. The MCPL field is 04, which says that the entry point of the subtask immediately follows the two-byte STCB.
7. The address of the STCB for the subtask to be activated is 042900. You now know enough to understand the priority field. Looking at the list of relative priorities in Figure 46, you can see that an E4 priority indicates tposting an ERB to the Activate QCB to request building an initial channel program and EXCP for the line.

The Formatted Table: The trace table in main storage contains the most current entries. You may find the formatted trace printed by IEDQXB useful when intermittent failures require a history of what has been happening in the TCAM system. Figure 48 shows the printed output. Field headings are:

Subtask trace: The title of the table.

Sequence: A sequential count of the number of tables written on the data set. If a number is skipped you have lost entries; that is, the table half in main storage wrapped because COMWRITE was busy and could not write it to the data set. Prevent this problem by increasing your table size.

AVT address: The address of the AVT (address vector table).

Date/time: The date and time at which the table was placed on the data set. You can dump selectively using the time.

First type QCB and second type QCB: Not significant, other than telling you that the right-most 16-byte entry was made first.

Pri: The priority of the dispatched element.

Ele: The address of the dispatched element.

Entry: The address of the entry point of the dispatched subtask.

Fl: The flag byte of the dispatched QCB.

QCB: The address of the dispatched QCB.

Ml: The subtask entry code (MCPL).

STCB: The address of the dispatched STCB.

Use the translation of the module name to follow the flow of activity. Remember, however, that these translations are general descriptions of the module activity, and are sometimes misleading. Do not depend on this translation; go to the addresses specified to see which module is acting on the element. The translation helps explain the module activity.

QCB POSTED TO ITSELF indicates an LCB being posted to itself.

ACTIVATE is the Activate-I/O Generator routine.

Using the Table: To further help you understand this trace and its use, a list of hints of what to look for in determining the cause of the problem follows.

```
**SUBTASK TRACE**         SEQUENCE- 03      AVT ADDRESS- C413F0                               DATE- 71.139    TIME- 13.44.51
   FIRST TYPE QCB       PRI  ELE    ENTRY  FL  QCB    ML  STCB      SECOND TYPE QCB     PRI  ELE    ENTRY  FL  QCB    ML  STCB

 QCB POSTED TO ITSELF   E0 072CF8 063BD2 E4 072CF8 0C 072100 QCB POSTED TO ITSELF   E0 0720F8 063BD2 00 0720F8 0C 072100
 AVAILABLE BUFFER       E4 072144 04762A 08 0416FC 04 047628 ACTIVATE               E4 072144 047BF2 02 041738 04 047BF0
                        C0 0416AC 047BF2 00 04156C 00 0416AC                        E7 072CF8 04B734 C9 04B728 06 04B730
 BUFFER RETURN          E4 0690C0 0477E8 02 0417C8 04 0477E6 BUFFER RETURN          E3 0720F8 04B96C 10 04B960 06 04B968
 QCB POSTED TO ITSELF   E0 072CF8 063BD2 E4 072CF8 0C 072100 QCB POSTED TO ITSELF   E0 0720F8 063BD2 00 0720F8 0C 072100
 AVAILABLE BUFFER       E4 072144 04762A 08 0416FC 04 ~~047628 ACTIVATE~~               ~~E4 072144 047BF2 02 041738 04 047BF0~~
                        C0 0416AC 047BF2 00 041560 00 0416AC                        E7 0720F8 04B734 C9 04B728 06 04B730
 BUFFER RETURN          E4 0690C0 0477E8 02 0417C8 04 0477E6 BUFFER RETURN          E3 0720F8 04B96C 10 04B960 06 04B968
 QCB POSTED TO ITSELF   E0 0720F8 063BD2 E4 072CF8 0C 072100 QCB POSTED TO ITSELF   E0 0720F8 063BD2 0C 0720F8 0C 072100
 AVAILABLE BUFFER       E4 072144 04762A 08 0416FC 04 047628 ACTIVATE               E4 072144 047BF2 02 041738 04 047BF0
                        C0 0416AC 047BF2 C0 041560 00 0416AC                        E0 0650B4 04B734 C9 04B728 06 04B730
                        E3 065C68 04B96C 10 04B96C 06 04B968 QCB POSTED TO ITSELF   ~~E0 065068 063B02 E0 065068 0C 065070~~
 QCB POSTED TO ITSELF   E0 065C68 063BD2 00 065C68 0C 065070 AVAILABLE BUFFER       E4 0650B4 04762A 08 0416FC 04 047628
 ACTIVATE               E4 065CB4 047BF2 02 041738 04 047BF0 ACTIVATE               00 0416AC 047BF2 00 04156C 00 0416AC
                        E7 0720F8 04B734 C9 04B728 C6 04B730 BUFFER RETURN          E4 0690C0 0477E8 02 041708 04 0477E6
                        E3 0720F8 04B96C 10 04B96C 06 04B968 QCB POSTED TO ITSELF   E0 0720F8 063BD2 E4 072CF8 0C 072100
 QCB POSTED TO ITSELF   E0 072CF8 063BD2 00 072CF8 0C 072100 AVAILABLE BUFFER       E4 072144 04762A 08 0416FC 04 047628
 ACTIVATE               E4 072144 047BF2 02 041738 04 ~~047BF0 ACTIVATE~~               00 0416AC 047BF2 00 ~~041560 00 0416AC~~
                        E7 072CF8 04B734 C9 04B728 06 04B730 BUFFER RETURN          E4 069000 0477E8 02 0417C8 04 0477E6
                        E3 072CF8 04B96C 10 04B96C 06 04B968 QCB POSTED TO ITSELF   E0 0720F8 C63BD2 E4 0720F8 0C 072100
 QCB POSTED TO ITSELF   E0 072CF8 063B02 00 072CF8 0C 072100 AVAILABLE BUFFER       E4 072144 04762A 08 0416FC 04 047628
 ACTIVATE               E4 072144 047BF2 02 041738 04 047BF0 ACTIVATE               00 0416AC 047BF2 C0 041560 00 0416AC
                        E7 072CF8 04B734 C9 04B728 C6 04B730 BUFFER RETURN          E4 0690C0 0477E8 02 041708 04 0477E6
                        E3 072CF8 04B96C 10 04B96C 06 04B958 QCB POSTED TO ITSELF   ~~E0 0720F8 063BD2 E4 0720F8 0C 072100~~
 QCB POSTED TO ITSELF   E0 072CF8 063BD2 00 072CF6 0C 072100 AVAILABLE BUFFER       E4 072144 04762A 08 0416FC 04 047628
 ACTIVATE               E4 072144 047BF2 02 041738 04 047BF0 ACTIVATE               00 0416AC 047BF2 00 041560 00 0416AC
                        E7 072CF8 04B734 C9 04B728 06 04B730 BUFFER RETURN          E4 069000 0477E8 02 0417C8 04 0477E6
                        E3 072CF8 04B96C 10 04B96C 06 04B968 QCB POSTED TO ITSELF   E0 0720F8 063BD2 E4 072CF8 0C 072100
 QCB POSTED TO ITSELF   E0 072CF8 063BD2 00 072CF8 0C 072100 AVAILABLE BUFFER       E4 072144 04762A 08 0416FC 04 047628
 ACTIVATE               E4 072144 047BF2 02 041738 04 047BF0 ACTIVATE               00 0416AC 047BF2 00 041560 0C 0416AC
                        E7 0720F8 04B734 C9 04B728 06 04B730 BUFFER RETURN          E4 0690C0 0477E8 C2 041708 04 0477E6
                        E3 072CF8 04B96C 10 04B96C 06 04B958 QCB POSTED TO ITSELF   E0 0720F8 063BD2 E4 0720F8 0C 072100
 QCB POSTED TO ITSELF   E0 072CF8 063BD2 00 072CF8 0C 072100 AVAILABLE BUFFER       E4 072144 04762A C8 0416FC 04 047628
```

Figure 48. Formatted Subtask Trace Table

1. Practice reading normal message-flow entries so you can more readily identify error flows.
2. Know the dispatching concept, control block linkages, and the data movement initiated by the subtasks and the dispatcher (see the *TCAM PLM.)*
3. Learn to recognize the scheduler MCPL fields.
4. Determine where the flow went wrong. Experience and the PLM will help here. Always try to associate the trouble with a line. This gives you an LCB to follow. Find the last time the LCB was shown free, and trace the flow forward to the failure.
5. Become familiar with pertinent MCPL fields, QCB addresses, and QCB flags. Priority fields in a buffer can tell you where the buffer should go.
6. Determine what negative polling and addressing responses look like. Use Figure 49 (Parts 1-3) to help diagnose the reason for line failures.
7. On program checks, the last two or three entries in the trace table will in most cases give you an idea of what was happening.
8. The SCB parameter list for the message handler macros tells you which functional macro routine has control if you get a program check in the message handler.
9. Check the buffer prefix fields for proper disposition of the buffer being handled at failure.
10. Identify the immediate entries before a loop or hard wait because they often lead to the source of the error.

Once you identify the LCB for the failure being traced by the subtask trace, you can note the interrupts and associate the line I/O entries. You can associate both line I/O and buffer traces with the subtask entries using the LCB and line information. Doing this, you can correlate the entire TCAM line activity.

```
                                    - Receive Operation -
05BA60   E0066038 00057082 00066038 0C066040   E4066084 0003C15A E40355D4 0403C158
05BA80   E4066084 0003C722 02035610 0403C720   00035584 0003C722 00035438 00035584
05BAA0   E8035640 0003AD02 02035640 0403AD00   00035584 000453A8 00035438 00035584
05BAC0   E8066084 0003C15A E40355D4 0403C158   00035584 00035423 00035438 00035584
05BAE0   E405D000 000408E0 0F036F38 0A0408D8   E405D000 0003F962 0F036F38 0403F960
05BB00   DF05D000 000404CC C90400C0 080404C8   E405D060 0003C318 020355E0 0403C310
05BB20   E405D000 00057B5A 62038B70 0E038B78   E405D000 0003DCF6 62038B70 0803DCF0
05BB40   E405D000 0003A772 02035634 0403A770   E4059580 0003C318 020355E0 0403C316
05BB60   E6066084 000404CC C90404C0 06040468   E8035640 0003AD02 02035640 0403AD00
05BBS0   E3066038 00040704 100406F8 06040700   E0066038 00057082 00066038 0C066040

                      Same Entry Type Denoting End of a Receive Operation

05C300   E0066038 00057082 00066038 0C066040   E4066084 0003C15A 080355D4 0403C158
05C320   E4066084 0003C722 02035610 0403C720   00035584 0003C722 00035438 00035584
05C340   E7066038 000404CC C90404C0 060404C8   E405CB20 0003C318 020355E0 0403C316
                                 - Negative Response to Polling -
05C360   E3066038 00040704 100406F8 06040700   E0066038 00057082 E4066038 0C066040
05C380   E0066038 00057B5A 00066038 0E038B78   E4066084 0003A772 02035634 0403A770
05C3A0   E4066084 0003C722 02035610 0403C720   00035584 0003C722 00035438 00035584
05C3C0   E4059580 000408E0 0F036F38 0A0408D8   E4059580 0003F962 0F036F38 0403F960
05C3E0   E405D000 0003C318 020355E0 0403C316   00035584 0003C318 00035438 00035584
05C400   00035584 000052C2 00035438 00035584   E4059580 000404CC C90404C0 060404C8
05C420   E0066084 0003A772 02035634 0403A770   E0066084 000404CC C90404C0 060404C8
05C440   E405CF40 00043842 120390C0 160399C8   E405CF40 0003DCFG 120390C0 0803CDF0
05C460   E005CF40 000461B4 000461A8 060461B0   00035584 00009C22 00035438 00035584
05C480   E8035640 0003AD02 02035640 0403AD00   00035584 00009C22 00035438 00035584
                                      - Send Operation -
       LINE 05C4A0 SAME AS ABOVE
05C4C0   E8035640 0003AD02 02035640 0403AD00   E005CF40 000404CC C90404C0 060404C8
05C4E0   EC059580 0003A772 02035634 0403A770   E405CF40 0003C318 020355E0 0403C316
05C500   E405CB20 0003C318 020355E0 0403C316   E405D060 0003C318 020355E0 0403C316
05C520   E3066038 00040704 100406F8 06040700   E0066038 00057B5A E4066038 0E038B78
05C540   E0066038 00057082 00066038 0C066040
```

Figure 49. A Receive, a Negative Response to Polling, and a Send Operation (Part 1 of 3)

RECEIVE OPERATION FROM A TERMINAL

LCB tposted to itself is the beginning of a Receive operation. First word is the LCB as an element. Third word is the LCB as the QCB. MCPL = 0C.	E0066038	00057082	00066038	0C066040
ERB in LCB is tposted to Buffer Request. Fourth word is the subtask IEDQGA address.	E4066084	0003C15A	E40355D4	0403C158
GA tposts the ERB to Activate, IEDQKA.	E4066084	0003C722	02035610	0403C720
KA started the line. Return to the dispatcher found nothing on the ready queue. WAIT issued. MCPL = 00.	00035584	0003C722	00035438	00035584
CPB Cleanup QCB tposted to itself. QCB is on the queue by Disk End Appendage for a previous open. I/O interrupt for Disk End Appendage gave dispatcher control.	E8035640	0003AD02	02035640	0403AD00
IGG019RC for the previous disk open started disk I/O. With no work yet to do for the receive, the CPB Cleanup is dispatched. The dispatcher, when it next gains control, issues a WAIT again. Data is still filling the buffer for the Receive.	00035584	000453A8	00035438	00035584
Result of first PCI interrupt. Request for BUFMAX. E8 priority in ERB means ERB tposted to GA for first PCI request.	E8066084	0003C15A	E40355D4	0403C158
WAIT after PCI service. Line now filling the last of the buffer to go on to the next or finishing last of the message.	00035584	00035423	00035438	00035584
Buffer has been tposted to EOB/ETB IEDQBT subtask on the STARTMH QCB. Some form of user option specified in STARTMH macro.	E405D000	000408E0	0F036F38	0A0408D8
	NOTE: If this is a one-buffer message, Line End Appendage tposts the buffer to STARTMH QCB; otherwise, PCI would tpost the buffer when handling subsequent PCI interrupts.			
The buffer is bypassed to IEDQAA by the dispatcher.	E405D000	0003F962	0F036F38	0403F960
All EOM buffers have 'DF' in the priority field of the RCB. After INHDR and INBUF processing, the buffer is tposted to Buffer Disposition, IEDQBD.	DF05D000	000404CC	C90404C0	060404C8
	NOTE: IEDQBD has 2 entry points. If C9 is in the QCB address, the buffers are being processed by the first entry point code of IEDQBD.			
BD tposts any unused buffers to Buffer Return, IEDQGB. This would be the extra buffer gotten by the PCI interrupt.	E405D060	0003C318	020355E0	0403C316
BD tposts the message buffer to the Destination QCB. QCB flag = 62. This shows main-storage queues with nonreusable disk backup.	E405D000	00057B5A	62038B70	0E038B78
	NOTE: The Send Scheduler is the first STCB in the STCB chain, signifying this destination awaits a full message.			
Send Scheduler bypasses control to the Destination Scheduler by the dispatcher entry point DSPBYPAS.	E405D000	0003DCF6	62038B70	0803DCF0
Destination Scheduler (IEDQHM) tposts the buffer to CPB Initialization, IEDQFA.	E405D000	0003A772	02035634	0403A770
FA swaps the buffer with the CPB unit. CPB unit is returned to the buffer-unit pool by being tposted to IEDQGB. EXCP is done to write on the disk queue.	E4059580	0003C318	020355E0	0403C316
Low-priority ERB on the ready queue for Buffer Disposition to process INMSG macros. This is done after FA starts the Disk I/O to utilize the time that the channel is writing on disk queue.	E0066084	000404CC	C90404C0	060404C8
CPB Cleanup QCB tposted to itself. Done by Disk End Appendage.	E8035640	0003AD02	02035640	0403AD00
INEND macro signals BD to tpost the LCB to BD's second entry point: E3 in priority field of LCB.	E3066038	00040704	100406F8	06040700
IEDQBD02 entry point of BD will tpost the LCB to itself to signify the end of receive. Line is free.	E0066038	00057082	00066038	0C066040

Figure 49. A Receive, A Negative Response to Polling, and a Send Operation (Part 2 of 3)

NEGATIVE RESPONSE TO POLLING

The LCB is tposted to itself signifying that the line is free.	E0066038	00057082	00066038	0C066040
LCB Receive Scheduler passes the ERB to Buffer management, IEDQGA, to request buffers.	E4066084	0003C15A	E40355D4	0403C158
IEDQGA gets buffers, completes them, and tposts the ERB to Activate, IEDQKA, to poll the line.	E4066084	0003C722	02035610	0403C720
WAIT issued by the dispatcher - waiting for an I/O interrupt. You may not see this if heavy line traffic. Only one line in this example.	00035584	0003C722	00035438	00035584
Line End tposts the LCB to Buffer Disposition on a negative response to polling. E7 priority in LCB equals a negative response situation.	E7066038	000404CC	C90404C0	060404C8
BD will free buffers by tposting the buffer to GB. This will only happen at the end of the invitation list if this is a multipoint line.	E405CB20	0003C318	020355E0	0403C316
BD will branch to do INMSG macro processing. The parameter list in the SCB will have the macros that are used in INMSG processing. At INEND the LCB is tposted to BD for finishing the line activity.	E3066038	00040704	100406F8	06040700
	NOTE: No actual INMSG processing is done. Control is passed through the macros to INEND.			
BD will tpost the LCB to itself. This designates that the line is again free.	E0066038	00057082	00066038	0C066040

SEND OPERATION TO A TERMINAL

Receive Scheduler in LCB tposted to itself.	E0066038	00057082	E4066038	0C066040
Receive Scheduler gives control to the Send Scheduler by DSPBYPAS.	E0066038	00057B5A	00066038	0E038B78
Send Scheduler tposts the ERB to CPB Initialization, IEDQFA, for Disk I/O to retrieve message from the queue. E4 in ERB is an initial request for a Send. FA tposts the ERB with the count to IEDQKA. Message is read from the core queue.	E4066084	0003A772	02035634	0403A770
	NOTE: If no core queuing, FA would have done a SIO by EXCP Driver to get the message from the disk queue and Disk End Appendage would tpost the CPB Cleanup QCB to itself. Core queuing with disk backup is used in this example, thus no SIO.			
Activate, IEDQKA, starts addressing the line. A SIO is done with a Write Idle loop.	E4066084	0003C722	02035610	0403C720
Wait on same interrupt -- either positive response to addressing or disk ending.	00035584	0003C722	00035438	00035584
Line End on positive response to addressing tposts buffer(s) filled by FA to the STARTMH QCB. IEDQBT EOB/ETB is the first subtask in the chain, as in receive open, indicates user logical error checking.	E4059580	000408E0	0F036F38	0A0408D8
BT bypasses to IEDQAA, which processes OUTHDR and OUTBUF macros. BALs to Buffer Association for CCW building. Not seen in trace as it does not go through dispatcher.	E4059580	0003F962	0F036F38	0403F960
Wait on buffer to finish being sent to the line.	00035584	0003C318	00035438	00035584
Buffer tposted by Line End Appendage to perform OUTMSG processing. Buffer has been sent to line and line interrupt has occurred. C9 signals EOM buffer. NOTE: PCI interrupt would occur if coded but is effective NOP on send if a one-buffer message or if initial request has enough buffers assigned to hold all the message. PCI can free any previously sent buffers but will not get any more if this is the case as in this send operation.	E4059580	000404CC	C90404C0	060404C8
At OUTEND BD will tpost the buffer to FA to write the message serviced flag in the queued record. EC in priority field is EOM buffer to be marked.	EC059580	0003A772	02035634	0403A770
At OUTEND, BD will also tpost the LCB to its second entry point to perform line-freeing activity.	E3066038	00040704	100406F8	06040700
LCB is tposted to itself with Send Scheduler first in the subtask chain. The next message on this queue would be sent if any more were queued. When no more messages, the Send Scheduler will bypass to the Receive Scheduler to poll the line. This is a CPRI = E situation.	E0066038	00057B5A	E4066038	0E038B78
LCB tposted to itself with the Receive Scheduler done by DSPBYPAS.	E0066038	00057082	00066038	0C066040

Figure 49. A Receive, a Negative Response to Polling, and a Send Operation (Part 3 of 3)

Buffer Trace

The buffer trace dumps TCAM buffer contents and status to a sequential data set. You can only trace buffers for a line being traced by the line I/O interrupt trace.

Activating the Trace: Whether this trace is available in your system depends on how you design your MCP. To include it, code on the INTRO macro instruction the operand COMWRTE=YES. The default is COMWRTE=NO. Include the operand at assembly time, or at INTRO execution time in response to the message

```
IED002A SPECIFY TCAM PARAMETERS
```

that is generated only if you omit one of the following INTRO operands at assembly time:

```
STARTUP=, LNUNITS=, KEYLEN=, and, if DISK=YES, CPB=.
```

The response keyword is G= or COMWRTE=. If you specify YES, include a DD statement in your MCP execution deck to create the COMWRITE data set. You must also specify a positive value for the TRACE= operand of the INTRO macro, either at assembly or INTRO execution time.

The trace table is internal to COMWRITE (an attached task), therefore, a dump of your MCP region does not contain the buffer trace table. The only way you can obtain the output of the buffer-trace table dump is to use the utility COMEDIT (IEDQXB). Activate the Buffer Trace routine by issuing the DEBUG operator command.

control characters	*operation*	*operand*
control chars	{MODIFY F}	{[procname.]id jobname} ,DEBUG=L,IEDQFE30

This loads (L) the dump routine for the buffer trace. If you want to deactivate the routine, replace the L with D; otherwise, the command is the same.

Example: Printing the buffer trace

```
//PRINT        JOB  MSGLEVEL=1
//STEP         EXEC PGM=IEDQXB,PARM='BUFF'
//SYSPRINT     DD   SYSOUT=A
//SYSUT1       DD   DSN=COMWRITE,UNIT=2400,DISP=OLD,*
//                  LABEL=( ,NL),VOL=SER=DUMMY
/*
```

The buffer trace table is also printed when no PARM= parameter is specified.

Use the buffer trace to learn the status of a message as STARTMH receives it, both incoming and outgoing. This helps you determine where your problem is, since you know the status of the message before you do any processing (trouble in transmission) and the status after incoming processing (trouble in your message handler). Dump the buffer trace for all data-oriented problems, such as lost, erroneous, or extraneous data. Dump it also for all line or line-oriented problems.

Using the Buffer Trace: The TCAM buffer trace table records buffer contents and status. Use this table to trace a message through your message handler. TCAM places an entry in the table as soon as it posts the buffer to STARTMH. Both header and text buffers are posted.

TCAM places an entry for an input buffer in the table before it does *any* message handling. Therefore, the buffer contents are in a hexadecimal format that represents the line code of the originating terminal. Buffer contents include, in front of message data, the buffer prefix and the number of bytes you reserved in the RESERVE= operand of the line group DCB macro. Each trace entry is only 96 bytes; therefore, if you have many reserve characters, the entry includes little or no message data.

The buffer prefix contains only the number of units in the buffer, the address of the LCB of the originating terminal, the status byte, and the amount of data in the buffer. TCAM fills in the remainder of the prefix during message processing.

By examining the input-buffer trace entry, you can learn the status of the buffer before *any* message handling. If you find an error, you have a problem either in the originating terminal or on the line over which the message was transmitted (a probable hardware error).

TCAM places an entry for an output buffer in the table after incoming message processing is complete and before any output processing. The buffer is already on the queue for the destination terminal. If you find an error in the trace entry, you have a problem either in the incoming subgroup of your message handler or in the queuing activity of TCAM (a probable software error).

Format of an Entry: The 96-byte buffer trace entry has the following format:

0	4	8	12	13	14	15
Buffer Address	*SCB Flags*	*Error CSW*	*Sense Byte*	*IOB Flag1*	*IOB Flag3*	*ERB Status*
LCB Status	*Line Address*	*Buffer Prefix and Data*				
16	18	20				95

Buffer address: The address of the buffer in main storage. The address of the input buffer and the output buffer may not be identical because of TCAM queuing. For example, if you send a message to a station with disk queuing, TCAM places the input buffer on the disk, and frees the buffer in main storage. When the buffer is ready to send to the destination, TCAM brings a copy of the buffer contents back into main storage to process in the outgoing message handler. This copy will probably not be in the same main-storage location as the input buffer.

SCB error flags: The first four bytes of the message error record assigned to the message.

CSW: The last half of the channel status word. It includes the status and count; each is two bytes. The two status bytes identify conditions in the device and channel. Bits 0 through 7 indicate conditions detected by the device or control unit. Bits 8 through 15 indicate conditions associated with the subchannel.

Bit	*Meaning*	*Bit*	*Meaning*
0	attention	8	program-controlled interruption
1	status modifier	9	incorrect length
2	control unit end	10	program check
3	busy	11	protection check
4	channel end	12	channel data check
5	device end	13	channel control check
6	unit check	14	interface control check
7	unit exception	15	chaining check

The two count bytes are the residual count for the last CCW used. See *Principles of Operation,* GA22-6821, for a complete discussion of the CSW and its bit settings.

Sense byte: For a description of the contents of the sense byte, see the component description publication for the transmission control unit that you are using.

IOB flag 1: The flag byte in the IOB with the following meanings:

Bits	*Meaning*
00..	no chaining
01..	command chaining
10..	data chaining
11..	both command and data chaining
..1.	error routine in control
...1	device is to be repositioned
.... 1...	cyclic redundancy check (CRC) needed - tape only
.... .1..	exceptional condition. After the error routine returns and this bit is on, the error is permanent
.... ..1.	IOB unrelated flag
....0	start
....1	restart

IOB flag 3: The I/O Supervisor routine flag byte. It is device dependent. See the *I/O Supervisor PLM,* for a description of this byte.

ERB status: The element request block (ERB) status byte. The ERB is a control area used to request buffers for a line group.

Bit	*Value*	*Meaning*
0	X'80'	end of initiate mode
1	X'40'	end of message read from disk
2	X'20'	logical read error
3	X'10'	ERB is waiting for buffers
4	X'08'	can never be set—distinguishes buffer from ERB
5	X'04'	error on send side
6	X'02'	disk request is complete (temporary)
7	X'01'	delink switch. ERB is not tposted, but is eligible to be tposted

LCB status: A two-byte field containing the status of the LCB.

Bit	*Value*	*Meaning*
0	X'80'	recall is being performed
1	X'40'	line is in control mode or this is the first BSC output conversational block

2	X'20'	non-immediate operator control operation is in progress
3	X'10'	receiving an initiate-mode message
4	X'08'	continue or reset operation in progress
5	X'04'	line is free
6	X'02'	line is receiving
7	X'01'	line is sending

If bits 5, 6, and 7 are off, the line is stopped.

8	X'80'	I/O trace is active for this line or the line is in lock mode
	X'7F'	mask to specify the I/O trace is not active for this line
9	X'40'	MSGGEN or start-up message
	X'BF'	mask to specify that this is not a MSGGEN or start-up message
10	X'20'	EOT from a buffered terminal, no EOM
	X'DF'	mask to specify a regular EOM if not EOT from a buffered terminal
11	X'10'	send priority switch set by the send scheduler
12	X'08'	negative response to polling
13	X'04'	line is binary synchronous (BSC)
14	X'02'	this is a dial LCB
15	X'01'	a response needs to be sent to the line

Line address: The hardware address of the line over which the message was transmitted.

Buffer prefix and data: The remaining 76 bytes of the trace entry contain the buffer prefix, the reserved space you requested in the RESERVE= operand of the line group DCB macro, and the actual message data. Figure 50 shows the format of the buffer prefix. The bit definitions for the status byte (PRFSTAT1) are:

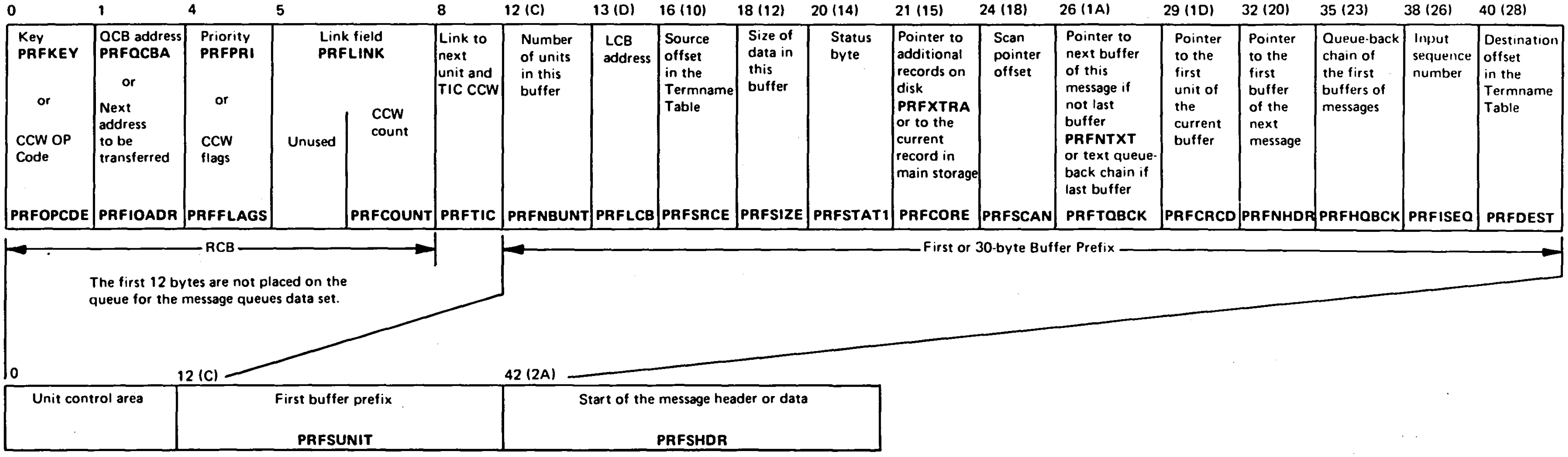
Value	*Meaning*
X'80'	message has been canceled
X'40'	this buffer contains an error message
X'20'	this message is being held
X'10'	this is a TSO buffer
X'08'	this is a duplicate-header buffer
X'04'	SETEOF was executed
X'02'	this is not the last buffer of the message
X'01'	this is not the first buffer of the message
X'00'	there is only one buffer in this message

The Formatted Table: Figure 51 shows the buffer trace as formatted by the utility program IEDQXB. The meaning of each field follows. Figure 50, a buffer prefix, is for your reference.

Buffer trace: The title of the dump.

Buffer Prefix

First buffer of a message:

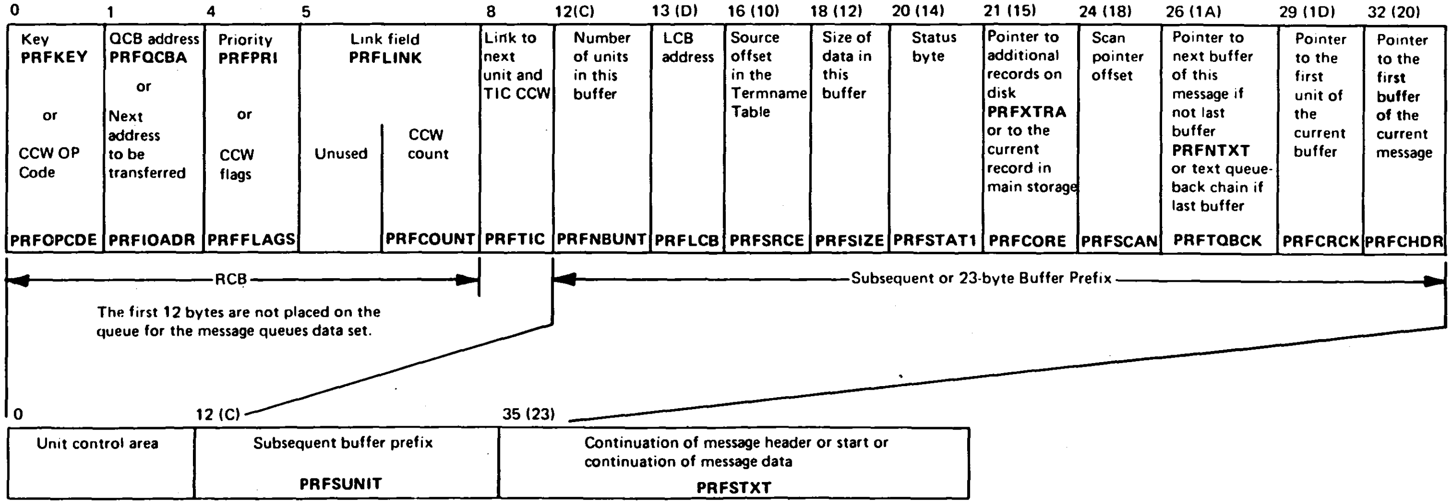


Offset

0	1	4	5		8	12 (C)	13 (D)	16 (10)	18 (12)	20 (14)	21 (15)	24 (18)	26 (1A)	29 (1D)	32 (20)	35 (23)	38 (26)	40 (28)
Key **PRFKEY** or CCW OP Code	QCB address **PRFQCBA** or Next address to be transferred	Priority **PRFPRI** or CCW flags	Link field **PRFLINK** Unused	CCW count	Link to next unit and TIC CCW	Number of units in this buffer	LCB address	Source offset in the Termname Table	Size of data in this buffer	Status byte	Pointer to additional records on disk **PRFXTRA** or to the current record in main storage	Scan pointer offset	Pointer to next buffer of this message if not last buffer **PRFNTXT** or text queue-back chain if last buffer	Pointer to the first unit of the current buffer	Pointer to the first buffer of the next message	Queue-back chain of the first buffers of messages	Input sequence number	Destination offset in the Termname Table
PRFOPCDE	**PRFIOADR**	**PRFFLAGS**		**PRFCOUNT**	**PRFTIC**	**PRFNBUNT**	**PRFLCB**	**PRFSRCE**	**PRFSIZE**	**PRFSTAT1**	**PRFCORE**	**PRFSCAN**	**PRFTQBCK**	**PRFCRCD**	**PRFNHDR**	**PRFHQBCK**	**PRFISEQ**	**PRFDEST**

0	12 (C)	42 (2A)
Unit control area	First buffer prefix **PRFSUNIT**	Start of the message header or data **PRFSHDR**

Subsequent buffer of a message:

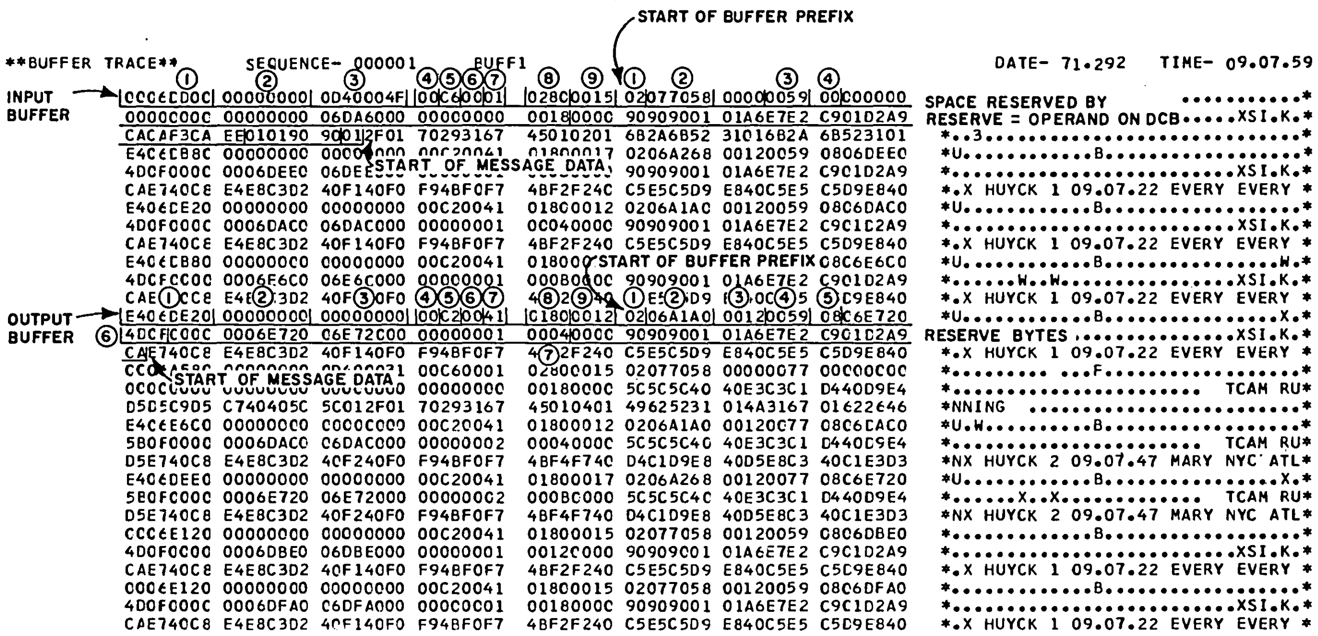


Offset

0	1	4	5		8	12(C)	13 (D)	16 (10)	18 (12)	20 (14)	21 (15)	24 (18)	26 (1A)	29 (1D)	32 (20)
Key **PRFKEY** or CCW OP Code	QCB address **PRFQCBA** or Next address to be transferred	Priority **PRFPRI** or CCW flags	Link field **PRFLINK** Unused	CCW count	Link to next unit and TIC CCW	Number of units in this buffer	LCB address	Source offset in the Termname Table	Size of data in this buffer	Status byte	Pointer to additional records on disk **PRFXTRA** or to the current record in main storage	Scan pointer offset	Pointer to next buffer of this message if not last buffer **PRFNTXT** or text queue-back chain if last buffer	Pointer to the first unit of the current buffer	Pointer to the first buffer of the current message
PRFOPCDE	**PRFIOADR**	**PRFFLAGS**		**PRFCOUNT**	**PRFTIC**	**PRFNBUNT**	**PRFLCB**	**PRFSRCE**	**PRFSIZE**	**PRFSTAT1**	**PRFCORE**	**PRFSCAN**	**PRFTQBCK**	**PRFCRCK**	**PRFCHDR**

0	12 (C)	35 (23)
Unit control area	Subsequent buffer prefix **PRFSUNIT**	Continuation of message header or start or continuation of message data **PRFSTXT**

Figure 50. A Buffer Prefix



```
**BUFFER TRACE**        SEQUENCE- 000001        BUFF1                                            DATE- 71.292   TIME- 09.07.59
          0C06CD0C 00000000 0D40004F 00C60001   0280C015 02077058 00000059 00C00000   SPACE RESERVED BY        ...........*
          0000C00C 00000000 06DA6000 00000000   0018000C 90909001 01A6E7E2 C901D2A9   RESERVE = OPERAND ON DCB.....XSI.K.*
          CACAF3CA EE010190 90012F01 70293167   45010201 6B2A6B52 31016B2A 6B523101     *..3..........................*
          E4C6CB8C 00000000 0000[illegible]000 00C20041   01800017 0206A268 00120059 0806DEE0     *U.............B..............*
          4DCF000C 0006DEE0 06DEE[illegible]                  90909001 01A6E7E2 C901D2A9     *........................XSI.K.*
          CAE740C8 E4E8C3D2 40F140F0 F94BF0F7   4BF2F24C C5E5C5D9 E840C5E5 C5D9E840     *.X HUYCK 1 09.07.22 EVERY EVERY *
          E406CE20 00000000 00000000 00C20041   01800012 0206A1AC 00120059 0806DAC0     *U.............B..............*
          4D0F000C 0006DACC 06DAC000 00000001   0C04000C 90909001 01A6E7E2 C9C1C2A9     *........................XSI.K.*
          CAE740C8 E4E8C3D2 40F140F0 F94BF0F7   4BF2F240 C5E5C5D9 E840C5E5 C5D9E840     *.X HUYCK 1 09.07.22 EVERY EVERY *
          E406CB80 000000C0 00000000 00C20041   01800[illegible]                   08C6E6C0     *U.............B.............W.*
          4DCFCC0C 0006E6C0 06E6C000 00000001   000B0[illegible]0C 90909001 01A6E7E2 C901D2A9     *......W..W...............XSI.K.*
          CAE[illegible]CC8 E4E[illegible]3D2 40F[illegible]0F0 [illegible]   4[illegible]2[illegible]40 [illegible]E[illegible]D9 [illegible]0C[illegible]5 [illegible]C9E840     *.X HUYCK 1 09.07.22 EVERY EVERY *
          E406DE20 00000000 00000000 00C20041   C1800012 0206A1A0 00120059 08C6E720     *U.............B..............X.*
          4DCFC00C 0006E720 06E72C00 00000001   0004000C 90909001 01A6E7E2 C901D2A9   RESERVE BYTES ...............XSI.K.*
          CAE740C8 E4E8C3D2 40F140F0 F94BF0F7   4[illegible]2F240 C5E5C5D9 E840C5E5 C5D9E840     *.X HUYCK 1 09.07.22 EVERY EVERY *
          CC0[illegible]A58[illegible] [illegible] [illegible]1 00C60001   02800015 02077058 00000077 00C00C0C     *.......... ...F....................*
          0C0C[illegible] [illegible] [illegible] 00000000   0018000C 5C5C5C40 40E3C3C1 D440D9E4     *......................   TCAM RU*
          D5D5C9D5 C740405C 5C012F01 70293167   45010401 49625231 014A3167 01622646     *NNING  ........................*
          E4C6E6C0 00000000 C000C000 00C20041   01800012 0206A1A0 00120C77 08C6DAC0     *U.W...........B..............*
          5B0F000C 0006DACC C6DAC000 00000002   0004000C 5C5C5C40 40E3C3C1 D440D9E4     *......................   TCAM RU*
          D5E740C8 E4E8C3D2 40F240F0 F94BF0F7   4BF4F74C D4C1D9E8 40D5E8C3 40C1E3D3     *NX HUYCK 2 09.07.47 MARY NYC ATL*
          E406DEE0 00000000 00000000 00C20041   01800017 0206A268 00120077 08C6E720     *U.............B..............X.*
          5B0FC00C 0006E720 06E72000 000000C2   000BC000 5C5C5C4C 40E3C3C1 D440D9E4     *......X..X.............   TCAM RU*
          D5E740C8 E4E8C3D2 40F240F0 F94BF0F7   4BF4F740 D4C1D9E8 40D5E8C3 40C1E3D3     *NX HUYCK 2 09.07.47 MARY NYC ATL*
          CCC6E120 00000000 00000000 00C20041   01800015 02077058 00120059 C806DBE0     *..............B..............*
          4D0F0C00 0006DBE0 06DBE000 00000001   0012C000 90909C01 01A6E7E2 C9C1D2A9     *........................XSI.K.*
          CAE740C8 E4E8C3D2 40F140F0 F94BF0F7   4BF2F240 C5E5C5D9 E840C5E5 C5C9E840     *.X HUYCK 1 09.07.22 EVERY EVERY *
          000€E120 00000000 00000000 00C20041   01800015 02077058 00120059 0806DFA0     *..............B..............*
          4D0F000C 0006DFA0 C6DFA000 000C0C01   0018000C 90909001 01A6E7E2 C9C1D2A9     *........................XSI.K.*
          CAE740C8 E4E8C3D2 40F140F0 F94BF0F7   4BF2F240 C5E5C5D9 E840C5E5 C5D9E840     *.X HUYCK 1 09.07.22 EVERY EVERY *
```

Figure 51. Formatted Buffer Trace

Sequence: A sequential count of the number of buffer trace tables printed. It is incremented by one each time a table is filled. If a number is skipped, you have lost records. Once a table is filled, the buffer trace dump routine increments the sequence count and gives the table to COMWRITE for writing. If this routine is busy, the table wraps, and entries are lost. The sequence field shows this internal wrapping.

Buff1: The first of the two trace tables is being dumped. This field alternates between BUFF1 and BUFF2. The buffer-trace dump routine fills one of the tables while the other is being placed on the data set.

Time and date field: The time and date the trace table was placed on the data set.

Input buffer:

1. Main-storage address is 06DD00.
2. No error bits are on in the message error record.
3. The CSW control unit status (0D40) is channel end, device end, unit exception, and a residual count X'4F' or 79 bytes.
4. The sense byte for the control unit is 00.
5. IOB FLAG1 is C6, indicating both command and data chaining and an exceptional condition (permanent error).
6. IOB FLAG3 contains no information.
7. ERB status is 01. The ERB is not tposted but is eligible to be tposted.
8. LCB status is 0280. The line is receiving and the I/O trace is active for this line.
9. The originating terminal is on line 0015.
10. The next 30 bytes are the buffer prefix. It contains the following information.
    1. Two units are in the buffer.
    2. The LCB address for the originating station in 077058.
    3. The size of the data in the buffer is X'59', or 89 bytes.
    4. The status byte contains 00, indicating that only one buffer is in this message.

On the DCB for this line, the RESERVE= operand has a value of 24. Therefore, the next 24 bytes are reserved, and at present contain no valid data.

The message is in line code. The terminal entering the message is a 1050. By examining the line code chart for the 1050 terminal in the *TCAM Programmer's Guide,* you can translate the message contents.

Byte	*Translation*
2F	x
01	space
70	H
29	U
31	Y
67	C
45	K
etc.	

Output buffer: There are several entries for output buffers, since one of the destinations in the input buffer is a distribution list. The output buffers are easy to find since they are translated into EBCDIC. This example shows one of the output buffer entries.

1. The main-storage address for the output buffer to this terminal is 6DE20. It is not the same as the input buffer location.
2. No SCB error bit flags are set, so the message is still correct.
3. There is no CSW information.
4. The sense byte for the control unit is 00.
5. IOB FLAG1 is C2, indicating both command and data chaining.
6. IOB FLAG3 is 00.
7. The ERB status is 41, indicating that end of message was read from disk and that the ERB is not tposted but is eligible to be tposted.
8. The LCB status is 0180, indicating that the line is receiving and that an I/O trace is active for this line.
9. The address of the receiving terminal is 0012.

The next 30 bytes are the buffer prefix, which contains the following information.

1. Two units are in the buffer.
2. The LCB address for the receiving terminal is 06A1A0.
3. Alphabetically, the originating terminal is the eighteenth terminal in the termname table (HUYCK).
4. The size of the data in the buffer is X'59', or 89 bytes.
5. The status is 08, indicating that this is a duplicate-header buffer.
6. The scan pointer is located at X'4D' from the beginning of the prefix. The X'0F' in the scan pointer field indicates that 15 reserve bytes are still left in the buffer. Nine of the original 24 reserve bytes were used to insert the time.
7. Alphabetically, the receiving terminal is the fourth terminal in the termname table.

The next 15 bytes are the reserve bytes; they contain no valid data. Following the reserve bytes is the EBCDIC translation of the message contents.

Cross-Reference Table

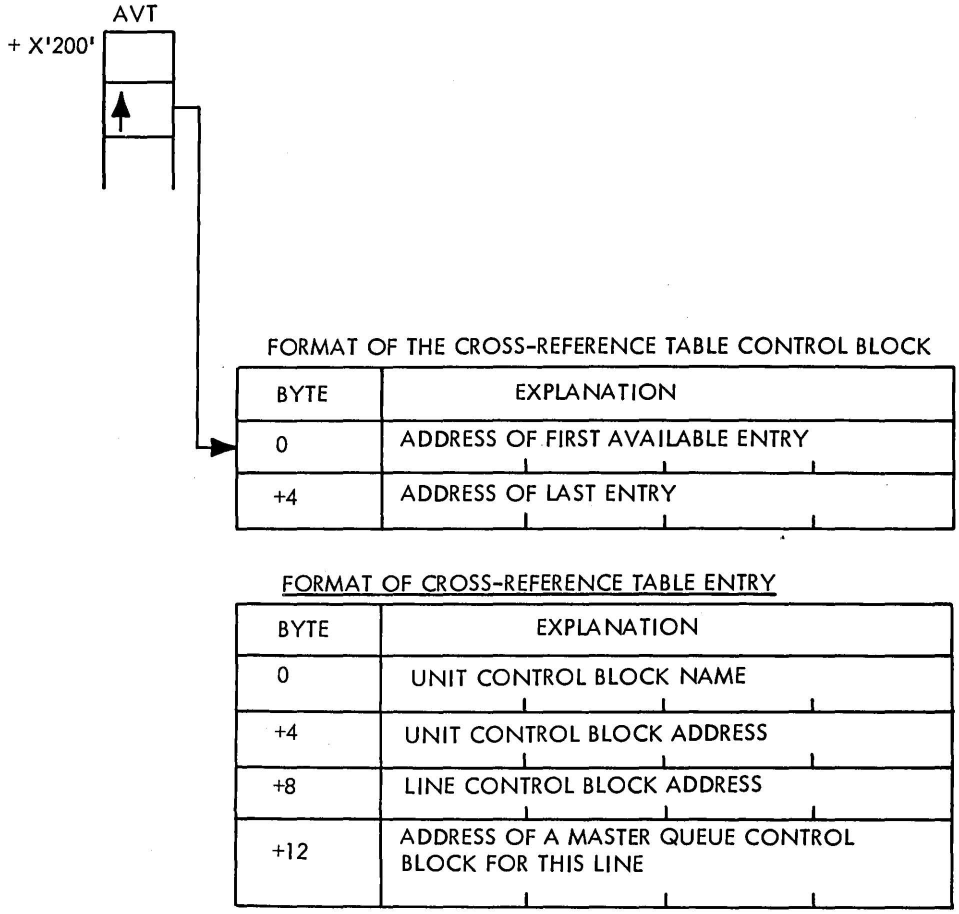
The TCAM cross-reference table contains the locations of all opened lines in your system and pointers to the major control blocks for each line. A formatted listing of this table is not available. If you include the table in your system, entries are created in it for each open line. Use it primarily as a quick reference, after system failure, to locate control blocks in a TCAM dump.

The TCAM cross-reference table is a convenient way to locate, in a dump, information for each open line. TCAM builds the cross-reference table if you code a positive integer in the CROSSRF= operand of the INTRO macro instruction.

At INTRO execution time, TCAM allocates 16n+8 contiguous bytes of main storage, where *n* is the integer specified in the CROSSRF= operand, and eight bytes is the length of the control block preceding the first entry for the table. AVT+X'200' contains the address of the table. Each time a line is opened, TCAM fills in the next available four-word entry in the table for that line.

The eight-byte control block preceding the first entry and the format of each entry is shown in Figure 52.
If you queue by line, only one master queue control block is assigned to the line, and TCAM places its address in the fourth word. If you queue by terminal, a master queue control block is assigned to each station on the line; in this instance, TCAM fills the fourth word with the address of the queue control block for the station whose entry appears in the terminal table before that of any other station on the line. If you open more lines than you provide entries for in the table, entries are made until the space is exhausted; no entries are made for lines opened after space runs out in the table.

If space permits, you should dynamically include the table at start-up time, specifying, CROSSRF=n or F=n where *n* is the number of lines to be opened.



FORMAT OF THE CROSS-REFERENCE TABLE CONTROL BLOCK

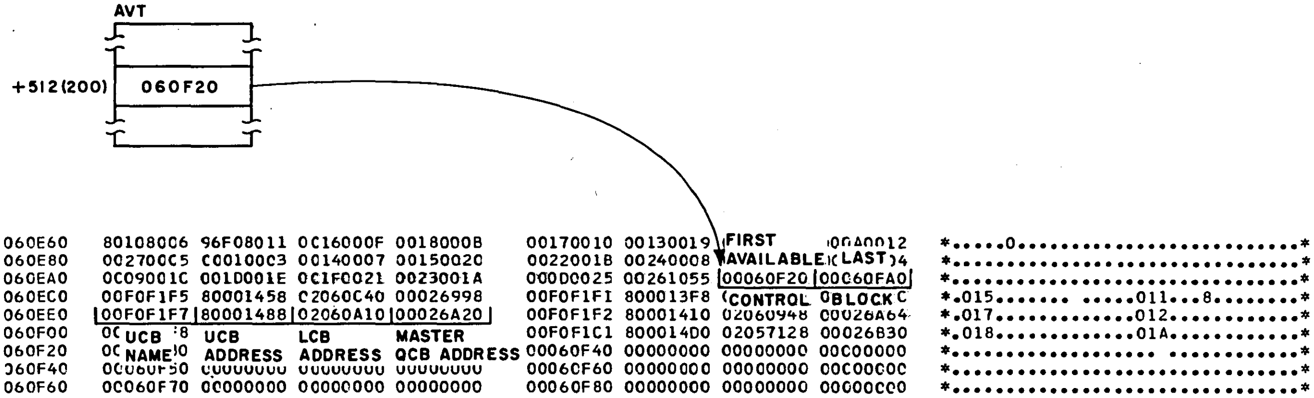
BYTE	EXPLANATION
0	ADDRESS OF FIRST AVAILABLE ENTRY
+4	ADDRESS OF LAST ENTRY

FORMAT OF CROSS-REFERENCE TABLE ENTRY

BYTE	EXPLANATION
0	UNIT CONTROL BLOCK NAME
+4	UNIT CONTROL BLOCK ADDRESS
+8	LINE CONTROL BLOCK ADDRESS
+12	ADDRESS OF A MASTER QUEUE CONTROL BLOCK FOR THIS LINE

Figure 52. Cross-Reference Table Format

The information in the table is a ready reference for each UCB. You will find this information especially helpful in the test/diagnose stages of implementing TCAM; it is a fast way to find out which line is using which UCB and where the QCB for a line is. It also shows which lines have been opened successfully. Figure 53 shows the printout of a cross-reference table in a main-storage dump.



```
060E60    80108006 96F08011 0C16000F 0018000B    00170010 00130019 FIRST      0040012    *.....0..........................*
060E80    002700C5 C00100C3 00140007 00150020    0022001B 00240008 AVAILABLE  LAST 4     *................................*
060EA0    0C09001C 001D001E 0C1F0021 0023001A    000D0025 00261055 00060F20 00C60FA0    *................................*
060EC0    00F0F1F5 80001458 C2060C40 00026998    00F0F1F1 800013F8 CONTROL  0BLOCK C    *.015....... .....011...8........*
060EE0    00F0F1F7 80001488 02060A10 00026A20    00F0F1F2 80001410 02060948 00026A64    *.017.............012............*
060F00    0C UCB  8 UCB      LCB      MASTER     00F0F1C1 800014D0 02057128 00026B30    *.018.............01A............*
060F20    0C NAME 0 ADDRESS  ADDRESS  QCB ADDRESS00060F40 00000000 00000000 00C00000    *................. ..............*
060F40    0C060F50 00000000 00000000 00000000    00060F60 00000000 00000000 00C00C0C    *................................*
060F60    0C060F70 0C000000 00000000 00000000    00060F80 00000000 00000000 00000CC0    *................................*
```

Figure 53. A Cross-Reference Table

Console and Terminal Listings

Have the system console listing available before you start *any* debugging. It contains a great deal of information about the system activities while TCAM is running, including the IEA000I error messages when permanent I/O errors occur. These messages can help you determine if your problem lies in the hardware of the line or the terminal.

In an ideal situation, you will also have all remote terminal listings when you start to debug a problem. Sometimes this is not possible. However, you should always have some remote terminal listings.

If you specified a terminal other than SYSCON, the system console, on the PRIMARY= operand of the INTRO macro, have the terminal listing for that terminal. All IEA000I error messages will be on this listing. You also need the terminal listing from every secondary terminal (a terminal for which you specified SECTERM=YES in its TERMINAL macro). These terminals can enter operator commands that allow them to reconfigure the network as they desire. An unexpected operator command can impact the entire system, and you may not have any idea why the system failed. Finally, you need terminal listings when there is any data or line problem.

When you collect the listings, mark the terminal name and type on the listings to make it easier to locate the LCB for the terminal in error.

Remember, for *all* problems, you should have:

1. the system console listing,
2. the primary terminal listing (if it is not the console), and
3. all secondary terminal listings.

Using Operator Commands

Using the TCAM operator control facility, you can enter operator commands to examine or alter the status of your telecommunications network. You can enter these commands from:

1. the system console;
2. a system input device (you must place the JCL identification // in the first two positions); or
3. any station or application program you designate as a secondary terminal (by coding SECTERM=YES on the TERMINAL or TPROCESS macro).

When you enter a command from a secondary station, remember:

1. you *must* precede the command with the characters specified in the CONTROL= operand of the INTRO macro instruction; and
2. you *must* follow the command with one or more blanks.

Figure 54 is a quick-reference chart of the TCAM operator commands and their functions.

Notes:

lineaddress may be entered either as the channel unit address or as *ddname,rln* where *ddname* is the name of the line group as specified on the DD statement and *rln* is the relative line number in the group.

statname is the specific station for which information or change is desired. It must correspond to the name of a TERMINAL macro and may be from one to eight characters beginning with an alphabetic character.

opfldname is the name of the specific option field, as specified in an OPTION macro for which information or change is desired. It may be from one to eight characters beginning with an alphabetic character.

procname is the name of the TCAM cataloged procedure in SYS1.PROCLIB.

id is the TCAM identifier used in the START command or the name of the job used to start TCAM in the system input stream.

jobname is identical to the *jobname* field in the JOB statement for the job.

Note 1: *The **sense** field may be any one of the following:*

BO	bus-out check
CR	command reject
DC	data check
EC	equipment check
IM	general intensive mode
IR	intervention required
LD	lost data
M2	leading graphics for 2740 Model 2 terminal
OR	overrun
TO	time-out
UE	unit exception

TYPE OF OPERATION	KEYWORD NAME	COMMAND	AREAS AFFECTED	FUNCTION
DISPLAY	DPRIOPCL	D TP, PRITERM	system, station	Displays the name of the current primary operator control station.
	DSECOPCL	D TP, SECTERM	system, station	Displays the names of all secondary operator control stations.
	INTRCEPT	D TP, INTER	system, station	Displays the names of all stations in the network that are intercepted (that is, stations that can enter messages but to which transmission of messages is suspended).
	ACTVATED	D TP, ACT, lineaddress	line, station	Displays the names of all active stations on the line addressed.
	INACTVTD	D TP, INACT, lineaddress	line, station	Displays the names of all inactive stations on the line addressed.
	LNSTATUS	D TP, LINE, lineaddress	line	Displays the status field and error record for the line (see Note 2).
	QSTATUS	D TP, QUEUE, statname	line, station	Displays the queue control block for the station; the information includes the number of messages queued, the queue status, and the priority levels permitted for messages.
	STATDISP	D TP, LIST, lineaddress	line, station	Displays whether the invitation list for the line may be polled and whether the Auto Poll feature is being used to poll the list.
	OPTFIELD	D TP, OPTION, statname, opfldname, $\left\{\begin{matrix}X\\C\\D\end{matrix}\right\}$	station	Displays the contents of the field that is reserved in the option table for the station. X is hexadecimal format; C, character format; D, decimal format.
	RLNSTATN	D TP, ADDR, statname	station	Displays the name of the line group of which the station is a part, the relative line number of the line on which the station is located, and the machine address of the line.
	STSTATUS	D TP, TERM, statname	station	Displays the status, input and output sequence numbers, and current intensive-mode recording status for the station.
HALT	SYSCLOSE	Z TP, QUICK	system	Stops message traffic on each line as soon as transmission of any message currently being sent or received on the line is completed. Messages remaining in the system are sent to the appropriate destinations after TCAM is restarted.
		Z TP, FLUSH	system	Stops message transmission from stations as soon as transmission of any message currently being sent is completed. All messages to stations are then sent before the system is halted. Intercepted messages that cannot be sent to stations are sent to the appropriate destination after TCAM is restarted.
HOLD	SUSPXMIT	H TP=statname	station	Suspends transmission to the station named. The station is intercepted, but can enter messages.
MODIFY	CPRIOPCL	F $\left\{\begin{matrix}\text{[procname.] id}\\ \text{jobname}\end{matrix}\right\}$, OPERATOR=statname	system, station	Changes the secondary operator control station specified to the primary operator control station.
	ERRECORD	F $\left\{\begin{matrix}\text{[procname.] id}\\ \text{jobname}\end{matrix}\right\}$, INTENSE=LINE, lineaddress, sense [, sensecount] (See Note 1)	system, station, line	Records recoverable I/O errors occurring on the line specified by lineaddress. Sensecount is the number of times error recording is to take place; default is 15.
		F $\left\{\begin{matrix}\text{[procname.] id}\\ \text{jobname}\end{matrix}\right\}$, INTENSE=TERM, statname, sense [, sensecount] (See Note 1)	system, station	Records recoverable I/O errors occurring on the station specified. Sensecount is the number of times error recording is to take place; default is 15.

Figure 54. Summary of Operator Commands (Part 1 of 2)

TYPE OF OPERATION	KEYWORD NAME	COMMAND	AREAS AFFECTED	FUNCTION
MODIFY	INTERVAL	F {[procname.] id jobname}, INTERVAL=SYSTEM	system, line	Causes the system to enter a delay for the duration specified on the INTVAL= operand of the INTRO macro.
	SYSINTVL	F {[procname.] id jobname}, INTERVAL=SYSTEM, value	system	Changes the duration of the system interval to the value specified. Value is a decimal number of seconds not exceeding 65535.
	POLLDLAY	F {[procname.] id jobname}, INTERVAL=POLL, statname, value	line	Changes the polling interval of the line group. Statname is the name of any station in the line group to be changed. Replace value with the decimal number of seconds less than 255.
	AUTOSTRT	F {[procname.] id jobname}, AUTOPOLL=lineaddress, ON	line	Changes the line from programmed poll to the Auto Poll facility if the automatic polling bit is on in the UCB for the line.
	AUTOSTOP	F {[procname.] id jobname}, AUTOPOLL=lineaddress, OFF	line	Changes the line from automatic polling to programmed polling.
	DATOPFLD	F {[procname.] id jobname}, OPT=statname, opfldname, data	station	Changes the contents of the option field for a station. Opfldname is the option field to be changed. Data is the data to be inserted.
	GOTRACE	F {[procname.] id jobname},TRACE=lineaddress, ON	line	Starts the TCAM I/O trace facility for the line.
	NOTRACE	F {[procname.] id jobname},TRACE=lineaddress, OFF	line	Deactivates the TCAM I/O trace facility for the line.
	DEBUG	F {[procname.] id jobname},DEBUG=L, routine	system	Starts the TCAM service aid routine that writes the dispatcher subtask trace table (IEDQFE10 is the routine), the I/O interrupt trace table (IEDQFE20), or the buffer trace (IEDQFE30).
RELEASE	RESMXMIT	A TP=statname	station	Releases the intercepted station so that messages can be transmitted to the station specified or for the line on which the station is located.
VARY	ACTVBOTH	V statname, ONTP, B (See Note 3)	station	Activates the nonswitched station named for both accepting and entering messages.
	ENTERING	V statname, ONTP, E (See Note 3)	station	Activates the nonswitched station specified for entering messages only.
	NOENTRNG	V statname, OFFTP, E (See Note 3)	station	Prevents the nonswitched station specified from entering messages.
	NOTRAFIC	V statname, OFFTP, B (See Note 3)	station	Prevents the nonswitched station specified from both entering and receiving messages.
	STARTLINE	V lineaddress, ONTP	line	Begins or resumes transmission on the line specified. Lineaddress may specify either for a line or for the entire line group.
	STOPLINE	V lineaddress, OFFTP, C	line	Stops transmission of messages on the line or line group specified after the current message.
		V lineaddress, OFFTP, I	line	Immediately stops transmission of messages on the line or line group specified.

Figure 54. Summary of Operator Commands (Part 2 of 2)

Note 2: *Possible responses in the LNSTAT= field of the response message are:*

BS	binary synchronous line
CM	line in control mode
CR	continue or reset operation
DL	switched (dial) line
IM	receiving initiate mode message
LF	line is free
MS	MSGGEN/start-up message
NR	negative response to polling
OC	operator control is stopping this line
RC	recall is being performed
RV	line is in receive mode
SD	line is in send mode
TB	EOT from a buffered terminal
TR	I/O trace active
NO BITS ON	

Possible responses in the ERR= field of the response message are:

ABR	abort—BSC line
CDC	connect/disconnect error
CHR	channel error
CUR	control unit error
CUT	CUTOFF error
FMT	format error
FWD	FORWARD error
HDR	incomplete header
HDW	hardware error
INV	invalid ID from station
ISB	insufficient buffers
LER	line error
LST.	message lost (overlaid)
MAX	main-storage maximum passed
MIN	main-storage minimum passed
MNS	message not sent/received
NOP	station inoperative
NTS	TSO not in the system
OLT	on-line test not in the system
ORG	invalid origin
SEL	selection error
SQH	sequence number is high
SQL	sequence number is low
TER	terminal error
TXT	text transfer error
UNR	undefined error
UNX	unit exception
USE	user error
NO BITS ON	

Note 3: *You must issue a STOPLINE command to stop the line before you enter this command, and, after receiving the response for the command, you must issue a STARTLINE command.*

Normal End-of-Day Closedown

Dump the TCAM data sets at the end of the day, since they contain important information about your system. The best way to handle your end-of-day cleanup is to place a procedure in SYS1.PROCLIB that dumps all necessary information.

You then have to start only that one procedure, rather than try to remember all the things you want to dump.

The following sample JCL creates an end-of-day procedure that dumps the nonreusable disk message queue, the reusable disk message queue, SYS1.LOGREC (the OBR/SDR data set), the log segment data set, and the COMWRITE data set. If you have experienced trouble during the day's execution, you should dump this output to the printer. However, if you are only dumping those data sets to keep a history of the system, you should dump to an output tape.

```
//PROC         JOB  MSGLEVEL=1
//STEP         EXEC PGM=IEBUPDTE
//SYSPRINT     DD   SYSOUT=A
//SYSUT1       DD   DSNAME=SYS1.PROCLIB,DISP=OLD
//SYSUT2       DD   DSNAME=SYS1.PROCLIB,DISP=OLD
//SYSIN        DD   DATA
./             ADD  LIST=ALL,NAME=ENDOFDAY,LEVEL=01,SOURCE=0
./             NUMBER NEW1=1000,INCR=1000
//STEP1        EXEC PGM=IEDQXC,PARM='Q=010,ALL'
//DISKQ01      DD   DSN=SAMP1,DISP=SHR
//SYSPRINT     DD   SYSOUT=A
//STEP2        EXEC PGM=IEDQXC,PARM='Q=010,ALL'
//DISKQ01      DD   DSN=REUSABLE,DISP=SHR
//SYSPRINT     DD   SYSOUT=A
//STEP3        EXEC PGM=IFCEREP0,PARM=(MCOS,PS)
//SERLOG       DD   DSNAME=SYS1.LOGREC,DISP=OLD,UNIT=2311, *
//                  VOL=SER=DT1010
//EREPPT       DD   SYSOUT=A
//STEP4        EXEC PGM=IEDQXB
//SYSUT1       DD   DSN=LOGSEG,VOL=SER=111111,UNIT=2311, *
//                  DISP=OLD
//SYSPRINT     DD   SYSOUT=A
//STEP5        EXEC PGM=IEDQXB
//SYSUT1       DD   DSN=COMWRITE,UNIT=2400,DISP=OLD, *
//                  LABEL=(,NL),VOL=SER=THANKS
//SYSPRINT     DD   SYSOUT=A
./     ENDUP
```

When you code this procedure:

1. You cannot include any step that requires a DD * or DD DATA statement (that is, you cannot include a utility that requires control statements); and
2. Each step has a space allocation. If you have trouble getting your output, either place a SPACE= parameter on each SYSPRINT statement, or allocate the SYSPRINT data set directly to the desired device.

After you close TCAM, issue the following commands for the system console:

```
      Z EOD
      S ENDOFDAY
```

The Z EOD command places an end-of-day marker in the SYS1.LOGREC data set. The START command starts the dumping procedure.

Appendix A. TCAM Control Block Relationships

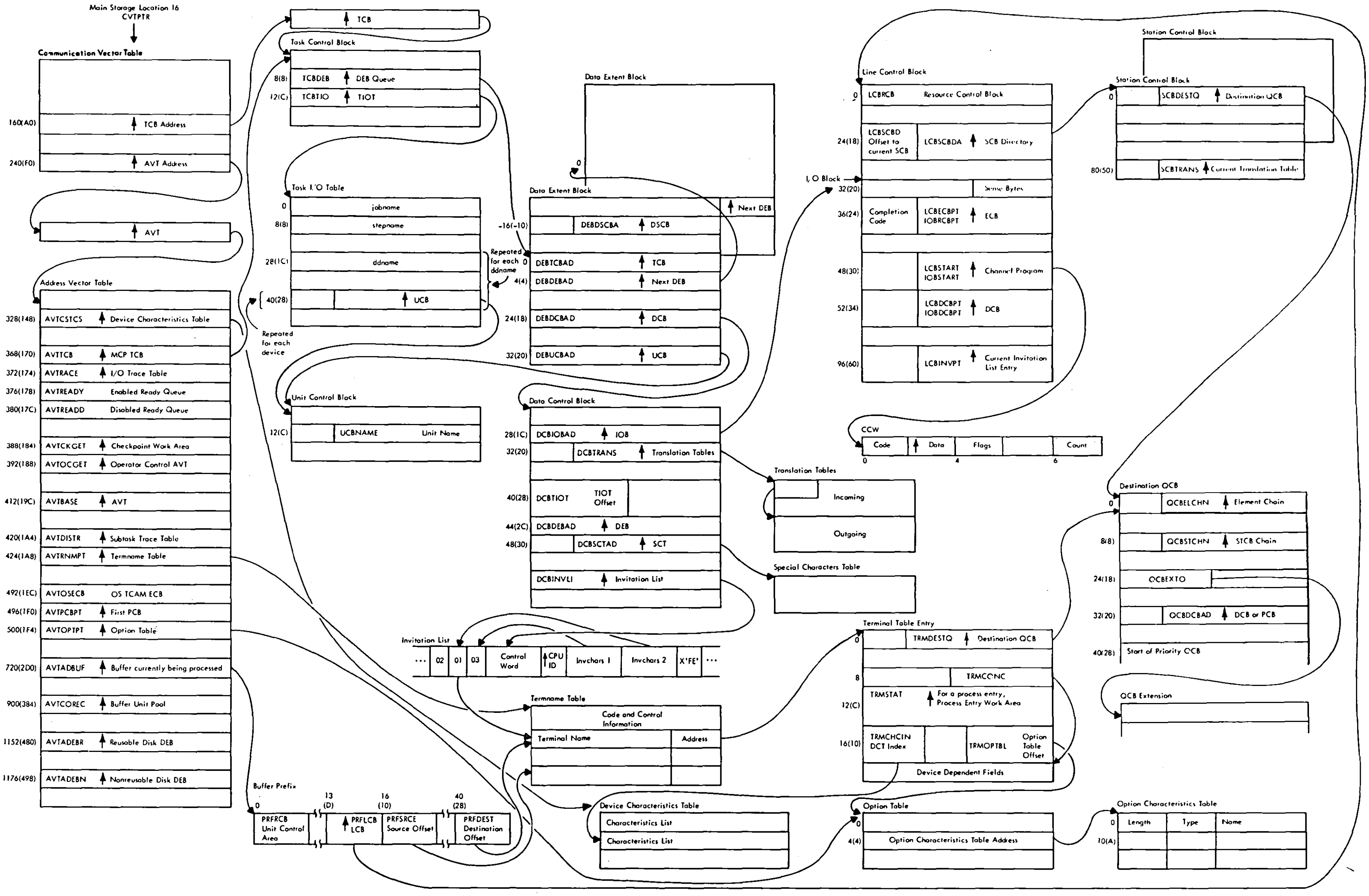


Figure 55. TCAM Control Block Linkages

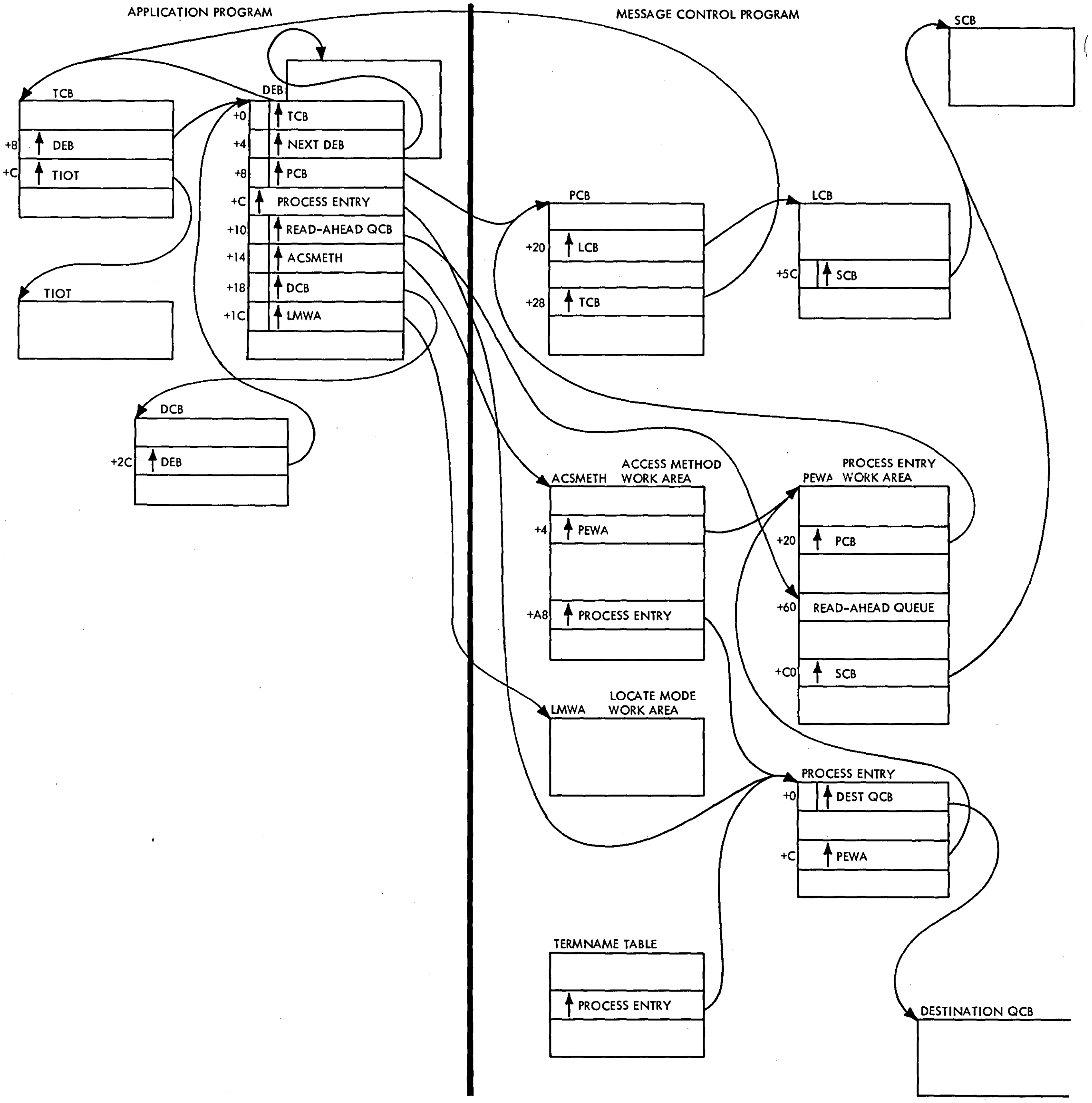


Figure 56. TCAM Control Block Linkages Between an Application Program and the MCP

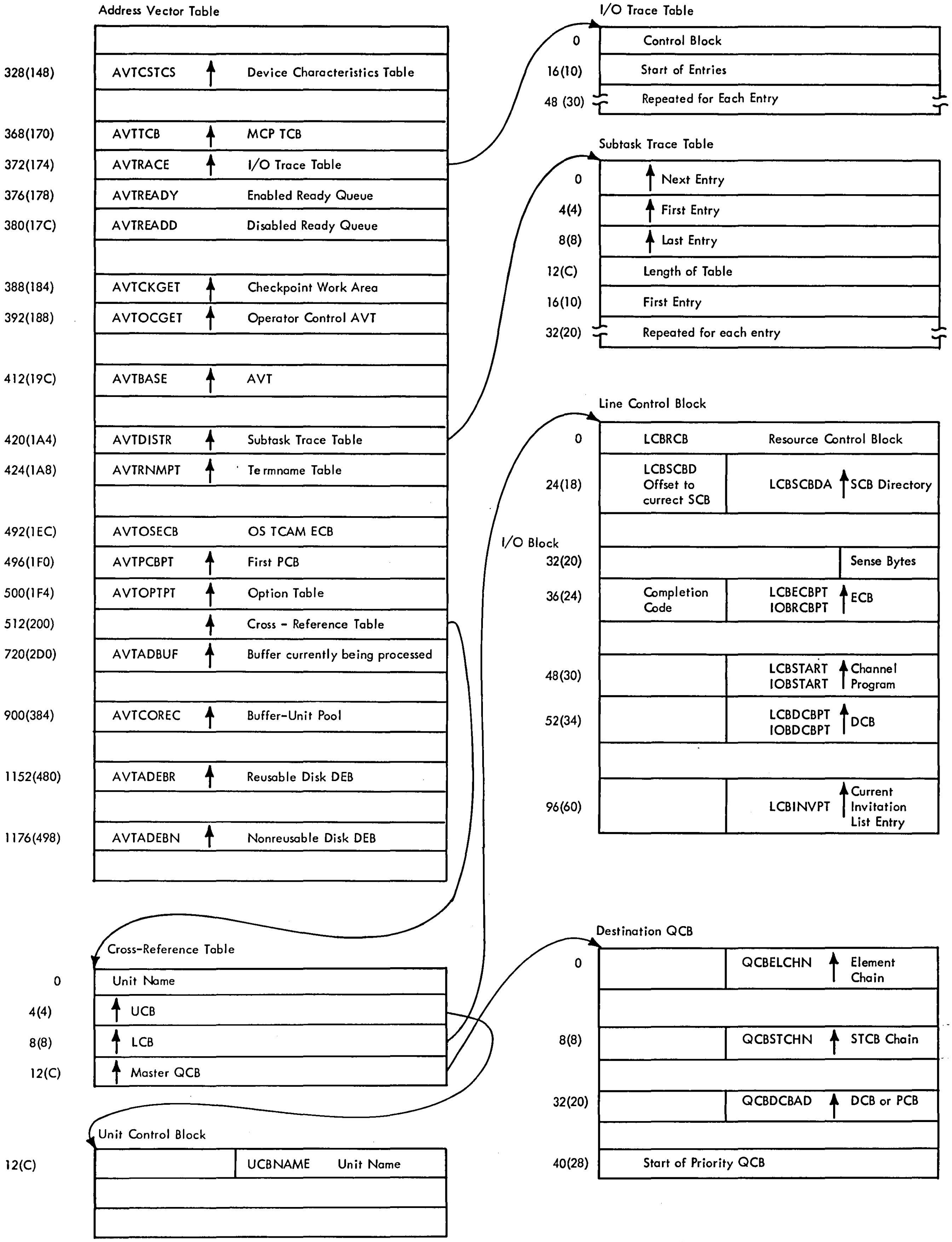


Figure 57. Linkages of TCAM Diagnostic Aids

Appendix B. TCAM Macro Operand Summary

The following figures (58-63) summarize the TCAM macros. They include the macros defining terminal and line control (Figure 58, three parts), the macros defining MCP data sets (Figure 59, three parts), the macros for activation and deactivation (Figure 60, five parts), the MH macros (Figure 61, 10 parts), the application program macros (Figure 62, seven parts), and other TCAM macros (Figure 63).

The abbreviations used in these figures are:

Abbreviation	*Meaning*
SYM	Any symbol valid in the assembler language.
DEC DIG	Any decimal digits, up to the value indicated in the associated macro description.
REG	A general register, always coded within parentheses.
RX-TYPE	Any address that is valid in an RX-type instruction.
A-TYPE ADCON	Any address that may be written in an A-type address constant.
HEX DIG	Any hexadecimal digits, up to the value indicated in the associated macro description.
CHARS	Framed or unframed hexadecimal characters, up to the maximum indicated in the associated macro description.

X indicates the appropriate column; for specific coding requirements, see the *TCAM Programmer's Guide.*

Note: *Defaults are underlined.*

DEFINE INVITATION LIST

symbol	INVLIST	ORDER = (statname+invchars,...) [, EOT = hexchar] [, CPUID = address]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
statname	X						
+	+ or -						
invchars						X	
EOT =						X	
CPUID =				X			

DEFINE A LOG ENTRY IN THE TERMINAL TABLE

logname	LOGTYPE	dcbname, BUFSIZE = integer, QUEUES = form

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
dcbname	X						
BUFSIZE =		X					
QUEUES =	MO, MR, MN, DR, or DN						

DEFINE AN OPTION FIELD

symbol	OPTION	typelength

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
typelength							X

Figure 58. TCAM Macros Defining Terminal and Line Control (Part 1 of 3)

DEFINE A TERMINAL OR LINE ENTRY

symbol	TERMINAL	QBY = T, DCB = dcbname, RLN = integer, TERM = type, QUEUES = form [, DIALNO = characters or DIALNO = NONE] [, ADDR = characters] [, LEVEL = (integer,...)] [, CLOCK = time] [, CINTVL = integer] [, BUFSIZE = integer] [, ALTDEST = symbol] [, BFDELAY = integer] [, TBLKSZ = integer] [, NTBLKSZ = (integer, integer)] [, OPDATA = (data,...)] [, SECTERM = YES or SECTERM = NO] [, COMP = YES or COMP = NO] [, UTERM = YES or UTERM = NO]

or

symbol	TERMINAL	QBY=L, DCB = dcbname, RLN = integer, TERM =type, QUEUES = form [, DIALNO = characters or DIALNO = NONE] [, ADDR = characters] [, LEVEL = (integer,...)] [, CLOCK = time] [, CINTVL = integer] [, BUFSIZE = integer] [, ALTDEST = symbol] [, BFDELAY = integer] [, TBLKSZ = integer] [, NTBLKSZ = (integer, integer)] [, OPDATA = (data,...)] [, SECTERM = YES or SECTERM = NO] [, COMP = YES or COMP = NO] [, UTERM = YES or UTERM = NO]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
QBY =	T or L						
DCBNAME =	X						
RLN =		X					
TERM =							X
QUEUES =	MO, MR, MN, DR, or DN						
DIALNO =		X					X
ADDR =						X	
LEVEL =		X					
CLOCK =		X					
CINTVL =		X					
BUFSIZE =		X					
ALTDEST =	X						
BFDELAY =		X					
NTBLKSZ =		X					
TBLKSZ =		X					
OPDATA =	X	X	X	X	X	X	X
SECTERM =	YES or NO						
COMP =	YES or NO						
UTERM =	YES or NO						

Figure 58. TCAM Macros Defining Terminal and Line Control (Part 2 of 3)

DEFINE A LIST ENTRY IN THE TERMINAL TABLE

symbol	TLIST	TYPE = D, LIST = (entry,...)

or

symbol	TLIST	TYPE = C, LIST = (entry,...)

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
TYPE =	D or C						
entry	X						

DEFINE A PROCESS ENTRY

symbol	TPROCESS	PCB = pcbname [, QUEUES = form] [, ALTDEST = entry] [, CKPTSYN = YES or CKPTSYN = NO] [, RECDEL = hexchar] [, SECTERM = YES or SECTERM = NO] [, LEVEL = (integer,...)] [, OPDATA = (data,...)]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
PCB =	X						
QUEUES =	MO, MR, MN, DR, or DN						
ALTDEST =	X						
CKPTSYN =	YES or NO						
SECTERM =	YES or NO						
RECDEL =						X	
LEVEL =		X					
OPDATA =	X	X	0-15	X	X	X	X

DEFINE TERMINAL TABLE BOUNDARIES

[symbol]	TTABLE	LAST = statname [, MAXLEN = integer]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
LAST =	X						
MAXLEN =		X					

Figure 58. TCAM Macros Defining Terminal and Line Control (Part 3 of 3)

DEFINE CHECKPOINT DATA CONTROL BLOCK

ckptdcb	DCB	DSORG = TQ, MACRF = (G, P), DDNAME = ddname, OPTCD = C [, EXLST = address]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
DSORG =	TQ						
MACRF =	(G,P)						
DDNAME =	X						
OPTCD =	C						
EXLST =				X			

Figure 59. TCAM Macros Defining MCP Data Sets (Part 1 of 3)

DEFINE LINE GROUP DATA CONTROL BLOCK

linedcb	DCB	DSORG = TX, MACRF = (G, P), CPRI = R, DDNAME = ddname, MH = mhname, INVLIST = (listname [, B, B or B, A or A, B or A, A],,...) [, INTVL = integer] [, EXLST = address] [, BUFIN = integer or BUFIN = 1] [, BUFOUT = integer or BUFOUT = 2] [, BUFMAX = integer or BUFMAX = 2] [, BUFSIZE = integer] [, PCI = (N, N) or PCI = (R, A) or PCI = (A, N) or PCI = (A, R) or PCI = (A, A)] [, RESERVE = (integer, integer)] [, SCT = table] [, TRANS = table or TRANS = EBCD]

or

linedcb	DCB	DSORG = TX, MACRF = (G, P) , CPRI = E, DDNAME = ddname, MH = mhname, INVLIST = (listname [, B, B or B, A or A, B or A, A],,...) [, INTVL = integer] [, EXLST = address] [, BUFIN = integer or BUFIN = 1] [, BUFOUT = integer or BUFOUT = 2] [, BUFMAX = integer or BUFMAX = 2] [, BUFSIZE = integer] [, PCI = (N, N) or PCI = (N, R) or PCI = (N, A) or PCI = (R, N) or PCI = (R, R) or PCI = (R, A) or PCI = (A, N) or PCI = (A, R) or PCI = (A, A)] [, RESERVE = (integer, integer)] [, SCT = table] [, TRANS = table or TRANS = EBCD]

or

linedcb	DCB	DSORG = TX, MACRF = (G, P), CPRI = S, DDNAME = ddname, MH = mhname, INVLIST = (listname [, B, B or B, A or A, B or A, A],...) [, INTVL = integer] [, EXLST = address] [, BUFIN = integer or BUFIN = 1] [, BUFOUT = integer or BUFOUT = 2] [, BUFMAX = integer or BUFMAX = 2] [, BUFSIZE = integer] [, PCI = (N, N) or PCI = (N, R) or PCI = (N, A) or PCI = (R, N) or PCI = (R, R) or PCI = (R, A) or PCI = (A, N) or PCI = (A, R) or PCI = (A, A)] [, RESERVE = (integer, integer)] [, SCT = table] [, TRANS = table or TRANS = EBCD]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
DSORG =	TX						
MACRF =	(G, P)						
INTVL =		X					
CPRI =	R, E, or S						
DDNAME =	X						
MH =	X						
EXLST =				X			
BUFIN =		X					
BUFOUT =		X					
BUFMAX =		X					
BUFSIZE =		X					
RESERVE =		X					
SCT =							X
TRANS =	X						X
PCI =	(N, R, or A) , (N, R, or A)						
INVLIST =	X						X

Figure 59. TCAM Macros Defining MCP Data Sets (Part 2 of 3)

DEFINE A LOG DATA CONTROL BLOCK

logdcb	DCB	DSORG = PS, MACRF = W, DDNAME = ddname, BLKSIZE = keylen, RECFM = F, NCP = integer, SYNAD = address

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
DSORG =	PS						
MACRF =	W						
DDNAME =	X						
BLKSIZE =		X					
RECFM =	F						
NCP =		X					
SYNAD =				X			

DEFINE A MESSAGE QUEUES DATA CONTROL BLOCK

diskdcb	DCB	DSORG = TQ, MACRF = (G, P), DDNAME = ddname, OPTCD = L [, EXLST = listname] [, THRESH = integer]

or

diskdcb	DCB	DSORG = TQ, MACRF = (G, P), DDNAME = ddname, OPTCD = R [, EXLST = listname]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
DSORG =	TQ						
MACRF =	(G, P)						
DDNAME =	X						
OPTCD =	L or R						
EXLST =				X			
THRESH =		X					

Figure 59. TCAM Macros Defining MCP Data Sets (Part 3 of 3)

CLOSE MCP DATA SET

symbol	CLOSE	(dcbname,,...)[, MF = L or MF = (E, listname)]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
CLOSE Regular Form							
dcbname	X						
CLOSE List Form							
dcbname	X						
MF =	L						
CLOSE Execute Form							
dcbname	X						
MF = (E,	X		(1)				

INITIALIZE TCAM MCP

symbol	INTRO	[PROGID = characters] [, DISK = NO or DISK = YES] [, CPB = integer] [, CIB = integer or CIB = 2] [, PRIMARY = statname or PRIMARY = SYSCON] [, CONTROL = characters or CONTROL = 0] [, KEYLEN = integer] [, UNITSZ = integer] [, LNUNITS = integer] [, MSUNITS = integer] [, MSMAX = integer or MSMAX = 70] [, MSMIN = integer or MSMIN = 50] [, DLQ = statname or DLQ = 0] [, USEREG = integer] [, INTVAL = integer] [, CPINTVL = integer or CPINTVL = 1800] [, CPRCDS = integer or CPRCDS = 2] [, STARTUP = C [Y] [I] or STARTUP = W [Y] [I]] [, CKREQS = integer] [, RESTART = integer] [, PASSWRD = characters or PASSWRD = 0] [, CROSSRF = integer] [, TRACE = integer] [, TREXIT = address] [, DTRACE = integer] [, OLTEST = integer or OLTEST = 10] [, COMWRTE = YES or COMWRTE = NO] [, WTTONE = integer] [, TOPMSG = NO or TOPMSG = YES] [, LINETYP = BISC or LINETYP = STSP or LINETYP = MINI or LINETYP = BOTH] [, FEATURE = (NODIAL or DIAL, NO2741 or 2741, NOTIMER or TIMER)]

Figure 60. TCAM Macros for Activation and Deactivation (Part 1 of 3)

INTRO - INITIALIZE TCAM MCP

Parameter Summary:

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
PROGID =							X
DISK =	YES or NO						
CPB =		X					
CIB =		X					
PRIMARY =	X						X
CONTROL =		0					X
KEYLEN =		X					
UNITSZ =		X					
LNUNITS =		X					
MSUNITS =		X					
MSMAX =		X					
MSMIN =		X					
DLQ =	X	0					
USEREG =		X					
INTVAL =		X					
CPINTVL =		X					
CPRCDS =		X					
STARTUP =	C [Y] [I] or W [Y] [I]						
CKREQS =		X					
RESTART =		X					
PASSWRD =		0					X
CROSSRF =		X					
TRACE =		X					
TREXIT =				X			
DTRACE =		X					
OLTEST =		X					
COMWRTE =	YES or NO						
WTTONE =		X					
TOPMSG =	YES or NO						
LINETYP =	BISC, STSP, MINI, or BOTH						
FEATURE =	NODIAL or DIAL, NO2741 or 2741, NOTIMER or TIMER						

Figure 60. TCAM Macros for Activation and Deactivation (Part 2 of 3)

OPEN MCP DATA SET

[symbol]	OPEN	(dcbname [, (OUTPUT or INOUT or INPUT [, IDLE])] ,...) [, MF = L or MF = (E, listname)]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
OPEN Regular Form							
dcbname	X						
type	OUTPUT, INOUT, or INPUT						
status	IDLE						
OPEN List Form							
dcbname	X						
type	OUTPUT, INOUT, or INPUT						
status	IDLE						
MF =	L						
OPEN Execute Form							
dcbname	X						
type	OUTPUT, INOUT, or INPUT						
status	IDLE						
MF = (E,	X		(1)				

COMPLETE TCAM INITIALIZATION AND ACTIVATION

[symbol]	READY	[GMMSG = address] [, RSMSG = address]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
GMMSG =				X			
RSMSG =				X			

Figure 60. TCAM Macros for Activation and Deactivation (Part 3 of 3)

CANCEL A MESSAGE

[symbol]	CANCELMG	[mask] [, CONNECT = AND or CONNECT = OR]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
mask		X				X	
CONNECT=	AND or OR						

TAKE INCIDENT CHECKPOINTS OF OPTION FIELDS

[symbol]	CHECKPT	

TRANSLATE A MESSAGE

[symbol]	CODE	[tablename or (register) or NONE]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
tablename	X		(0-12,15)				X

KEEP A COUNT OF MESSAGES

[symbol]	COUNTER	opfld

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
opfld	X						

CUT OFF RECEPTION OF A MESSAGE

[symbol]	CUTOFF	integer

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
integer		X				X	

Figure 61. TCAM Message Handler Macros (Part 1 of 10)

INSERT DATE OR TIME IN A MESSAGE

[symbol]	DATETIME	[DATE = NO or DATE = YES] [, TIME = NO or TIME = YES]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
DATE =	YES or NO						
TIME =	YES or NO						

SEND AN ERROR MESSAGE

[symbol]	ERRORMSG	[mask] [, CONNECT = AND or CONNECT = OR] , DATA = message [, DEST = destname or DEST = opfld or DEST= ORIGIN or DEST = DESTIN] [, EXIT = address]

or

[symbol]	ERRORMSG	[mask] [, CONNECT = AND or CONNECT = OR] , DATA = address [, DEST = destname or DEST = opfld or DEST = ORIGIN or DEST = DESTIN] [, EXIT = address]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
mask		X				X	
DATA =				X			X
DEST =	X					X	X
EXIT =				X			
CONNECT=	AND or OR						

Figure 61. TCAM Message Handler Macros (Part 2 of 10)

FORWARD A MESSAGE

[symbol]	FORWARD	[DEST = destname or DEST = opfld or DEST = (number) or DEST = PUT or DEST = **] [, EOA = characters] [, EXIT = address]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
DEST =	X	(X)					X
EOA =						X	X
EXIT =				X			

SUSPEND MESSAGE TRANSMISSION

[symbol]	HOLD	[mask] [, RELEASE] [, INTVL = integer] [, CONNECT = AND or CONNECT = OR]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
mask		X				X	
RELEASE	Written as shown						
INTVL =		X				X	
CONNECT =	AND or OR						

DEFINE START OF INBUFFER SUBGROUP

[symbol]	INBUF	[PATH = (opfld, switch)]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
opfld	X						
switch		X				X	

DEFINE END OF INCOMING GROUP

[symbol]	INEND	

DEFINE START OF INHEADER SUBGROUP

[symbol]	INHDR	[PATH = (opfld, switch)]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
opfld	X						
switch		X				X	

Figure 61. TCAM Message Handler Macros (Part 3 of 10)

EXPEDITE MESSAGE DISTRIBUTION

[symbol]	INITIATE	[conchars [, BLANK = character or BLANK = NO or BLANK = YES]]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
conchars						X	X
BLANK =						X	X

DEFINE START OF INMESSAGE SUBGROUP

[symbol]	INMSG	[PATH = (opfld, switch)]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
opfld	X						
switch		X				X	

LOCK STATION TO APPLICATION PROGRAM

[symbol]	LOCK	[EXTEND or MESSAGE] [, conchars [, BLANK = character or BLANK = NO or BLANK = YES]]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
type	MESSAGE or EXTEND						
conchars						X	X
BLANK =						X	X

LOCATE OPTION FIELDS

[symbol]	LOCOPT	opfld [, (register) or (15)]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
opfld	X						
register			(2-12,15)				

LOG MESSAGES OR SEGMENTS

[symbol]	LOG	dcbname or typename

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
dcbname	X						
typename	X						

Figure 61. TCAM Message Handler Macros (Part 4 of 10)

EDIT A MESSAGE

[symbol]	MSGEDIT	((I or R [A] [T] , characters or (hexform,n) or DELIMIT or CONTRACT] [, characters or offset or (integer, opfld) or SCAN] [, characters or offset or SCAN or (count) or (0)]),...) [, BLANK = character or BLANK = NO or BLANK = YES]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
function	I or R [A] [T]						
data	DELIMIT or CONTRACT						
characters						X	X
(hexform						X	
,n)		X					

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
AT	SCAN						
characters						X	X
offset		X					
(integer		X					
,opfld)	X						
TO	SCAN or (0)						
characters						X	X
offset		X					
(count)		X					

FORMAT A MESSAGE

[symbol]	MSGFORM	[BLOCK = integer] [, SUBBLCK = integer] [, SENDTRP = YES or SENDTRP = NO]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
BLOCK =		X				X	
SUBBLCK =		X				X	
SENDTRP =	YES or NO						

Figure 61. TCAM Message Handler Macros (Part 5 of 10)

GENERATE A MESSAGE

[symbol]	MSGGEN	[mask] , message [, CONNECT = AND or CONNECT = OR] [, CODE = tablename or CODE = NO]

or

[symbol]	MSGGEN	[mask] , fldname [, CONNECT = AND or CONNECT = OR] [, CODE = tablename or CODE = NO]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
mask		X				X	
message						X	X
CONNECT =	AND or OR						
CODE =				X			X
fldname				X			

LIMIT MESSAGES

[symbol]	MSGLIMIT	integer or opfld

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
integer		X				X	
opfld	X						

DEFINE MESSAGE TYPE

[symbol]	MSGTYPE	[conchars [, BLANK = character or BLANK = NO or BLANK = YES]]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
conchars						X	X
BLANK =						X	X

IDENTIFY MESSAGE ORIGIN

[symbol]	ORIGIN	[integer or X'FF']

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
length		X				X	

Figure 61. TCAM Message Handler Macros (Part 6 of 10)

DEFINE START OF OUTBUFFER SUBGROUP

[symbol]	OUTBUF	[PATH = (opfld, switch)]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
opfld	X						
switch		X				X	

DEFINE END OF OUTGOING GROUP

[symbol]	OUTEND	

DEFINE START OF OUTHEADER SUBGROUP

[symbol]	OUTHDR	[PATH = (opfld, switch)]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
opfld	X						
switch		X				X	

DEFINE START OF OUTMESSAGE SUBGROUP

[symbol]	OUTMSG	[PATH = (opfld, switch)]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
opfld	X						
switch		X				X	

SET A PATH SWITCH

[symbol]	PATH	switch, opfld [, conchars [, BLANK = character or BLANK = NO or BLANK = YES]]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
switch		X				X	
opfld	X						
conchars						X	X
BLANK =						X	X

Figure 61. TCAM Message Handler Macros (Part 7 of 10)

DEFINE MESSAGE PRIORITY

[symbol]	PRIORITY	[integer] [, conchars [, BLANK = character or BLANK = NO or BLANK = YES]]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
integer		X					
conchars						X	X
BLANK =						X	X

REDIRECT A MESSAGE

[symbol]	REDIRECT	[mask] [, CONNECT = AND or CONNECT = OR] [, DEST = destname or DEST = opfld or DEST = ORIGIN]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
mask		X				X	
CONNECT =	AND or OR						
DEST =	X						X

SET DISPLAY SCREEN

[symbol]	SCREEN	[WRE or WLA or WDC] [, conchars [, BLANK = character or BLANK = NO or BLANK = YES]]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
type	WRE, WLA, or WDC						
conchars						X	X
BLANK =						X	X

INSERT OR VERIFY MESSAGE SEQUENCE

[symbol]	SEQUENCE	

SET END OF FILE

[symbol]	SETEOF	[conchars [, BLANK = character or BLANK = NO or BLANK = YES]]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
conchars						X	X
BLANK =						X	F/U

Figure 61. TCAM Message Handler Macros (Part 8 of 10)

SET SCAN POINTER

[symbol]	SETSCAN	skipchars [, BLANK = character or BLANK = NO or BLANK = YES] [, POINT = FORWARD] [, MOVE = RETURN or MOVE = KEEP] [, RESULT = (register) or RESULT = (15)]

or

[symbol]	SETSCAN	integer [, BLANK = character or BLANK = NO or BLANK = YES] [, POINT = BACK or POINT = FORWARD] [, MOVE = KEEP] or MOVE = RETURN [, RESULT = (15) or RESULT = (register)]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
skipchars						X	X
integer		X					
BLANK =						X	X
POINT =	BACK or FORWARD						
MOVE =	RETURN or KEEP						
RESULT =			(2-12,15)				

Figure 61. TCAM Message Handler Macros (Part 9 of 10)

DEFINE START OF MH

symbol	STARTMH	LC = IN [, STOP = YES or STOP = (opfld, switch)] [, CONV = YES or CONV = (opfld, switch) or CONV = NO] [, LOGICAL = opfld or LOGICAL = (opfld, switch, opfld)] [, BREG = integer or BREG = 1] [, CONT = YES or CONT = (opfld, switch)]

or

symbol	STARTMH	LC = OUT [, STOP = YES or STOP = (opfld, switch)] [, CONV = YES or CONV = (opfld, switch) or CONV = NO] [, LOGICAL = opfld or LOGICAL = (opfld, switch, opfld)] [, BREG = integer or BREG = 1] [, CONT = YES or CONT = (opfld, switch)]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
LC =	IN or OUT						
STOP =	YES or						
(opfld	X						
,switch)		X				X	
CONT =	YES or						
(opfld	X						
,switch)		X				X	
CONV =	YES or NO or						
(opfld	X						
,switch)		X				X	
LOGICAL = opfld	X						
LOGICAL = switch		X				X	
BREG =		X					

SET USER ERROR BIT

[symbol]	TERRSET	

UNLOCK A LOCKED STATION

[symbol]	UNLOCK	[conchars [, BLANK = character or BLANK = NO or BLANK = YES]]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
conchars						X	X
BLANK =						X	X

Figure 61. TCAM Message Handler Macros (Part 10 of 10)

WAIT FOR AND TEST COMPLETION OF A READ OR WRITE

[symbol]	CHECK	decbname

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
decbname	X						

REQUEST A CHECKPOINT

[symbol]	CKREQ	

CLOSE APPLICATION PROGRAM DATA SET

[symbol]	CLOSE	(dcbname,,...) [, MF = L or MF = (E, listname)]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
CLOSE Regular Form							
dcbname	X						
CLOSE List Form							
dcbname	X						
MF =	L						
CLOSE Execute Form							
dcbname	X						
MF = (E,	X		(1)				

Figure 62. TCAM Application Program Macros (Part 1 of 7)

DEFINE INPUT DATA CONTROL BLOCK

dcbname	DCB	DSORG = PS, MACRF = GM [T] , DDNAME = ddname, BLKSIZE = integer [, BUFL = integer] [, LRECL = integer] [, RECFM = F or RECFM = V [B] or RECFM = U] [, OPTCD = [W] [U] [C]] [, EODAD = address] [, SYNAD = address] [, EXLST = address]

or

dcbname	DCB	DSORG = PS, MACRF = GL [T] , DDNAME = ddname, BLKSIZE = integer [, BUFL = integer] [, LRECL = integer] [, RECFM = F or RECFM = V [B] or RECFM = U] [, OPTCD = [W][U][C]] [, EODAD = address] [, SYNAD = address] [, EXLST = address]

or

dcbname	DCB	DSORG = PS, MACRF = R [P] , DDNAME = ddname, BLKSIZE = integer [, BUFL = integer] [, LRECL = integer] [, RECFM = F or RECFM = V [B] or RECFM = U] [, OPTCD = [W][U][C]] [, EODAD = address] [, SYNAD = address] [, EXLST = address]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
DSORG =	PS						
MACRF =	[GM, GMT, GL, GLT, R, RP]						
DDNAME =	X						
BLKSIZE =		X					
BUFL =		X					
LRECL =		X					
RECFM =	F, V, VB, or U						
OPTCD =	W, WU, WC, U, UC, C, or WUC						
EODAD =				X			
SYNAD =				X			
EXLST =				X			

Figure 62. TCAM Application Program Macros (Part 2 of 7)

DEFINE OUTPUT DATA CONTROL BLOCK

dcbname	DCB	DSORG = PS, MACRF = PM, DDNAME = ddname [, BLKSIZE = integer] [, LRECL = integer] [, OPTCD = [W][U][C]] [, SYNAD = address] [, RECFM = F or RECFM = V [B] or RECFM = U] [, EXLST = address] [, BUFL = integer]

or

dcbname	DCB	DSORG = PS, MACRF = PL, DDNAME = ddname [, BLKSIZE = integer] [, LRECL = integer] [, OPTCD = [W][U][C]] [, SYNAD = address] [, RECFM = F or RECFM = V [B] or RECFM = U] [, EXLST = address] [, BUFL = integer]

or

dcbname	DCB	DSORG = PS, MACRF = W, DDNAME = ddname [, BLKSIZE = integer] [, LRECL = integer] [, OPTCD = [W][U][C]] [, SYNAD = address] [, RECFM = F or RECFM = V [B] or RECFM = U] [, EXLST = address] [, BUFL = integer]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
DSORG =	PS						
MACRF =	PM, PL, or W						
DDNAME =	X						
BLKSIZE =		X					
LRECL =		X					
OPTCD =	W, WU, WC, WUC, U, UC, or C						
SYNAD =				X			
RECFM =	F, V, VB, or U						
EXLST =				X			
BUFL =		X					

GET A WORK UNIT

[symbol]	GET	dcbname [, areaname]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
dcbname	X						
areaname				X			

Figure 62. TCAM Application Program Macros (Part 3 of 7)

CHANGE INVITATION LIST

[symbol]	ICHNG	grpname, rln, areaname [, PASSWRD = characters]

or

[symbol]	ICHNG	grpname, rln, ACT [, PASSWRD = characters]

or

[symbol]	ICHNG	grpname, rln, DEACT [, PASSWRD = characters]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
grpname	X		(0-14)				
rln	X		(0-14)				
type			(0-14)	X			X
PASSWRD =							X

COPY INVITATION LIST

[symbol]	ICOPY	grpname, rln, areaname

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
grpname	X		(1-15)				
rln		X					
areaname			(0, 1)	X			

CLOSE THE TCAM SYSTEM

[symbol]	MCPCLOSE	[QUICK or FLUSH] [, PASSWRD = characters]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
type	QUICK or FLUSH						
PASSWRD =							X

RELEASE A HELD STATION

[symbol]	MRELEASE	statname [, PASSWRD = characters]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
statname	X						
PASSWRD =							X

Figure 62. TCAM Application Program Macros (Part 4 of 7)

OPEN APPLICATION PROGRAM DATA SET

[symbol]	OPEN	(dcbname,,...) [, MF = L or MF = (E, listname)]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
OPEN Regular Form							
dcbname	X						
OPEN List Form							
dcbname	X						
MF =	L						
OPEN Execute Form							
dcbname	X						
MF = (E,	X		(1)				

DEFINE PROCESS CONTROL BLOCK

symbol	PCB	MH = mhname, BUFSIZE = integer [, BUFIN = integer or BUFIN = 2] [, BUFOUT = integer or BUFOUT = 2] [, RESERVE = (integer, integer)]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
MH =	X						
BUFSIZE =		X					
BUFIN =		X					
BUFOUT =		X					
RESERVE =		X					

POINT TO A RECORD TO BE RETRIEVED

[symbol]	POINT	dcbname, address

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
dcbname	X						
address				X			

PUT A WORK UNIT

[symbol]	PUT	dcbname [, areaname]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
dcbname	X						
areaname				X			

Figure 62. TCAM Application Program Macros (Part 5 of 7)

COPY QUEUE CONTROL BLOCK

[symbol]	QCOPY	statname, areaname

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
statname	X		(0, 2-15)				
areaname			(1-15)	X			

READ A WORK UNIT

[symbol]	READ	decbname, SF, dcbname, areaname [, length or 'S']

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
decname	X						
SF	written as shown						
dcbname	X						
areaname				X			
length		X					'S'

CHANGE TERMINAL-TABLE ENTRY

[symbol]	TCHNG	statname, areaname [, PASSWRD = characters]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
statname	X						
areaname				X			
PASSWRD =							X

COPY TERMINAL-TABLE ENTRY

[symbol]	TCOPY	statname, areaname

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
statname	X		(1)				
areaname			(0)	X			

Figure 62. TCAM Application Program Macros (Part 6 of 7)

WRITE A WORK UNIT

[symbol]	WRITE	decbname, SF, dcbname, areaname [, length or 'S']

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
decbname	X						
SF	written as shown						
dcbname	X						
areaname				X			
length		X					'S'

Figure 62. TCAM Application Program Macros (Part 7 of 7)

START QTAM APPLICATION PROGRAM TO RUN WITH TCAM

[symbol]	QSTART	

EDIT IBM 50 MDI CONTROL CHARACTERS

[symbol]	TPEDIT	MINLN = N [, EDIT = EDITR or EDIT = EDITD] [, RECFM = U or RECFM = V] [, ERROPT = name or ERROPT = IGNORE] [, VERCHK = VOKCHK or VERCHK = NOCHK] [, REPLACE = X'nn' or REPLACE = X'19'] [, BUFFER = YES or BUFFER = NO]

PARAMETER WRITTEN AS	SYM	DEC DIG	REG	RX-TYPE	A-TYPE ADCON	HEX DIG	CHARS
MINLN=		X					
EDIT=	EDITR or EDITD						
RECFM=	U or V						
ERROPT=					X		X
VERCHK=	VOKCHK or NOCHK						
REPLACE=						X	
BUFFER=	YES or NO						

Figure 63. Other TCAM Macros

Appendix C. TCAM Formatted ABEND Dump

A formatted TCAM dump is automatically produced as a part of the OS ABEND/SNAP storage dump when TCAM is resident in the system. ABEND/SNAP storage dumps occur immediately after an abnormal termination, provided that the control program or problem program has issued an ABEND or SNAP macro instruction, or when the operator issues a CANCEL command that requests a dump, and the proper dump data sets have been defined.

The TCAM part of an MFT dump starts after the TRACE TABLE entries, and in an MVT dump, the TCAM part starts after the SAVE AREA TRACE entries. For a complete discussion of the OS portion of the dump, see the *Guide to Reading Dumps.*

The following discussion of the TCAM part of either the OS MFT or MVT dump is interspersed with sample sections from an ABEND dump. Capital letters represent the headings found in all dumps, and lowercase letters represent information that varies. The lowercase letter used indicates the mode of the information, and the number of letters indicate the length of the information.

- h represents 1/2 byte of hexadecimal information
- d represents one byte of decimal information
- c represents a one-byte character

```
TCAM ADDRESS VECTOR TABLE  hhhhhh

SAVE AREA 1

0000     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0020     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0040     hhhhhhhh hhhhhhhh

SAVE AREA 2

0048                       hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0060     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0080     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh

SAVE AREA 3

0090                                            hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
00A0     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
00C0     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh

SAVE AREA 4

00D8                                                              hhhhhhhh hhhhhhhh
00E0     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0100     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh

DISABLED SAVE AREA

0120     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0140     hhhhhhhh hhhhhhhh
```

TCAM ADDRESS VECTOR TABLE hhhhhh is the starting address of the TCAM address vector table (AVT), which is generated by the INTRO macro instruction. The formatted dump of the AVT beginning with the first save area, labeled save area 1, follows the TCAM ADDRESS VECTOR TABLE hhhhhh heading.

SAVE AREA 1 is the contents of the first save area defined in the AVT. The registers are saved in and restored from this area according to standard linkage conventions. Along the left-hand side of the dump are the relative offsets of this save area from the beginning of the AVT.

SAVE AREA 2 is the contents of the second save area defined in the AVT. The registers are saved in and restored from this area according to standard linkage conventions. Along the left-hand side of the dump are the relative offsets of this save area from the beginning of the AVT.

SAVE AREA 3 is the contents of the third save area defined in the AVT. The registers are saved in and restored from this area according to standard linkage conventions. Along the left-hand side of the dump are the relative offsets of this save area from the beginning of the AVT.

SAVE AREA 4 is the contents of the fourth save area defined in the AVT. The registers are saved in and restored from this area according to standard linkage conventions. Along the left-hand side of the dump are the relative offsets of this save area from the beginning of the AVT.

DISABLED SAVE AREA is the contents of the fifth save area defined in the AVT. When a disabled TCAM routine gains control from the I/O supervisor, it saves and restores consecutively the I/O supervisor's registers 0 through 9 in this save area.

```
TABLE POINTERS

0148                       hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0160     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh
```

TABLE POINTERS are the addresses of the first device characteristics table; three work areas used by the internal TCAM logic; the operator control message identification string; the scrambled password character string; the TCAM MCP TCB; and the TCAM I/O trace table. The following chart shows the different fields, their offsets relative to the beginning of the AVT (which are also given on the left-hand side of the dump), their length, and their contents.

+0148	Address of the first DCT entry
+014C	Disabled parameter list
+0150 +0154	Disabled doubleword scratch area
+0158 +015C	Enabled doubleword scratch area
+0160 +0164	The operator control message identification character string
+0168 +016C	The scrambled password character string
+0170	Address of the TCB of the TCAM MCP
+0174	Address of the TCAM line I/O trace table

```
DISPATCHER READY QUEUES

0178                                                               hhhhhhhh hhhhhhhh
0180      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
01A0      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
01C0      hhhhhhhh hhhhhhhh hhhhhhhh
```

DISPATCHER READY QUEUES gives the contents of the TCAM dispatcher ready queues (one enabled, one disabled) and various other fields in this section of the AVT. The different fields, their offsets relative to the beginning of the AVT (which are also given on the left-hand side of the dump), their length, and their contents are illustrated below.

+0178	Enabled ready queue (points to first element to be dispatched)				
+017C	First word of the disabled FIFO ready queue				
+0180	Second word of the disabled FIFO ready queue				
+0184	Checkpoint work area				
+0188	Operator control work area				
+018C +0190 +0194	Executable instructions to save the user's registers, if requested				
+0198	Parameter list				
+019C	Protection key	+019D	Address of the AVT		
+01A0	Address of additional optional parameters				
+01A4	Address of the TCAM dispatcher subtask trace table				
+01A8	Address of the termname table				
+01AC	User exit address in the READY macro expansion				
+01B0	Address of the Line End Appendage BSC message scan subroutine (SCAN)				
+01B4	Address of the Line I/O Interrupt Trace routine (IGG019Q0)				
+01B8 +01BC	Tpost parameter list used by operator control				
+01C0	Address of start parameter list				
+01C4	Number of CIBs	+01C5	Number of checkpoint requests	+01C6	Number of line units
+01C8	Address of Hold/Release Terminal routine (IEDQAS)				

```
TCB POINTERS

01CC                               hhhhhhhh     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
```

TCB POINTERS give the addresses of the TCBs for checkpoint, operator control, on-line test, and the FE Common Write (COMWRITE) task. These tasks are attached tasks of the TCAM MCP. The following chart shows the fields containing the addresses and the offsets of the fields relative to the beginning of the AVT.

+01CC	Address of the Checkpoint TCB
+01D0	Address of the Operator Control TCB
+01D4	Address of the On-Line Test TCB
+01D8	Address of the FE Common Write TCB

```
ECBS

01DC                                                                          hhhhhhhh
01E0    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0200    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0220    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0240    hhhhhhhh hhhhhhhh hhhhhhhh
```

ECBs contain the addresses of some of the internal routines and subtasks of the TCAM MCP, the addresses of certain TCAM tables, the checkpoint ECB, the on-line test ECB, the operator control ECB, the ECB used by the TCAM dispatcher to cause TCAM to enter a wait state when the ready queues are empty, and the address of the FE Common Write (COMWRITE) ECB. The following example gives a list of the different fields, their contents, and their relative offsets from the beginning of the AVT (which are also given on the left-hand side of the dump).

+01DC	Address of the FE Common Write ECB
+01E0	Checkpoint ECB
+01E4	On-Line Test ECB
+01E8	Operator Control ECB
+01EC	ECB used by the dispatcher to cause TCAM to be in wait state
+01F0	Address of the first process entry control block
+01F4	Address of the option table
+01F8	Address of the I/O Generator in the Activate subtask
+01FC	Address of the user trace exit
+0200	Address of the cross-reference table
+0204	Address of the communications parameter list
+0208	Address of the User Interface routine (IEDQUI)
+020C	Address of the Return Interface routine (IEDQLM)
+0210	Address of the routine to remove an element from the Time Delay QCB (IEDQHG02)
+0214	Address of the Address Finder routine (IEDQAL)
+0218	Address of the Buffer Association routine (IEDQGD)
+021C	Address of the Transparency CCW Builder routine (IEDQGT)
+0220	Address of the Buffer Step routine (IEDQAX)
+0224	Address of the TCAM Dispatcher (IGG019RB or IGG019RO)
+0228	Address of the Leased Receive Scheduler (IGG019R3)

+022C	Address of the Send Scheduler (IGG019R4)
+0230	Address of the Get Scheduler (IEDQEW)
+0234	Address of the Put Scheduler (IEDQEC)
+0238	Address of the Get FIFO Scheduler (IEDQEZ)
+023C	Address of the Log Scheduler (IEDQBZ)
+0240	Address of the Dial Receive Scheduler (IGG019R1)
+0244	Address of the Buffered Terminal Scheduler (IGG019RD)
+0248	Address of the Retrieve Scheduler (IEDQE7)

```
SPECIAL ELEMENTS

024C                                      hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0260     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0280     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
02A0     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
02C0     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh
```

SPECIAL ELEMENTS contain the interval checkpoint element, a special element to request removal of the interval checkpoint element for the time delay queue, the incident checkpoint element, and several address and constant areas used by the internal TCAM logic. The following chart gives a list of the different fields, their contents, their size, and their relative offsets from the beginning of the AVT.

+024C	Reserved
+0250	Reserved
+0254	Reserved
+0258	Reserved
+025C	Reserved
+0260	Reserved
+0264	Reserved
+0268	Reserved
+026C	Reserved
+0270	Dummy Line ECB
+0274	Address of the translation list for IEDQA3
+0278	Address of the World Trade tone characters
+027C	Address of the Operator Awareness Message Router routine (IEDQNX)
+0280	Address of the I/O Trace Table Handler routine (IGG019Q0)
+0284	Address of the System Delay QCB
+0288	Address of the Stop Line QCB
+028C	Special element to cause removal of the checkpoint element from the time delay queue
+029C +02A0	Element to request interval checkpoint
+02A4	Size of SCB
+02A5	Address of Checkpoint QCB
+02A8	Checkpoint request element flags
+02A9	Number of checkpoint records
+02AA	Checkpoint time interval
+02AC	Time of day of interrupt
+02AE	Offset to Checkpoint QCB
+02AF	Open error locator
+02B0	Open module ID having error
+02B2	Type of Open error
+02B3	Checkpoint time delay status
+02B4	Open translate byte
+02B5	Address of Time Delay subroutine (IEDQHG01)

+02B8	Offset to Binary Search routine	+02B9	Link field on time queue
+02BC	Dummy last element		
+02C0	Address of dummy last element		
+02C4 +02C8	Incident checkpoint element		
+02CC	Halfword constant X'0000'	+02CE	Halfword constant X'FFFF'
+02D0	Address of current buffer being processed (by message handler)		
+02D4	Address of the 2260 Local Line End Appendage (IGG019R5)		
+02D8	System error flag byte	+02D9	Address of list of V-type address constants

```
QCB POINTERS

02DC                                                                           hhhhhhhh
02E0      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0300      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0320      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0340      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0360      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0380      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
```

QCB POINTERS contain the available buffer QCB, the buffer return QCB, the checkpoint QCB, the operator control QCB, the on-line test QCB, the activate QCB, the closedown QCB, the QCB to remove the checkpoint element from the time-delay queue, the disk I/O QCB, the CPB cleanup QCB, the address of the start-up message QCB, the address of the time-sharing input QCB, the address of the application program OPEN/CLOSE routine, the address of the first byte of main storage obtained by GETMAIN for the buffer-unit pool, a word containing the number of buffer units being used by the main-storage message queues data set, and a fullword constant of zeros. The following chart gives a list of the different fields, their contents, their size, and their relative offsets from the beginning of the AVT.

+02DC	Queue of available insert blocks	
+02E0	Address of the Start-up Message QCB	
+02E4	Address of the Time Sharing Input QCB	
+02E8	Address of the application program OPEN/CLOSE routine (IEDQEU)	
+02EC	Time Delay QCB	
+02FC	Reference time	+02FE Dummy INEND/OUTEND AVT
+0300	SVC 102 parameter list, used to cause SVC 102 to tpost the time	
+0304	Delay QCB to itself when a timer interrupt occurs	
+0308	Time delay queue	
+030C	Available Buffer QCB	
+0318	Buffer Return QCB	
+0324	Checkpoint QCB	
+0330	Operator Control QCB	
+033C	On-Line Test QCB	
+0348	Activate QCB	
+0354	Closedown QCB	
+0360	QCB to remove checkpoint element from time delay queue	
+036C	Disk I/O QCB	
+0378	CPB Cleanup QCB	
+0384	Address of area obtained by GETMAIN for buffer-unit pool	
+0388	Number of buffer units being used by main-storage message queues	
+038C	Fullword constant of zero	

```
INTERFACE

0390                                                hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
03A0     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
03C0     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
03E0     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0400     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
```

INTERFACE contains a GETMAIN parameter list used to obtain the buffer-unit pool, the key length specified for the message queues, the number of lines opened, the number of lines in the system delay, the offset into the termname table of the primary operator control terminal, the number of buffer units in the buffer pool, the number of lines serviced by the Start-up Message subtask, the number of seconds of the system delay, the offset into the termname table of the dead-letter queue terminal, three flags and several constants used by the internal TCAM logic, the number of restart checkpoint records, the number of buffers or CPBs on the EXCP or retry queue, and the address of the FE Patch module used for additional serviceability routines. Also, there are an FE work area, two parameter list pointers, two ECBs, and four flag bytes all used by the FE Common Write (COMWRITE) subtask. The following chart gives a list of the different fields, their contents, their size, and their relative offsets from the beginning of the AVT.

+0390	Address of the FE Patch module (IEDQFE)
+0394	First parameter list pointer
+0398	First ECB
+039C	FE flag byte 1
+039D	FE flag byte 2
+039E	FE flag byte 3
+039F	FE flag byte 4
+03A0	Second parameter list pointer
+03A4	Second ECB
+03A8	FE work area
+03AC	
+03B0	
+03B4	
+03B8	
+03BC	
+03C0	
+03C4	
+03C8	
+03CC	
+03D0	
+03D4	
+03D8	
+03DC	
+03E0	
+03E4	
+03E8	

+03EC					
+03F0					
+03F4	GETMAIN parameter list				
+03F8					
+03FC		+03FE	Halfword constant of 2		
+0400	Halfword constant of 3	+0402	Halfword constant of 4		
+0404	Halfword constant of 7	+0406	Halfword constant of 16		
+0408	Key length on message queues	+040A	Number of lines opened		
+040C	Number of lines in system delay	+040E	Offset to primary operator control terminal		
+0410	Number of buffer units in buffer-unit pool	+0412	Number of lines serviced by Start-up Message subtask		
+0414	Number of seconds of system delay	+0416	Offset to dead-letter terminal		
+0418	BR instruction	+041A	Flag byte 1	+041B	Flag byte 2
+041C	Flag byte 3	+041D	Number of restart checkpoint records	+041E	Number of buffer or CPBs on EXCP or retry queue

Note: *This is the end of the AVT when ENVIRON=TSO has been specified on the INTRO macro instruction.*

```
CORE QUEUE
0420      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh     hhhhhhhh hhhhhhhh
```

CORE QUEUE contains the address of the Destination Assignment routine, the values specified by the MSMIN=, MSMAX=, and MSUNITS= operands of the INTRO macro instruction, and a queue of buffers and ERBs waiting to be processed. The following chart gives a list of the different fields, their contents, their size, and their relative offsets from the beginning of the AVT.

+0420	Address of the Destination Assignment routine (IEDQHM02)
+0424	MSMIN=integer
+0428	MSMAX=integer
+042C	Number of units usable in main-storage queues (MSUNITS=integer)
+0430 +0434	Queue of buffers and ERBs waiting to be processed

```
DISK

0438                                                      hhhhhhhh hhhhhhhh
0440      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0460      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0480      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
04A0      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
04C0      hhhhhhhh hhhhhhhh hhhh
```

DISK contains the queues and control information for the disk message queues (reusable and nonreusable) for the TCAM MCP. The following chart gives a list of the different fields, their contents, their size, and their relative offsets from the beginning of the AVT.

+0438	Address of the Disk EXCP Driver routine (IGG019RC)
+043C	Address of the Reusability subtask (IGG019RP - REUS)
+0440	Address of the Copy subtask QCB (IGG019RP - COPY)
+0444 +0448	Disabled queue of CPBs to be processed by CPB Cleanup
+044C +0450	Enabled queue of CPBs to be processed by CPB Cleanup
+0454 +0458	Queue of CPBs waiting for buffers
+045C +0460	Reserved
+0464 +0468	Queue of CPBs requesting I/O to be done by the Disk EXCP Driver
+046C	Queue of inactive CPBs, called the CPB free pool
+0470	Address of the CPB free pool
+0474	Address of list of IOBs for reusable disk queues
+0478	Address of list of IOBs for nonreusable queues
+047C	Reusable disk queue when Reusability subtask activated
+0480	Address of DEB (reusable disk)
+0484	Number of extents (reusable disk)
+0488	Number of records per track (reusable disk)
+048C	Number of tracks per cylinder (reusable disk)

+0490	Number of records in entire data set (reusable disk)
+0494	Product of number of extents times number of records per track (reusable disk)
+0498	Address pf DEB (nonreusable disk)
+049C	Number of extents (nonreusable disk)
+04A0	Number of tracks per cylinder (nonreusable disk)
+04A4	Number of records per track (nonreusable disk)
+04A8	Number of records in entire data set (nonreusable disk)
+04AC	Product of number of extents times number of records per track (nonreusable disk)
+04B0	Absolute record number that is the threshold to cause closedown due to filling of the nonreusable disk queue
+04B4	Nonreusable disk queue
+04B8	Reusable disk queue
+04BC	Nonreusable Threshold Closedown Element
+04C8	CPB = integer

Note: *This is the end of the AVT.*

```
TNT  hhhhhh  CODE  hhhhhhhh        hhhhhhhh        hhhhhhhh        hhhhhhhh        hhhhhhhh
                   hhhhhhhh        hhhhhhhh        hhhhhhhh        hhhhhhhh        hhhh
             SRCHX hhhh      ENLEN hh        MIDEN hhhhhh     LEN  hhhh
             DCODE hhhhhhhh        hhhhhhhh        hhhhhhhh        hhhhhhhh        hhhhhhhh
                   hhhhhhhh        hhhhhhhh        hhhhhhhh        hhhhhhhh
```

TNT hhhhhh is the address of the TCAM termname table, which contains the names and addresses of all of the terminal-table entries. (Each of the terminal-table entries is displayed following this section of the dump.)

CODE is the executable termname table code that converts the invitation list relative position field into the absolute address of the terminal-table entry. This code is used only by enabled routines.

SRCHX hhhh is the search extent factor.

ENLEN hh is the number of bytes in each entry.

MIDEN hhhhhh is the absolute address of the middle entry.

LEN hhhh is the total number of entries.

DCODE is the executable termname table code that converts the invitation list relative position field into the absolute address of the terminal-table entry. This code is used only by disabled routines.

Following the TNT section of the dump are each of the terminal-table entries along with their option-table entries (if any exist) and contents. Some additional fields in each of the terminal-table entries may or may not be present according to the optional parameters specified on the TERMINAL macro instruction. These are discussed where applicable. Four different types of entries are in the terminal table. They are single entries, list entries (cascade and distribution), process entries, and line entries. The following four sections give an example of each type of entry. Each of the four types of entries has a STATE field, which is the status byte of the terminal-table entry. The following example is a list of the bit meanings of this one-byte status field.

Bit(s)	*Meaning*
0-2	000 = single entry 001 = process entry 010 = list entry (cascade or distribution) 100 = line entry
3	always 0 for a GET-type list or a GET-type process entry always 1 for a single entry, line entry, or a PUT-type process entry
4	0 = a PUT-type process entry (if process entry) 1 = a GET-type process entry (if process entry) (always 1 for other type entries)
5	0 = terminal is not in hold mode 1 = terminal is in hold mode (If this is a process entry, it indicates CKPTSYN=YES was specified on the TPROCESS macro.)
6	0 = no option fields used 1 = option fields used
7	0 = not secondary operator control terminal 1 = secondary operator control terminal

An example of a single entry follows.

```
NAME cccccccc
TRM  hhhhhh  STATE/DESTQ hhhhhhhh   IN/OUTSEQ hhhhhhhh   ALTD/DEVFL hhhhhhhh   STAT hhhhhhhh   CHCIN/OPNO/OPTBL hhhhhhhh

             NAME      ADDR  OPTION FIELD
             cccccccc hhhhhh hhhhhhhh
             cccccccc hhhhhh hhhhhhhh

             BUFFSIZE        hhhh
             DIAL DIGITS     hhhhhhh
             ADDR CHAR       hhhhhh
             BLOCK           hhhh
             SUBBLOCK        hh
             TRANS BLOCK     hhhh
             BFDELAY         hhhh
             TIME SHARING    hhhh
```

NAME cccccccc is the name in the termname table of this terminal-table entry.

TRM hhhhhh is the address of the terminal-table entry.

STATE/DESTQ hhhhhhhh

The first byte is the status byte of the terminal-table entry. The last three bytes contain the address of the destination QCB for this entry.

IN/OUTSEQ hhhhhhhh

The first two bytes contain the next expected input sequence number. The second two bytes contain the next output sequence number to be used.

ALTD/DEVFL hhhhhhhh

The first two bytes contain the offset into the terminal table of the alternate destination for this entry. The last two bytes are flag bytes used by the internal TCAM logic. The bits and their meanings are:

Bit(s)	*Meaning*
0	BUFSIZE= specified
1	dial digits present
2	addressing characters present
3	BLOCK= specified
4	SUBBLCK= specified
5	TRANS= specified
6	BFDELAY= specified
7	TSO field present
8-15	Reserved

STAT hhhhhhhh is a word for error statistics.

CHCIN/OPNO/OPTBL hhhhhhhh

The first byte is the index to the device characteristics table for this entry. The second byte gives the number of option fields for this entry. The next two bytes contain the offset into the option table for the option fields for this entry.

NAME ADDR OPTION FIELD cccccccc hhhhhh hhhhhhhh gives a list of the names, addresses, and contents of each of the option fields for this entry.

BUFFSIZE hhhh is the output-buffer size for this entry. This value is given in the dump only when a nonzero value has been specified on the BUFSIZE= operand of the TERMINAL macro.

DIAL DIGITS hhhhhhh is the telephone number of this terminal. This field is given in the dump only when the CALL= operand of the TERMINAL macro has been specified, except where CALL=NONE was specified.

ADDR CHAR hhhhhh is the addressing characters for the terminal as specified on the ADDR= operand of the TERMINAL macro.

BLOCK hhhh is the number of bytes to be transmitted in each block of data in nontransparent mode for messages sent to this terminal. The value corresponds to the value specified in the BLOCK= operand of the TERMINAL macro and is not given in the dump if the value was not specified.

SUBBLOCK hh is the number of bytes to be transmitted in each subblock of data in nontransparent mode for messages sent to this terminal. The value corresponds to the value specified in the SUBBLCK= operand of the TERMINAL macro and is not given in the dump if the value was not specified.

TRANS BLOCK hhhh is the number of bytes to be transmitted in each block of data in transparent mode for messages sent to this terminal. The value corresponds to the value specified in the TBLKSZ= operand of the TERMINAL macro and is not given in the dump if the value was not specified.

BFDELAY hhhh is the number of seconds of delay to be used between message blocks being sent to a buffered terminal. This field is given in the dump only if the BFDELAY= operand of the TERMINAL macro has been specified.

TIME SHARING hhhh is a field used by TSO. In the case that this entry is an IBM 2260 or an IBM 2265, the first byte is the number of lines that can be displayed and the second byte is the number of characters per line. If the terminal is not an IBM 2260 or an IBM 2265, both bytes are zero. This field is given in the dump only when TSO is being used.

An example of a list entry follows.

```
NAME cccccccc
TRM  hhhhhh  STATE/DESTQ hhhhhhhh  TLISTCNT hhhh

             LIST ENTRIES
               cccccccc
               cccccccc
```

NAME cccccccc is the name in the termname table of this terminal-table entry.

TRM hhhhhh is the address of the terminal-table entry.

STATE/DESTQ hhhhhhhh
The first byte is the status byte of the terminal-table entry. The last three bytes contain the address of the destination QCB.

TLISTCNT hhhh is the number of entries in this distribution or cascade list.

LIST ENTRIES is a list of the names that appear in the cascade or distribution list.

An example of a line entry follows.

```
NAME cccccccc
TRM  hhhhhh  STATE/DESTQ hhhhhhhh   IN/OUTSEQ hhhhhhhh   ALTD/DEVFL hhhhhhhh   STAT hhhhhhhh   CHCIN/OPNO/OPTBL hhhhhhhh

             NAME     ADDR   OPTION FIELD
             cccccccc hhhhhh hhhhhhhh
             cccccccc hhhhhh hhhhhhhh

             ADDR CHAR       hhhhhh
```

NAME cccccccc is the name in the termname table of this terminal-table entry.

TRM hhhhhh is the address of the terminal-table entry.

STATE/DESTQ hhhhhhhh
The first byte is the status byte of the terminal-table entry. The last three bytes contain the address of the destination QCB for this entry.

IN/OUTSEQ hhhhhhhh
The first two bytes contain the next expected input sequence number. The second two bytes contain the next output sequence number to be used.

ALTD/DEVFL hhhhhhhh

The first two bytes contain the offset into the terminal table of the alternate destination for this entry. The last two bytes are flag bytes used by the internal TCAM logic. The following table is a list of the bits and their meanings.

Bit(s)	*Meaning*
0	BUFSIZE= specified
1	dial digits present
2	addressing characters present
3	BLOCK= specified
4	SUBBLCK= specified
5	TRANS= specified
6	BFDELAY= specified
7	TSO field present
8-15	Reserved

STAT hhhhhhhh is a word for error statistics.

CHCIN/OPNO/OPTBL hhhhhhhh

The first byte is the index to the device characteristics table for this entry. The second byte gives the number of option fields for this entry. The next two bytes contain the offset into the option table for the option fields for this entry.

NAME ADDR OPTION FIELDS cccccccc hhhhhh hhhhhhhh gives a list of the names, addresses, and contents of each of the option fields for this entry.

ADDR CHAR hhhh is the addressing characters for the terminal as specified on the ADDR= operand of the TERMINAL macro.

An example of a process entry follows.

```
NAME cccccccc
TRM  hhhhhh  STATE/DESTQ hhhhhhhh   IN/OUTSEQ hhhhhhhh   ALTD/DEVFL hhhhhhhh   STAT hhhhhhhh   CHCIN/OPNO/OPTBL hhhhhhhh

             NAME       ADDR  OPTION FIELD
             cccccccc hhhhhh hhhhhhhh
             cccccccc hhhhhh hhhhhhhh
```

NAME cccccccc is the name in the termname table of this terminal-table entry.

TRM hhhhhh is the address of the terminal-table entry.

STATE/DESTQ hhhhhhhh

The first byte is the status byte of the terminal-table entry. The last three bytes contain the address of the destination QCB for this entry.

IN/OUTSEQ hhhhhhhh

The first two bytes contain the next expected input sequence number. The second two bytes contain the next output sequence number to be used.

ALTD/DEVFL hhhhhhhh

The first two bytes contain the offset into the terminal-table of the alternate destination for this entry. The last two bytes are flag bytes used by the internal TCAM logic. The following table is a list of the bits and their meanings.

Bit(s)	*Meaning*
0	BUFSIZE= specified
1	dial digits present
2	addressing characters present
3	BLOCK= specified
4	SUBBLCK= specified
5	TRANS= specified
6	BFDELAY= specified
7	TSO field present
8-15	Reserved

STAT hhhhhhhh is the address of the process-entry work area (IEDQPEWA) if the corresponding application program DCB is opened.

CHCIN/OPNO/OPTBL hhhhhhhh

The first byte is the index to the device characteristics table for this entry. The second byte gives the number of option fields for this entry. The next two bytes contain the offset into the option table for the option fields for this entry.

NAME ADDR OPTION FIELDS cccccccc hhhhhh hhhhhhhh gives a list of the names, addresses, and contents of each of the option fields for this entry.

```
TCAM DESTINATION QCB'S

QCB  hhhhhh  DSFLG/ELCHN hhhhhhhh   PRI/LINK    hhhhhhhh    STVTO/STCHN hhhhhhhh   STPRI/SLINK hhhhhhhh
             EOLTD/STAT  hhhhhhhh   SCBOF/INSRC hhhhhhhh    INTVL/MSGCT hhhhhhhh   PRLVL/LKRRN hhhhhhhh
             RELLN/DCBAD hhhhhhhh   FLAG/QBACK  hhhhhhhh

     PRIORITY QCB  hhhhhh
             DNHDR hhhhhh   FHDLZ hhhhhh   FHDTZ hhhhhh    INTFF hhhhhh   INTLF hhhhhh
             FFEFO hhhhhh   LFEFO hhhhhh   CFHDR hhhhhh    PRIPQ hh       CPVHD hhhhhh
```

TCAM DESTINATION QCB'S gives the destination QCBs for all of the terminal-table entries. These QCBs are used to control the message queuing for the terminals in the TCAM system. Each QCB may service one or more terminals depending upon the type of queuing specified in the TERMINAL macro. Each of these QCBs consists of a master QCB and one or more priority-level QCBs. Priority QCBs are generated by the LEVEL= operand of the TERMINAL macro. If this operand is omitted, only one priority level QCB is generated and its priority is X'00'. Whether or not the LEVEL= operand is specified, the X'00' priority-level QCB is generated.

QCB hhhhhh is the starting address of the master QCB.

DSFLG/ELCHN hhhhhhhh

The first byte is a flag byte indicating the type of queuing being used by this QCB. The next three bytes contain the address of the next element in the chain.

PRI/LINK hhhhhhhh

The first byte is the priority of this QCB. The last three bytes contain the address if the next STCB in the chain.

STVTO/STCHN hhhhhhhh

The first byte is the index to the entry in the subtask vector table. The last three bytes contain the STCB chain.

STPRI/SLINK hhhhhhhh

The first is the priority of the STCB. The last three bytes contain the address of the next STCB in the chain.

EOLTD/STAT hhhhhhhh

The first two bytes contain the interrupt time used by the time-delay routine. The third byte is the LOCK relative line number, and the fourth byte is the QCB status byte.

SCBOF/INSRC hhhhhhhh

The first byte is the offset to the proper SCB for the current transmission. The next three bytes contain the address of the first LCB in the source LCB chain.

INTVL/MSGCT hhhhhhhh

The first two bytes contain the value as specified on the CLOCK= or INTVL= operand of the TERMINAL macro. The second two bytes contain the count of the messages on this queue.

PRLVL/LKRRN hhhhhhhh

The first byte is the priority of the highest-priority message in the queue. The last three bytes contain the LOCK relative record number.

RELLN/DCBAD hhhhhhhh

The first byte is the relative line number for the line that this QCB represents. The last three bytes contain the address of the DCB.

FLAG/QBACK hhhhhhhh

The first byte is an additional status byte for the QCB. The last three bytes contain the QBACK message chain.

PRIORITY QCB hhhhhh is the address of this priority-level QCB.

DNHDR hhhhhh is the disk record number assigned to the next header that is received.

FHDLZ hhhhhh is the disk record number of the first header placed in the last zone used by this queue.

FHDTZ hhhhhh is the disk record number of the first header placed in the current zone.

INTFF hhhhhh is the disk record number of the first held message in this queue (placed in FEFO order).

INTLF hhhhhh is the disk record number of the last held message in this queue (placed in FEFO order).

FFEFO hhhhhh is the disk record number of the first message that has not been sent (placed in FEFO order).

LFEFO hhhhhh is the disk record number of the last message that has not been sent (placed in FEFO order).

CFHDR hhhhhh is the main-storage queue address of the first header appearing in this queue.

PRIPQ hh is the priority level of this priority-level QCB.

CPVHD hhhhhh is the main-storage queue address of the last header appearing in this queue.

```
TCAM DCB'S

DCB  hhhhhh  (LINE GROUP)
             DEVICE INTERFACE   hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh
             D/S INTERFACE      hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh
             FOUNDATION         hhhhhhhh    hhhhhhhh    hh
             EXTENSION                                    hhhhhh    hhhhhhhh    hhhhhhhh
             INVITATION LISTS   hhhhhhhh

     LCB  hhhhhh  KEY/QCBA    hhhhhhhh   PRI/LINK    hhhhhhhh   RSKEY/STCBA hhhhhhhh   RSPRI/RSLNK hhhhhhhh
                  EOLTD/TSOB  hhhhhhhh   CHAIN/INSRC hhhhhhhh   SCBO/SCBDA  hhhhhhhh   ISZE/FSBFR  hhhhhhhh
                  FLAGS/SENSE hhhhhhhh   ECBCC/ECBPT hhhhhhhh   FLAG3/CSW   hhhhhhhh               hhhhhhhh
                  SIOCC/START hhhhhhhh   DCBPT       hhhhhhhh   RCQCB       hhhhhhhh   INCAM/ERRCT hhhhhhhh
                  UCBX/RCBFR  hhhhhhhh   RECOF/STATE hhhhhhhh   TSTSW/RECAD hhhhhhhh   ERBKY/ERBQB hhhhhhhh
                  ERBPY/ERBLK hhhhhhhh   ERBST/ERBCH hhhhhhhh   ERBCT/TTCIN hhhhhhhh   MSGFM/SCBA  hhhhhhhh
                  ERMSK/INVPT hhhhhhhh   TPCD        hhhhhhhh               hhhhhhhh               hhhhhhhh
                  SNSV/CSWSV  hhhhhhhh               hhhhhhhh   ERCCW       hhhhhhhh               hhhhhhhh
                              hhhhhhhh               hhhhhhhh               hhhhhhhh               hhhhhhhh
```

TCAM DCBs give the three different types of TCAM DCBs: the line group DCBs (along with their related LCBs), the message queues DCBs, and the checkpoint DCB. (The message queues DCBs are not given in the dump if the TCAM system does not use disk queuing, and the checkpoint DCB is not given in the dump if the checkpoint/restart facility is not being utilized.)

DCB hhhhhh (LINE GROUP) is the starting address of this line group DCB.

DEVICE INTERFACE
This section is reserved.

D/S INTERFACE contains the number of buffers assigned initially for input operations, the number of buffers assigned initially for output operations, the address of the message handler for this line group, the polling delay interval, the program-controlled interruption options, the data set organization, the maximum number of buffers to be used at any given time for data transfer for each line in the line group, the base for addressing IOBs for the line group (initialized at open time), the relative priority of send and receive operations, the address of the translation table, the extended IOB index (size of an LCB), and the address of the exit list. The following table shows these fields, their relative offsets from the beginning of the DCB, their contents, and their size.

+14 Initial Receive Allocation ¦ Initial Send Allocation	+15 Address of the Message Handler	
+18 Polling Delay Interval	+19 PCI Options	+1A Data Set Organization
+1C Maximum Send or Receive Allocation	+1D Open-Base for Addressing IOBs	
+20 Priority of Send/ Receive Operations	+21 Address of the Translation Table	
+24 IOB Index	+25 Address of the Exit List	

For more detailed information on these fields, see *System Control Blocks,* GC28-6628.

FOUNDATION contains fields that are changed during open time. Before open, these fields contain the DDNAME character string, the open flags, the IOS error flags, and the macro instruction reference. After open, they contain the offset of the DD entry from the beginning of the TIOT, the macro instruction reference, the IOS error flags, the address of the DEB, and the open flags. The following two charts show this area and its contents before and after open.

Before Open:

+28 DDNAME character string		
+2C		
+30 Open flags	+31 IOS error flags	+32 Macro instruction reference

Note: *During open, the IOS error flags field and the macro instruction reference field are relocated and the last three bytes of the last word become part of the EXTENSION section.*

After Open:

+28 Offset of DD entry from beginning of the TIOT	+2A Macro instruction reference
+2C IOS error flags	+2D Address of the DEB
+30 Open flags	

For more detailed information on these fields, see *System Control Blocks.*

EXTENSION contains the address of the special characters table, the number of invitation lists, the number of units for each buffer, the size of all buffers used by this line group, and the number of reserve characters. The following example shows these fields, their relative offsets from the beginning of the DCB, their contents, and their size.

	+31 Address of the special characters table	
+34 Number of invitation lists	+35 Number of units per Buffer	+36 Buffer size
+38 Four one-byte reserve values		

For more detailed information on these fields, see *System Control Blocks.*

INVITATION LISTS gives the addresses of the different invitation lists for the different lines in the line group. Each list is pointed to by a one-word address. These addresses are given in order by relative line number.

Following each line group DCB is one or more LCBs (line control blocks), which are used by the internal TCAM logic to perform line management. The LCBs in the dump are given in order by relative line number.

LCB hhhhhh is the starting address of this LCB.

KEY/QCBA hhhhhhhh
The first byte is the key of this LCB. The next three bytes contain the address of its QCB.

PRI/LINK hhhhhhhh
The first byte is the priority of this LCB. The next three bytes contain the link address to the next element.

RSKEY/STCBA hhhhhh
The first byte is the receive scheduler key. The next three bytes contain the address of the first STCB when the LCB is a QCB.

RSPRI/RSLNK hhhhhhhh
The first byte is the receive scheduler priority. The next three bytes contain the address of the next item in the chain.

EOLTD/TSOB hhhhhhhh
The first two bytes contain the end-of-polling list, and the time-delay reference time. The third byte is the time-delay queue offset to the QCB address (always X'14' for an LCB). The fourth byte is a status byte used by TSO.

CHAIN/INSRC hhhhhhhh
The first byte is a status byte used by TCAM. The next three bytes contain the in-source chain.

SCBO/SCBDA hhhhhhhh
The first byte is the offset to the current SCB (station control block). The next three bytes contain the address of the SCB directory.

ISZE/FSBFR hhhhhhhh
The first byte is the count of reserved idles. The next three bytes contain the address of the first buffer assigned to this line.

FLAGS/SENSE hhhhhhhh is the start of the IOB contained in the LCB. The first and second bytes are IOS flags. The last two bytes are the sense bytes.

ECBCC/ECBPT hhhhhhhh
The first byte is the ECB completion code. The next three bytes contain the address of the ECB.

FLAG3/CSW
The first byte is an IOS flag byte. The next seven bytes are the last seven bytes of the CSW.

SIOCC/START hhhhhhhh
The first byte is the start I/O condition code. The last three bytes contain the address of the start of the channel program area.

DCBPT hhhhhhhh is the address of the DCB for this line.

RCQCB hhhhhhhh is the address of the QCB to tpost a recalled buffer to IOS.

INCAM/ERRCT hhhhhhhh are two halfword IOS error counters.

UCBX/RCBFR hhhhhhhh
The first byte is the UCB index. The last three bytes contain the address of a recalled buffer or the last buffer serviced by a PCI.

RECOF/STATE hhhhhhhh
The first two bytes contain the offset into the current block. The last two bytes are the LCB status bytes.

TSTSW/RECAD hhhhhhhh
The first byte is a test-and-set switch. The last three bytes contain the address of the current message block.

ERBKY/ERBQB hhhhhhhh
The first byte is the key of the ERB. The next three bytes contain the address of the QCB to which the ERB is tposted.

ERBPY/ERBLK hhhhhhhh
The first byte is the priority of this ERB. The next three bytes contain the address of the next item in the chain.

ERBST/ERBCH hhhhhhhh
The first byte is the ERB status byte. The next three bytes contain the address of a chain of assigned buffers.

ERBCT/TTCIN hhhhhhhh
The first two bytes contain the count of buffers requested by this ERB. The second two bytes contain the index into the termname table of the currently connected terminal.

MSGFM/SCBA hhhhhhhh
The first byte is used to control BSC lines. The next three bytes contain the address of the current SCB.

ERMSK/INVPT hhhhhhhh
The first byte is an error-recording mask. The next three bytes contain the address of the current entry in the invitation list.

TPCD is a three-word list of TP operation codes for the CCWs.

SNSV/CSWSV hhhhhhhh hhhhhhhh
The first byte is a save area for the sense byte. The last seven bytes comprise a save area for the CSW.

ERCCW is a three-doubleword area for ERP (error-recovery procedure) CCWs.

The following section gives the checkpoint DCB.

```
DCB   hhhhhh   (CHECKPOINT)
               DEVICE INTERFACE    hhhhhhhh      hhhhhhhh      hhhhhhhh      hhhhhhhh      hhhhhhhh
               D/S INTERFACE       hhhhhhhh      hhhhhhhh      hhhhhhhh      hhhhhhhh      hhhhhhhh
               FOUNDATION          hhhhhhhh      hhhhhhhh      hh
               EXTENSION                                         hhhhhh      hhhhhhhh      hhhhhhhh
```

DCB hhhhhh (CHECKPOINT) is the starting address of the checkpoint DCB.

DEVICE INTERFACE

This section is reserved.

D/S INTERFACE contains the data set organization, the address of the AVT, and the address of the exit list. The following table shows these fields, their relative offsets from the beginning of the DCB, their contents, and their size.

+14	Reserved		
+18	Reserved	+1A	Data set organization
+1C	Reserved	+1D	Address of the AVT
+20	Reserved		
+24	Reserved	+25	Address of the exit list

For more detailed information of these fields, see *System Control Blocks.*

FOUNDATION contains fields that are changed during open time. Before open, these fields contain the DDNAME character string, the open flags, the IOS error flags, and the macro instruction reference. After open, they contain the offset of the DD entry from the beginning of the TIOT, the macro instruction reference, the IOS error flags, the address of the DEB, and open flags. The following two tables show this area and its contents before and after open.

Before Open:

+28	DDNAME character string				
+2C					
+30	Open flags	+31	IOS error flags	+32	Macro instruction reference

Note: *During open, the IOS error flags field and the macro-instruction reference field are relocated and the last three bytes become part of the EXTENSION section.*

+28	Offset of DD entry from beginning of the TIOT	+2A	Macro instruction reference
+2C	IOS error flags	+2D	Address of the DEB
+30	Open flags		

For more detailed information on these fields, see *System Control Blocks.*

EXTENSION contains the OPTCD= value of the DCB. The remainder of this area is reserved. The following table shows these fields, their relative offsets from the beginning of the DCB, their contents, and their size.

	+31 Reserved
+34 OPTCD=value	+35 Reserved
+38 Reserved	

For more detailed information on these fields, see *System Control Blocks.*

The message queues DCB follows.

```
DCB  hhhhhh  (MESSAGE QUEUE)
             DEVICE INTERFACE  hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh
             D/S INTERFACE     hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh
             FOUNDATION        hhhhhhhh   hhhhhhhh   hh
             EXTENSION                                 hhhhhh   hhhhhhhh   hhhhhhhh
```

DCB hhhhhh (MESSAGE QUEUES) is the starting address of the message queues DCB.

DEVICE INTERFACE
This section is reserved.

D/S INTERFACE contains the data set organization, the address of the AVT, the threshold value of the percentage of the nonreusable disk message queue records to be used before a flush closedown of the system is initiated, and the address of the exit list. The following table shows these fields, their relative offsets from the beginning of the DCB, their contents, and their size.

+14 Reserved	
+18 Reserved	+1A Data set organization
+1C Reserved	+1D Address of the AVT
+20	+21 Reserved
+24 Reserved	+25 Address of the exit list

For more detailed information about these fields, see *System Control Blocks.*

FOUNDATION contains fields that are changed during open time. Before open, these fields contain the DDNAME character string, the open flags, the IOS error flags, and macro instruction reference. After open, they contain the offset of the DD entry from the beginning of the TIOT, the macro instruction reference, the IOS error flags, the address of the DEB, and open flags. The following two tables show this area and its contents before and after open.

Before Open:

+28 +2C	DDNAME character string	
+30 Open flags	+31 IOS error flags	+32 Macro instruction reference

Note: *During open, the IOS error-flags field and the macro instruction field are relocated and the last three bytes become part of the EXTENSION section.*

After Open:

+28 Offset of DD entry from beginning of the TIOT	+2A Macro instruction reference
+2C IOS error flags	+2D Address of the DEB
+30 Open flags	

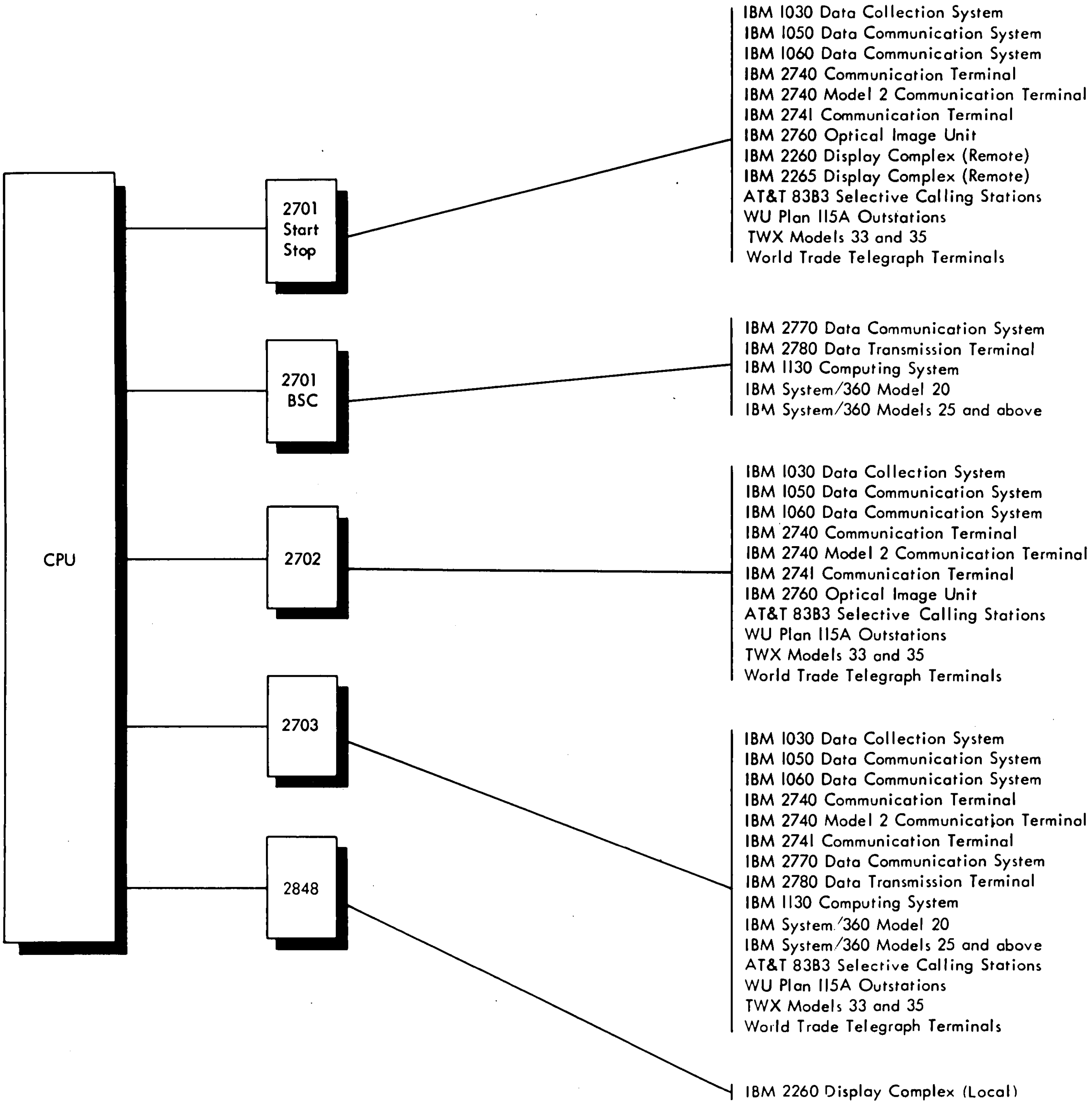
For more detailed information about these fields, see *System Control Blocks.*

EXTENSION contains the OPTCD= value of the DCB. The remainder of this area is reserved. The following table shows these fields, their relative offsets from the beginning of the DCB, their contents, and their size.

	+31 Reserved
+34 OPTCD=value	+35 Reserved
+38 Reserved	

For more detailed information on these fields, see *System Control Blocks.*

Appendix D. Device Configurations Supported by TCAM



Appendix D-Part 1

Station Type		Channel Type		TCU			Audio Response Unit	Line Type		Notes
		Multiplexer	Selector	IBM 2701 Data Adapter Unit	IBM 2702 Transmission Control	IBM 2703 Transmission Control	IBM 7770 Model 3	Switched	Nonswitched	
IBM 1030 Data Collection System	Auto Poll	X			X	X			X	The IBM Digital Time Out feature cannot be attached through an IBM 2701 TCU.
		X		X	X	X			X	
IBM 1050 Data Communication System	Auto Poll	X			X	X			X	
		X		X	X	X		X	X	
IBM 1060 Data Communication System	Auto Poll	X			X	X			X	
		X		X	X	X			X	
IBM 2260-2848 Display Complex (Remote)		X		X					X	
IBM 2260-2848 Display Complex (Local)		X	X							
IBM 2265-2845 Display Complex (Remote)		X		X					X	
IBM 2740 Model 1 Communication Terminal	Auto Poll	X			X	X			X	Two Types: 2740 with station control 2740 with station control and record checking
		X		X	X	X			X	Four Types: 2740 basic 2740 with station control 2740 with record checking 2740 with station control and record checking
		X		X	X	X		X		Four Types, all with dial: 2740 2740 with transmit control 2740 with record checking 2740 with transmit control and record checking
IBM 2740 Model 2 Communication Terminal	Auto Poll	X			X	X			X	Four Types: 2740 2740 with record checking 2740 with buffer receive 2740 without buffer receive (requires line slowdown feature)
		X		X	X	X			X	Four Types: 2740 2740 with record checking 2740 with buffer receive 2740 without buffer receive
IBM 2741 Communication Terminal		X		X	X	X		X	X	The attention feature is not supported, and the break feature is supported only if the CPU is sending and the terminal has not entered data when the break is issued.

Appendix D-Part 2

Station Type		Channel Type		TCU			Audio Response Unit	Line Type		Notes
		Multiplexer	Selector	IBM 2701 Data Adapter Unit	IBM 2702 Transmission Control	IBM 2703 Transmission Control	IBM 7770 Model 3	Switched	Nonswitched	
IBM 2760 Optical Image Unit								X	X	Attached to a 2740 Model 1 with record checking
IBM 2770 Data Communication System		X		X		X		X	X	BSC transmission using either ASCII or EBCDIC code
IBM 2780 Data Transmission Terminal		X		X		X		X	X	BSC transmission using ASCII, EBCDIC, or 6-bit code
IBM 1130 Computing System		X		X		X		X	X	BSC transmission
IBM System/360 Model 20		X		X		X		X	X	BSC transmission using either ASCII or EBCDIC code
IBM System/360 Models 25 and above		X		X		X		X	X	BSC transmission and point-to-point lines only
AT&T 83B3 Selective Calling Stations		X		X	X	X			X	
Western Union Plan 115A Outstations		X		X	X	X			X	
TWX Models 33 and 35		X		X	X	X		X		Teletype terminals, dial service (8-level code)
World Trade Telegraph Terminals		X		X	X	X			X	Control unit must incorporate a WTTA
Audio terminals		X					X	X		Example: IBM 2721 Portable Audio Terminal

Appendix D-Part 3

Glossary

accepting: the process in which a destination station acquires a message transmitted to it from the central computer. Entering and accepting are functions of a station.

access method: a combination of an access technique (either queued or basic) and a given data set organization (for instance, sequential, partitioned, indexed sequential, or direct) that allows the programmer to transfer data between main storage and I/O devices.

addressing characters: identifying characters, sent by the computer, that cause a particular station (or component) to be selected to accept a message sent by the computer.

application program: a user-provided program that processes the text portions of messages. Application programs run asynchronously with the message control program, and are usually located in another partition or region of main storage. TCAM application programs are optional; there may be many or none, depending on the needs of the user.

available-unit queue: a queue in main storage to which all buffer units are assigned initially (that is, before allocation to TCAM lines and application programs requiring buffers). *Empty* buffer units (that is, buffer units whose contents have been processed by the incoming or outgoing group of an MH, and that are not assigned to the main-storage message queues data set) are returned to the available-unit queue, from which they are reallocated.

binary synchronous communications (BSC): data transmission in which character synchronization is controlled by timing signals generated by the device that originates a message (and the device that obtains the message recognizes the *sync pattern* at the beginning of the transmission—the devices are locked in step with one another); contrast with *start-stop transmission.*

block: that portion of a message terminated by an EOB or ETB line-control character or, if this is the last block in the message, by an ETX or EOT line-control character. When end-of-block checking is specified in the STARTMH macro, messages are checked for certain types of transmission and user-specified logical errors on a block-by-block basis.

buffer: an area in main storage into which a message segment is read, or from which a message segment is written. Buffers are temporary data-holding areas that are used to compensate for the difference between the rate at which data can be entered from or accepted by a station and the rate at which it can be processed by the central processing unit; buffers also may be used as work areas in TCAM. The size of TCAM buffers is designated by the user. (See also *hardware buffer.)*

buffer allocation: the assignment of buffers by TCAM to lines or application programs in preparation for reception of message segments from stations on the lines or from application programs. (See also *dynamic buffer allocation* and *static buffer allocation.)*

buffer deallocation: for a sending operation, deallocation consists of returning the units that compose the buffer to the available-unit queue after the data in these units has been sent to its destination station or application program; for a receiving operation, deallocation consists of transferring full buffers from the line or application program to which they were assigned to the incoming group of the MH that is to process the message segments they contain.

buffer prefix: a control area contained within each TCAM buffer. The prefix for the buffer containing the first segment of a message is 30 bytes, while the prefix for each buffer containing a subsequent segment of the message is 23 bytes. The user must allow room for the buffer prefix when he specifies his buffer size. TCAM fills the prefix area with buffer control information.

buffer unit: the basic building block from which TCAM buffers are constructed. All units in a particular TCAM system are the same size; this size is specified by the KEYLEN= operand of the INTRO macro.

buffer-unit pool: all the buffer units in a particular TCAM system together constitute the buffer-unit pool for that system. The number of units in the pool is equal to the sum of the integers specified by the LNUNITS= and MSUNITS= operands of the INTRO macro.

buffered terminal: a terminal having a hardware buffer. As used in this book, a buffered terminal is an IBM 2740 Model 2 station or IBM 2770 station whose TERMINAL macro specifies BFDELAY=integer. When the BFDELAY= operand of TERMINAL is coded, messages are sent to the station segment-by-segment; after a segment is sent, the message control program pauses before sending the next segment to allow the station's buffer to empty. During this pause, the MCP may send segments to other stations on the line.

central processing unit (CPU): a unit of a computer that controls interpretation and execution of instructions.

channel program block (CPB): a TCAM control block used in the transfer of the data between buffer units and message queues maintained on disk. The CPB= operand of the INTRO macro specifies the number of CPBs to be provided in a TCAM system.

checkpoint data set: an optional TCAM data set that contains the checkpoint records used to reconstruct the MCP environment after closedown or system failure, when the TCAM checkpoint/restart facility is utilized.

checkpoint records: records, located in the checkpoint data set, that are used to reconstruct the MCP environment upon restart following closedown or system failure. The four types of checkpoint records are: environment records, incident records, checkpoint request records, and a control record.

checkpoint request record: a checkpoint record taken as a result of execution of a CKREQ macro issued in an application program; the record contains the status of a single destination queue for the application program. The latest checkpoint request record for a message queue is used during restart to cause sending from that queue to the application program to begin with the message that follows the last message sent to the program from that queue at the time the checkpoint request record was taken, rather than with the message following the last message marked serviced.

checkpoint/restart: a TCAM facility that records the status of the teleprocessing network at designated intervals or following certain events. Following system failure or closedown, the checkpoint/restart facility uses the records it has taken to restore the message control program environment as nearly as possible to its status before the failure or closedown.

closedown: an orderly deactivation of the MCP by either an MCPCLOSE macro instruction issued in an application program or an operator command. See *quick closedown* and *flush closedown.*

cold restart: start-up of a TCAM message control program 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 start-up), and is the only type of restart possible when no checkpoint/restart facility is used.

component: an I/O device associated with a station.

computer: in this publication, the central processing unit in which the TCAM message control program is located.

continuation restart: a restart of the TCAM message control program following termination of the message control program because of system failure; the TCAM checkpoint/restart facility restores the MCP environment as nearly as possible to its condition before failure.

control characters: characters transmitted over a line that are not message data, but which cause certain control operations to be performed when encountered by the computer, transmission control unit, or station; among such operations are polling and addressing, message delimiting and blocking, transmission-error checking, and carriage return.

control record: a record, included in a checkpoint data set, that keeps track of the correct environment, incident, and checkpoint request records to use for reconstructing the message control program environment during restart.

data control block (DCB): an area of main storage that serves as a logical connector between the problem program and a data set. The data control block also can provide control information for any transfer of data. A data control block must be created for each TCAM data set except a message queues data set residing in main storage; a DCB macro instruction creates a data control block.

data set:

1. a named, organized collection of logically related records (program data set). The information is not restricted to a specific type, purpose, or storage medium. Among the data sets specifically related to TCAM are the line group data sets, the message queues data sets, the checkpoint data set, the message log data set, and the input and output data sets for a TCAM-compatible application program.
2. a device containing the electrical circuitry necessary to connect data processing equipment to a communication channel; also called a subset, Data-Phone*, modulator/demodulator, or modem.

dead-letter queue: the destination queue for the station or application program named by the DLQ= operand of the INTRO macro instruction. If an invalid destination is detected in a message header by a FORWARD macro instruction, and if no user exit is specified in the FORWARD macro, that message is sent to the dead-letter queue.

delimiter macro instruction: a TCAM macro instruction that classifies and identifies sequences of functional macro instructions and directs control to the appropriate sequence of functional macro instructions.

descriptor code: under Multiple Console Support, indicates the means of message presentation and message deletion on display devices.

destination: the place to which a message being handled by a TCAM message handler is to be sent. A destination may be either a station defined by a TERMINAL macro, a group of stations defined by a TLIST macro, or an application program

---

*Trademark of the American Telephone & Telegraph Co.

defined by a TPROCESS macro. One or more destinations may be specified in fields of the message header that are checked by a FORWARD macro, or a single destination may be specified for all messages handled by a particular inheader subgroup by means of the DEST= operand of a FORWARD macro issued in that subgroup.

destination field: a field in a message header containing the name of a station or application program to which a message is directed.

destination queue: a queue on which messages bound for a particular destination are placed after being processed by the incoming group of a message handler. A separate destination queue is created for each station defined by a TERMINAL macro specifying queuing by terminal; one for each line whose stations are defined by TERMINAL macros specifying queuing by line; and one for each application-program process entry (defined by a TPROCESS macro) to which the application program may direct GET or READ macros. Destination queues are maintained in message queues data sets, which may be located on disk or in main storage. Queuing messages by destination permits overlap of line usage in I/O operations. See also *process queue.*

destination station: a station that accepts a message sent to it by the outgoing group of the message handler that is specified for the line to which the accepting station is assigned.

distribution entry: an entry in the terminal table associated with a distribution list. A distribution entry is created by a TLIST macro.

distribution list: a list of single, group, cascade, or process entries; when a message is directed to the distribution entry associated with this list, TCAM sends the message to each destination named in the list.

dynamic buffer allocation: the assignment of buffers to a line on an as-needed basis, after a message has started coming in over the line. Dynamic allocation occurs following program-controlled interruptions, and is specified by the PCI= operand of the line group DCB macro. See also *static buffer allocation.*

end-of-address (EOA) character:

1. a hardware generated line-control character or characters transmitted on a line to indicate the end of nontext characters (for example, addressing characters).
2. a TCAM character that must be placed in a message if the system is to accommodate routing of that message to several destinations; the character must immediately follow the last destination code in the message header; and must also be specified by the EOA= operand of the FORWARD macro for the message.

entering: the process in which a station places on the line a message to be transmitted to the central computer (a station *enters* and *accepts* messages, while a computer *sends* and *receives* messages).

environment record: a record of the total teleprocessing environment at a single point in time. The environment record resides in the checkpoint data set; at restart time, an environment record is updated by the contents of incident records that were taken after the environment record was taken, and the updated environment record is then used to reconstruct the message control program environment as it existed before MCP closedown or system failure.

error-recovery procedures (ERP): a set of internal TCAM routines that attempt to recover from transmission errors.

FEFO (first-ended first-out): a queuing scheme whereby messages on a destination queue are sent to the destination on a *first-ended first-out* basis within priority groups. That is, higher-priority messages are sent before lower-priority messages; when two messages on a queue have equal priority, the one whose final segment arrived at the queue earliest is sent first.

FIFO (first-in first-out): a queuing scheme whereby equal-priority messages on the same destination queue are sent in the order that their first segments arrived at the queue.

flush closedown: a closedown of the TCAM message control program during which incoming message traffic is suspended and queued outgoing messages are sent to their destinations before closedown is completed; this form of termination is known as a *flush* closedown because unsent messages are flushed from the message queues. See also *quick closedown.*

folded table: a table that recognizes as valid either uppercase or lowercase characters.

functional macro instructions: TCAM macros that perform the specific operations required for messages directed to the message handler. See also *delimiter macro instructions.*

group entry: an entry in the terminal table associated with a group of terminals having the group-addressing hardware feature.

hardware buffer: a buffer that is located in a station, as opposed to the buffers for the TCAM MCP, which are located in the computer. The IBM 2740 Communication Terminal Model 2, for example, contains a hardware buffer that accommodates up to 120 characters. See also *buffered terminal.*

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, a priority level for the message, etc. The message header is operated on by macros in the inheader and outheader subgroups of the message handler.

header buffer: a buffer containing a header segment.

header segment: a message segment containing all or part of the message header.

identification characters (ID characters): characters sent by a BSC station on a switched line to identify the station. ID characters can also be assigned to the computer (by the CPUID= operand of the INVLIST macro); in this case, the computer and the station can exchange ID sequences. TWX stations also use ID characters.

idle: describes a line that is not currently available for transmission of data because IDLE was coded in the OPEN macro for the line group data set containing the line. Such a line may be activated by a STARTLINE operator command.

inactive station: a station that is currently ineligible for entering and/or accepting messages. A station may be inactive for entering or inactive for accepting, or both; the status of a station is determined by the status of the line it is on, by a special character (+ or −) coded in the invitation-list entry for the station, by the presence or absence of a HOLD macro in the outgoing group of the message handler handling outgoing messages for this station, and by the five operator commands (ACTVBOTH, ENTERING, NOENTRNG, NOTRAFIC, SUSPXMIT) that directly affect the station's status.

incident record: a checkpoint record residing in the checkpoint data set on a DASD; an incident record logs a change in station status or in the contents of an option field that occurred since the last environment record was taken. Incident records update the information contained in environment records at restart time after a closedown or system failure.

incoming group: that portion of a message handler designed to handle messages arriving for handling by the message control program. See also *outgoing group.*

incoming message: a message being transmitted from a station to the computer.

input: of or related to a message transmission that involves entering data at a station or receiving data at the computer.

input data set: a logical data set for a TCAM-compatible application program. The input data set contains all messages or records being sent to the application program from a single process queue. Though it is not located in a physical medium, the input data set requires a DD statement and a DCB macro for its definition and must be activated and deactivated by OPEN and CLOSE macros. See also *output data set.*

input sequence number: a means of ensuring that messages are received from a source in the correct order; the user may place a sequence number in the header of each message entered by a station or application program, and code a SEQUENCE macro in the incoming group of his message handler. The SEQUENCE macro checks the sequence number for each message; if the number is not one more than that assigned to the previous message received from that origin, a bit is turned on in the message error record.

inquiry processing: a TCAM application in which the message control program receives a message from a station, then routes it to an application program that processes the data in the message and generates a reply; the reply is routed by the message control program to the inquiring station. Response time often may be shortened by specifying lock mode (by a LOCK macro in the message handler) and by locating the message queues data set containing the queues for the application program in main storage.

intercepted station: a station to which no messages may be sent. A station is intercepted by issuing a HOLD macro instruction in the outmessage subgroup of a message handler; the suspension is either for a specified time interval or until either an operator command or an application-program macro instruction-is issued to release messages held for the intercepted station.

invalid destination: a destination specified for a message that does not correspond to a valid terminal-table entry.

invitation: the process in which the computer contacts a station in order to allow the station to transmit a message if it has one ready.

invitation delay: a period of time during which invitation is suspended to allow transmission of outgoing messages for lines whose line group DCB has CPRI=R specified. This delay is observed for all such stations on a line when the end of the invitation list for that line is reached. The delay in polling is observed for such stations whether or not the computer has any messages to send them. If no invitation delay is specified for such stations, no messages can be sent to them.

invitation list: a series of sets of polling characters or identification sequences associated with the stations on a line; the order in which sets of polling characters are specified (in the INVLIST macro for the line) determines the order in which polled stations are invited to enter messages on the line.

line: the communications medium linking the computer to one or more remote stations; message transmission occurs over this medium.

line control: the scheme of operating procedures and control signals by which a telecommunications system is controlled.

line control block (LCB): an area of main storage containing control information for operations on a line; one LCB is maintained by TCAM for each line in the system.

line-control characters: characters that control transmission of data over a line or control the state of the devices on the line; for example, line-control characters delimit messages, cause transmission-error checking to be performed, and indicate whether a station has data to send or is ready to receive data.

line group: a set of one or more communications lines of the same type, over which stations with similar characteristics can communicate with the computer.

line group data set: a message control program data set consisting of all the lines in a line group; the messages that are transmitted on these lines constitute the data in this data set. A line group data set is defined by a line group DCB macro instruction, and by a DD statement for each line in the line group.

line group DCB: a data control block created by a line group DCB macro instruction; information in the data control block defines the line group to TCAM.

local station: a station whose control unit is connected directly to a computer data channel by a local cable. See *remote station.*

lock mode: a TCAM facility, invoked in a message handler by the LOCK macro, whereby a station entering an inquiry message for an application program is held on the line by the message control program until a response has been returned to it by the application program. Using lock mode decreases response time because there are no interruptions on the line before a response is returned. If LOCK is executed and CONV=YES is coded in the STARTMH macro, lock mode is in effect for the station. A station may be placed in lock mode either for the duration of a single inquiry and response (*message* lock mode) or for the duration of several inquiry-response cycles (*extended* lock mode). The type of lock mode is specified in the LOCK macro.

log: a collection of messages or message segments placed on a secondary-storage device for accounting or data collection purposes. The TCAM logging facility is invoked by a functional macro instruction issued in a message handler.

log data set: a data set consisting of the messages or message segments recorded on a secondary-storage medium by the TCAM logging facility. A log data set is defined by means of a BSAM DCB macro instruction that is issued with the DCB macro instructions defining the line group data sets, the message queues data sets, and the checkpoint data set.

logtype entry: an entry in the terminal table associated with a queue on which complete messages reside while awaiting transfer to the logging medium (a logtype entry is not needed if message segments only are to be logged). A logtype entry is created by a LOGTYPE macro.

message: a unit of data received from or sent to a station that is terminated by an EOT or ETX control character or, if the CONV= operand of the STARTMH macro is coded CONV=YES, by an EOB or ETX control character. A TCAM message is often divided into a header portion, which contains control information, and a text portion, which contains the part of the message of concern to the party ultimately receiving it.

message control program (MCP): a set of user-defined TCAM routines that identifies the teleprocessing network to the System/360 Operating System, establishes the line control required for the various kinds of stations and modes of connection, and controls the handling and routing of messages to fit the user's requirements.

message error record: five bytes assigned to each message being processed by a message handler; these bytes indicate physical or logical errors that have occurred during transmission on the line or during subsequent processing or queuing of the message, and are checked by error-handling macros in the inmessage and outmessage subgroups of a message handler.

message handler (MH): a sequence of user-specified TCAM macro instructions in the message control program that examine and process control information in message headers, and perform functions necessary to prepare message segments for forwarding to their destinations. One message handler must be assigned to each line group by the MH= operand of the line group DCB macro, and one must be assigned to each TCAM-compatible application program by the MH= operand of the PCB macro. The incoming group of an MH handles messages received from either an originating station or an application program; the outgoing group of an MH handles messages before their being sent to a destination station or application program.

message priority: refers to the order in which messages in a destination queue are transmitted to the destination, relative to each other. Higher-priority messages are forwarded before lower-priority messages. Up to 255 different priority levels may be assigned to a single destination (by the LEVEL= operand of the TERMINAL or TPROCESS macro). The priority for each message sent to the destination may be specified in the message header or assigned by a PRIORITY macro; in either case, a PRIORITY macro should be coded in the inheader subgroup handling the message.

message queue: see *destination queue.*

message queues data set: a TCAM data set that contains one or more destination queues. A message queues 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 queues data sets (one in main storage, one on reusable disk, one on nonreusable disk) may be specified for a TCAM message control program.

message segment: the portion of a message contained in a single buffer.

message switching: a telecommunications application in which a message is received from a remote station, stored until a suitable outgoing line is available, and then transmittted to its destination station. TCAM message switching can be handled entirely by the message control program.

nontransparent mode: a mode of binary synchronous transmission in which all control characters are treated as control characters (that is, not treated as text). See *transparent mode.*

on-line test (OLT): an optional TCAM facility that permits either a system console operator or a remote-station operator to test transmission control units and remote stations to find out if they work properly.

operator command: a command entered either at an operator control station or at the system console to examine or alter the status of the telecommunications network during execution.

operator control station: a station eligible to enter operator commands. An application program and the system console may also serve as operator control stations. Operator control stations are designated as such by the PRIMARY= operand of the INTRO macro and by the SECTERM= operand of the TERMINAL and TPROCESS macros.

option field: a storage area containing data relating to a particular station, component, line, or application program; certain message handler routines that need source- or destination-related data to perform their functions have access to data in an option field. User-written routines also have access to data in an option field. Option fields are defined by OPTION macros and initialized for each station, line, component, or application program by the OPDATA= operand of the TERMINAL or TPROCESS macro.

origin: a station or application program from which a message, or other data originates. See also *destination.*

outgoing group: that section of a message handler that manipulates outgoing messages after they have been removed from their destination queues. The outgoing group has three types of subgroup—the outheader subgroup, which executes on outgoing header segments; the outbuffer subgroup, which executes on each outgoing segment; and the outmessage subgroup, which executes on the entire message. See also *incoming group.*

output data set: a logical data set for a TCAM-compatible application program. The output data set contains the messages or records returned from the application program to the message control program by a process entry in the terminal table. An output data set is defined by a DD statement and a DCB macro, and must be activated and deactivated by OPEN and CLOSE macros. See also *input data set.*

output DCB: a data control block created by an output DCB macro. One output DCB is required for each output data set.

output sequence number: a number placed in the header of a message by TCAM that determines the order in which messages were sent to a destination by the computer. When specified in an outheader subgroup, the SEQUENCE macro causes an output sequence number to be placed in the header of each outgoing message; this sequence number is one greater than the sequence number for the last message sent to this destination. See also *input sequence number.*

polling: a non-contention line management method whereby the computer invites stations to enter messages. The computer contacts stations in the order specified by the invitation list; each station contacted is invited to enter messages.

polling characters: a set of identifying characters peculiar to either a station or a component of that station; a response to these characters indicates to the computer whether the station has a message to enter.

priority: see *message priority* and *transmission priority.*

problem determination: The act of pointing to the malfunctioning hardware unit or program and ultimately determining who has the responsibility for fixing the trouble.

process queue: a destination queue for an application program (see *destination queue*). A process queue is defined by a TPROCESS macro.

queue: a set of items consisting of:

1. a queue control block (an area in main storage containing control information for the queue), and
2. one or more ordered arrangements of items (the items may be messages, main-storage addresses, etc.).

quick closedown: a closedown of the TCAM message control program that entails stopping message traffic on each line as soon as transmission is complete for any messages being sent or received at the time of the request for closedown.

read-ahead queue: an area of main storage in which the message control program plans work units in advance of their being requested by the application programs.

receiving: the process in which the central computer obtains a message from a remote station (the message is *entered* by the station). Receiving and sending are functions of the central computer.

record: a logical unit of data, the length of which is defined by the user through the use of operands on the input or output DCB macro and delimiting characters in the message.

relative line number: a number assigned by the user to a communications line of a line group at system generation time or MCP execution time. If a line group is defined at system generation time by a UNITNAME macro, the lines in the group are assigned relative line numbers according to the order in which their hardware addresses are specified in the UNIT= operand of UNITNAME; the line whose address is specified first is relative line number one, that address specified second is relative line number two, etc. If a line group is defined at MCP execution time by concatenated DD statements, the order in which the DD statements for the lines in the line group are arranged determines the relative line numbers for the lines. The line whose DD statement appears first is relative line number one, the statement that appears second is relative line number two, etc.

remote station: a station that is connected to a computer data channel through either a transmission control unit, an audio response unit or common carrier facilities. See also *local station.*

retry: an error-recovery procedure in which the current block of data (from the last EOB or ETB) is re-sent a prescribed number of times, or until accepted or entered correctly.

routing code: under Multiple Console Support, indicates the consoles to which the messages should be sent.

segment: the portion of a TCAM message contained in a single buffer.

selection: the process whereby the computer contacts a remote station to send it a message.

sending: the process in which the central computer places a message on a line for transmission to a station (the station *accepts* the message). Sending and receiving are functions of the central computer.

sequence number: see *input sequence number* and *output sequence number.*

single entry: an entry in the terminal table associated with a single station or station component; one such entry must be created (by a TERMINAL macro) for each station in the TCAM system not defined by a group entry.

start-stop transmission: data transmission in which each character being transmitted is preceded by a special control signal indicating the beginning of the sequence of data bits representing the character, and is followed by another control signal indicating the end of the data-bit sequence (character recognition by the device that obtains the data depends on the presence of these control signals for each character); contrast with *binary synchronous communications.*

static buffer allocation: the assignment to a line, before transmission over that line, of all buffers to contain the transmitted data. When PCI=N or PCI=R is coded in the line group DCB macro, the number of buffers specified by the BUFIN= or BUFOUT= operand of the line group DCB macro instruction is assigned to a line before incoming or outgoing transmission begins on that line; once transmission has started, no more buffers are available to handle the data involved in the transmission.

station: either a remote terminal, or a remote computer used as a terminal.

subblock: that portion of a BSC message terminated by an ITB line-control character.

switched line: a communications line on which the connection between the computer and a remote station is established by dialing. Also known as a dial line.

symbol: in assembler language, a character or character string used to represent addresses or arbitrary values. A symbol must meet the following requirements:

1. 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.
2. No blanks or special characters are allowed in a symbol.

system interval: a user-specified time interval during which polling and addressing are suspended on multipoint lines to polled stations. The system interval is specified by the INTVAL= operand of the INTRO macro, and may be changed during TCAM initialization, by a SYSINTVL operator command. The INTERVAL operator command tells TCAM to begin the system interval. The system interval minimizes unproductive polling, minimizes CPU meter time, and synchronizes polling on the polled lines in the system. See also *invitation delay.*

terminal table: an ordered collection of information consisting of a control field for the table and blocks of information on each line, station, component, or application program from which a message can originate or to which a message can be sent.

text: that part of the message of concern to the party ultimately receiving the message (that is, the message exclusive of the header, or control, information).

text segment: a portion of a message that contains no part of the message header.

transmission: the transfer of coded data by an electromagnetic medium between two points in a telecommunications network.

transmission control unit (TCU): a control unit that serves as an interface between communications lines and a computer for logical operations. The transmission control units supported by TCAM are the 2701 Data Adapter Unit Model 1, the 2702 Transmission Control Model 1, and the 2703 Transmission Control Model 1.

transmission priority: refers to the order in which sending and receiving occur, relative to each other, for a particular station. Transmission priority is specified on a line-group basis by the CPRI= operand of the line group DCB macro. The three transmission priorities possible in TCAM are send priority, equal priority, and receive priority. The exact meaning of each priority depends upon the line configuration and type of station. See also *message priority.*

transparent mode: a mode of binary synchronous transmission in which all data, including normally restricted data-link control characters, is transmitted only as specific bit patterns. Control characters that are intended to be effective are preceded by a DLE character.

unit: see *buffer unit.*

warm restart: a restart of the TCAM message control program following either a quick or a flush closedown; the TCAM checkpoint/restart facility restores the MCP environment as nearly as possible to its condition before failure. See checkpoint/restart.

work area: an area of storage related to an application program that receives messages or records transferred to the application program from the message control program by GET or READ macros, and from which messages or records are transferred to the MCP by PUT or WRITE macros. The size of the work area must be specified in the BLKSIZE= operand of the input or output DCB macro associated with the data set whose contents are being transferred to or from the work area. A work area may be defined either statically (by a DC or DS assembler instruction) or dynamically (by specifying locate mode in the MACRF= operand of the input DCB macro).

work unit: the amount of data transferred from the message control program to an application program by a single GET or READ macro, or transferred from an application program to the MCP by a single PUT or WRITE macro. The work unit may be a message or a record (or, for QTAM-compatible application programs, a segment).

Index

READER'S COMMENT FORM

OS TCAM User's Guide Order No. GC30-2025-0

- How did you use this publication?

  As a reference source ☐
  As a classroom text ☐
  As . . . . ☐

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

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

- What is your occupation? . . . .

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

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

YOUR COMMENTS, PLEASE . . .

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

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

Cut Along Line

Fold Fold

FIRST CLASS
PERMIT NO. 569
RESEARCH TRIANGLE PARK
NORTH CAROLINA

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

POSTAGE WILL BE PAID BY . . .

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

Attention: Publications Center, Dept. E01

Fold Fold

IBM®

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

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

GC30-2025-0

OS TCAM User's Guide Printed in U.S.A. GC30-2025-0

IBM®

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

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