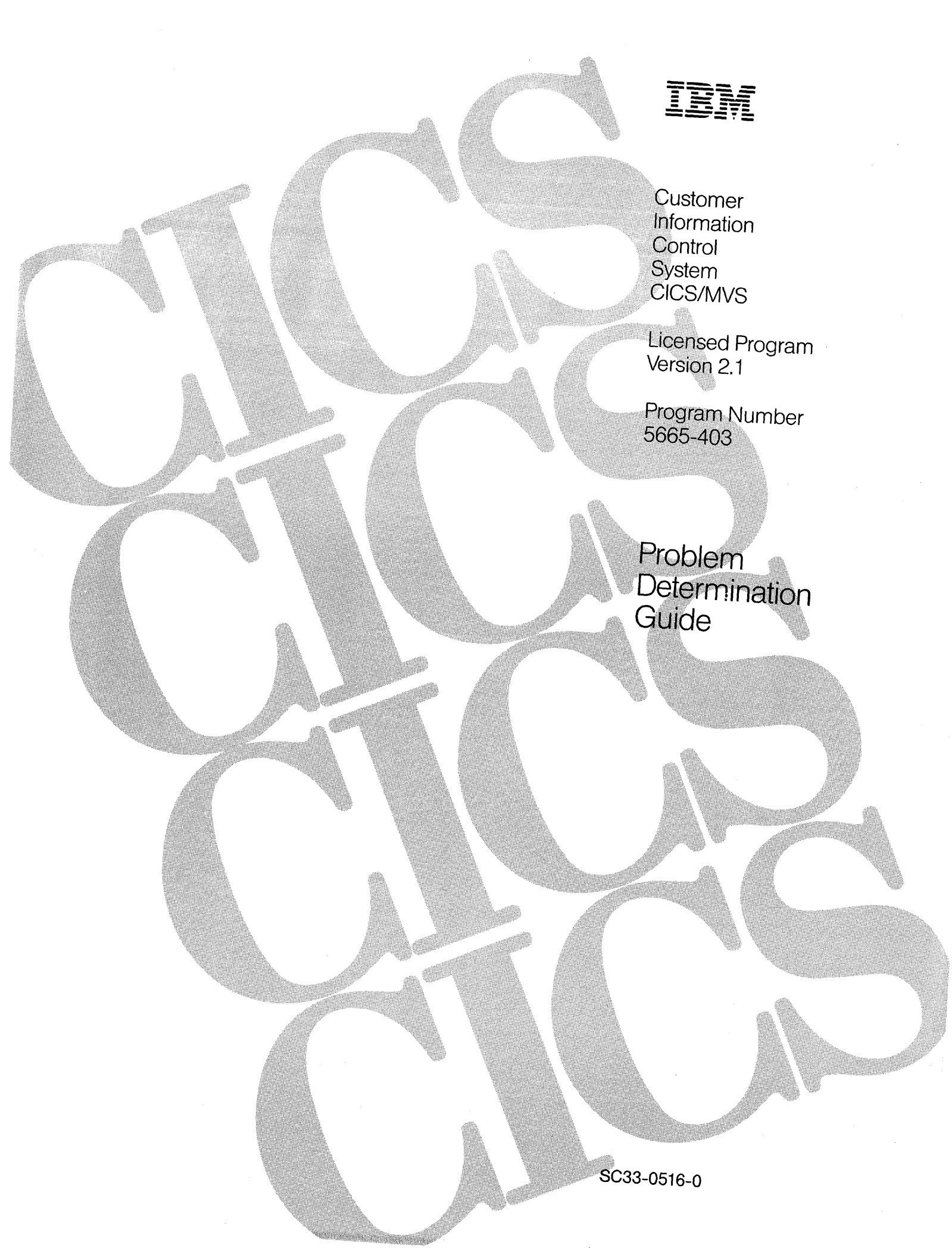


IBM

Customer
Information
Control
System
CICS/MVS

Licensed Program
Version 2.1

Program Number
5665-403

Problem Determination Guide

SC33-0516-0

First Edition (December 1987)

This edition applies to Version 2 Release 1 (Version 2.1) of the licensed program Customer Information Control System/MVS (CICS/MVS), program number 5665-403.

This edition is based on the earlier book for CICS/VS Version 1.7, SC33-0242-0 (which remains applicable and current for users of Version 1.7). Changes in OS information are indicated by vertical lines to the left of the changes.

Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead.

Publications are not stocked at the addresses given below. Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed either to:

International Business Machines Corporation, Department 6R1H,
180 Kost Road, Mechanicsburg, PA 17055, U.S.A.

or to:

IBM United Kingdom Laboratories Limited, Information Development,
Mail Point 095, Hursley Park, Winchester, Hampshire, England, SO21 2JN

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

What this book is about

This book helps you to debug problems in CICS application programs or in the CICS system.

Who this book is for

This book is for:

IBM Service personnel
CICS application programmers
CICS system programmers.

What you need to know to understand this book

This book assumes a good knowledge of CICS. You should be familiar with the books that tell you how to install and use a CICS system. For particular problems, you may need information from other books. The names of these books are quoted when required.

How to use this book

This book is designed as a reference book, so you should look at the appropriate chapter for your problem. "Chapter 1.1. Techniques of problem determination" on page 3, "Chapter 1.2. Problem determination flowcharts" on page 11, "Chapter 2.1. Trace" on page 45, "Chapter 2.3. Formatted dump" on page 131, and "Chapter 2.4. Control block linkages" on page 141 are useful for many problems.

Notes on terminology

CICS is used throughout this book to mean CICS/MVS. The term **VTAM** refers to ACF/VTAM and to the Record interface of ACF/TCAM. The term **TCAM** refers to both TCAM and the DCB interface of ACF/TCAM.

Book structure

Part 1. Approach to Problem Determination on page 1

Describes the first steps in problem determination.

Part 2. Aids to Problem Determination on page 43

Describes the most useful tools and methods for problem determination.

Part 3. Review of CICS Operation on page 233

Gives a high-level explanation of how CICS works if you need to go deeper into a problem.

Part 4. Interfaces on page 327

Describes the interfaces between CICS and associated licensed programs.

Appendixes on page 363

The appendixes contain:

Information you need to send to IBM if you wish to submit an APAR
Tips on avoiding errors
A list of abbreviations used in this book.

Index on page 395

Bibliography

CICS/MVS Version 2 Release 1 library

Evaluation and Planning

- Brochure GC33-0503
- General Information GC33-0155
- Facilities and Planning Guide SC33-0504
- Release Guide GC33-0505

General

- Library Guide GC33-0356
- Master Index SC33-0513
- User's Handbook SX33-6061
- Messages and Codes SC33-0514

Administration

- Installation Guide SC33-0506
- Customization Guide SC33-0507
- Resource Definition (Online) SC33-0508
- Resource Definition (Macro) SC33-0509
- Operations Guide SC33-0510
- CICS-Supplied Transactions SC33-0511

Programming

- Application Programming Primer SC33-0139
- Application Programmer's Reference SC33-0512

Service

- Problem Determination Guide SC33-0516
- Diagnosis Reference LC33-0517
- Diagnosis Handbook LX33-6062
- Data Areas LC33-0518

Special Topics

- Intercommunication Guide SC33-0519
- Recovery and Restart Guide SC33-0520
- Performance Guide SC33-0521
- XRF Guide SC33-0522

Version 1 books

Application Programmer's Reference Manual (Macro Level) *SC33-0079*

IBM 3270 Data Stream Device Guide *SC33-0232*

IBM 4700/3600/3630 Guide *SC33-0233*

IBM 3767/3770/6670 Guide *SC33-0235*

IBM 3650/3680 Guide *SC33-0234*

IBM 3790/3730/8100 Guide *SC33-0236*

Books from related libraries

MVS/XA

Installation: System Generation Reference, GC26-4009
System Programming Library: System Macros and Facilities Volume 1, GC28-1150
System Programming Library: System Macros and Facilities Volume 2, GC28-1151 (or GB0F-1014 to order both volumes)
MVS Supervisor Services and Macro Instructions, GC28-0683
Data Administration Macro Instructions Reference, GC26-4014
Debugging Handbook Volumes 1 to 5, LC28-1164 to LC28-1168 (or LBOF-1015, to order all five volumes)
Diagnostic Techniques, LY28-1199
System Programming Library: Service Aids, GC28-1159
Message Library: System Messages, GC28-1156
Interactive Problem Control System: User Guide and Reference, GC28-1297.

IMS/VS Version 1

IMS/VS Release Guide, SH20-9207
IMS/VS General Information Manual, GH20-1260
IMS/VS Installation Guide, SH20-9081
IMS/VS Primer Function Installation Guide, SH20-9208
IMS/VS Application Programming, SH20-9026
IMS/VS Application Programming for CICS/VS Users, SH20-9210
IMS/VS Application Programming Reference Summary, SX26-3727
IMS/VS Data Base Administration Guide, SH20-9025
IMS/VS System Administration Guide, SH20-9178
IMS/VS Message Format Service User's Guide, SH20-9053
IMS/VS Utilities Reference Manual, SH20-9029
IMS/VS System Programming Reference, SH20-9027
IMS/VS Programming Guide for Remote SNA Systems, SH20-9054
IMS/VS Operations and Recovery Guide, SH20-9209
IMS/VS Operator's Reference Manual, SH20-9028
IMS/VS Data Base Recovery Control: Guide and Reference, SH35-0027
IMS/VS Messages and Codes Reference Manual, SH20-9030
IMS/VS Failure Analysis Structure Tables (FAST) for Dump Analysis, LY20-8050
IMS/VS Diagnosis Guide, LY20-8063
IMS/VS Diagnosis Reference, LY20-8069
IMS/VS Data Base Recovery Control: Logic, LY35-0028
IMS/VS Master Index and Glossary, SH20-9085.

IMS/VS Version 2 Releases 1 and 2

IMS/VS General Information Manual, GC26-4180
Introducing the IMS/VS Library, GC26-4294
IMS/VS Installation Guide, SC26-4172
IMS/VS Application Programming, SC26-4178
IMS/VS Application Programming for CICS/VS Users, SC26-4177
IMS/VS Application Programming Reference Summary, SX26-3746
IMS/VS Data Base Administration Guide, SC26-4179
IMS/VS System Administration Guide, SC26-4176
IMS/VS Message Format Service User's Guide, SC26-4181

IMS/VS Utilities Reference Manual, SC26-4173
IMS/VS Customization Guide, SC26-4187
IMS/VS Programming Guide for Remote SNA Systems, SC26-4186
IMS/VS Operator's Reference Manual, SC26-4175
IMS/VS Data Base Recovery Control: Guide and Reference, SC26-4209
IMS/VS Messages and Codes Reference Manual, SC26-4174
IMS/VS Failure Analysis Structure Tables FAST for Dump Analysis, LY26-3992
IMS/VS Master Index and Glossary, SC26-4182
IMS/VS Installation Listings, SC26-4215
IMS/VS System Definition Reference, SC26-4216
IMS/VS Licensed Program Specifications, GC26-4171
IMS/VS Summary of Operator Commands, SX26-3754.

IMS/VS Version 2 Release 1

IMS/VS Release Guide, SC26-4185
IMS/VS Operations and Recovery Guide, SC26-4183
IMS/VS Data Base Recovery Control (DBRC) Guide and Reference, SC26-4029
IMS/VS Diagnosis Guide, LY26-3991
IMS/VS Diagnosis Reference, LY26-3993.

IMS/VS Version 2 Release 2

IMS/VS Release Guide, GC26-4325
IMS/VS Operations Administration Guide, SC26-4323
IMS/VS Sample Operating Procedures, SC26-4324
IMS/VS Diagnosis Guide and Reference, LY27-9526
IMS/VS Program Organization, LY27-9527.

Development management System/CICS/VS

General Information Manual, SH20-0125
User's Guide, SH20-0001
Program Reference, SH20-2209
Installation and Operations Manual, SH20-0004
Problem Determination Manual, SH20-0003
Messages and Codes, SH20-0005.

Database 2

Guide to Publications, GC26-4111
Diagnosis Guide, SY26-3935
Messages and Codes, SC26-4113.

BTAM

OS/VS BTAM, GC27-6980.

ACF/VTAM

VTAM Messages and Codes for MVS and VSE, SC23-0114
Introduction to VTAM, GC27-6987
System Programmer's Guide, SC38-0258
Advanced Communication Function for VTAM: Diagnostics Techniques, SY38-3029.

SNA

Concepts and Products, GC30-3072
Format and Protocols Reference Manual: Architecture Logic for LUType 6.2, SC30-3269
Format and Protocols Reference Manual: Architecture Logic, SC30-3112.

VSAM

MVS/XA VSAM Administration Guide, GC26-4015
MVS/XA VSAM Administration: Macro Instruction Reference, GC26-4016.

PL/I

OS PL/I Optimizing Compiler: Execution Logic, SC33-0025
OS PL/I Optimizing Compiler: Programmer's Guide, SC33-0006
OS PL/I Optimizing Compiler: Messages, SC33-0027.

VS COBOL II

General Information, GC26-4042
Debugging, SC26-4048
Compiler and Library: Diagnosis Guide, SY26-3911
Compiler and Library: Diagnosis Reference, SY26-3910.

IBM 3270 information display system

An Introduction to the 3270 Information Display System, GA27-2739
3274 Control Unit Description and Programmer's Guide, GA23-0061.

General

IBM System/370, 30xx, and 4300 Processors Bibliography, GC20-0001
IBM System/370 Principles of Operation, GA22-7000.

Contents

Customer Information Control System
CICS/MVS Version 2 Release 1
Problem Determination Guide

QUESTIONNAIRE

To help us produce books that meet your needs, please fill in this questionnaire. It would help us if you provide your name and address in case we need to clarify any of the points you raise. Please understand that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

1. Please rate the book on the points shown below
   The book is:

accurate	1	2	3	4	5	inaccurate
readable	1	2	3	4	5	unreadable
well laid out	1	2	3	4	5	badly laid out
well organized	1	2	3	4	5	badly organized
easy to understand	1	2	3	4	5	incomprehensible
adequately illustrated	1	2	3	4	5	inadequately illustrated
has enough examples	1	2	3	4	5	has too few examples

   And the book as a whole?

excellent	1	2	3	4	5	poor

2. When using this book, did you find what you were looking for? _____
   What were you looking for? _____
   What led you to this book? _____
   Did you come straight to this book? _____

3. Which topics does the book handle well?

4. And which does it handle badly?

5. How could the book be improved? _____

6. How often do you use this book?
   Less than once a month? ☐ Monthly? ☐ Weekly? ☐ Daily? ☐

7. What sort of work do you use CICS for? _____

8. How long have **you** been using CICS? ____ years/months

9. Have you any other comments to make?

Thank you for your time and effort. No postage stamp necessary if mailed in the USA. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to either address in the Edition Notice on the back of the title page.)

Note: *Staples can cause problems with automated mail-sorting equipment.*
Please use pressure-sensitive or other gummed tape to seal this form.

SC33-0516-0

Questionnaire

Fold and tape | Please do not staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 6R1H
180 Kost Road
Mechanicsburg, PA 17055, USA

Fold and tape | Please do not staple | Fold and tape

IBM®

Name .....

Job Title ..... Company .....

Address .....

..... Zip .....

Part 1. Approach to problem determination

Chapter 1.1. Techniques of problem determination

CICS is an interactive online terminal system catering for many simultaneous users. As a result, determining the cause of a problem may require techniques that are different from those you would use for finding the cause of a problem in a compiler, for instance, where a problem may be reproduced by attempting the compilation again.

With CICS, in contrast to a compiler, several users may be sharing the same area of storage so that one user may accidentally overwrite another user's data. Or there may be an unexpected interaction when two users access the same data base file. To reproduce these types of problems, which may also be time-dependent, requires the use of special techniques.

These techniques, requiring a logical approach, are introduced in this chapter under the following headings:

- Preliminary checks
- CICS service aids
- Service aids for other components
- Miscellaneous techniques.

Preliminary checks

You should always look at the console output when an error occurs. Before considering dumps, traces, and other problem determination aids, it is important to eliminate, or at least to recognize, possible external causes of the failure. The following check list may be useful:

1. Has the system run successfully before? If not, then some sort of setup problem should be suspected. Are you sure that no error messages have been ignored or overlooked during system generation or initialization?

2. Have any changes (fixes, PTFs, user modifications, table changes, hardware) been made since the last successful run? If so, does removal of the fixes allow successful execution? Have all affected modules been reassembled and/or link-edited or was there any problem when applying the fix or PTF?

3. Are the same parameters being used to start the system (JCL, CICS SIT, console overrides, VTAM CONFIG/LIST, NCP load module, 3600 application) as when the system last ran successfully?

4. Have there been any CICS application changes since the last successful run?

5. Is the problem a solid failure? Can it be re-created? Does it always show the same symptoms? Is the problem related to a certain application, CICS task, or application request, for example, SEND or RECEIVE? Is it a time-dependent problem relating to processing unit load or network activity?

6. Is the whole network affected?

Note: If CICS is started by an OS/VS procedure, there is no SYSOUT listing of the JCL or the error messages.

Even if the answers to the questions above do not suggest the cause of the problem, it should still be remembered that even a small telecommunications network is a very complex system in which all components depend on each other. If one component fails and presents incorrect information to the other components, they may fail more severely. Sometimes the failure may be considerably delayed so that error indicators may be lost before the error is detected, or the error may have side effects which obscure its original nature.

The most important aspect of problem determination is to find the **beginning** of a failure.

The beginning of a failure is not normally when the first unexplained failure occurs but may be much earlier. A trace table from a successful run may be invaluable for spotting unexpected changes from the normal.

If you are reasonably certain that you have a CICS problem, or that you need to examine a CICS manifestation of a problem from another component, then the problem determination flowcharts in "Chapter 1.2. Problem determination flowcharts" on page 11 are designed to help you to start logically.

CICS service aids

Trace table

CICS maintains an internal trace table to record activity in the system. An entry is made in the trace table each time a CICS management program is entered. Entries may also be made by user application programs. The trace table is described in "Chapter 2.1. Trace" on page 45.

Auxiliary trace

The trace table mentioned above uses a table in main storage that wraps around whenever it is full, overwriting old entries. To avoid losing entries in this way, auxiliary trace may be used. Auxiliary trace puts entries into a buffer and, when the buffer is full, writes it to a data set. The data set may later be printed using the trace utility program, DFHTUP, as described in the *Operations Guide*.

XRF trace

If you are using XRF, there is an additional trace of the XRF processes which always takes place. For details of the trace entries, see "Chapter 2.2. XRF trace."

Formatted dump

Under certain conditions a formatted dump of the CICS address space may be produced. (To save time and paper, programs and areas of unallocated storage are not printed.) The dump is intended primarily for the system programmer, rather than the application programmer, and is described in "Chapter 2.3. Formatted dump" on page 131. You can also specify an unformatted dump, a SNAP dump, or an SDUMP dump.

Program check and abend trace table

Included in a formatted dump (and in a system dump) is a program check and abend trace table which contains information relating to the last six program checks and abends; the table is in the form of a wraparound table and its layout is described in "Chapter 2.3. Formatted dump" on page 131.

Transaction dump

At any time, a transaction can request that a transaction dump be taken and then continue execution. A transaction dump is also taken if a transaction terminates abnormally. A transaction dump contains only those areas directly related to the transaction itself and is intended primarily for use by the application programmer in debugging his application program.

Short symptom string

A short symptom string is written at the head of every formatted dump and transaction dump. The short symptom string gives the abend code, the name of the CICS component, the name of the module being executed when the dump is taken, and the module identification from LIFO storage. This information is used as the initial search argument in the Software Support Facility. For a description of the short symptom string, see "Chapter 2.3. Formatted dump" on page 131.

CSMT, CSTL, and CXRF transient data destinations

During execution, the terminal (or node) abnormal condition program, DFHTACP (or DFHZNAC), and the abnormal condition program, DFHACP, write terminal error and transaction abend messages, respectively, to the transient data destination CSMT. In addition DFHTACP (or DFHZNAC) writes terminal I/O error messages to the transient data destination CSTL. During or after every execution of CICS, the user should print the contents of these transient data destinations. If they are extrapartition destinations, this can be achieved by assigning the data set to a printer (UNIT=) or to a SYSOUT data set, or by assigning it to a disk or tape and using a utility program. If they are intrapartition destinations then either DFH$TDWT or a user-written transaction must be used to retrieve the messages and send them to a terminal – either on operator request or by means of a trigger level.

The CXRF transient data destination is used by an alternate system for terminal control messages, because the CSMT and CSTL destinations may not be available to it.

Statistics

During execution, the CICS system keeps a record of many statistics which may be useful for determining the cause of performance problems in particular. The statistics include the frequencies of use of transactions, programs, files, the numbers and types of transaction abends, storage violations, and transaction restarts. Statistics can be obtained on **request** (at destination CSSL or a user-specified destination) or **automatically** at user-defined intervals (at destinations CSSM and CSSN). Statistics are also printed when CICS shuts down. See the *CICS-Supplied Transactions* manual, or the *Operations Guide* for further information.

Command interpreter

The command interpreter assists the user in writing syntactically correct CICS commands, and shows the results of execution of a command entered on a display. (For details of how to use the command interpreter, see the *Application Programmer's Reference.*)

Execution (command-level) Diagnostic Facility (EDF)

The execution (command-level) diagnostic facility (EDF) is designed to help an application programmer test and debug command-level application programs. EDF can also help system programmers and IBM Customer Engineers to determine if a particular program failure is due to an application program error or a control program error. (For details of how to use EDF, see the *Application Programmer's Reference.*)

Note: The PCT entry for CEDF has the default value of TRACE = NO. If trace entries are required, the PCT entry should be changed to TRACE = YES. The trace entries for the transaction being executed under EDF are not affected.

CSFE transaction

The transaction CSFE is designed to help the IBM field engineer to diagnose terminal problems and software problems. The terminal test function is applicable to all terminals supported by CICS, except the IBM 2780 Data Transmission Terminal, the IBM 3600 Finance Communication System, the IBM 3614 Consumer Transaction Facility, the IBM 3735 Programmable Buffered Terminal, output-only printers (for example, the IBM 3270 Information Display System printers), and terminals communicating with the IBM 7770 Audio Response Unit. The terminal function allows all valid characters for a terminal to be printed at the terminal.

In addition, you can use CSFE to activate any of the following kinds of trace: FE, system, user; and to activate two special purpose trace facilities: the system spooling interface trace (SPOOLFE) and the DFHZCP activate scan trace.

When the DEBUG operand is specified, the storage used by a transaction can be held until the end of the transaction, or a regular check can be made of the free area queue element (FAQE) chains and transaction storage chains.

Full details of the CSFE transaction can be found in the *CICS-Supplied Transactions* manual.

Storage freeze

Certain classes of CICS storage which are normally freed during the processing of a transaction can, optionally, be kept intact and freed only at the end of the transaction. Thus, in the event of an abend, the dump contains a record of the storage that would otherwise have been lost (including the storage used by CICS service modules). The classes of storage that can be frozen in this way are those in subpool 2 (teleprocessing subpool, except for line storage), and subpool 4 (task subpool).

The storage freeze function is invoked by the CSFE transaction; see the *CICS-Supplied Transactions* manual for information on how to use CSFE.

This function uses storage freeze bits in the program control table (PCT) and in the TCA. The storage freeze bit in the PCT is initially turned off, but can be turned on or off by the CSFE transaction.

During transaction initialization, DFHKCP sets the TCA storage freeze bit to be the same as the PCT bit. Prior to handling a FREEMAIN request, DFHSCP tests the TCA bit and, if the bit is on, bypasses the FREEMAIN request. During transaction termination, DFHKCP turns the TCA's storage freeze bit off to ensure that frozen storage gets freed correctly.

Storage violation trap

Segments of unused storage in a CICS system are chained together in free area queue element (FAQE) chains. If the CSFE transaction is invoked with the operands DEBUG,FAQE=ON, then each time the trace program is entered, a scan is made of these chains to check that they have not been corrupted.

You can also use the storage violation trap in the trace program to check the addresses and storage accounting areas on transaction storage chains. Do this by invoking the CSFE transaction with the operands DEBUG,TASKSTG=ON.

When a storage violation is detected by the trap, a formatted dump is produced, and the trap turned off.

For further details, see "Using the storage violation trap" on page 267.

Temporary storage browse

For all application programs, the temporary storage browse (CEBR) transaction is useful. If you are using VS COBOL II, you can also look at the diagnostics. More details on this are given in "VS COBOL II application programs" on page 180, and the *Application Programmer's Reference.*

PTF number

The PTF number is imbedded in a module, near the beginning of the module, so you can easily see it in a dump.

Service aids for other components

MVS/XA

MVS/XA service aids and debugging procedures are described in *Diagnostic Techniques*, and in *System Programming Library: Service Aids*.

The Generalized Trace Facility (GTF) is likely to be the most useful service aid.

VTAM

VTAM service aids are described in the *VTAM OS/VS2 MVS Debugging Guide*.

The most useful VTAM service aids are the I/O and Buffer traces.

External security manager

CICS provides an interface to an external security manager, which can be the Resource Access Control Facility (RACF) licensed program.

Messages produced by RACF are described in the *MVS/XA Message Library: System Messages* manual.

Database 2

For Database 2 problem determination information, see the *Database 2 Diagnosis Guide*, and the *Database 2 Messages and Codes* manual.

Miscellaneous techniques

CEDA transaction

The CEDA transaction enables the interactive definition and dynamic addition of transactions, profiles, programs, map sets, partition sets, terminals, connections, and sessions while CICS is executing. For a description of the CEDA transaction, see the *Resource Definition (Online)* manual.

CEMT transaction

If, as a result of a problem, a program or table defined in the processing program table (PPT) needs to be reassembled, it is not necessary to terminate CICS in order to test the new version of the program. After link-editing the new version into the appropriate library, use the master terminal command, CEMT SET PROGRAM(id) NEWCOPY.

Future requests to load or link to the specified program will then cause the new version to be obtained. If the program is resident, it will become nonresident when you do the NEWCOPY.

The CEMT transaction has other options that are useful for debugging. You can examine or change the system parameters for tasks, transactions, programs, data sets, queues, terminals, traces and dumps. You can also perform snap dumps using this transaction.

Chapter 1.2. Problem determination flowcharts

The flowcharts in this chapter are intended primarily to help the newcomer to CICS problems; the intention is to help you start in the right direction and to approach each problem logically. (Experienced personnel may not need to use these charts – the decisions will probably be performed automatically anyway.) For more information on examining the contents of a control block, see the chapter on failure analysis structure tables in the *Messages and Codes* manual.

Errors can be classified in the following categories:

Abends

Waits

Loops

Incorrect output.

Abends

Abends are the commonest and most easily recognizable errors. You are informed of an abend by a message at a terminal, at the console, or in the CSMT destination. A dump can optionally be produced. The key to fixing an abend problem is to look at a description of the abend in the *Messages and Codes* manual. If this is not enough, you can look at a dump, or use the execution diagnostic facility (EDF) to solve the error.

Some abends are caused by a program check. A program check is detected by the hardware, for example, trying to write over protected storage. Other abends are detected by the software, for example, trying to write on a file which has not been defined.

Note: The abend code AICA is a CICS-detected loop (see "Loops" on page 12).

Waits

A wait is recognized by the operator's observations. Symptoms of a wait are: terminal(s) locked, slow response time, or a transaction active for a long time or suspended. If a wait is suspected, you can use the CEMT transaction to check if there is a **task** associated with the terminal. If there is no task, the problem is incorrect output. If there is a task, you can use CEMT to abnormally terminate the task. You should look at the trace table to see what happened to the task with the same number.

When CICS is in a wait state, the task dispatcher could be searching for work to do. Use the Generalised Trace Facility (GTF) trace to see if there is any activity in the CICS address space. You should at least see activity in DFHKCP.

Loops

A loop produces an AICA abend code if detected by CICS, or else the system appears to be tied up. You can use CEMT to find the status of tasks, and abnormally terminate the task that is looping. You can look at the trace table to find what happened, or, if this is not enough, insert your own trace commands in the suspect portion of the application program.

It is useful to see if the address spaces with lower priority are also tied up. If so, this is a good indication of a loop.

Incorrect output

Incorrect output can be no output, or garbled output. The cause is usually an error in an application program, or a mistake in setting up your system, such as having an incorrect map. You should insert trace commands, or use the execution diagnostic facility (EDF) to step through the program.

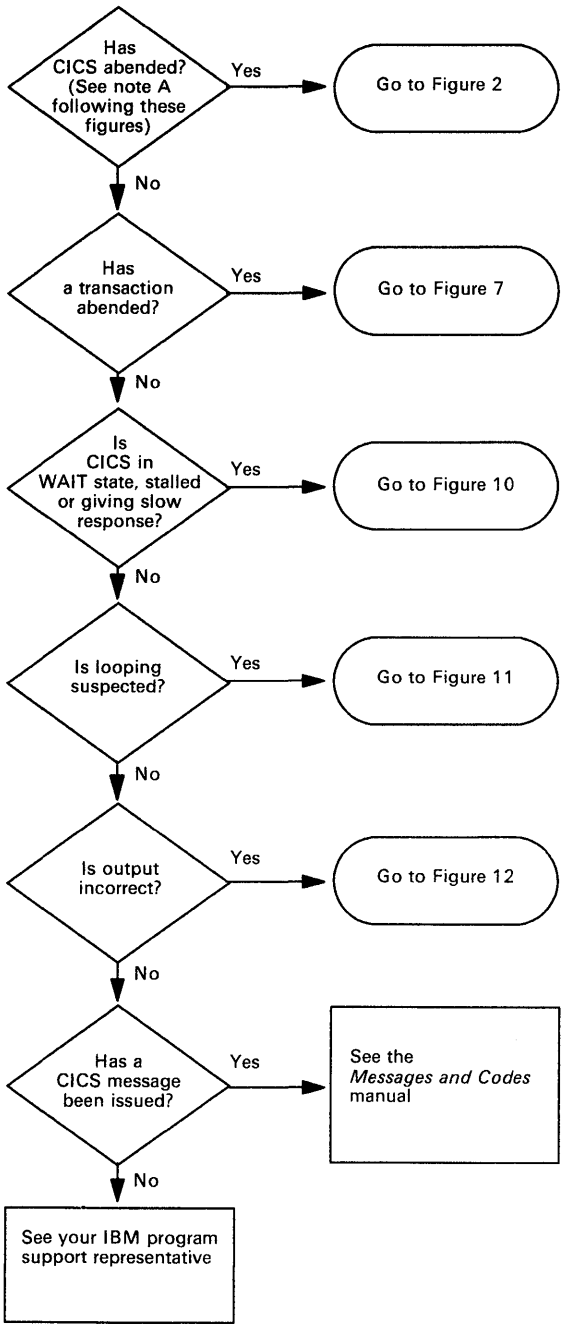


Figure 1. A CICS problem is suspected

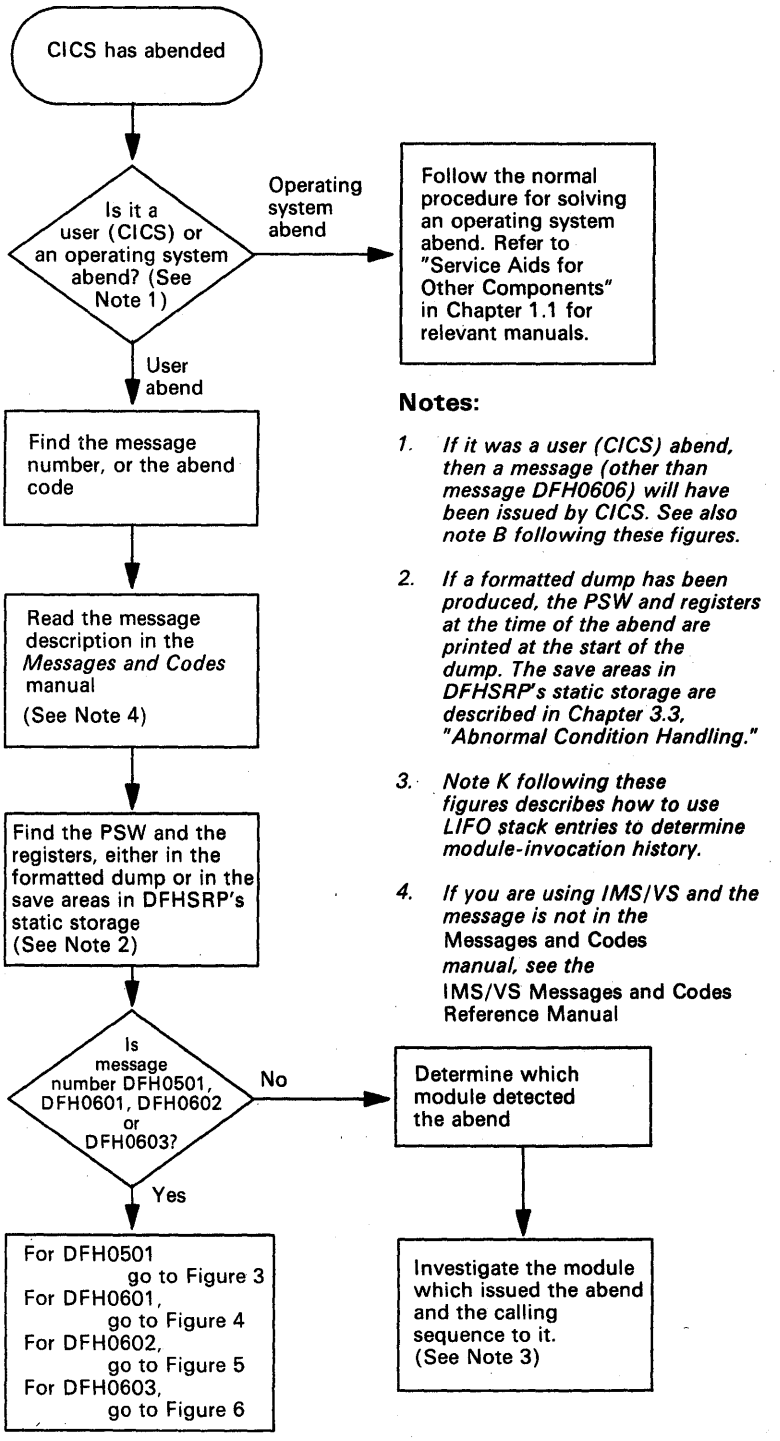


Figure 2. CICS has abended

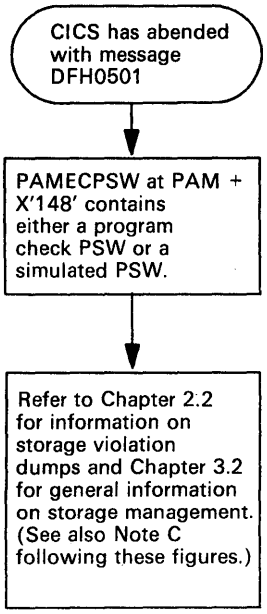


Figure 3. CICS has abended with message DFH0501

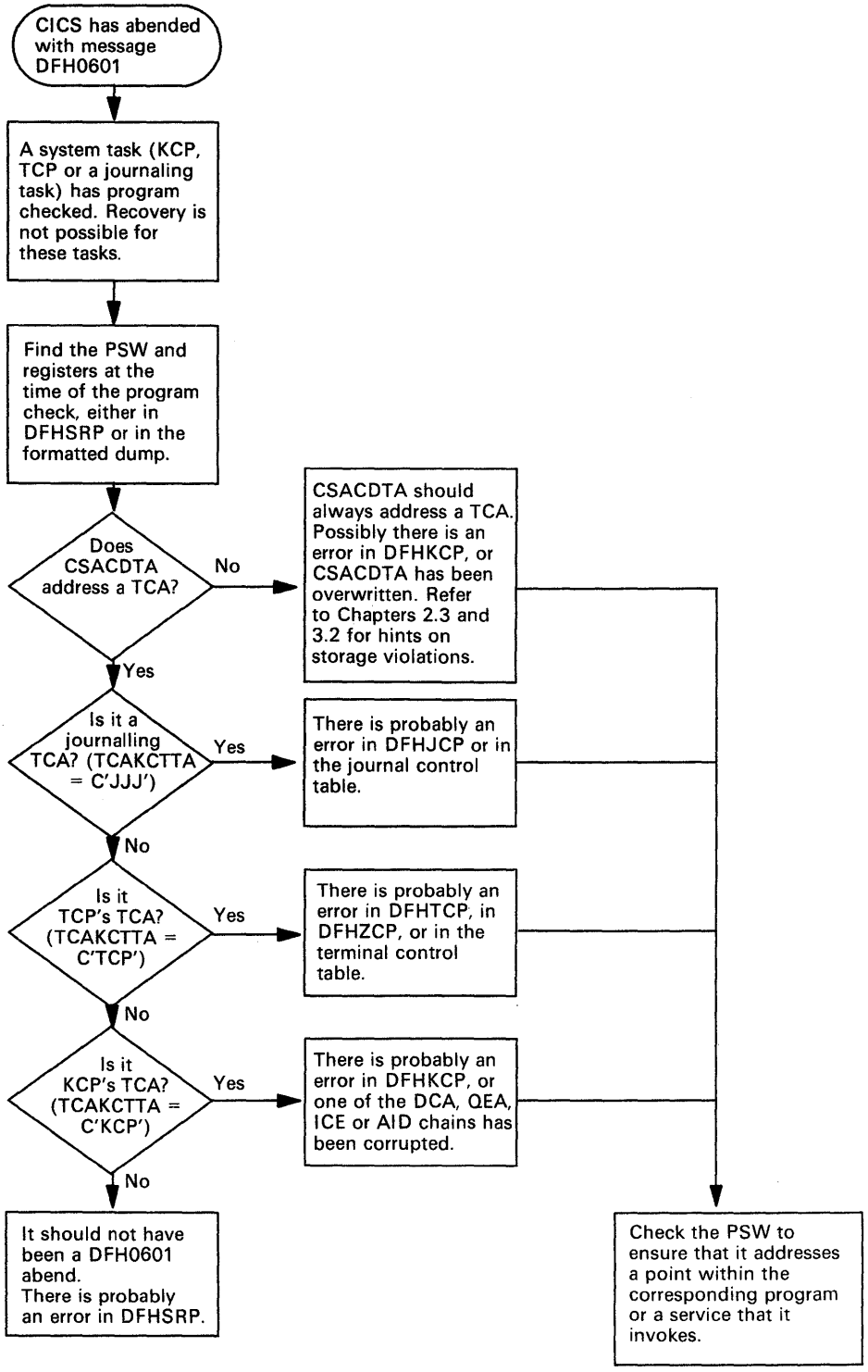


Figure 4. CICS has abended with message DFH0601

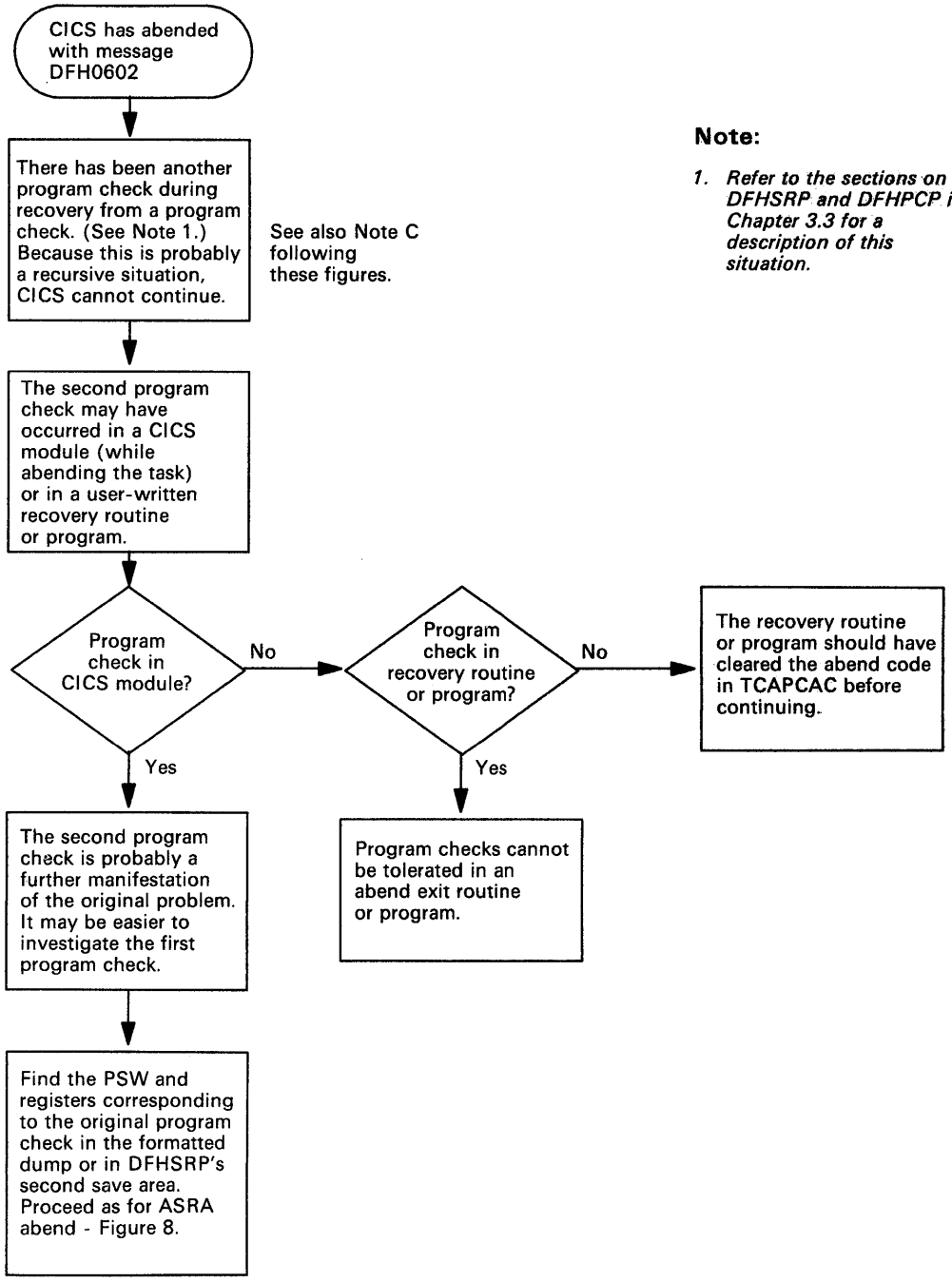


Figure 5. CICS has abended with message DFH0602

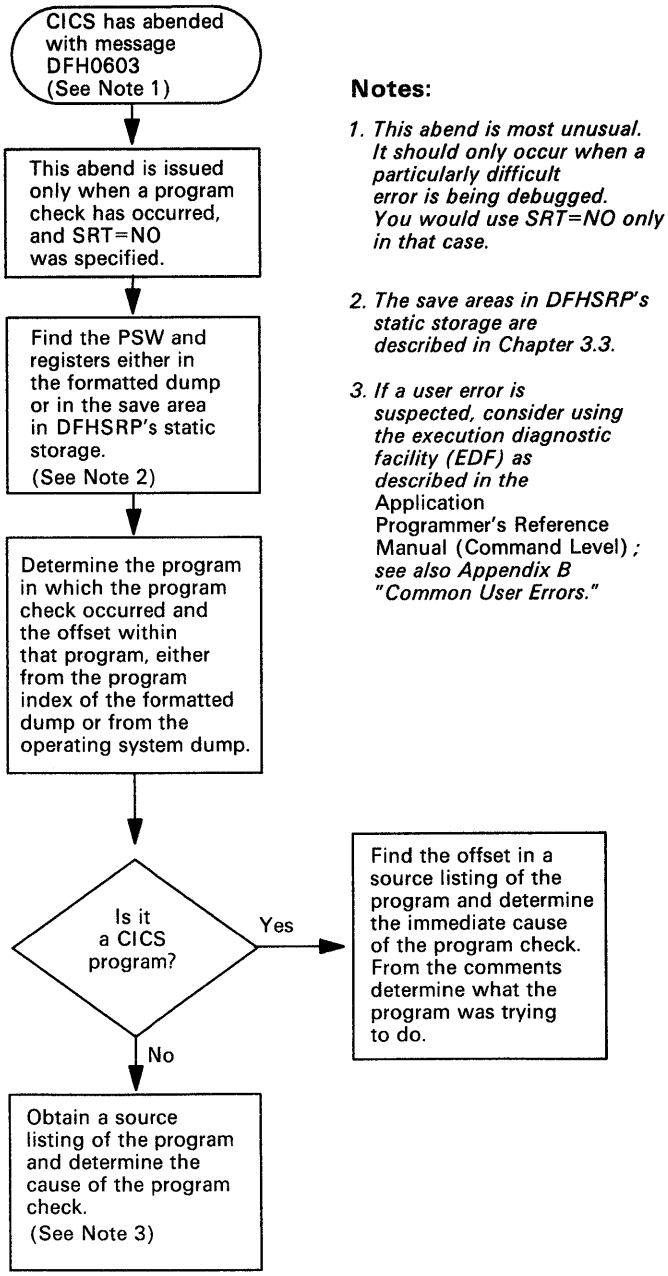


Figure 6. CICS has abended with message DFH0603

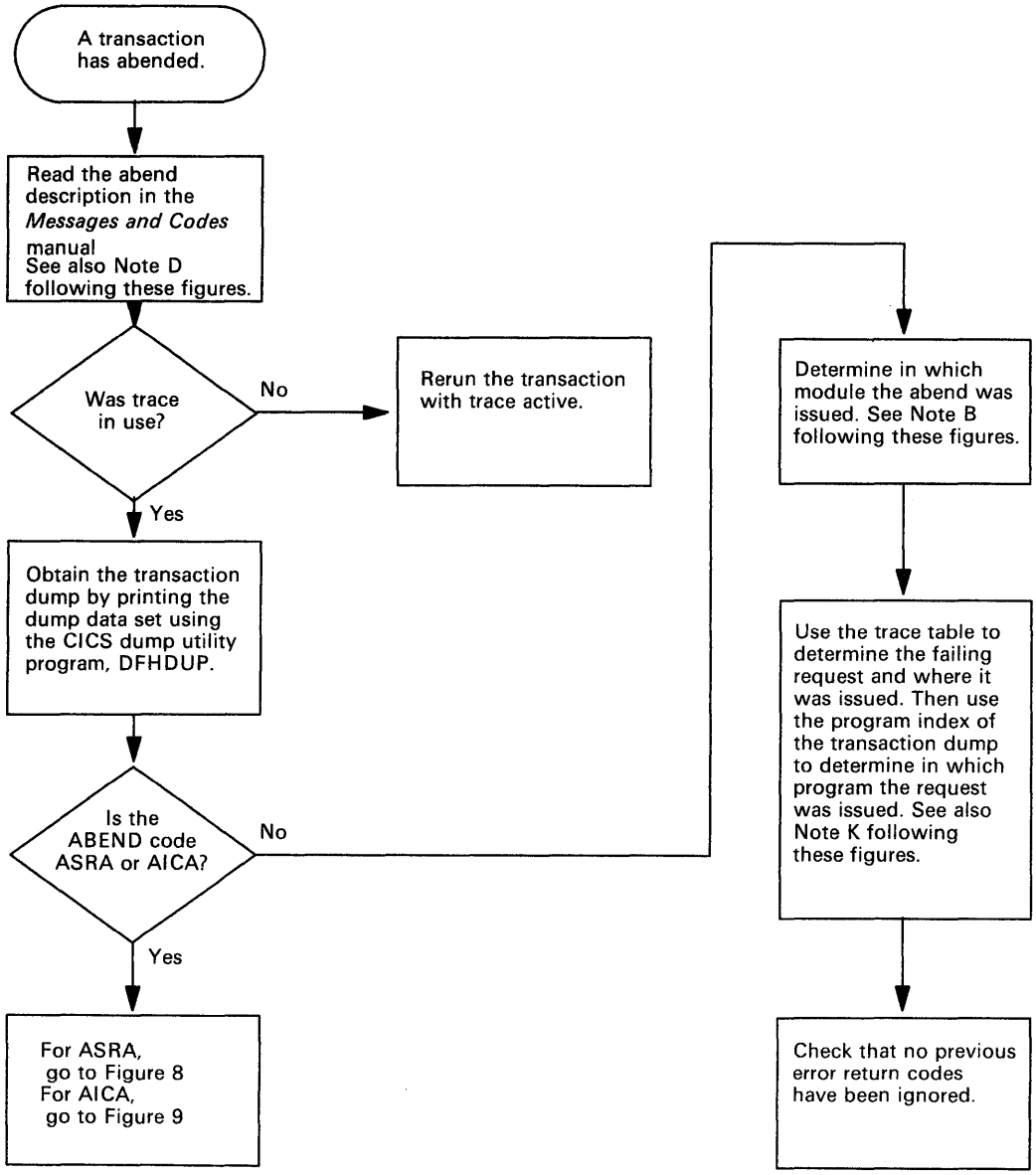
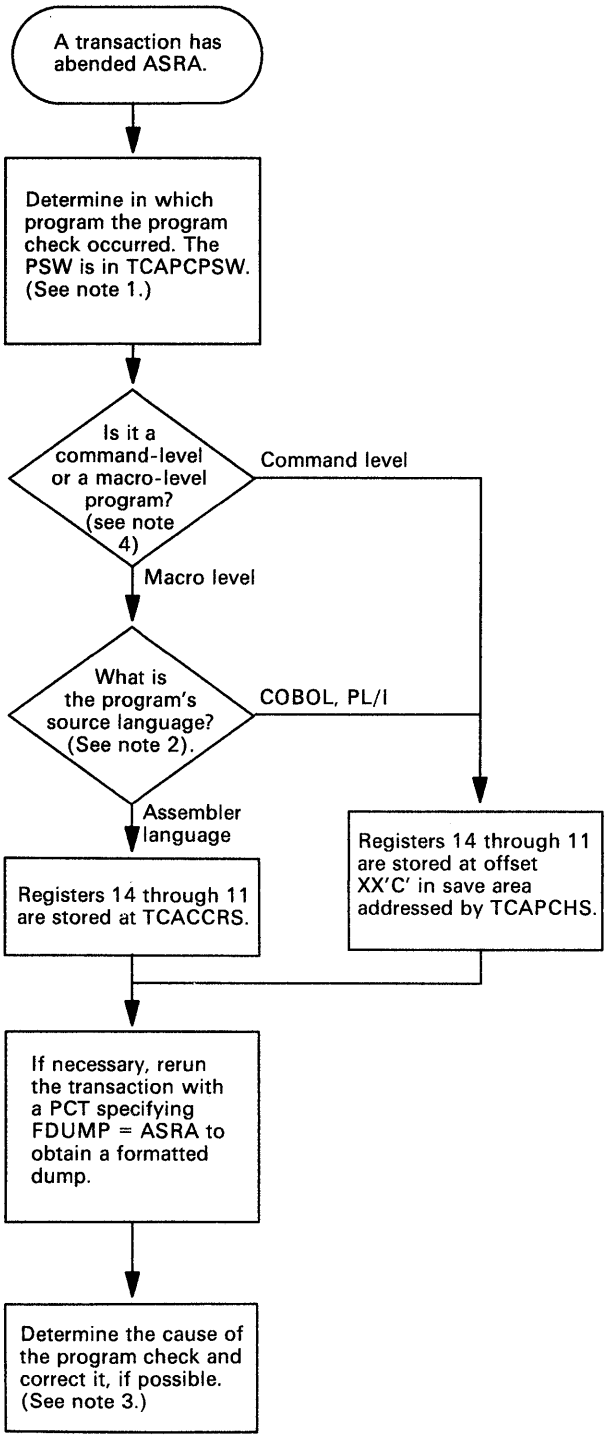


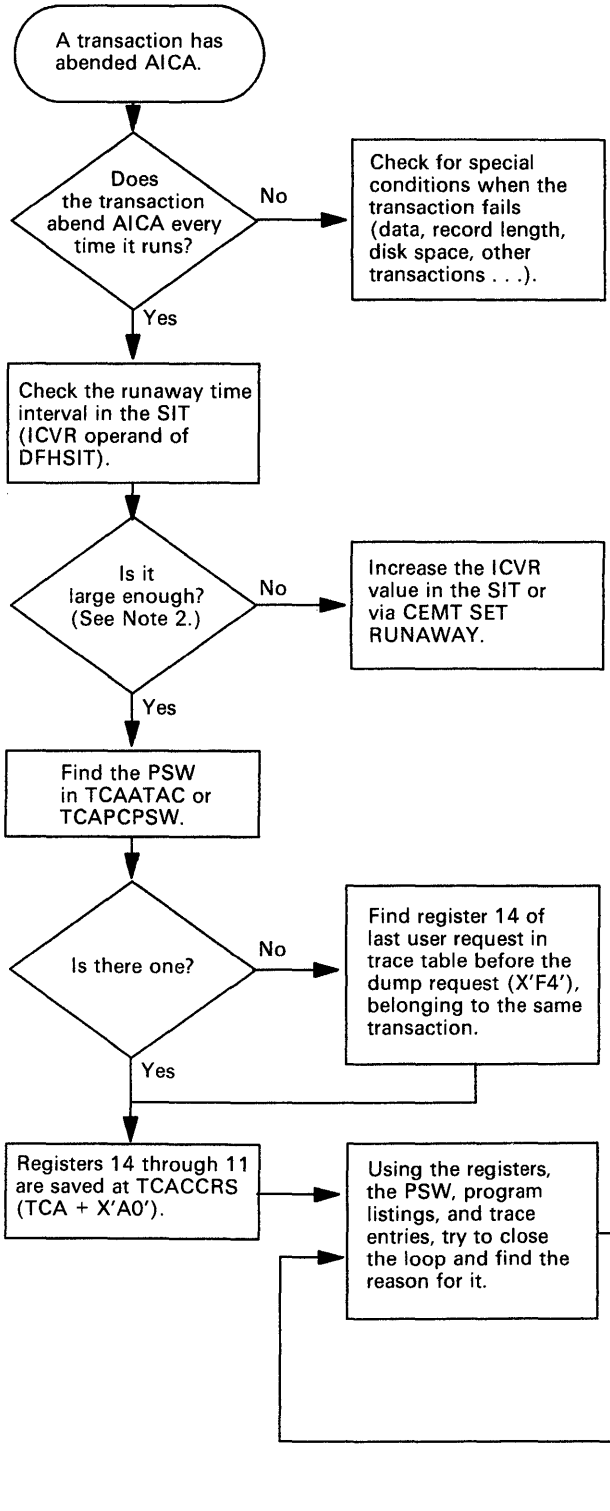
Figure 7. A transaction has abended



Notes:

1. *If the PSW does not address a location in the user's program, then it will be necessary to use the program index of the transaction dump.*
2. *The program's source language can be determined from bits 1, 2, and 3 in the PPTTLR byte of the PPT:*
   Bit 2 on = PL/I
   Bit 3 on = COBOL
   Bits 1, 2, and 3 all off = Assembler
3. *Note K following these figures describes how to use LIFO stack entries to determine module invocation history.*
4. *Command-level programs start with 'DFHYxxxx' at the load point in the dump.*

Figure 8. A transaction has abended ASRA (program check)



Notes:

1. *For further information concerning runaway task abend processing see the section on the task management program in the* Diagnosis Reference Manual
2. *The value required for ICVR will depend on many factors. A large value may be required when running on a small processor, when running under VM/370, and so on.*

Figure 9. A transaction has abended AICA (runaway)

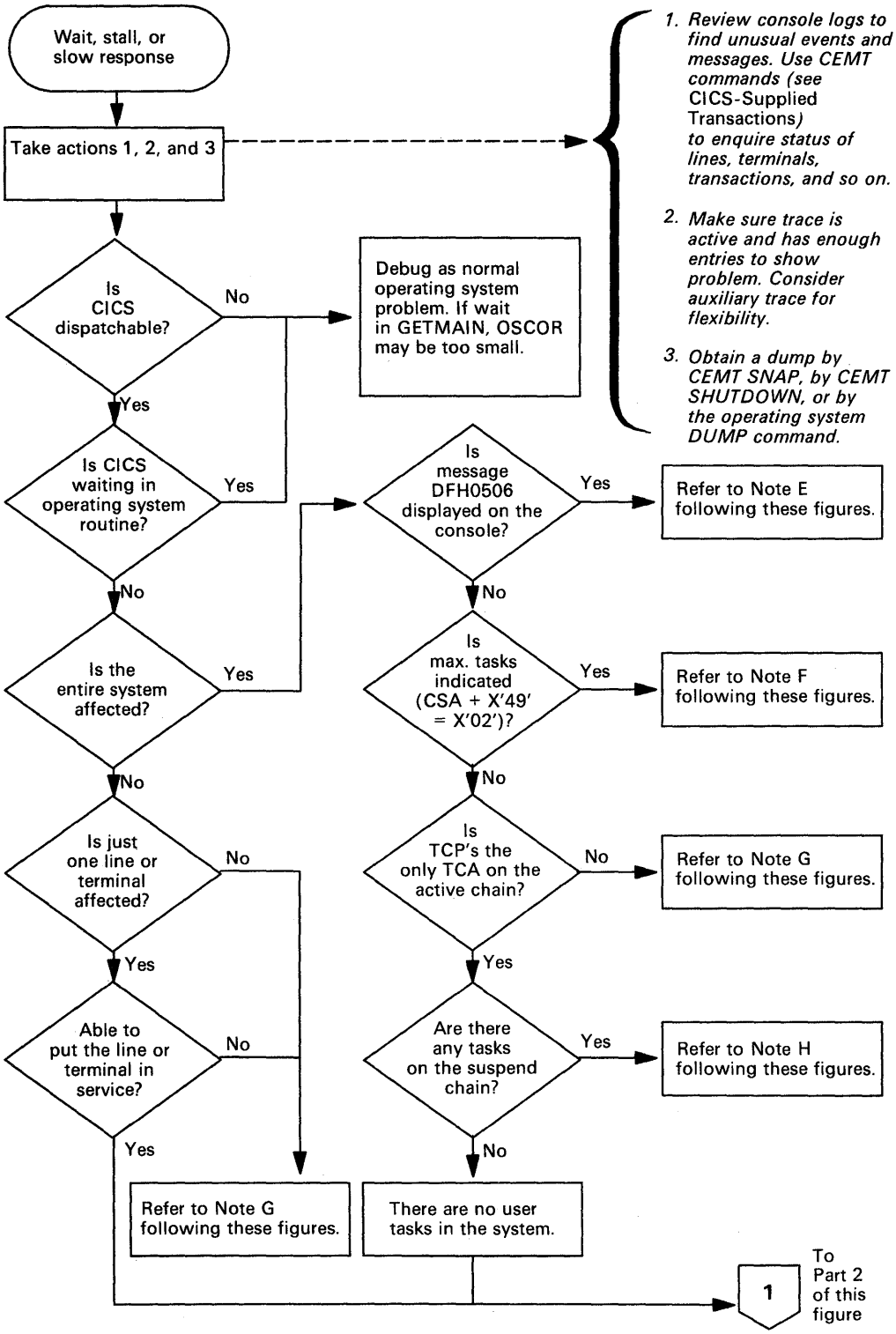


Figure 10 (Part 1 of 2). CICS is in a wait state, has stalled, or is giving slow response

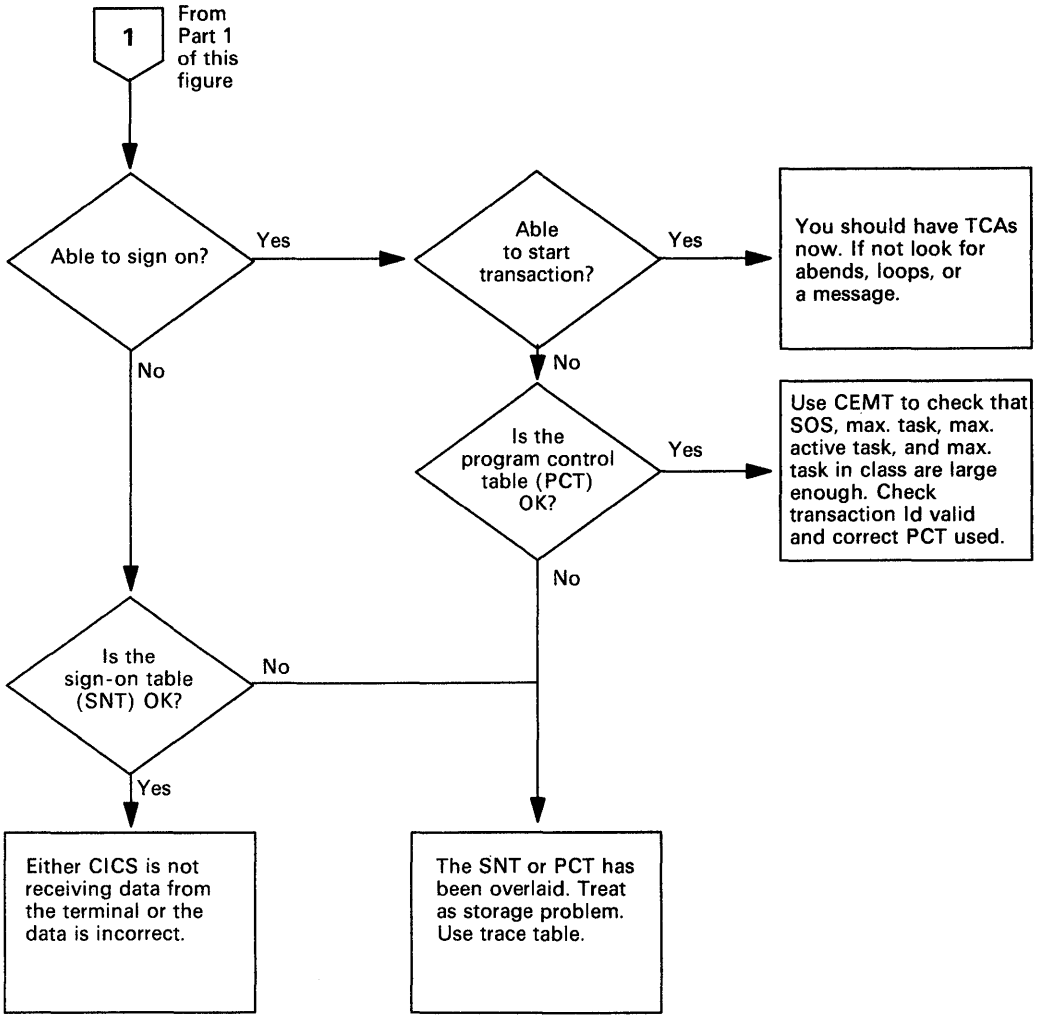


Figure 10 (Part 2 of 2). CICS is in a wait state, has stalled, or is giving slow response

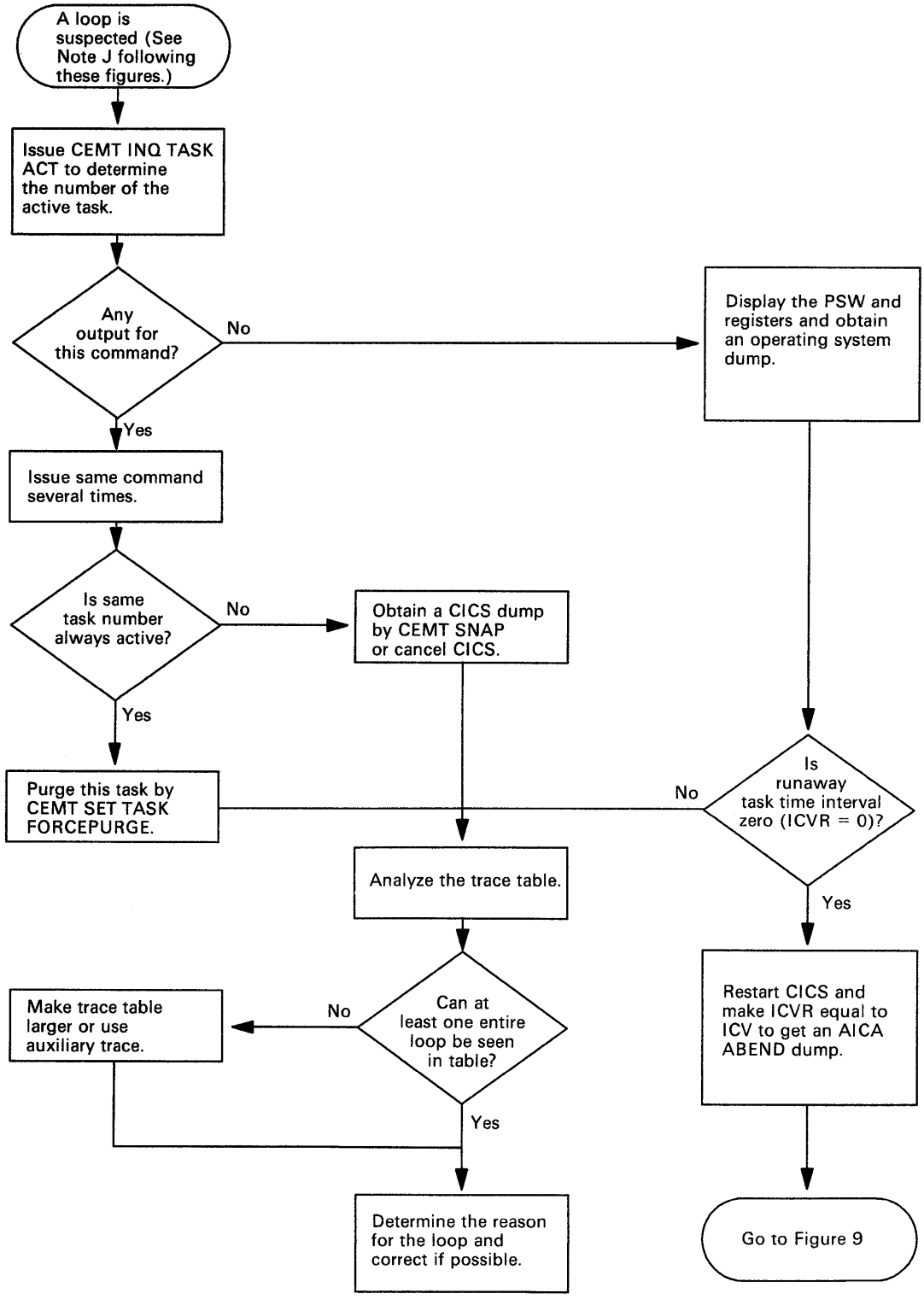


Figure 11. A loop is suspected

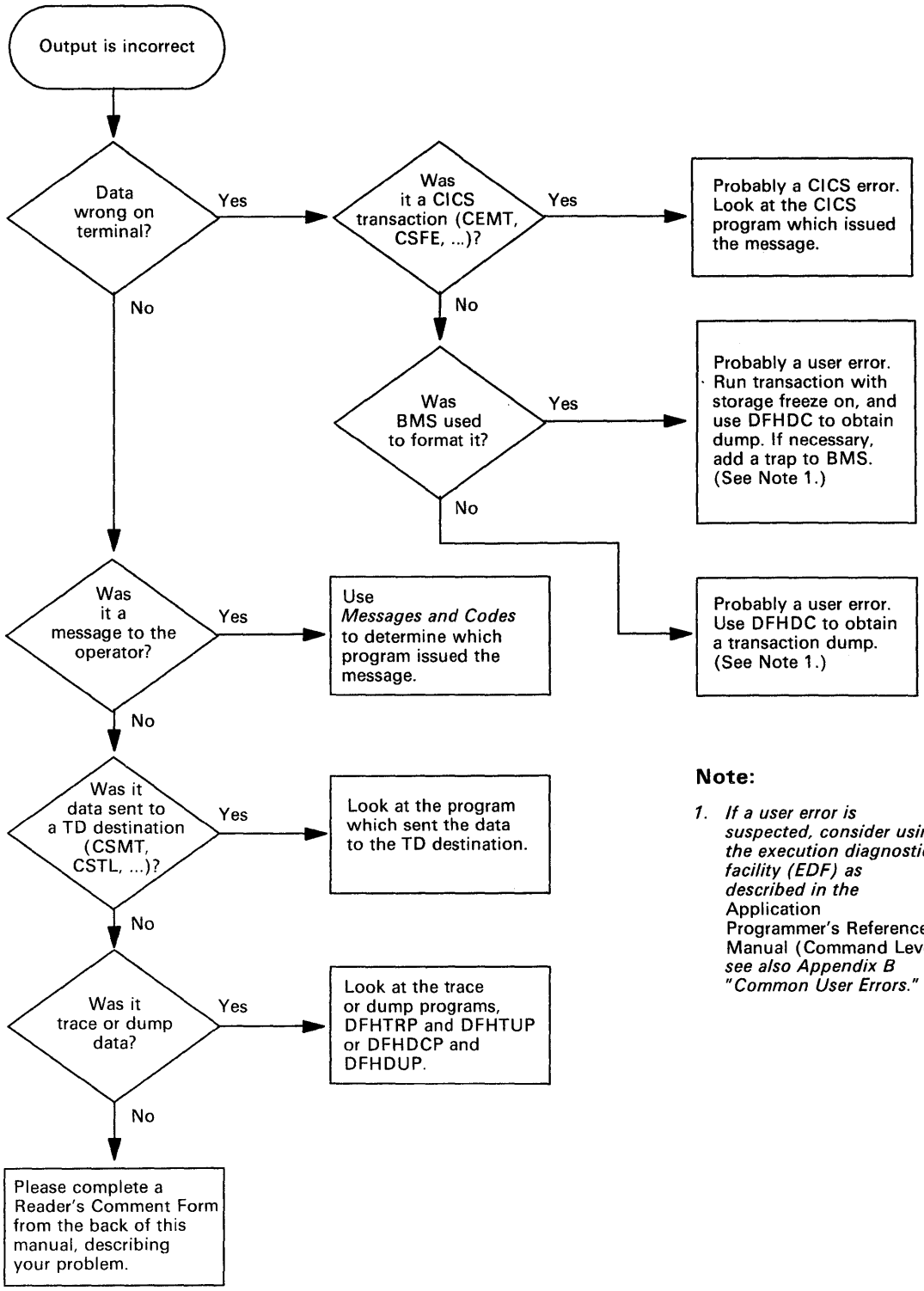


Figure 12. Output is incorrect

Note A

There are two levels of abnormal termination in CICS. The first level abnormally terminates a single transaction, dumps that transaction and its associated storage areas to the CICS dump file, and logs error messages to the master terminal and to the terminal attached to that transaction. The second level is more serious in that CICS itself terminates abnormally. A message is sent to the system console, and a dump goes to the appropriate dump data set. The CICS dump file should always be printed in order to have all available information for diagnosing the cause of the abend. Optionally, a formatted dump is provided to indicate the reason for the system abend.

An example of an abend that terminates CICS is associated with the message DFH1310. This message is produced when CICS is attempting to compress a temporary storage data set and an error occurs. For guidance in debugging this problem, see the chapter on failure analysis structure tables in the *Messages and Codes* manual.

Note B

CICS messages are documented in *Messages and Codes* manual. A standard message identifier is displayed as the first part of each message text. The message text and identifier have the following format:

DFHccnn message

where:

DFH = An IBM-assigned identification for CICS modules.

cc = The applicable CICS component code shown in Figure 13 on page 27.

nn = Any unique 2-digit code assigned by CICS to identify the particular message or group of messages within a given assembled program. Progress messages (for example, DFH1500 – CICS START-UP IS IN PROGRESS and DFH1500 LOADING CICS NUCLEUS) may be grouped under the same identifier. For all other messages, unique identifiers are assigned.

```
03 - DFHKCP    Task management
04 - DFHPCP    Program control
05 - DFHSCP    Storage control
06 - DFHSRP    System recovery
07 - DFHDCP    Dump control
08 - DFHTAJP   Time adjustment
09 - DFHFCP,   File control
     DFHVSP,
     and DFHVAP
10 - DFHTCP    Terminal control
11 - DFHSKP    Subtask control
12 - DFHTDP    Transient data
13 - DFHTSP    Temporary storage
14 - DFHTRP    Trace control
15 - DFHSIP    System initialization
16 - DFHDUP    Dump utility
17 - DFHSTP    System termination
18 - DFHSTKC   Statistics
20 - DFHACP    Abnormal end
21 - DFHZNAC   Network error
22 - DFHACP    Abnormal end
23 - DFHZCP    Terminal control
24 - DFHZNAC   Network error
25 - DFHTACP   Terminal error
26 - DFHZEMW   Error msg writer
28 - DFHRUP    Recovery utility
29 - DFHTEOF   Tape end-of-file
30 - DFHMTPA   Master terminal
31 - Keypoint modules
32 - DFHLFO    LIFO storage
33 - DFHFEP    Field engineering program
34 - DFHZNAC   Network error
35 - DFHSNP    Sign-on
36 - DFHXSP    Security
37 - Interregion (IRC) modules
38 - DFHGAP    Graphics attention
39 - DFHDLI    DL/I
40 - DFHMCP    Mapping control
41 - DFHTPR    BMS page retrieval
42 - DFHZCNR   Console read request
43 - DFHCRS    Remote scheduler
44 - DFHRTE    Transaction routing
45 - DFHJCP    Journal control
46 - DFHDBP    Dynamic backout
47 - DFHVCP    Volume control
48 - DFHAMP    Allocation management
49 - LU6.2 modules
50 - DFHFDP    Formatted dump
51 - DFHCSDUP  CSD utility
52 - DFHCSDUP  CSD utility
53 - DFHPSP    System spooling interface
54 - DFHPSP    System spooling interface
56 - DFHCMP    Monitoring
57 - Emergency restart backout
58 - DFHAKP    Activity keypoint
59 - DFHBSS    TCT entry builder
60 - DFHTOR    Terminal object resolution
61 - DFHFTAP   Format tape
62 - DFHTBS    Table builder services
70 - DFHECP    COBOL HLPI translator
```

Figure 13. CICS component codes

Note C

DFH0501

The storage chain has been broken or is invalid.

1. Find the TCA.

2. Ensure that the program has not exceeded the size of the TWA (transaction work area) as specified in the PCT. The literal 'LIFOSTOR' is placed at the end of the TWA; ensure that it is not overwritten. (See "Debugging problems caused by overwriting LIFO stack information" on page 272.)

3. Find the trace table.

4. The last trace table entry of type X'F1' (storage control) identifies the type of storage. If it is terminal storage, check the terminal storage from TCTTESC. If it is user storage, check the storage chained from TCASCCA. The storage accounting area describes the storage chain. If there was a storage violation, the trace entry X'CA' follows X'F1'.

5. If any of the storage chains is broken, determine the source of the data that overlaid the chain, and correct the program.

6. A common error is to obtain a map area, file area, or TWA that is too small. When data is moved into the storage area, the storage chains are overlaid.

7. For message DFH0501, if recovery has not been specified, DFHSCR includes "RECOVERY NOT SPECIFIED" in the message. If recovery is not attempted, "RECOVERY NOT POSSIBLE" is included. See Abend U501 in the failure analysis structure tables in the *Messages and Codes* manual.

   Look at PAMECINT in the PAM data area. The second half of PAMECINT is the interrupt code, and if this = X'011C', then the upper boundary of the area being freed overlaps an existing FAQE. In this case, register 1 contains the length of the area, register 7 the address of the area, and register 14 the address of the FAQE.

DFH0602

A program interrupt has been encountered while the system recovery program is running.

1. Check the storage chains as described in DFH0501 above.

2. Find the reason for the program check.

3. If the above steps do not identify the error, it may be a CICS system error.

Note D

All CICS transaction abend codes are 4-character alphabetic codes:

```
AXXY

A  =  Fixed IBM assigned designation.
XX =  Component code, for example, BM for basic mapping support
      or IC for interval control.
Y  =  1-character alphanumeric code assigned by CICS.
```

Many of these codes are described in the *Messages and Codes* manual and need no further explanation. Other codes are described in the corresponding manual for the program that was executing with CICS.

Optionally a dump is provided by CICS on one of the dump data sets which can be formatted and printed with the CICS dump utility (DFHDUP). This dump can assist in debugging abends which need further investigation.

Note E

The message "DFH0506 – CICS IS UNDER STRESS – SHORT-ON-STORAGE" appearing on the console indicates that some GETMAINs cannot be satisfied. Any task that has issued unsuccessful unconditional GETMAINs is on the suspend chain.

Some common reasons for this situation are:

- Not enough storage specified for the address space.
- User applications not releasing storage.
- Storage chains destroyed, causing lost storage areas – as much as one page of the task subpool could be lost at one time. Look for previous DFH05xx type messages for storage violations.
- Storage cushion not specified properly.
- Max task (MXT), max active task (AMXT) or max task in class (CMXT) value too high.
- Storage freeze in use.

Sometimes, a map of all CICS storage in use can be helpful. Locate the page allocation map (PAM) which is addressed by CSAPAMA. From this, the allocated and/or free storage in various subpools can be mapped out. For individual TCAs, storage freed and allocated are chained from TCASCFFC and TCASCCA respectively.

Note F

Either the value of max task (MXT operand on the DFHSIT macro) has been set incorrectly, or tasks are not terminating. Check the active and suspended chains as described in "Chapter 1.3. Approach to wait problems" on page 33 of this manual.

Note G

Examine each DCA on the active DCA chain and see if it is dispatchable (DCATCDC). If some DCAs are dispatchable and no external activity is noticed, it is probable that the highest priority task (other than the terminal control task) is looping, and the runaway task value is not being used or is set too high.

When not dispatchable, the tasks are waiting for some events indicated by TCATCEA. From this and TCAFCAAA and fields in the TCA common section, the type of activity the task is waiting for can be determined. In general, a task waits when the application issues a DFHKC TYPE = WAIT macro or waits for an I/O operation.

Refer to "Chapter 1.3. Approach to wait problems" on page 33 for further information.

Note H

A task is put on the suspended chain when it is waiting for some resources to be released by other tasks. TCATCDC indicates the reason for suspension. DCATCDC also gives the reason for suspending the task.

Some common reasons for suspending a task are:

DFHSCP Waiting for storage. The amount and type of storage requested are in the TCA in TCASCNB (number of bytes) and TCASCTR.

DFHKCP Enqueued on some resources. The resource name is in TCATCQA which is also available from the trace table. CSAQCAA contains the address of the QCA that addresses queues of resources (QEAs), from which the DCAs owning them can be found. See Figure 26 on page 148.

DFHTCP Tasks requesting I/O with WAIT specified.

DFHJCP Each journal task itself is suspended when there is no outstanding I/O request for its journal. Remember there is one journal task for each open journal.

DFHICP A DFHIC request has been issued and the task is waiting for an ICE to expire. Task will wait until the expiration time has elapsed.

DFHALP There are no sessions free to satisfy an ALLOCATE command. The task is suspended until one becomes free.

DFHTSP When there is no auxiliary storage, DFHTSP suspends a task if the request is not conditional. When some temporary storage space is freed, the task will be resumed.

A task can be suspended for reasons other than lack of resources:

The mirror program, DFHMIR, suspends itself when it has completed its work.

The EXEC diagnostic facility, DFHEDF, suspends the subject task.

The new connections transaction, CSNC, suspends itself when it has nothing more to do.

Refer to "Chapter 1.3. Approach to wait problems" on page 33 and "Chapter 3.5. CICS in a wait state" on page 297 for further information.

Note J

Loops can produce various effects, such as:

- Reduced (or no) activity on terminals.
- Other address spaces do not continue, or at least only slowly.
- SYS-lamp on the console is permanently on.
- No messages on system console.
- The same message always appears.
- Data files are filled with identical data.

Note K

LIFO stack entries can be used to discover which LIFO-using modules have been invoked, and in what sequence. (Many CICS management modules use LIFO stack entries in which to save registers and to obtain work space; for further information, see "LIFO storage" on page 269; the main LIFO fields are illustrated in Figure 66 on page 270 and Figure 67 on page 272.)

To determine a module invocation history, proceed thus:

1. Find the current (latest) LIFO stack entry using the pointer in TCALCDSA.
2. If this entry contains X'42' in byte 0, this is DFHKCP's initial stack entry and there are no CICS modules active.
3. If the entry contains other than X'42' in byte 0, determine the owner of the entry by referring to offset X'50' of the entry (which contains the identity of the owning module).

   The register save area in general contains the register values corresponding to the last invocation of another CICS module. (Offset X'0C' in this stack entry corresponds to register 14, offset X'10' to register 15, offset X'14' to register 0, and so on.)
4. To determine the invoker of the current module, use the chain-back field at offset X'4'. The register save area in this stack gives the register values as they were on entry to the current module.
5. Continue the process of chaining back through each LIFO stack element until DFHKCP's initial LIFO stack entry is found.

Chapter 1.3. Approach to wait problems

This chapter suggests an approach to dealing with wait problems. (However, see "Chapter 3.5. CICS in a wait state" on page 297 for details of the task dispatcher mechanism and the various sources of waits and suspends.)

This chapter refers to various fields in CICS data areas. The name of the data area is usually the first 3 or 4 characters of the name of the field; for example, DCATCDC is in the dispatch control area (DCA).

There are two different reasons for the CICS address space to be in a wait state:

1. No CICS tasks are dispatchable so DFHKCP has issued an operating system wait for an ICV interval (operand on the DFHSIT macro).
2. A wait has been issued somewhere else in CICS or an SVC has been issued.

In case 1, the wait is issued at label KCPWAIT in DFHKCP and it is necessary to investigate each task to determine what it is waiting for, or why it is suspended. There may also be some reason why new tasks are not being attached. This could be either: (1) the short-on-storage indicator is on (CSA + X'48' = B'00000001'); (2) the maximum number of tasks has been reached (see operands MXT, AMXT and CMXT on the DFHSIT macro; or (3) terminal input is not being accepted or processed.

In case 2, it is necessary to find out why the operating system is not posting the ECB or giving control back after the SVC. In such a case, the CICS dispatcher is not getting control and no tasks will run even if their ECBs are posted. The only operating system wait issued by CICS during normal running is the one in DFHKCP described above; but during initialization and termination of CICS, operating system waits are often issued directly. An operating system wait is issued by the journaling system subtask, but this does not directly hold up the main CICS task. Operating system requests made during the normal running of CICS should be asynchronous and should not normally involve operating system waits and/or take a long time to service. Exceptions are the OPEN SVC, BDAM READ EXCLUSIVE (in DFHFCP), and the wait in DFHKCP.

CICS task switches cannot take place as a result of a page exception, so page I/O therefore holds up the whole of CICS.

Indications of types of transaction waits

The first indication of the reason for a wait or suspension is given by a dispatch control indicator (DCI), which is a special byte that is set up in the fields DCATCDC and TCATCDC whenever a task issues DFHKC TYPE=WAIT or TYPE=SUSPEND.

The possible values (and meanings) for the dispatch control indicator for tasks on the active and suspended chains are shown in Figure 14 on page 35. The descriptions in this figure are intended as a first-level overview of the reason for the wait or suspend. Most waits and suspends issued by CICS are described in "Chapter 3.5. CICS in a wait state" on page 297 (which also describes how and when the wait is satisfied or when a resume is issued by an active task).

If the CICS address space has control of the processor, all the CICS tasks except the active one will be in one of the wait or suspend states shown in Figure 14 on page 35. The address of the active task is in CSACDTA. If CICS has relinquished control of the processing unit to another system task as a result of an operating system wait, then (1) all CICS tasks will be in one of the above states; (2) register 12 in the TCB will normally carry the address of the task control program's TCA (a dummy TCA, which never appears on the active or suspend chain, but which is used by the task dispatcher mechanism in DFHKCP); and (3) CSACDTA, being the **last** dispatched task, will be largely irrelevant.

TCATCDC or DCATCDC value	Meaning
X'10'	If a task is on the active chain, it issued a DFHKC TYPE=WAIT,DCI=NON macro. If a task is on the suspend chain, it issued a DFHKC TYPE=SUSPEND macro without indicating (using the DCI operand) the reason for the suspend.
X'11'	The task issued DFHKC TYPE=ATTACH macro. but there was not enough storage for the new task. The request is retried each time the dispatcher selects this task to run.
X'13'	The task issued a DFHKC TYPE=WAIT,DCI=TERMINAL macro, that is, it is waiting for terminal I/O to complete. These tasks stay on the suspend chain until resumed by terminal control on completion of the I/O.
X'14'	The task has been attached or resumed, but is nondispatchable for maximum active task reasons (see the AMXT operand in the SIT). This value is set at task attach time, if the task has no TCLASS. It is also set when a task is resumed.
X'15'	The task has been attached, but not dispatched yet. It belongs to a class (see CMXT in the SIT and TCLASS in the CEDA DEFINE TRANSACTION command) and will not be dispatched until the number of tasks in the class that are still in the system, and have been dispatched at least once, falls below the class limit.
X'16'	The task issued an EXEC CICS RETRIEVE WAIT command.
X'17'	An anticipatory paging task is waiting for a page control area (PCA) to become free.
X'18'	The task was suspended by storage control (DFHSCP) because there was not enough storage to satisfy its GETMAIN request.
X'19'	The task issued an EXEC CICS DELAY command (DFHIC TYPE=WAIT).
X'1A'	A mirror task has suspended itself to wait for work. It will be automatically purged after two seconds if no work arrives.
X'1C'	The task was suspended due to lack of temporary storage.
X'1D'	As X'15', but tasks in the X'1D' state do not count towards the maximum task count (see MXT in the SIT).
X'20'	The task is dispatchable. It may have issued an EXEC CICS SUSPEND command (DFHKC TYPE=WAIT,DCI=DISP).
X'21'	The task is dispatchable and is to be abended. Field TCAPCAC in the TCA contains the abend code.
X'40'	The task issued DFHKC TYPE=WAIT,DCI=LIST macro.
X'41'	The anticipatory paging task is waiting for page I/O to complete before it can be given control.
X'42'	The task is in SRB mode.
X'43'	The task is waiting on short term I/O, for example, on DASD I/O. The task counts toward the IOCP count (see IOCP in the SIT).
X'44'	The terminal control task is waiting.
X'80'	The task issued a DFHKC TYPE=WAIT,DCI=SINGLE macro, or a DFHKC TYPE=WAIT,DCI=ECB.
X'88'	The task issued a DFHKC TYPE=WAIT,DCI=CICS macro to wait on an internal event, that is, one that will be posted by another CICS task. These ECBs are not included in the MVS wait list.

Figure 14. Meanings of dispatch control indicator (DCI) byte in TCATCDC and DCATCDC

How to find the reason for each wait or suspend

Make a list of the TCAs and DCAs in the system. See Figure 24 on page 144.

For each task, note the following:

1. The dispatch control indicator at DCATCDC and at TCATCDC. For the meaning of this byte see Figure 14 on page 35.

   If a task is on the active chain and its dispatch control indicator is X'10', X'11', X'12', X'14', X'15', X'1D', or X'88', or is on the suspended chain, then it is waiting for action by another CICS task. It may be that a number of tasks are all waiting for the same action – such as the DEQ of a resource. In this case there will be a key task (possibly the owner of the resource) which is waiting, probably for a completely different and unrelated reason, and only this key task need be investigated in detail.

   For each waiting task, use Figure 14 on page 35 to determine the meaning of the dispatch control indicator. If the DCI is X'80' or X'88', then TCATCEA addresses the ECB that the task is waiting to be posted. Determine where this is (in what sort of control block, or in which program) using the control block and program indexes of a formatted dump. Check that this is compatible with the location where the request was issued as determined in 2, and use the descriptions in "Chapter 3.5. CICS in a wait state" on page 297 to find out the purpose of the ECB, and when, and by whom, it should be posted. If the DCI is X'40', TCATCEA addresses a list of addresses of ECBs, and the task would be dispatched if any one of them were posted.

   For each task on the suspended chain, use the descriptions in "Suspended tasks" on page 312 to determine the reason for suspension, and the task which should issue DFHKC TYPE=RESUME.

2. The register 14 value saved by DFHKCP at TCATCRS (User TCA+X'20'). This is the address of the next instruction of the task when it is next given control.

   Provided that it was not issued by a COBOL or PL/I application program, this is the address of the code which issued the DFHKC TYPE=WAIT or SUSPEND. This address may also be found from the trace table or from the auxiliary trace. Determine in which program this is by using the program index in the formatted dump or by scanning an address-space dump.

   If the request was issued by a **command-level** program, the field TCAPCHS in the system TCA addresses the application save area (register 13) at the time the last EXEC command was issued. The first 18 words of the addressed area constitute a standard save area in which register 14 (at offset X'C') addresses the point in the program where the request was issued.

3. The service module indicator at TCASVMID. These two bytes are used by runaway task control. Their meanings are as follows:

```
X'0001' in journal control program
X'0002' in BMS
X'0004' in DL/I interface program
X'0008' in terminal control program
X'0010' in data interchange program
X'0020' in COBOL exec program
X'0040' in deferred DL/I abend
X'0080' in system task
X'0100' in runaway task, don't flush
X'0200' in system task, don't flush
X'0400' in storage control program
X'0800' in trace control program or monitoring control program
X'1000' in program control program
X'2000' in dump control program
X'3000' in file control program
X'4000' in transient data program
X'5000' in temporary storage program
X'6000' in interval control program
X'8000' in task control program
```

4. The task identifier at TCAKCTTA in the system area of the TCA. This may be required for matching trace entries.

5. The transaction identifier. This is found in the first four bytes of the program control table entry which is addressed by TCATCPC in the system area of the TCA.

6. The facility address at TCAFCAAA (User TCA + X'8'). The first byte of this field indicates the type of facility:

X'00'	Undefined	TCAFCAAA may point to any resource or no resource.
X'01'	Terminal	TCAFCAAA addresses the TCTTE.
X'02'	File control	TCAFCAAA addresses the FCT.
X'04'	Interval control	TCAFCAAA addresses the ICE. The CSPG task used by BMS to purge temporary storage periodically is an example of this type of task.
X'08'	Queue control	TCAFCAAA addresses the DCT. These tasks are normally initiated by DFHTDP when the associated queue reaches the specified length.
X'10'	Automatic initiation	TCAFCAAA addresses the AID.

7. The lock address, TCALCKAD. If nonzero, the task is waiting for or owns the resource represented by the ECB whose address is in this slot.

Chapter 1.4. Special considerations for performance problems

Performance problems in CICS installations (for example, one or more transactions are unexpectedly slow) can usually be attributed to one or more of the following causes:

1. Poor application design and implementation from a performance point of view.
2. Incorrect use of CICS and its associated components from a performance point of view.
3. System overload or allocation of insufficient resources.
4. An error in an IBM program.

In general terms, items 1 to 3 are the user's responsibility and item 4 is IBM's responsibility. However, to diagnose a problem, it is usually necessary to make some measurements to isolate the problem area. It is often found that a performance problem is caused by a collection of minor problems. Discovery and correction of a single fault is usually not sufficient to correct the problem.

Measurement and evaluation

Techniques available for evaluating the performance of a CICS system, and for isolating the general category of a problem are discussed in the *Performance Guide.*

CICS monitoring facility

The CICS monitoring facility is a powerful tool for tuning and debugging the CICS system, and application programs. Detailed information is collected at specified frequencies and can be analyzed later. Measurements include usage statistics, accounting data, timing data, and problem determination data. For a description of how to interpret the CICS monitoring facility, see the *Performance Guide*, and how to activate it, see the *CICS-Supplied Transactions* manual, and the *Resource Definition (Online)* manual or the *Resource Definition (Macro)* manual.

Poor application design and implementation

Some general guidelines on performance considerations to be taken into account when designing an application are given in the *Performance Guide*. In some cases it may be found that performance considerations lead to guidelines which are in opposition to guidelines generated by other considerations, for example, programmer productivity.

Some examples of faults caused by poor design and implementation are:

- Heavy use of the more costly CICS functions where alternative techniques using less expensive functions are adequate. The data presented in the *Performance Guide* should assist in making these trade-off decisions. Examples would be the relative merits of (1) auxiliary and main temporary storage; and (2) the use of transient data queues with a trigger level of one against a high trigger level or a time-initiated task retrieving data from main temporary storage.

- Bottlenecks caused by application design or coding – for example, the failure to release a heavily used resource as soon as the application has finished using that resource.

- Interleaving of heavily used and lightly used code. This practice can lead to unnecessarily large working sets that could cause a high paging rate.

Incorrect use of CICS and associated licensed programs

During the generation and initialization of CICS and associated licensed programs it is necessary to specify a number of parameters that can have a significant impact on the performance of the CICS system. A description of the major parameters known to have an impact is given in the *Performance Guide*. A general evaluation of the CICS tables and generation parameters is probably necessary apart from any specific pointers to trouble areas obtained during the evaluation process.

System overloading or insufficient allocation of resources

Some general discussion on the subject of loading and performance characteristics is given in the *Performance Guide*. Resources can be categorized into two types – those that are under the control of the user (for example, virtual storage buffer sizes) and those that are bound by the limitations of the hardware (for example, processing unit speed, real storage, and number of communication lines). In general, problems due to insufficient allocation of the first type of resource can easily be diagnosed and corrected. The more common problems are often diagnosed in CICS statistics output, for example, short on (virtual) storage, insufficient VTAM receive any RPLs, or too small a VSAM STRNO. The other type, where the resource is basically fixed, is more difficult to solve.

It is necessary to estimate the required amounts of the resource and see whether this estimate agrees with the actual values. If not, then it is essential to understand where the discrepancy lies. In some cases this may be due to a misunderstanding of how the application works, or possibly a coding error. When agreement is obtained one can estimate how much extra resource is needed or how much saving in the use of the resource is required. The data given in the *Performance Guide*, together with the *Resource Definition (Macro)* manual may be of use in determining whether the problem can be solved by tuning techniques.

The basic cause of this kind of problem is usually that either too little attention is paid at the design stage to matching the system's capacity to the application requirements, or that applications have been added or the system loading increased without due consideration of the increase in resources required to accommodate the extra load.

An error in an IBM program

If the problem cannot be identified as being due to one or more of the above causes, then it might be due to an error in an IBM program.

Appendix A, "Documentation required when submitting an APAR" on page 365 indicates the type of information required for an APAR. For a performance problem, it is probable that measurement data and a general description of the main application program structure will be required.

Part 2. Aids to problem determination

Chapter 2.1. Trace

The CICS trace facility is a debugging aid for application programmers and IBM field engineers. It maintains in main storage a **trace table** consisting of standard CICS entries and entries defined by the user or licensed programs. The table is filled in a wraparound manner; when the end is reached, subsequent entries overwrite the entries at the start of the table. When you print the trace entries, the print transaction is given higher priority, so that the dump is printed before wraparound and overwriting occur.

Tracing can be activated and deactivated by an EXEC CICS TRACE command or a DFHTR macro instruction in an application program, by the master terminal transaction CEMT, or, at initialization, by the SIT parameter TRACE.

If you want to trace CICS operation when using the execution diagnostic facility (EDF), add the option TRACE = YES to the CEDF transaction in the program control table (PCT). You can see the trace table in a dump. If you use the MVS/XA SDUMP facility, the dump goes to the SYS1.DUMP data set.

The trace table does not have to be written to the CICS dump data set when an SDUMP is taken. XRF provides a PRINTDUMP exit that formats the trace table, regardless of the options you specify.

The XRF trace, described in "Chapter 2.2. XRF trace" on page 127, is independent of the normal trace facility. It is used by the CICS availability manager (CAVM), and always takes place. It makes about 12 entries every 2 seconds, and wraps around.

As well as being recorded in the trace table, trace entries can be stored in a sequential data set by the CICS **auxiliary trace facility**. The auxiliary trace facility is activated either by the master terminal transaction CEMT, or, at system initialization, by the SIT parameter AUXTRACE. It is deactivated by the CEMT transaction. Auxiliary trace entries are recorded only when main storage trace is also active.

The auxiliary trace data set does not wrap around; all entries are preserved so that a complete history is obtained. The CICS trace utility program (DFHTUP) can be used to print the contents of the auxiliary trace data set, or selected entries from it.

In addition to the standard CICS system entries, entries produced by the terminal control program (DFHTCP) for non-VTAM terminals can be recorded in the trace table. These entries, called field engineering (FE) entries, are normally inhibited but can be activated by the DFHTR macro instruction or by the CSFE transaction.

A third class of entry is the user entry, which can be defined by the application programmer with the DFHTR macro instruction or the EXEC CICS ENTER command.

Remote server trace

When you invoke the remote server by typing CEHS on the screen, you can add the option TRACE to the command. This causes the remote server to write entries to a trace log, which is kept in temporary storage. Main storage is used for this queue, and not auxiliary storage, so you should use this facility with care, as it can take up large amounts of storage.

The trace records are written to a queue whose name is created by concatenating "CEHS" with the 4-character terminal identifier. You can use the CEBR transaction to browse this queue. To find out more about the CEHS trace, see the *CICS/VS Remote Server Diagnosis* manual.

Trace table

The CICS trace table, which is built during system initialization, consists of a trace header and a number of fixed-length entries that can be used to trace the flow of transactions through the system. The number of trace table entries is specified by the TRACE parameter in the SIT or as a startup override.

The trace table can be initially disabled by specifying OFF in the TRACE parameter in the SIT. The master terminal can be used to turn trace, or auxiliary trace, on or off during CICS execution.

Each entry in the trace table is 32 bytes long and is aligned on a 32-byte boundary. The table is used cyclically, so that when the last entry is used, the next entry is placed at the beginning of the table.

The address of the trace header is held in CSATRTBA in the CSA. The trace header, which is separate from the trace table, contains pointers to the trace table as follows:

Field	Contents
Bytes 0-3	Address of last-used entry
Bytes 4-7	Address of first entry in table
Bytes 8-11	Address of last entry in table

For a complete description of the trace table header format, see the DSECT ZTRHEADR in the *Data Areas* manual.

When the trace table is printed in a transaction dump, it should be noted that the last-used entry may be some way after the trace entry for the dump invocation because other transactions can run while the dump is being output to the dump data set. In a CICS formatted dump, by contrast, the trace table is printed showing the last-used entry as the last entry before the formatted dump was invoked, since no transactions can run during a formatted dump.

The format of an entry in the trace table, together with the names of fields as printed in a dump and in auxiliary trace output, is shown in Figure 15.

Field	**Contents**
ID Byte 0	Trace identification of entry
REQD Byte 1 and byte 2, bits 0-3	For system and FE entries, byte 1 and byte 2 (bits 0-3) usually contain the type of request code relating to the CICS management, or the service program involved. For user entries, these bits are unused.

Figure 15 (Part 1 of 2). Format of trace table entries

Field	Contents
REQD Byte 2 bits 4-7	These bits indicate the type of entry as follows: X'0' Reserved X'1' FE entry X'2' User entry X'3' LIFO system entry X'4' System entry X'5' LIFO response/return X'6' Reserved X'7' Reserved X'8' Reserved X'9' Reserved X'A' Reserved X'B' Reserved X'C' Reserved X'D' On/off entry X'E' Reserved X'F' Reserved
Bytes 3-4	Reserved
TASK Bytes 5-7	Task identification as found in TCAKCTTA (a 3-byte field in the system part of the TCA).
FIELD A Bytes 8-11	Data relevant to this entry
FIELD B Bytes 12-15	Data relevant to this entry
RESOURCE Bytes 16-23	When provided, this is usually the name of a resource associated with the request being traced. For example, for program control requests, it is the program name. See Figure 18 on page 111. For user entries, the resource name can be set by the EXEC CICS ENTER command (but not by the DFHTR macro instruction).
REG 14 Bytes 24-27	If byte 0 contains other than one of the values from X'F0' through X'FC', these bytes contain the contents of register 14 at entry to the trace control program. If byte 0 contains a value from X'F0' through X'FC', these bytes contain the contents of register 14 at entry to the CICS management or service program involved. For an EXEC CICS command, register 14 contains the address of the point in the application program immediately after the last call to CICS. (For a macro request, from a COBOL or PL/I program, register 14 contains an address in DFHPCP.)
TIME OF DAY Bytes 28-31	Timestamp, which is the time of day in units of 32 micro-seconds since midnight.

Figure 15 (Part 2 of 2). Format of trace table entries

The contents of byte 0 and the two data fields (bytes 8 through 15) of user trace entries are determined by the EXEC CICS ENTER command, or the DFHTR TYPE=ENTRY macro instruction.

For system trace entries only, if consecutive, identical entries are generated, the first entry only is entered into the table. In these cases, a special trace control entry with trace identification X'FD' (with X'01' in byte 1) is created, and a count of the number of times the previous entry is repeated is stored therein.

Trace control entries with trace identification X'FE' or X'FF' indicate the turning on or turning off of the trace facility, respectively.

For formatted dumps and transaction dumps, the trace table entry is printed in a different format, as described in "Chapter 2.3. Formatted dump" on page 131.

The same display format is used when auxiliary trace output is printed. See "Chapter 3.1. Normal operation" on page 235 for an annotated illustration of the auxiliary trace output obtained from running a sample transaction.

Trace identification

The application programmer can make entries in the trace table by using the DFHTR macro instruction or the EXEC CICS ENTER command. A trace identification number from 0 through 199 (X'00' through X'C7') and accompanying data may be assigned for each trace entry. Thus, by defining several unique trace entries, the programmer can trace the logical path through a particular application or group of application programs.

Each CICS entry contains a trace identification number in the range 192 to 255 (X'C0' through X'FF') that is unique to the functional area concerned, together with information to aid the application programmer in determining where the macro instruction was issued and what type of request was made. When you look at a trace entry, you should first look at byte 1, bits 4-7, to determine whether it is a system, user, or FE entry. See Figure 15 on page 47 for details of byte 1. The name of the CICS program corresponding to each trace identification is shown in Figure 16 on page 51.

The contents of bytes 0 to 15 of the CICS trace table entries are shown in Figure 17 on page 53. The contents of any fields characterized as "Reserved" in the descriptions should be ignored during the analysis of a trace table entry. Figure 18 on page 111 gives the contents of bytes 16-23 where applicable. Figure 19 on page 112 gives the formats of DFHZCP trace entry fields A1-11, B1, and B2. Figure 20 on page 117 gives the EXEC interface command and response codes.

Trace entries are also made in the CICS trace table by DL/I.

Auxiliary trace

You can use one or two auxiliary trace data sets. If two data sets are defined, you can alternate between them during CICS execution. Each trace entry, in the format described above, is placed in an output buffer. When the buffer is full, it is written to an auxiliary trace data set before being reused. In this way, no trace entries are lost as is the case when the main storage trace table wraps around. This is one reason why the auxiliary trace is often more useful than the trace table.

When an auxiliary trace data set is full, the operator is informed by means of the message DFH1401 and the data set is closed.

The contents of an auxiliary trace data set can be printed using the trace utility program, DFHTUP. This program formats each trace entry to show the CICS component which made the entry and the function being performed. The 4-byte time stamp is also printed in hours, minutes, seconds, and microseconds, and the time difference between consecutive entries is printed in seconds and microseconds. If the time difference between consecutive entries exceeds 12800 microseconds, the entry is flagged with an asterisk. This is particularly useful when dealing with performance problems.

When using the trace utility program, not all entries need be printed. The entries to be printed may be selected by task identifier, transaction identifier, terminal identifier, time or trace identifier.

Full details of auxiliary trace and the use of the trace utility program can be found in the *Operations Guide* and the *Facilities and Planning Guide.*

Program	Trace Identification
CAVM services	X'C4',X'C5',X'C6', X'C7'
Storage control	X'C8',X'C9',X'CA'
Dynamic transaction backout	X'CB'
Catalog control	X'CC'
Basic mapping support	X'CD',X'CF'
Recovery control	X'CE'
Task control	X'D0'
Volume management	X'D2'
LU6.2 services manager	X'D3'
DMS interface	X'D4'
User exit interface	X'D5'
Allocation program	X'D6'
Data interchange	X'D7'
Sync point	X'D8'
Function request shipping data transformation	X'D9'
Statistics	X'DA'
Transaction routing data transformation	X'DB'
Abnormal condition program	X'DC'
Interregion communication	X'DD'
Subtask management program	X'DE'
Intersystem program	X'DF'
Message processor	X'E0'
EXEC interface	X'E1'
Sign-on	X'E2'
System spooling interface	X'E3'
Master terminal	X'E4'
Security program	X'E5'
Terminal control (DFHTCP, DFHTACP)	X'E6'
Task-related user exit interface program	X'E7'
Terminal sharing	X'E8'
Table management program	X'EA'
Allocation management program	X'EB'
Definition management program	X'EC'
LIFO overflow program	X'ED'
VTAM I/O trace	X'EE'
Terminal object resolution	X'EF'
Task control	X'F0'
Storage control	X'F1'
Program control	X'F2'
Interval control	X'F3'
Dump control	X'F4'
File control	X'F5'
Transient data control	X'F6'
Temporary storage control	X'F7'
CICS-DL/I interface	X'F8'
Journal control	X'F9'
Basic mapping support	X'FA'
Built-in functions	X'FB'
Terminal control (DFHZCP)	X'FC'
Trace control	X'FD',X'FE',X'FF'

Figure 16. Trace identification

Controlling the trace

The EXEC CICS TRACE command or the DFHTR macro

The EXEC CICS TRACE command or the trace control macro instruction DFHTR can be used to activate and deactivate tracing. Tracing of the standard CICS system entries, FE entries, and user entries are separately controllable. For the standard CICS system entries the master trace flag can be turned on or off, or the flags for individual system components can be turned on or off. A standard trace entry is added to the table only if both the master flag and the appropriate system component flag are on. FE tracing can be turned on or off by the DFHTR macro instruction, but not by the EXEC CICS TRACE command. For user entries, tracing can be turned on or off for all tasks, or for the task issuing the request only.

Full details can be found in the *Application Programmer's Reference*, and *CICS/VS Application Programmer's Reference Manual (Macro Level)*.

The CEMT transaction

The trace and auxiliary trace facilities can be turned on and off, and the auxiliary trace data set can be opened, closed, or switched, by means of a CEMT command issued by the master terminal operator.

Details of the CEMT command can be found in the *CICS-Supplied Transactions* manual.

The CSFE transaction

System, user, and FE traces can be turned on or off by the CSFE transaction. CSFE also controls two special-purpose trace facilities that can result in the creation of additional trace entries of system type. These are the system spooling interface trace and the activate scan trace, which are controlled by the SPOOLFE and ZCPTRACE operands of CSFE, respectively.

The CSFE transaction can be used to turn on or off the debugging (DEBUG) functions provided by the storage violation trap and the global trap/trace exit, both of which are entered from the trace program. Details of the storage violation trap are given in "Chapter 3.2. Storage management" on page 259. The global trap/trace exit is described in this chapter.

Full details of the CSFE command can be found in the *CICS-Supplied Transactions* manual.

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'C4' DFHWMS	**Bytes 1-2** X'00037 entry	Message manager request ID (WMSREQID)	Data area address (WMSDATAD)
	X'0005' return	Message manager return code (WMSRC)	**Bytes 12-13** Data area size (WMSDATSZ) **Bytes 14-15** Data length (WMSDATLN)
X'C5' DFHXRCP	**Byte 1** X'00' Entry **Byte 2** X'04'	Reserved	Reserved
	X'01' Exit	Reserved	Reserved
	X'02' Resumption of DFHXRCP	Reserved	Reserved
	X'03' Suspension of DFHXRCP	Reserved	Reserved
	X'04' CIB prefix	**Byte 8** CIB-VERB **Byte 9** CIB-LEN **Bytes 10-11** CIB-ASID or TJID	**Byte 12** CIB-CONID **Byte 13** Reserved **Bytes 14-15** CIBDATLN
	X'05' CIB data for modify command	CIB data bytes 0-3	CIB data bytes 4-7
	X'06' CIB data for modify command - continuation 1	CIB data bytes 8-11	CIB data bytes 12-15
	X'07' CIB data for modify command - continuation 2	CIB data bytes 16-19	CIB data bytes 20-23

Figure 17 (Part 1 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'C6' DFHXRSP	**Byte 1** X'00' Entry **Byte 2** X'04'	Reserved	Reserved
	X'01' Exit	Reserved	Reserved
	X'02' Resumption of DFHXRSP	Reserved	Reserved
	X'03' Suspension of DFHXRSP	Reserved	Reserved
	X'04' Work element	**Byte 8** XRWERQ X'01' - Sign on X'02' - Sign off normal X'03' - Sign off abnormal X'07' - Clock difference changed X'08' - Status data changed X'09' - Surveillance signal overdue X'0A' - Surveillance signal resumed X'0F' - Takeover requested X'10' - Post IA (incipient active) ECB X'11' - Post TC (takeover complete) ECB X'12' - Post SS (synchronized with respect to signoff) ECB X'13' - Post ST (synchronized with respect to termination) ECB X'18' - CAVM failed X'19' - Invalidated **Byte 9** XRWERQM Bit 0 - Implicit request (SIGNON/SIGNOFF NORM) Bit 1 - DUMP=YES (takeover requested) **Bytes 10-11** Reserved	A(work element)
	X'05' Work element - continuation 1	**Bytes 8-11** XRWEINS instance number	XRWEVER version number
	X'06' Work element - continuation 2 For signon: For clock difference changed: For surveillance signal overdue: For CAVM failed: Any other event:	APPLID - first four characters Lower bound for difference in seconds Number of seconds overdue Abend code Reserved	APPLID - second four characters Upper bound for difference in seconds
	X'07' Abend imminent	Abend code	Reserved

Figure 17 (Part 2 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'C7' DFHXRA	**Byte 2, bits 4-7** X'3' Entry	**Byte 8** XRRQTR X'01' DFHXR CTYPE=INITIALIZE X'02' DFHXR CTYPE=TERMINATE X'03' DFHXR CTYPE=SIGNON X'04' DFHXR CTYPE=SIGNOFF X'05' DFHXR CTYPE=QUERY-TAKEOVER-INIT X'06' DFHXR CTYPE=SET-TAKEOVER-INIT X'07' DFHXR CTYPE=WAIT-TAKEOVER-INIT X'08' DFHXR CTYPE=QUERY-TAKEOVER=COMP X'09' DFHXR CTYPE=SET-TAKEOVER-COMP X'0A' DFHXR CTYPE=WAIT-TAKEOVER-COMP X'0B' DFHXR CTYPE=QUERY-RSD-AVAIL X'0C' DFHXR CTYPE=SET-RSD-AVAIL X'0D' DFHXR CTYPE=WAIT-RSD-AVAIL X'0E' DFHXR CTYPE=QUERY-SYNC-SIGNOFF X'0F' DFHXR CTYPE=SET-SYNC-SIGNOFF X'10' DFHXR CTYPE=WAIT-SYNC-SIGNOFF X'11' DFHXR CTYPE=QUERY-SYNC-TERM X'12' DFHXR CTYPE=SET-SYNC-TERM X'13' DFHXR CTYPE=WAIT-SYNC-TERM X'18' DFHXR CTYPE=INIT-SURVEILLANCE X'19' DFHXR CTYPE=TERM-SURVEILLANCE X'1A' DFHXR CTYPE=INIT-CONSOLE-COMM X'1B' DFHXR CTYPE=TERM-CONSOLE-COMM X'1C' DFHXR CTYPE=SET-HEALTH-DATA	
	X'5' Exit	**Byte 8** See entry trace XRRQTR immediately above **Byte 11** XRRQRC For DFHXR CTYPE=xxx X'00' Normal response X'08' Invalid request For DFHXR CTYPE=SIGNON X'14' Retry possible X'1C' Rejected by CAVM For DFHXR CTYPE=SIGNOFF X'24' Dump requested X'2C' Rejected by CAVM For DFHXR CTYPE=SET-TAKEOVER-INIT X'34' Takeover in progress X'35' Shutdown in progress X'3C' Rejected by CAVM For DFHXR CTYPE=WAIT-RSD-AVAIL X'44' Restart data set not available	

Figure 17 (Part 3 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'C8'	DFHSCP GETMAIN exit trace - see under X'F1' DFHSCP		
X'C9'	DFHSCP FREEMAIN exit trace - see under X'F1' DFHSCP		
X'CA'	DFHSCR storage violation trace - see under X'F1' DFHSCP		
X'CB' DFHDBP	Reserved	Address of dynamic log	Address of first DWE, TCADWLBA
X'CC' DFHCCP	**Byte 2, bits 4-7** X'3' Entry	**Byte 8** Request type X'01' OPEN X'02' CLOSE X'03' CONNECT X'04' DISCONNECT X'05' STARTBROWSE X'06' ENDBROWSE X'07' GETNEXT X'08' WRITE X'09' READ X'0A' DELETE X'0B' PURGE **Byte 9** Request Modifier 1..... SYNC=YES **Bytes 10-11** Reserved	Addr of parm list Addr of parm list Addr of parm list Addr of parm list Addr of parm list Addr of parm list Addr of parm list Addr of parm list Addr of parm list Addr of parm list
	Byte 2, bits 4-7 X'5' Exit	**Byte 8** Request type Same as on entry trace **Bytes 9-10** Reserved **Byte 11** Response code X'00' Normal response X'04' Not found X'06' Length error X'08' Duplicate found X'0C' Invalid request X'10' Disastrous error	Reserved
X'CD'	BMS temporary storage error - see under X'FA' BMS		

Figure 17 (Part 4 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'CE' DFHRCP	**Byte 2, bits 4-7** X'3' Entry	**Byte 8** Request type X'01' OPEN X'02' CLOSE X'03' CONNECT X'04' DISCONNECT X'05' STARTBROWSE X'06' ENDBROWSE X'07' GETNEXT X'08' WRITE X'09' READ X'0A' DELETE X'0B' PURGE X'0C' LOG X'0D' INITIALIZE X'0E' WAITINIT X'0F' RESTART TASK **Byte 9** Request Modifier ...1.... CREATE=YES ..1..... Resource type specified .1..... Record number specified 1..... Backward recovery **Bytes 10-11** Reserved	Addr of parm list Addr of parm list Addr of parm list Addr of parm list Addr of parm list Addr of parm list Addr of parm list Addr of parm list Addr of parm list Addr of parm list Addr of parm list Addr of parm list Addr of parm list Reserved
	Byte 2, bits 4-7 X'5' Exit	**Byte 8** Request type Same as on entry trace **Bytes 9-10** Reserved **Byte 11** Response code X'00' Normal response X'04' Not found X'06' Length error X'08' Duplicate found X'0C' Invalid request X'0E' Warning issued X'10' Disastrous error	Reserved
X'CF' DFHMCP (See note 1 on page 110)	Reserved	Same as for X'FA' with byte 2, bits 4-7=X'5'	

Figure 17 (Part 5 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)

X'D0' DFHKCP auxiliary trace - see under X'F0' DFHKCP

X'D2' DFHVCP

Byte 2, bits 4-7
X'3' Entry
Byte 1

		FIELD A	FIELD B
...0 0000	Locate vol descriptor	Addr of volume ID	Buffer address
...0 0001	Create SDT		
...0 0010	Add series to SDT		
...0 0011	Locate series name		
...0 0100	Read & merge keypoint recs		
...0 0101	Build keypoint rec		
...0 0110	Prompt operator for scratch tape		
...0 0111	Examine & merge label record		
...0 1000	Build label record		
...0 1001	Tally contents of series		
...0 1010	Warn operator of deficient series		
...0 1011	Copy vol descriptor		
...0 1100	Delete vol descriptor		
...0 1101	Add vol to end of series		
...0 1110	Add vol to series in LIFO order		
...0 1111	Accept vol ID from restart resolution logic		
...1 0001	Mark as failed on closing		
...1 0010	Mark as failed on opening		
...1 0011	Mark as failed on input		
...1 0100	Mark as failed on output		
...1 0101	Make read only		
...1 0111	Mark vol clean & available		
...1 1011	Mark as occupied		
...1 1100	Mark open with occupancy unchanged		
...1 1101	Mark closed		
...1 1110	Mark pending (about to open)		
...1 1111	Mark open and occupied		

Figure 17 (Part 6 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'D2' DFHVCP	**Byte 1** (continued) 0.10 0000 Update op sys control block ..10 0001 Merge op sys control block to resolve SCRTCH vol ..10 0010 Build data in journal control note block ..10 0011 Merge NOTE info into a volume descriptor ..10 0100 Build a CICS catalog record 0.11 .... Update op sys control block 01.. .... Make vol current in series **Byte 2, bits 0-3** Type of locate X'0' Given vol ID X'1' Next vol or confirm series name X'2' First vol or next series name X'3' Previous vol or greater-or-equal series name X'4' Subject of NOTE X'6' Predicted vol for next output X'7' Vol available for output X'8' Current vol X'9' Next-after-current vol X'A' Next-before-current vol X'C' Vol referenced by op sys **Byte 2, bits 4-7** X'5' Exit **Byte 1** X'00' NORESP X'20' DEFCSER X'21' INSUFF X'22' DECLINE X'40' CONFLT X'41' NOSERS X'43' NODESC X'45' UNAVAIL X'46' BADDATA X'80' INVREQ	First four characters of name of volume last handled, if any	Next two characters of name of volume last handled, if any

Figure 17 (Part 7 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'D3' DFHLUP	Reserved	Reserved	Reserved
X'D4'	DMS trace entry, see the *Development Management System/CICS/VS: Problem Determination Manual*.		
X'D5' DFHUEH	**Byte 1** X'01' Before exit X'02' After exit	- - - - Exit identifier - - - - Return code (R15)	Address of current EPB
X'D6' DFHALP	**Byte 2, bits 4-7** X'3' Entry **Byte 1**		
	X'01' Unchain		
	X'02' BMS TPQ call 2	MCRID	
	X'05' BMS TPR call 2	Address of AID	
	X'06' CRS call 1	Reserved	System ID
	X'07' CRS call 2	Tranid	Termid
	X'08' ICP call 1	Reserved	Reserved
	X'09' ICP call 2	Address of AID	Reserved
	X'0A' BMS TPQ call 1	Normal threshold	Adjusted threshold
	X'0C' BMS TPQ call 3	Msg ID	Reserved
	X'0D' BMS TPR call 1	(Supplied AID)	Addr of TCTTE
	X'0E' CRQ call 1	Normal threshold	Adjusted threshold
	X'12' Schedule	Tranid	Termid
	X'14' Avail	Termid	Terml address
	X'15' Unavail	Termid	Terml address
	X'16' Release	Termid	Terml address
	X'17' Allocate	Tranid	System address
	X'18' Release abnormal	Termid	Terml address
	Byte 2, bits 4-7 X'5' Exit	Reserved	**Byte 12** X'00' Normal return X'04' Error return **Byte 13** Secondary code X'01' BMS TPQ call 1, no AIDs found X'02' BMS TPQ call 2, no AIDs found X'03' BMS TPQ call 3, no AIDs found X'04' ICP call 1, no AIDs found X'05' ICP call 1, AID found, but does not match X'06' No AID found for generalized locate function X'0C' Error return X'31' Error return

Figure 17 (Part 8 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'D7' DFHDIP	**Byte 2, bits 4-7** X'3' Entry **Byte 1** X'01' ADD X'02' ERASE X'03' REPLACE X'04' ABORT X'05' QUERY X'06' END X'07' RECEIVE X'08' NOTE X'09' DETACH X'0A' ATTACH X'0B' SEND X'0C' WAIT **Byte 2, bits 0-3** X'1' DSN not specified X'2' Profile specified X'4' SELECT X'8' VOLADDR	) **Byte 8**) X'01' WAIT) X'08' KEYNUMB) X'10' RRNADDR) X'20' KEYADDR) X'40' TYPE=SAVE) X'80' DEFRESP=YES)) **Byte 9** Reserved)) **Byte 10**) Value of NUMREC) (valid for ADD,) ERASE, and REPLACE)))))	TIOA address, TCTTEDA
	Byte 2, bits 4-7 X'5' Exit	**Bytes 8-9** Return code, TCADIRC **Bytes 10-11** Deferred return code (set by next call to DFHDIP if nonzero)	**Bytes 12-13** System sense code from DFHZNAC **Bytes 14-15** User sense code from DFHZNAC
X'D8' DFHSPP	**Byte 2, bits 4-7** X'3' Entry	**Byte 8** Type of request (from TCASPTR) X'01' USER X'02' SYSTEM X'09' ROLLBACK X'10' RESYNC request X'20' LUC request (see byte 9) X'60' LUC SEND-PREPARE **Byte 9** X'01' BIS (INBOUND) X'02' UNBIND X'04' RESYNC X'08' Exchange log names received X'10' CONTACT X'20' QPEND X'40' CLPEND **Bytes 10-11** Reserved	Address of first deferred work element (DWE) from TCADWLBA (or zeros if no DWEs)
	Byte 2, bits 4-7 X'5' Exit	**Bytes 8-11** Reserved	
X'D9' DFHXFP DFHXFX	Reserved	**Byte 8** X'00' Transform 1 X'02' Transform 2 X'04' Transform 3 X'06' Transform 4 **Bytes 9-11** Address of PLIST	**Byte 12** Error code X'04' Transform 1 error X'08' Transform 1 error **Byte 13** Error code 2 X'04' Transform 2 error X'08' Transform 2 error **Bytes 14-15** (EIBFN) command functions

Figure 17 (Part 9 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'DA' DFHST** (Statistics)	Reserved	Reserved	Reserved
X'DB' DFHXTP	**Byte 2, bits 4-7** X'3' Entry	**Byte 8** X'00' Transform 1 X'02' Transform 2 X'04' Transform 3 X'06' Transform 4 **Bytes 9-11** Address of DFHXTSTG	**Byte 12** Status code X'10' FLUSH X'20' DETACH X'40' ATTACH X'80' Application reques **Byte 13** Address of TCTTE
	Byte 2, bits 4-7 X'5' Exit	**Byte 8** X'00' Transform 1 X'02' Transform 2 X'04' Transform 3 X'06' Transform 4 **Bytes 9-11** Address of DFHXTSTG	**Byte 12** Response code X'00' Normal response X'08' Error response **Bytes 13-15** Address of TCTTE
X'DC' DFHACP	Reserved	Reserved	Reserved
X'DD' DFHIR	**Byte 1** X'00' RESUME	**Bytes 9-11** Address of TCA for CSNC	Reserved
	X'01' CONNECT	**Bytes 8-11** Return code from connect request	**Bytes 13-15** Address of TCTTE
	X'02' INBOUND X'04' OUTBOUND	**Bytes 8-9** Sequence number **Byte 10** X'01' End chain X'02' Begin chain X'04' Sense included X'10' FMH X'80' Response **Byte 11** X'10' RQE2 X'20' RQD2 X'40' RQE1 X'80' RQD1	**Byte 12** X'20' Change direction X'40' End bracket X'80' Begin bracket **Bytes 13-15** Address of TCTTE
	X'04' OUTBOUND	**Bytes 10-11** X'0000' Relinquish control (no data, no RH setting passed, sequence number unchanged)	**Byte 12** X'00'
	X'03' Inbound data X'05' Outbound data	**Bytes 8-9** Length of data **Bytes 10-11** First two bytes of data	**Bytes 12-15** Next four bytes of data
	X'06' Batch LUWID part 1 X'07' Batch LUWID part 2		

Figure 17 (Part 10 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'DE' DFHSKP	**Byte 2, bits 4-7** X'3' Entry	**Byte 8** Request type X'01' PERFORM X'02' WAIT X'03' RETURN X'04' TERMINATE X'05' DWE PROCESS **Byte 9** Request modifier ....1 AUTH=YES ......1. CLASS=I/O .....1.. SAVAREA ....1... SYNC=YES **Bytes 10-11** Reserved	 Addr of exit routine Reserved Reserved Reserved Reserved
	Byte 2, bits 4-7 X'5' Exit	**Byte 8** Request type X'01' PERFORM X'02' WAIT X'03' RETURN X'04' TERMINATE X'05' DWE PROCESS **Bytes 9-10** Reserved **Byte 11** Response code X'00' Normal response X'04' User code failed X'08' Subtask code failed X'0C' Unable to perform request X'10' Request never completed X'14' Invalid request X'18' Invalid ECB address X'1C' User task was canceled	 Addr of parm list Addr of parm list Addr of parm list Addr of parm list Addr of parm list
X'DF' DFHISP	**Byte 2, bits 4-7** X'3' Entry	**Byte 8** X'03' Shut down X'04' Converse **Bytes 9-11** Reserved	 TCTSE name
	Byte 2, bits 4-7 X'5' Exit	**Byte 8** (ISCRQTR) X'00' Successful X'04' Retry possible X'08' Permanent error X'0C' Terml out of service **Bytes 9-11** Reserved	 Reserved

Figure 17 (Part 11 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'E0' DFHMGP	Reserved	**Byte 8** X'01' **Byte 9** X'01' CSCS (TD queue) X'02' TIOA to be used X'04' No number to be used X'08' Return the message X'10' Output to console X'20' Output to terminal X'40' CSTL (TD queue) X'80' CSMT (TD queue) **Bytes 10-11** Msg no in binary	Addr(TCTTE)
X'E1' DFHEIP	**Byte 1** Reserved **Byte 2, bits 0-3** X'0' Entry	Value of reg 13, i.e: PL/I DSA Assembler's DFHEISTG COBOL TGT	**Bytes 12-13** Reserved **Bytes 14-15** Command code, EIBFN (see note 4 on page 110)
	Byte 1 EIBGDI **Byte 2, bits 0-3** X'F' Exit	**Bytes 8-13** Response code, EIBRCODE (Zeros indicate no exceptional condition)	**Bytes 14-15** Command code, EIBFN (see note 4 on page 110)
	The command codes and response codes are shown in Figure 20 on page 117.		
X'E2' DFHSNP	Reserved	Reserved	Reserved
X'E3' DFHPSP	Note: The RESP and RESP2 codes are given in the *Customization Guide*.		
	Byte 1 X'A0' Entry	Addr of DFHPS parameter list	Reserved
	X'A1' Response	Addr of DFHPS parameter list	Reserved
	X'A2' Unsupported function	Addr of DFHPS parameter list	RESP return code

Figure 17 (Part 12 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'E3' DFHPSP	**Byte 1** (continued) The following trace entries are provided only if the system spooling interface trace facility has been activated (by means of the SPOOLFE operand of the CSFE transaction), or if an abnormal response is given:		
	X'10' DFHEPS SPOOLCLOSE response	RESP return code	RESP2 return code
	X'11' DFHEPS SPOOLCLOSE DELETE	Reserved	Reserved
	X'14' DFHEPS SPOOLCLOSE PRINT	Reserved	Reserved
	X'20' DFHEPS SPOOLREAD response	RESP return code	RESP2 return code
	X'21' DFHEPS SPOOLREAD STANDARD	Reserved	Reserved
	X'22' DFHEPS SPOOLWRITE response	RESP return code	RESP2 return code
	X'28' DFHEPS SPOOLWRITE STANDARD	Reserved	Reserved
	X'40' DFHEPS SPOOLOPEN response	RESP return code	RESP2 return code
	X'41' DFHEPS SPOOLOPEN OUTPUT	Reserved	Reserved
	X'42' DFHEPS SPOOLOPEN INPUT	Reserved	Reserved
	X'88' DFHEPS SPOOLOPEN STANDARD	Reserved	Reserved
	X'B0' DFHPSPCK entry	Reserved	Reserved
	X'B1' DFHPSPCK invalid request	RESP return code	Reserved
	X'B2' DFHPSPCK interface halting	RESP return code	Reserved
	X'B3' DFHPSPCK interface disable	Reserved	Reserved
	X'B4' DFHPSPCK interface terminate	Reserved	Reserved
	X'E0' DFHPSPST entry	Addr of DFHPS parameter list	Reserved
	X'E1' DFHPSPST open input response	Addr of DFHPS parameter list	Spool data block addr
	X'E2' DFHPSPST open output response	Addr of DFHPS parameter list	Spool data block addr
	X'E3' DFHPSPST read response	Addr of DFHPS parameter list	Spool data block addr
	X'E4' DFHPSPST write response	Addr of DFHPS parameter list	Spool data block addr
	X'E5' DFHPSPST close response	Addr of DFHPS parameter list	Reserved
	X'E6' DFHPSPST interface halting	Addr of DFHPS parameter list	RESP return code
	X'E7' DFHPSPST invalid request	Addr of DFHPS parameter list	Reserved
	X'E8' DFHPSPST invalid open request	Addr of DFHPS parameter list	RESP return code
	X'E9' DFHPSPST invalid read request	Addr of DFHPS parameter list	RESP return code
	X'EA' DFHPSPST invalid write request	Addr of DFHPS parameter list	RESP return code
	X'EB' DFHPSPST invalid close request	Addr of DFHPS parameter list	RESP return code

Figure 17 (Part 13 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'E4' DFHMTP	Reserved	Reserved	Reserved
X'E5' DFHXSP	**Byte 2, bits 4-7** X'3' Entry **Byte 1** X'00' Initialization X'04' Sign-on with password X'08' Sign-on without password X'0C' Check authorization X'10' Sign-off X'14' Time-out sign-off X'18' Return USERID X'1C' Wait for initialization X'20' Min time-out from SNT X'24' Build SNTTE block X'28' Free SNTTE block X'2C' Rebuild resource profile X'30' Internal init req	Pointer to parm list	Reserved
	Byte 2, bits 4-7 X'5' Exit **Byte 1** Return code (See the *Customization Guide*)	Reserved	Reserved

Figure 17 (Part 14 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'E6' DFHTCP	**Byte 2, bits 4-7** X'1' FE type trace		
	Byte 1 X'01' (TCATRID1) This trace occurs during line scan.	Addr(TCTLE)	Addr(TCTTE)
	X'02' (TCATRID1)	**Byte 8** TCTLESI - Line status ....1 Out of service .... ..1. Initiated .... .1.. Terml READ initiated .... 1... Switchd line connected ...1 .... Interruptble READ ..1. .... Line perm out of serv .1.. .... Dial line ack 1... .... Error pendng **Byte 9** TCTLEMI - Access method flags ....1 Sequential access .... ..1. Local line .... .1.. Telecomm access .... 1... Error task init (CSTE) ...1 .... First-pool-line flag ..1. .... TCAM access .1.. .... Wrap list flag 1... .... Last-in-pool flag **Bytes 10-11** TCTLEAL - Input area length	TCTLEECB - Event control block **Byte 12** X'00' Initialization image X'40' Event completion X'41' I/O error X'44 I/O req rejected X'48' Enable halted or I/O purged X'7F' Normal completion **Bytes 13-15** Reserved
	This trace occurs during line scan.		

Figure 17 (Part 15 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'E6' DFHTCP	**Byte 1** (continued) X'03' (TCATRID1)	Addr(TCTLE)	**Bytes 12-13** TCTLETOP - Type of operation X'0000' Write break X'0000' Init image X'0001' Read initial X'0002' Write initial X'0003' Read continue X'0004' Write continue X'0005' Read conv X'0006' Write conv X'0006' Write erase alternate X'0008' Write structured field X'0008' Write +ve ack X'0009' Read inquiry moni X'000A' Write -ve ack X'000A' Write reset X'000A' Write disconnect X'000B' Write reset monit X'000B' Read buffer X'000C' Write at line address X'000C' Write initial transparent bloc X'000D' Write initial cor X'000E' Write erase X'000E' Write continue transparent bloc X'000F' Write cont conv X'000F' Read with ID exch X'0010' Write disconnect X'0011' Read modified X'0011' Read connect X'0012' Write initial transparent X'0012' Write unprotectec erase X'0014' Write continue transparent X'0015' Read inquiry X'0016' Write inquiry X'001A' Write wait before transmitting X'001B' Read interrupt X'001C' Write connect X'0080' Init with reset X'0081' Read init with reset X'0082' Write init with reset X'0086' Write conv with reset **Bytes 14-15** TCTLEIOL I/O data length
This trace occurs at I/O event initiation.			

Figure 17 (Part 16 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'E6' DFHTCP	**Byte 1** (continued) X'04' (TCATRID1)	See definition of bytes 8-11 for byte 1 = X'02'	BTAM return code (reg 15) (see *OS/VS BTAM* manual) **Byte 15** X'00' Normal completion X'04' Busy X'08' Invalid relative number X'0C' Invalid type code X'10' All skip bits on X'14' Line error at open X'18' No buffers available X'1C' No buffer pool X'20' No buffer management X'24' Invalid SBA order X'28' Invalid control block detected X'2C' Device not available, used by OLTEP
	This trace occurs at I/O event initiation.		
	X'05' (TCATRID1) This trace occurs at I/O event termination.	Addr(TCTLE)	Addr(TCTTE)
	X'06' (TCATRID1)	See definition of bytes 8-11 for byte 1 = X'02'	BTAM return code (reg 15) **Byte 15** X'00' Normal return (halt I/O successful) X'04' Complete (enable already completed) X'08 Illegal request, non-TP device X'0C Unsuccessful X'10 Enable command not issued
	This trace occurs at I/O event termination.		
	X'07' (TCATRID1)	**Byte 8** TCTTELT - Terml ctl indicators X'01' Skip flag status X'02' Skip terminal read X'04' Terminal connected X'08' Specific poll X'10' Control unit permanently out of service X'20' Control unit out of service X'40' Compatible terminal X'80' Last terminal in group	Addr(TCTTE)

Figure 17 (Part 17 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)

X'E6'
DFHTCP

Byte 1
X'07' (continued)

Byte 9
TCTTEIO - Internal operation request byte
....1 Write ack
....1 Segmented write
....1 Transparent transaction
.... ..1. Task to be initiated
.... .1.. Time control transaction
.... 1... Read pending
.... 1... Graphics attention flag
...1 .... Cond GETMAIN for READ attention
..1. .... Auto output transaction
.1.. .... Auto output message
1... .... Negative response

Bytes 10-11
Reserved

This trace occurs at entry to the autopoll change.

X'E6' (continued)
DFHTACP

Byte 2, bits 4-7
X'4' System trace

Byte 1
X'10' (TCATRID1)

Bytes 8-9
Op code TCTLETOP — Terminal ID

Byte 10
TP code TCTLETPO

Byte 11
Action flags
TCTLEECB+1
(set by DFHTACP)

This trace occurs in DFHTACP just prior to transferring control to DFHTEP.

X'20' (TCATRID1)

Byte 8
Completion code
TCTLEECB

Byte 9
BTAM return code
TCTLEECB+3

Byte 10
Error flags
TCTLEPFL

Byte 11
Action flags
TCTLEECB+1

Byte 12
Command code
TCTLECC

Byte 13
Status byte TCTLESF

Byte 14
Sense byte TCTLESB

Byte 15
Sense byte TCTLESB+1

This trace occurs in DFHTACP just after returning from DFHTEP.

Figure 17 (Part 18 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'E7' DFHERM (For task-related user exit interface)	Reserved	First four characters of resource manager's name	**Bytes 12-14** Next three characters of resource manager's name **Byte 15** X'00' Indicates entry to DFHERM X'09' RM unavailable X'0F' Exit to RM X'F0' Return from RM X'FF' Indicates exit from DFHERM
X'E8' DFHCRP DFHCRQ DFHCRS DFHRTE	Reserved	Reserved	Reserved
X'EA' DFHTMP	**Byte 1** Reserved **Byte 2, bits 4-7** X'3' Entry	**Byte 8** Request Type X'01' LOCATE X'02' GET NEXT X'03' GET NEXT ALIAS X'04' ADD X'05' DELETE X'06' ALIAS X'07' LOCK X'08' UNLOCK X'09' CREATE INDEX X'0A' INDEX X'0B' QUIESCE X'0C' TRANSFER LOCK X'0D' DWE X'0E' RESET **Byte 9** Request Modifier 00000001 Nonunique elements allowed 0.....10 Unlock previous (get next only) .....1.0 Conditional request ....1.00 Table element is protected ...1..00 Table element is not freeable .01...00 Do not copy table element 01000000 Local lock 10....00 Commit **Byte 10** Table type X'01' PCT X'02' PCTR X'03' PPT X'04' PFT X'05' FCT X'06' DCT X'07' TCTE X'08' TCTN X'09' TCTS	Addr of name Addr of name Addr of name Addr of name Addr of name Addr of name Addr of name Addr of name Reserved Addr of name Addr of name Addr of name Reserved Reserved

Figure 17 (Part 19 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'EA' (continued)			
DFHTMP	**Byte 2, bits 4-7**	**Byte 8** Request Type	
	X'5' Exit	X'01' LOCATE	Addr of element
		X'02' GET NEXT	Addr of element
		X'03' GET NEXT ALIAS	Addr of element
		X'04' ADD	Addr of element
		X'05' DELETE	Reserved
		X'06' ALIAS	Addr of element
		X'07' LOCK	Reserved
		X'08' UNLOCK	Reserved
		X'09' CREATE INDEX	Reserved
		X'0A' INDEX	Addr of element
		X'0B' QUIESCE	Addr of element
		X'0C' TRANSFER LOCK	Reserved
		X'0D' DWE	Reserved
		X'0E' RESET	Reserved
		Byte 10 Table type	
		X'01' PCT	
		X'02' PCTR	
		X'03' PPT	
		X'04' PFT	
		X'05' FCT	
		X'06' DCT	
		X'07' TCTE	
		X'08' TCTN	
		X'09' TCTS	
		Byte 11 Response code	
		X'00' Normal response	
		X'04' Not found	
		X'08' Duplicate	
		X'0C' Invalid request	
		X'10' Element is busy	
		X'14' Protected entry	
		X'18' Lock held	
		X'1C' Lock noted	

Figure 17 (Part 20 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'EB' DFHAMP	**Byte 1** Reserved **Byte 2, bits 4-7** X'3' Entry	**Bytes 8-9** Function code **Bytes 10-11** Reserved	Reserved
	Byte 2, bits 4-7 X'5' Exit	Reserved	Reserved
X'EC' DFHDMP	**Byte 2, bits 4-7** X'3' Entry **Byte 1** X'00' Router X'01' Connect X'02' Disconnect X'03' Write X'04' Read X'05' Delete X'06' (Un)lock X'08' Start browse X'09' Get next X'0A' End browse X'0B' Create set X'0D' Query set X'0F' Access primary control record X'10' Build KWA X'11' Release KWA X'12' Tokenize key X'13' Free token(s) X'15' Generic qualify X'16' Seq ordered set X'17' Verify KWA X'63' Adapter	**Bytes 8-9** Function code **Bytes 10-11** Reserved	Reserved
	Byte 2, bits 4-7 X'5' Exit	Reserved	Reserved
X'ED' DFHLFO	Reserved	LIFO stack NAB value (LFDSOFNB)	**Byte 12** Module ID (=trace ID) **Byte 13** Submodule ID **Bytes 14-15** Length of LIFO stack required

Figure 17 (Part 21 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'EE' DFHZCP	**Byte 2, bits 0-3** X'1' **Byte 1** X'11' WRITE X'12' RESET X'16' SIMLOGON X'17' OPNDST X'19' CHANGE X'1F' CLSDST X'21' Close ACB X'22' SEND X'23' RECEIVE X'24' RESET X'25' Session command X'27' Send CMD X'28' Receive CMD X'29' REQSESS X'2A' OPNSEC X'2C' TERMSESS	**Bytes 8-9** Sequence number field **Byte 10** X'01' OIC (Only in chain) X'02' LIC (Last in chain) X'04' MIC (Middle in chain) X'08' FIC (First in chain) X'10' FMH X'20' Change direction (CD) X'40' End bracket (EB) X'80' Begin bracket (BB) **Byte 11** ....1 RQD2 or RQE2 .... ..1. RQD1 or RQE1 .... .0.. RQD request .... .1.. RQE request (exception response) .... 1... Negative response ...1 .... Command ..1. .... Data .0.. .... RU (data or command) .1.. .... Response 0... .... Outbound 1... .... Inbound	**Byte 12** X'01' Data or SNA command X'04' Lustat X'05' RTR X'31' Bind X'32' Unbind X'70' BIS X'71' SBI X'80' QEC X'81' QC X'82' Release queue X'83' Cancel X'84' Chase X'A0' SDT X'A1' Clear X'A2' STSN X'A3' RQR X'C0' Shutdown X'C1' Shutdown complete X'C2' Request shutdown X'C8' Bid X'C9' Signal
	Byte 2, bits 0-3 X'2'	**Bytes 8-9** Number of bytes of data **Bytes 10-11** First two bytes of data	**Bytes 12-15** Next four bytes of data
	Byte 2, bits 0-3 X'3'	**Byte 8** X'00' **Bytes 9-11** 'NEG'	**Bytes 12-15** Negative response code (sent or received)
	Byte 2, bits 0-3 X'4'	**Byte 8** X'01' **Bytes 9-11** 'LUS'	**Bytes 12-15** Lustat code (sent or received)
	Byte 2, bits 0-3 X'5'	**Byte 8** X'02' **Bytes 9-11** 'SIG'	**Bytes 12-15** Signal code (sent)

Figure 17 (Part 22 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'EE' (continued)			
DFHZCP	**Byte 2, bits 0-3** X'6' SESSIONC	**Byte 8** X'01' STSN **Bytes 9-10** RPLIBSQV **Byte 11** RPLIBSQ X'02' Clear X'03' SDT X'04' Bind X'05' Switch	**Byte 12** X'01' **Bytes 13-14** RPLOBSQV **Byte 15** RPLOBSQ
X'EF' DFHTOR	**Byte 2, bits 4-7** X'3' Entry **Byte 1** X'01' Main program X'02' Add nonpooled terminal X'03' Add pooled terminal X'04' Add TYPETERM X'05' Add connection X'06' Add sessions X'07' End group event 1 X'08' End group event 2 X'09' DWE commit X'0A' DWE cancel X'0B' Utility 1 X'0C' Utility 2 X'10' Model retrieval X'11' Create terminal BPS X'12' Recovery program **Byte 2, bits 4-7** X'5' Exit	**Bytes 8-9** Function code **Bytes 10-11** Reserved Reserved	Reserved Reserved

Figure 17 (Part 23 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'F0' DFHKCP	**Byte 1**		
	X'01' ENQ	Queue name address, TCATCQA	Reserved
	X'02' DEQ	Queue name address, TCATCQA	Reserved
	X'03' DEQALL	Reserved	Reserved
	X'04' SUSPEND	**Byte 8**	
		Reason for suspend, TCATCDC	Reserved
		Bytes 8-11	
	X'05' ENQ conditional	Queue name address, TCATCQA	Reserved
	X'06' ATTACH with STARTCODE	Facility address, TCAKCFA	Transaction ID, TCAKCTI
	X'07' ATTACH COND with STARTCODE	Facility address, TCAKCFA	Transaction ID, TCAKCTI
	X'08' RESUME	TCA address of resumed transaction	**Byte 12** TCA priority **Bytes 13-15** Task number
	X'0A' Change MXT value	Reserved	**Bytes 12-15** Reserved
	X'0D' Attach HTA (HPO only)	Address of HTA, TCARSTSK	Reserved
	X'0E' CANCEL with FORCE=NO	Original transaction ID	**Bytes 13-15** Task no of canceled task
	X'0F' CANCEL with FORCE=YES	Original transaction ID	Task no of canceled task
	X'1x', X'3x', X'9x', X'Bx' ATTACH as follows:		**Bytes 12-15**
	...1 .... Bit set for all ATTACH requests:	Facility address (if any) from TCAKCFA or terminal name if facility is a terminal	Transaction ID from TCAKCTI
	...1 ...1 Conditional request		
	...1 ..1. STARTCODE passed		
	...1 .1.. System task (AMXT and MXT limits ignored)		
	...1 1... Return to caller if PCT entry not found		
	..11 .... Userid passed		
	1..1 .... ACTION=DELAYED		

Figure 17 (Part 24 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'F0' DFHKCP	**Byte 1** (continued)	**Byte 8**	
	X'20' CHAP	New priority, TCATCDP **Bytes 8-11**	Reserved
	X'28' LOCATE PCT entry DOMAIN=ALL	Reserved	Reserved
	X'29' LOCATE PCT entry DOMAIN=REGION	Reserved	Reserved
	X'2A' BROWSE	Addr PCT entry	Reserved
	X'2B' BROWSE UNLOCK	Addr PCT entry	Reserved
	X'2C' LOCATE PROFILE	Reserved	Reserved
	X'2D' BROWSE PROFILE	Reserved	Reserved
	X'2E' BROWSE PROFILE UNLOCK	Reserved	Reserved
	X'2F'	**Byte 8** X'01' REPLACE X'02' INITIALIZE X'03' WAITINIT X'04' RESTART TASK **Bytes 9-11** Addr parameter list	Reserved Reserved Reserved Reserved
	X'3x' ATTACH (see X'1x')		
	X'40' WAIT	**Byte 8** Dispatch control, TCATCDC **Bytes 8-11**	Event control address, TCATCEA
	X'80' DETACH	Reserved **Byte 8**	Reserved
	X'84' SUSPEND + CANCEL exit	Reason for suspend, TCATCDC	Reserved
	X'9x' ATTACH (see X'1x')		
	X'Bx' ATTACH (see X'1x')		
	X'C0' WAIT + CANCEL exit	Dispatch control, TCATCDC	Event control address, TCATCEA
X'D0' DFHKCP	**Byte 1**		
	X'05' Task dispatched	Reserved	Reserved
	X'06' Task created	Terminal ID (or zeros for non-terminal tasks)	Transaction ID
	X'07' Task terminated	**Bytes 8-9** Initial LIFO stack segment size (PCTISA01) for the transaction **Bytes 10-11** Overflow LIFO stack segment size (PCTISA02) for the transaction	Transaction ID
	X'09' System RESUME	**Byte 8** Timer ECB **Byte 9** No. of ECBs to be posted for MVS to dispatch CICS **Bytes 10-11** No. of AMXT active tasks	Time when wait was issued
	X'0A' Suspend	Reserved	Reserved

Figure 17 (Part 25 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'F1' DFHSCP	**Byte 1**		
	0000 0000 Extended interface	First word of extended interface parameter list	Address of parameter list, TCASCXPA
	001. .... Cushion change	Reserved	Reserved
	01.. .... FREEMAIN	Address of storage (or owning TCA)	Facility address, TCAFCAAA
	011. .... RELEASE =ALL	Reserved	Reserved
	1..s ssss GETMAIN	) **Byte 8**	Facility address,
	1.1s ssss Conditional GETMAIN	) Reserved) **Byte 9**	TCAFCAAA
	11.s ssss Initialize storage	) Initialization) byte) **Bytes 10-11**) Number of bytes) requested	
	The s ssss bit pattern specifies the storage class:		
	00000 1WD	01100 USER	11000 TSTABLE
	00001 DCA	01101 TRANSDATA	11001 MAP
	00010 QEA	01110 TEMPSTRG	11010 PERMANENT
	00011 ISC TIOA	01111 FILE	11011 JCA
	00100 LINE	10000 RPL	11100 Reserved
	00101 TERMINAL	10001 WRE	11101 DWE
	00110 ICE	10010 BCA	11110 MAPCOPY
	00111 AID	10011 SHARED	11111 DL/I
	01000 PROGRAM	10100 CONTROL	
	01001 RSA	10101 EXTPGM	
	01010 TCA	10110 TACLE	
	01011 LLA	10111 TSMAIN	
X'C8'	Reserved	Address of storage acquired	Storage accounting
X'C9'	Reserved	Address of storage released	Storage accounting
X'CA'	Reserved	Address of active TCA when storage control recovery was entered	Instruction address where storage error found, if program check occurred

Figure 17 (Part 26 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'F2' DFHPCP	**Byte 1 and byte 2, bits 0-3**		
	X'010' LINK		Reserved
	X'020' XCTL		
	X'030' CTYPE LOCATE		
	X'031' CTYPE BROWSE	Addr PPT entry	
	X'032' CTYPE LOCATE NOWAIT		
	X'035' CTYPE BROWSE UNLOCK	Addr PPT entry	
	X'040' Unconditional LOAD		
	X'051' CTYPE REPLACE	Addr parameter list from TCAPCEA	
	X'052' CTYPE INITIALIZE		
	X'053' CTYPE WAITINIT		
	X'080' DELETE		
	X'100' RETURN		
	X'120' RETRY		
	X'200' SETXIT, CANCEL		
	X'201' SETXIT, PROGRAM		
	X'202' SETXIT, ROUTINE	Routine address, TCAPCERA (or 0 if command-level COBOL)	
	X'204' BLDL		
	X'208' RESETXIT		
	X'240' Uncond LOAD, LOADLST=NO		
	X'400' ABEND, no dump		
	X'410' ABEND, no dump, CANCEL=YES		
	X'440' Uncond LOAD (RMODE ANY)		
	X'600' ABEND with dump	ABEND code from TCAPCAC	
	X'610' ABEND with dump, CANCEL=YES	ABEND code from TCAPCAC	
	X'640' Uncond LOAD, LOADLST=NO (RMODE ANY)		
	X'810' Conditional LINK		
	X'820' LOCATE		
	X'840' Conditional LOAD		
	X'880' Conditional XCTL		
	X'A40' Conditional LOAD, LOADLST=NO		
	X'C40' Conditional LOAD (RMODE ANY)		
	X'E40' Conditional LOAD, LOADLST=NO (RMODE ANY)		

Figure 17 (Part 27 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'F3' DFHICP	**Byte 2, bits 4-7** X'3' Entry **Byte 1**		
	0001 00.. GETIME	Return time to user address, TCAICDA	Reserved
	0001 ...0 Binary format		
	0001 ...1 Packed format		
	0001 ..0. Refresh CSA time only		
	0001 ..1. Return time to user		
	0010 ...0 WAIT	) INTRVAL or TIME	Reserved
	0011 ...0 POST	) value, TCAICRT	
	001. .00. No request ID provided		
	001. .01. Request ID prefix in TCAICQPX		
	001. .10. Request ID in TCAICQID		
	001. 0... INTRVAL specified		
	001. 1... TIME specified		
	0100 .... INITIATE	) INTRVAL or TIME	Reserved
	0101 .... PUT	) value, TCAICRT	
	0110 .... INITIATE	)	
	0111 .... PUT	)	
	01.. ...0 Nonterminal dest		
	01.. ...1 Terminal dest		
	01.. .00. No request ID supplied		
	01.. .01. Request ID prefix in TCAICQPX		
	01.. .10. Request ID in TCAICQID		
	01.. 0... INTRVAL supplied		
	01.. 1... TIME supplied		

Figure 17 (Part 28 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
	1000 00.. GET	User-provided data address	Reserved
	1000 ...1 RELEASE=NO		
	1000 ..0. Data address specified		
	1000 ..1. Return address to user		
	1001 1... WAIT		
	1001 000. RETRY		
	1001 ...1 RELEASE=NO		
	1010 0000 RESET	Reserved	Reserved
	1011 0000 SCHEDULE	Reserved	Reserved
	1100 .... EXPIRY ANALYSIS	Reserved	Reserved
	1101 .... RECOVERY CALL	Reserved	Reserved
	1110 .... Reserved	Reserved	Reserved
	1111 0.00 CANCEL		
	1111 .0.. No request ID provided		
	1111 .1.. Request ID specified		
	X'5' Exit		
	Byte 1 Return code	Reserved	Reserved
	X'00' NORESP		
	X'01' ENDDATA		
	X'04' IOERROR		
	X'11' TRNIDER		
	X'12' TRMIDER		
	X'14' TSINVLD		
	X'20' EXPIRD		
	X'81' NOTFND		
	X'FF' INVREQ		

Figure 17 (Part 29 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'F4' DFHDCP	**Byte 1** ....1 User-specified area .... .1.. Transaction areas .... 1... Terminal I/O areas ...1 .... Trace table ..1. .... Program areas .1.. .... TCA 1... .... CSA	**Byte 8** ....1 DCT .... ..1. FCT .... .1.. TCT .... 1... PCT ..1. .... PPT .1.. .... SIT **Bytes 9-11** Reserved	Dump code
X'F5' DFHFCP	**Byte 2, bits 4-7** X'3' Entry **Byte 1** X'00' DWE phase 2 processor X'01' DELETE ** X'02' CTYPE OPEN X'03' CTYPE IMPLICIT_ OPEN X'04' CTYPE MAKE_ME_A_ USER X'05' CTYPE CLOSE, no DISABLE X'06' CTYPE CLOSE, DISABLE, WAIT X'07' CTYPE CLOSE, DISABLE, NOWAIT X'08' CTYPE CLOSE, DISABLE, FORCE X'09' CTYPE ENABLE X'0A' CTYPE DISABLE X'0B' CTYPE TESTUSER X'0C' DFHFCM OPEN X'0D' DFHFCM CLOSE X'0E' DFHFCN OPEN X'0F' DFHFCN CLOSE X'10' RELEASE ** X'11' ESETL ** X'20' GETAREA ** X'24' GETAREA_MASS_ INSERT ** X'28' GETAREA_INITIALIZE_ STORAGE ** X'2C' GETAREA_MASS_INSERT_ INIT_STORAGE ** X'40' PUT UPDATE **	Addr(FCTE) Addr(FCTE) Addr(FCTE) Addr(FCTE) Addr(FCTE) Addr(FCTE) Addr(FCTE) Addr(FCTE) Addr(FCTE) Addr(FCTE) Addr(FCTE) Addr(FCTE) Addr(FCTE) Area addr from TCAFCAA	Addr(DWE)

Figure 17 (Part 30 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)

```
X'F5'      Byte 1 (continued)
DFHFCP     X'41' PUT DELETE **
           X'44' PUT NEW **
           X'80' GET **
           X'81' GET BDAM DEBLOCKING *
           X'84' GET UPDATE **
           X'85' GET BDAM DEBLOCKING
                 UPDATE *
           X'A0' SETL **
           X'A1' SETL BDAM DEBLOCKING *
           X'A4' RESETL **
           X'B0' GETNEXT **
           X'B4' GETPREV **
           X'F0' CTYPE SET              Addr(FCTE)

                                                        Byte 12
                                                        Request byte 1 to DFHFCS
                                                        X'03' OPEN
                                                        X'04' CLOSE
                                                        X'05' ENABLE
                                                        X'06' DISABLE
                                                        X'07' Set DSNAME
                                                        X'09' Set STRINGS
                                                        X'0A' Set LSRPOOLID
                                                        X'0B' Set DISPOSITION
                                                        X'0C' Set EMPTY CVDA
                                                        X'0D' Set READ CVDA
                                                        X'0E' Set UPDATE CVDA
                                                        X'0F' Set BROWSE CVDA
                                                        X'10' Set ADD CVDA
                                                        X'11' Set DELETE CVDA
                                                        X'12' Reorganize DSN
                                                              block chains
                                                        X'13' Set EXCLUSIVE CVDA
                                                        X'15' Set base cluster
                                                        X'16' Release base
                                                              cluster

                                                        Byte 13
                                                        Request byte 2 to DFHFCS
                                                        .... 1... Implicit
                                                                  (OPEN only)
                                                        ...1 .... EMPTY
                                                        ..1. .... FORCE
                                                        .1.. .... NOWAIT
                                                        1... .... WAIT
           X'F1' CTYPE LOCATE
           X'F4' CTYPE BROWSE           Addr(FCTE)
           X'F5' CTYPE INITIALIZE
           X'F6' CTYPE WAITINIT
           X'F7' CTYPE RESTART_TASK

         * For deblocking a BDAM data set, byte 2, bits 0-3, contains
            X'8' for deblocking by key, or X'4' for deblocking by relative record.
         ** If a VSAM data set is being processed, byte 2, bits 0-3, can have
             the following settings:
             ...1  Locate mode
             ..1.  Search key greater than or equal to
             .1..  Argument is generic key
             1...  Argument is RBA
```

Figure 17 (Part 31 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)

X'F5' (continued)

DFHFCP

Byte 2, bits 4-7

X'5' Exit

Byte 1 Return code

REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'00' NORESP	) For requests to	
X'01' DSIDER	) VSAM data sets:	
X'02' ILLOGIC	) **Bytes 8-11**	**Bytes 12-14**
X'08' INVREQ	) VSAM RPL	VSAM option flags
X'0C' NOTOPEN	) feedback area	(first 3 bytes of
X'0F' ENDFILE	) (VSWARESP/RPLFDBWD)	VSWAOPTC/RPLOPTCD)
	) **Byte 8**	**Byte 15**
	) RPL status flags	VSAM request type
	) **Byte 9**	(VSWAREQ/RPLREQ)
	) VSAM return code	
	) **Byte 10**	
	) VSAM component	
	) issuing code	
	) **Byte 11**	
	) VSAM error code	
X'20' Error		
X'40' DFHFCN normal		
X'41' DFHFCN warning	**Bytes 8-9**	**Bytes 12-13**
	Internal DFHFCN return code *	Error information if applicable
	Bytes 10-11	
	Register 15 return code if applicable	
X'42' DFHFCN error	**Bytes 8-9**	**Bytes 12-13**
	Internal DFHFCN return code *	Error information if applicable
	Bytes 10-11	
	Register 15 return code if applicable	

* In most cases, there is a corresponding console message and an explanation is given in the *Messages and Codes* manual.

REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'80' IOERROR	) For requests to	
X'81' NOTFND	) VSAM data sets:	
X'82' DUPREC	) **Bytes 8-11**	**Bytes 12-14**
X'83' NOSPACE	) VSAM RPL	VSAM option flags
X'84' DUPKEY	) feedback area	(first 3 bytes of
	) (VSWARESP/RPLFDBWD)	VSWAOPTC/RPLOPTCD)
	) **Byte 8**	**Byte 15**
	) RPL status flags	VSAM request type
	) **Byte 9**	(VSWAREQ/RPLREQ)
	) VSAM return code	
	) **Byte 10**	
	) VSAM component	
	) issuing code	
	) **Byte 11**	
	) VSAM error code	

Figure 17 (Part 32 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'F6' DFHTDP	**Byte 2** X'03' Appl request **Byte 1**		
	0000 .100 TYPE=PURGE	See below	Reserved
	0001 0000 TYPE=LOCATE	Reserved	Reserved
	0010 .000 TYPE=FEOV	See below	Reserved
	0100 .000 TYPE=PUT	See below	Data addr (from TCATDAA)
	1000 .000 TYPE=GET	See below	Reserved
	1100 .000 TYPE=GET, QUEBUSY=YES	See below	Reserved
	1110 0001 CTYPE=LOCATE	Reserved	Reserved
	1110 0100 CTYPE=BROWSE	Addr(DCTE) or 0	Reserved
	1111 0000 CTYPE =INITIALIZE	Reserved	Reserved
	1111 0001 CTYPE=WAITINIT	Reserved	Reserved
	1111 0010 CTYPE=FLUSH	Reserved	Reserved
	1111 0011 CTYPE =PROCESS_DWE	Addr(DCTE)	Addr(DWE)
	1111 0100 CTYPE=RECOVER	Addr(DCTE)	RBA
	.... 0... Name supplied	Reserved	
	.... 1... Addr supplied	Addr(DCTE)	
	Byte 2 X'13' Start of cancel logic **Byte 1**		
	Reserved	Addr(cancel token)	Reserved
	Byte 2 X'23' Start of I/O error logic X'33' Start of init phase 2 logic X'43' Start of init phase 1 logic		
	Byte 2 X'05' Appl response **Byte 1**		
	X'00' NORESP	Reserved	Reserved
	X'01' QUEZERO		
	X'02' IDERROR		
	X'04' IOERROR		
	X'08' NOTOPEN		
	X'10' NOSPACE		
	X'40' Invalid chain		
	X'C0' QUEBUSY		
	Byte 2 X'15' End of cancel logic X'25' End of I/O error logic X'35' End of init phase 2 logic X'45' End of init phase 1 logic		

Figure 17 (Part 33 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'F7' DFHTSP	**Byte 2** X'03' Appl request **Byte 1** 0000 0010 FLUSH 0010 00.. RELEASE 0010 ...1 Queue-type request (PURGE) 0010 ..1. Buffer flush request 010. .... PUT 010. ...1 Queue-type request (PUTQ) 010. ..1. System request 010. .1.. Replace 010. 1... Main storage 0101 .... Conditional request 10.. ..0. GET 10.. ...1 Queue-type request (GETQ) 10.. ..1. Storage class=terminal 10.. .1.. Exclusive control 10.. 1... Entry number supplied on GETQ 10.1 .... Input area address 101. .... RELEASE	- - - - Data identification, TCATSDI - - - - - - - - Data identification, TCATSDI - - - - - - - - Data identification, TCATSDI - - - -	
	Byte 2 X'13' Start of DWE logic X'23' Start of cancel logic		
	Byte 2 X'05' Appl response **Byte 1** X'00' NORESP X'01' ENERROR X'02' IDERROR X'04' IOERROR X'08' NOSPACE X'20' INVREQ	Reserved	Reserved
	Byte 2 X'15' End of DWE logic X'25' End of cancel logic		

Figure 17 (Part 34 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'F8' DL/I	**Byte 2, bits 4-7**		
	X'3' Entry	DL/I function code:	
		'PCB'	Addr(PSB)
		'TERM' or 'T'	Reserved
		Data base access call	Addr(PCB)
		Other	Reserved
	X'4' IMS/VS IWAIT	Reg 14 at point of call from IMS/VS module to IWAIT routine	Addr(ECB) (usually PST)
	X'5' Exit	**Byte 8** Response code TCAFCTR	DL/I function code
		Byte 9 Response code TCADLTR	
		Bytes 10,11 PCB status code, if any. If byte 8 = X'08' and byte 9 = X'05' (that is, the response is 'PSB schedule failure') and the PSB is local to this CICS system, then byte 10 contains the value of the PSTSCHDF field in the IMS/VS PST control block. (See the IMS/VS DSECTs for a description of PSTSCHDF, and the *IMS/VS Application Programming for CICS/VS Users* manual for a description of the PCB status code.)	

Figure 17 (Part 35 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'F9' DFHJCP	**Byte 2, bits 4-7** X'3' Entry	**Byte 8** Journal identifier Addr(JCA) X'01' system log X'02' to X'63' user journal number (binary) X'FF' dynamic log **Byte 9** Reserved **Bytes 11,10** Type of request X'01' WRITE X'01' COND=YES X'02' STARTIO=YES X'04' User prefix specified X'08' DL/I request X'10' USING clause passes data address (COBOL) X'80' CICS request X'02' WAIT X'02' STARTIO=YES X'03' PUT Same as (WRITE, WAIT) X'04' OPEN X'01' Output X'02' Input X'04' VOL=FIRST (output) or VOL=PREVIOUS (input) X'08' VOL=NEXT X'10' VOL=CURRENT X'20' SIVOL=YES (Tape journals) X'28' VOL=EMEREXT X'48' VOL=NEXT, POSN=ASIS X'80' CICS request X'08' CLOSE X'01' LEAVE=YES X'44' VOL=PREVIOUS, POSN=ASIS X'48' VOL=NEXT, POSN=ASIS X'10' NOTE X'20' POINT X'40' GETF X'80' GETB	

Figure 17 (Part 36 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
	Byte 2, bits 4-7 X'5' Exit	**Byte 8** Return code X'00' Normal response X'01' IDERROR X'02' INVREQ X'03' STATERR X'04' VOLERR X'05' NOTOPEN X'06' LERROR X'07' IOERROR X'08' End of data set X'09' Insufficient buffer space for a conditional request X'0A' Dynamic log error **Bytes 9-11** Same as entry trace	Reserved

Figure 17 (Part 37 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FA' BMS	**Byte 1** X'00' **Byte 2, bits 4-7** X'3' Entry	**Byte 8** X'01' OPCLASS= operator class X'02' LIST=symbolic address X'04' LIST=ALL X'08' TIME=numeric X'10' INTERVAL= numeric X'20' ERRTERM=termid X'40' ERRTERM=ORIG X'80' TYPE=ROUTE **Byte 9** X'01' TYPE=PURGE X'04' Send partition set X'08' IOTYPE=IMMED X'10' LDC=mnemonic or YES X'20' REQID=alphanum. X'40' PROPT=NLEOM X'80' TITLE=symbolic address **Byte 10** X'01' MAPSET=map set name X'02' MSETADR= symbolic addr X'04' MAP=map name X'08' CTRL=any 3270 WCC X'10' CURSOR=number X'20' TYPE=TEXT X'40' Receive partition set X'80' TYPE=LAST **Byte 11** X'01' TYPE=IN X'02' TYPE=ERASE X'04' TYPE=MAP X'08' TYPE=WAIT X'10' MAPADR=symbolic address X'20' TYPE=SAVE X'40' DATA=NO X'C0' DATA=YES	**Byte 12** X'01' TYPE=RETURN X'02' TYPE=STORE X'04' TYPE=OUT X'20' TYPE=ERASEAUP X'40' OFLOW=symbolic address X'80' TYPE=PAGEBLD **Byte 13** X'01' EODPURG=OPER X'02' WRBRK=ALL X'04' WRBRK=CURRENT X'08' CTRL=RELEASE X'10' CTRL=RETAIN X'20' CTRL=PAGE X'40' CTRL=AUTOPAGE X'80' TYPE=PAGEOUT **Byte 14** X'01' TYPE=NOEDIT X'04' Active partition X'08' Out partition X'10' JUSTIFY specified X'20' TRAILER=symbolic addr. X'40' HEADER=symbolic addr. X'80' TYPE=TEXTBLD **Byte 15** X'01' Control X'04' WRBRK=symbolic address X'08' RDATT=symbolic address X'10' FMHPARM=YES or parameter X'20' Request by EXEC interface X'40' Magnetic stripe reader X'80' In partition

Figure 17 (Part 38 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FA' BMS	**Byte 1** (continued) X'00' **Byte 2, bits 4-7** X'5' Exit	**Byte 8** Reserved X'00' Normal response X'01' Invalid request X'02' Page returned X'04' I/O area cannot be mapped X'08' Map too large X'20' Invalid error terminal X'40' Route worked, some resolutions X'80' Route failed, no resolutions **Byte 9** X'02' Partition fail X'04' Invalid partition X'08' Invalid partition set X'10' Invalid LDC mnemonic X'80' Temporary storage I/O error **Byte 10** X'01' PAGEBLD overflow (See note 2 on page 110) X'02' Inbound FMH in last input X'04' End-of-data-set in last input X'08' End-of-chain in last input X'10' Specified REQID ignored **Byte 11** Terminal code from TCAMSRI1 when byte 8 contains X'08'.	

Figure 17 (Part 39 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FA' BMS	**Byte 1** (continued)		
	X'01' DFHRLR entry	Addr(OSPWA)	Addr(TCTTE)
	X'02' DFHPBP entry	Addr(TTP)	Addr(TIOA)
	X'03' DFHM32 entry	Addr(TTP)	Addr(TIOA)
	X'04' DFHML1 entry	Addr(TTP)	Addr(TIOA)
	X'05' DFHDSB entry	Addr(TTP)	Addr(TIOA)
	X'06' DFHTPP entry	Addr(TTP)	Addr(TIOA)
	X'07' DFHIIP entry	Addr(TTP)	Addr(TIOA)
	X'41' DFHRLR exit	Addr(OSPWA)	Addr(TTP)
	X'42' DFHPBP exit	Addr(TTP)	Addr(TIOA)
	X'43' DFHM32 exit	Addr(TTP)	Addr(TIOA)
	X'44' DFHML1 exit	Addr(TTP)	Addr(TIOA)
	X'45' DFHDSB exit	Addr(TTP)	Addr(TIOA)
	X'46' DFHTPP exit	Addr(TTP)	Addr(TIOA)
	X'47' DFHIIP exit	Addr(TTP)	Addr(TIOA)
	X'81' Query entry 1	**Byte 8** TCTE32EF **Byte 9** TCTE32E2 **Byte 10** TCTE32E3 **Byte 11** TCTE32SF	**Bytes 12-13** TCTEASCZ **Byte 14** TCTEASCL **Byte 15** TCTEASCC
	X'82' Query entry 2	**Byte 8** TCTETXTF **Byte 9** TCTTEFMB **Bytes 10-11** Reserved	**Bytes 12-13** TCTECSG1 **Bytes 14-15** TCTECSG2
X'CD' BMS	Reserved (See note 3 on page 110)	First four bytes of temporary storage key	Next four bytes of temporary storage key

Figure 17 (Part 40 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FB' DFHBIF	**Byte 1**	**Byte Contents**	Byte address
	X'01' BITEST	8 Reserved	
		9 Function type X'10' BITEST X'20' BITFLIP X'40' BITSETOFF X'80' BITSETON	
		10 Bit pattern	
		11 Reserved	
	X'02' DEEDIT	8-9 Field length	Field address
		10-11 Reserved	
	X'03' INFORMAT	8 Reserved	
		9 Name list indicator X'00' No list	
		10-11 TIOA size	
	X'04' PHONETIC	Bytes 1-4 of the name	Bytes 5-8 of the name
	X'05' FVERIFY	8-9 Field length	Field address
		10-11 Reserved	
	X'06' TSEARCH	8 Reserved	Argument address
		9 Function code X'01' Target address X'02' Right side X'04' Left side X'08' Mixed table X'10' Descending order X'20' Ascending order X'40' Range	
		10-11 Number of entries in arg table	
	X'07' WTRETST	8 Reserved	Key address
		9 SETL indicator	
		10-11 Max. number of records	
	X'08' WTRTPARM	8-9 Reserved	VSWA address
		10-11 FIELD1 -second operand	
	X'09' WTRETGET	Same as WTRETST	VSWA address
	X'0A' WTRETREL	Same as WTRETST	VSWA address

Figure 17 (Part 41 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FC' DFHZCP	**Byte 2, bits 4-7** X'3' or X'4' except where stated **Byte 1** X'00' Exit X'01' Application req		
	Byte 2, bits 4-7 X'3' Entry	Format A2	Format B2 if byte 8 = X'00', otherwise not used
	Byte 2, bits 4-7 X'5' Exit	Format A2	Format B2 with bytes 13-15 = TCTTEDA value
	X'02' Locate		
	Byte 2, bits 4-7 X'3' Entry	Format A11	TCTTE name (locate ID) or TCTTE address
	Byte 2, bits 4-7 X'5' Exit	**Bytes 8-9** Format A11 **Byte 10** X'08' Option=NOWAIT X'20' Select request X'40' Status request X'80' TRMIDNT supplied **Byte 11** X'00' Normal X'F0' Last entry X'F1' Invalid request X'F2' Invalid ID X'F3' Invalid address X'F9' Not LU6.2 X'FA' Busy	Addr located entry, or X'FFFFFFFF'
	X'03' Detach	Format A1	Format B1
	X'04' Reserved		
	X'05' Startup task	Format A1	Format B1
	X'06' CTYPE request	Format A1	Format B1
	X'07' Reserved		
	X'08' Status change	Format A11	
	X'09' Transaction routing		
	Byte 2, bits 4-7 X'3' Entry	**Byte 8** X'0B' ATTACH X'0C' Applic request X'0D' DETACH X'0E' FLUSH X'0F' Route transaction	**Bytes 13-15** Relay link TCTTE address
	Byte 2, bits 4-7 X'5' Exit	**Byte 8** X'00' Normal response X'08' Error response	Surrogate TCTTE address
	X'0A' RPL executor,HPO	Format A1	Format B1

Figure 17 (Part 42 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FC' DFHZCP	**Byte 1** (continued) X'0B' Allocate/free		
	Byte 2, bits 4-7 X'3' Entry	**Byte 8** X'01' Allocate X'02' Point X'03' Free X'04' Free detach X'05' Free all X'06' LU6.2 allocate X'07' LU6.2 free	Format B1
	Byte 2, bits 4-7 X'5' Exit	**Byte 8** X'01' Allocate X'02' Point X'03' Free X'04' Free detach X'05' Free all X'06' LU6.2 allocate X'07 LU6.2 free **Byte 9** Return code X'00' Normal return X'04' Error return **Byte 10** Secondary code X'01' Invalid call to DFHZISP, function not supported X'02' Invalid call to DFHZISP, no function given X'03' Invalid free call X'04' Invalid LU6.2 parameters passed X'05' No profile exists X'06' Allocate for a session already owned X'07' Noqueue req and all sessions busy X'08' MODENAME not found X'0A' Free req for session not owned by task X'0B' Free req for system entry X'0C' MODENAME invalid X'0E' Detach for remote terminal failed X'10' Task canceled during alloc process X'14' Mode group out of service X'18' DRAIN=ALL set, mode group terminating X'21' Point req failed, no sessions available X'22' Point req failed, session not usable X'23' No CICS region block (IRC) X'24' No SCCB (IRC)	**Bytes 13-15** Reserved

Figure 17 (Part 43 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FC' DFHZCP	**Byte 1** (continued)		
	X'0C' CICS function request shipping sync point	**Byte 8** X'01' Prepare X'02' SPR X'03' Commit X'04' Abort	Format B1
	X'0D' IRC **Byte 2, bits 4-7** X'3' Entry	**Byte 8** X'01' Receive X'02' Disconnect X'05' I/O request **Byte 10** X'08' RESET X'10' LAST X'20' READ X'40' WAIT X'80' WRITE X'08' Normal STOP X'09' Immediate stop X'0A' Logoff X'10' Get data X'11' Operative X'12' Receive abort	Addr(TCTTE)
	X'0E' Abend	Format A1	Format B1
	X'0F' Close ACB	Format A9	Reserved
	X'10' Auto task init	Format A1	Format B1
	X'11' Attach **Byte 2, bits 4-7** X'3' Entry X'5' Exit	Format A1 Reserved	Format B1 Reserved
	X'12' Freemain	Format A1	Format B1
	X'13' Getmain	Format A1	Format B1
	X'14' Receive any	Format A8	Format B1
	X'15' RESETSR	Format A1	Format B1
	X'16' Receive specific	Format A8	Format B1
	X'18' Send normal	Format A8	Format B1
	X'1A' Translation	Format A1	Format B1
	X'1B' User exit	Format A1	Format B1
	X'1C' Activate scan	Format A10	Format B1
	X'1D' Send response	Format A8	Format B1
	X'1E' Reserved		
	X'1F' Reserved		
	X'20' Attach initialization	Format A1	Format B1
	X'22' CLSDST	Format A1	Format B1
	X'24' DWE processor	Format A1	Format B1
	X'27' Logical record presentation	Format A6	Format B1
	X'29' OPNDST	Format A1	Format B1
	X'2B' Read-ahead queuing	Format A1	Format B1
	X'2C' Read-ahead retrieval	Format A1	Format B1
	X'30' Resync	Format A1	Format B1
	X'33' Send async	Format A1	Format B1
	X'34' Send cmnd resp	Format A1	Format B1
	X'35' SessionC	Format A1	Format B1
	X'37' SIMLOGON	Format A1	Format B1
	X'39' SETLOGON start	Format A1	Format B1
	X'3E' Open ACB	Format A1	Format B1

Figure 17 (Part 44 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FC' DFHZCP	**Byte 1** (continued)		
	X'3F' Shutdown	Format A9	Format B1
	X'40' TCTTE queuing		
	X'41' Error message writer	Format A5	Format B1
	X'42' VTAM sync point handler	Format A2	
	X'43' VTAM trace handler	Reserved	Reserved
	X'44' ZARQ abend handler		
	X'45' Console input handler		
	X'46' Console request handler	**Byte 8** Console ID **Bytes 9-11** Reserved	
	X'47' Console abnormal condition handler	**Byte 8** Console ID **Bytes 9-11** Reserved	
	X'48' Attach user exit	Reserved	Reserved
	X'49' Output user exit	Reserved	Reserved

Figure 17 (Part 45 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
	X'4A' DFHZARL, LU6.2 appl request		
	Byte 2, bits 4-7	**Byte 8**	Addr(TCTTE)
	X'3' Entry	X'01' ALLOCATE	
		Byte 9	
		X'40' LUCASYS valid	
		X'80' NOQUEUE specified	
		Byte 10	
		X'40' ATI ALLOCATE	
		X'80' LUCMODNM valid	
		X'03' EXTRACT PROCESS	
		X'05' FREE	
		X'06' ISSUE ABEND	
		Byte 9	
		X'80' User invocation	
		Byte 10	
		X'20' LUCMSGNO valid	
		X'40' LUCSENSE valid	
		X'80' LUCAMSG, LUCLMSG valid	
		X'07' ISSUE ATTACH	
		Byte 9	
		X'80' TPN check not required	
		X'08' ISSUE CONFIRMATION	
		X'09' ISSUE ERROR	
		Byte 9	
		X'80' User invocation	
		Byte 10	
		X'20' LUCMSGNO valid	
		X'40 LUCSENSE valid	
		X'80' LUCAMSG, LUCLMSG valid	
		X'0A' ISSUE SIGNAL	
		X'0B' RECEIVE	
		X'80' SET	
		Byte 10	
		X'40' BUFFER	
		X'80' LLID	

Figure 17 (Part 46 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FC' DFHZCP	**Byte 1** X'4A' DFHZARL **Byte 2, bits 4-7** X'3' Entry (continued)	**Byte 8** (continued) X'0C' SEND **Byte 9** X'80' Application data **Byte 10** X'10' WAIT X'20' CONFIRM X'40' LAST X'80' INVITE X'0D' WAIT X'10' FREE STORAGE X'11' INITIAL CALL **Byte 9** X'80' Data provided X'12' ALLOCATE_ PRIVILEGED **Byte 9** X'40' LUCASYS valid X'80' NOQUEUE X'13' SYNC PREPARE **Byte 10** X'40' LAST X'14' SYNC REQ COMMIT **Byte 10** X'40' LAST X'15' SYNC COMMITTED **Byte 9** X'40' Implicit FORGET X'80' Explicit FORGET X'16' SYNC FORGET X'17' ABORT X'18' GET LUNAME X'19' SYNC ROLLBACK X'1A' SEND FMH **Byte 9** X'80' Application data X'1B' RECEIVE FMH **Byte 9** X'80' SET X'20' ERP FMH received X'21' Negative response received	

Figure 17 (Part 47 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FC' DFHZCP	**Byte 1** X'4A' DFHZARL (continued) **Byte 2, bits 4-7** X'5' Exit	**Byte 8** X'00' Normal response Addr(TCTTE) X'01' SYSIDERROR **Byte 9** X'04' **Byte 10** X'04' No bound contention-winner session available (SYSBUSY) X'08' Modename not known on this system X'0C' Attempt to use reserved modename (SNASVCMG) X'14' Available count set to zero for this modegroup **Byte 9** X'08' SYSID is out of service **Byte 10** X'00' Local queuing was not attempted X'04' Local queuing did not succeed **Byte 9** X'0C' SYSID not known in TCT **Byte 10** X'00' SYSID name is not known X'04' SYSID name is not that of a TCTSE X'08' SYSID.MODENAME is not known X'0C' SYSID.PROFILE is not known **Byte 8** X'03' INVREQ error **Byte 9** X'00' Session is not defined as LU6.2 X'04' Conversation level is wrong X'08' State error **Byte 10** X'xx' xx=state **Byte 9** X'0C' SYNCLEVEL cannot be supported X'10' LL count error **Byte 10** X'xx' xx=remaining LL value **Byte 9** X'14' Invalid request X'18' TPN SEND check failed X'04' NOTALLOC error X'05' LENGERR error	

Figure 17 (Part 48 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FC' DFHZCP	**Byte 1** (continued) X'4B' DFHZARM, LU6.2 migration		
	Byte 2, bits 4-7 X'3' Entry	**Byte 8** X'01' SEND X'02' WAIT X'03' RECEIVE X'04' SIGNAL X'05' FLUSH X'06' FREE	Addr(TCTTE)
	Byte 1 X'4C' DFHZRVL, LU6.2 receive and X'4E' DFHZSDL, LU6.2 send		
	Byte 2, bits 4-7 X'3' Entry	**Byte 8** X'01' FMH to be sent X'02' DR1 to be sent X'04' DR2 to be sent X'08' CD received X'10' CD to be sent X'20' CEB received X'40' CEB to be sent X'80' Partial LL count set **Byte 9** X'01' Implicit send X'02' LL set by caller X'04' DFHZRVL recalled by DFHZRLX X'08' Buffer type RECEIVE **Byte 10** X'01' Send with ER1 X'02' Send with RQD1 X'04' Send with RQD2 X'08' Attach FMH generated X'10' Response to be sent X'80' LUSTAT for null RU **Byte 11** X'20' -ER* received	Addr(TCTTE)

Figure 17 (Part 49 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FC' DFHZCP	**Byte 1** (continued) X'50' DFHZERH, LU6.2 error recovery		
	Byte 2, bits 4-7 X'3' Entry	**Byte 8** X'06' ABEND X'09' ISSUE ERROR X'19' BACKOUT X'20' FMH received X'21' Neg response received **Byte 9** X'80' User call **Byte 10** X'20' LUCMSGNO valid X'40' LUCSENSE valid X'80' LUCAMSG, LUCLMSG valid **Byte 11** X'80' System call	Addr(TCTTE)
	Byte 1 X'51' DFHZLUS, LU6.2 services		
	Byte 2, bits 4-7 X'3' Entry	**Byte 8** X'01' Initialize change session X'02' Change session X'03' Receive change session X'04' Shutdown X'05' Resynchronize **Byte 9** X'20' Immediate shutdown X'40' Released issued X'80' Acquired issued	System name
	Byte 1 X'52' DFHZBKT, LU6.2 bracket state machine	The following states correspond to byte TCTEBKTS **Byte 8** X'00' SET X'80' CHECK **Byte 9** Request type X'01' BB_SEND X'02' CEB, RQ*1_SEND X'03' CEB, RQD2_SEND X'04' DR2_RCV X'05' -RSP_RECEIVE X'06' BB_RECEIVE X'07' CEB, RQ*1_RECEIVE X'08' CEB, RQD2_RECEIVE X'09' DR2_SEND X'0A' -RSP_SEND X'0B' SESS_CLOSE **Byte 10** Current state X'01' BETWEEN_BKTS X'02' IN_BRACKET X'03' IN_BKT_TERM_SEND X'04' IN_BKT_TERM_RCV	Addr(TCTTE)

Figure 17 (Part 50 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FC' DFHZCP	**Byte 1** (continued) X'53' DFHZCNT, LU6.2 contention state machine	The following states correspond to byte TCTECNTS	
		Byte 8 X'00' SET X'80' CHECK	Addr(TCTTE)
		Byte 9 Request type X'01' LUS_CON_WIN X'02' LUS_CON_LOSE X'03' LUS_CON_RESET X'04' BIND_CON_WIN X'05' BIND_CON_LOSE X'06' UNBIND X'07' LOCAL_ALLOCATE X'08' DETACH X'09' BB_RCV_+RSP X'0A' BB_RCV_-R(0814) X'0B' BB_RCV_-R(0813) X'0C' -R(0819, RTR)_ RCV X'0D' -R(0882, RTR)_ RCV X'0E' +R(RTR)_RECEIVE X'0F' +R(BB)_RECEIVE X'10' RTR_RECEIVE X'11' -R(BB,0814)_ RCV X'12' -R(BB,0813)_RCV X'13' -R(BB,088B)_RCV) X'14' RTR_RCV_-R(0819) X'15' DATA_INBOUND	
		Byte 10 Current state X'01' NOT_BOUND X'02' NOT_BOUND_CON_WIN X'03' NOT_BOUND_CON_LOSE X'04' BOUND_CON_WIN X'05' CON_WIN_ALLOCATE X'06' CON_WIN_RTR_SEND X'07' CON_WIN_RTR_PEND X'08' BOUND_CON_LOSE X'09' CON_LOSE_ALLOC X'0A' CON_LOSE_BIDDING X'0B' CON_LOSE_BB_ CROSS X'0C' CON_LOSE_RTR_ PEND X'0D' CON_LOSE_REBID_ PEND X'0E' CON_LOSE_AWAIT_ ACTIVITY X'0F' BOUND_CON_WIN_ BID_ACCEPTED	

Figure 17 (Part 51 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FC' DFHZCP	**Byte 1** (continued) X'54' DFHZCHS, LU6.2 chain send	The following states correspond to byte TCTECHSS	
		Byte 8	Addr(TCTTE)
		X'00' SET X'80' CHECK	
		Byte 9 Request type	
		X'01' BC_SEND X'02' EC_SEND_RQD X'03' EC_SEND_RQE, CD X'04' EC_SEND_RQD, CD X'05' EC_SEND_RQE, CEB X'06' +RSP (FMD/ LUSTAT)_RCV X'07' -RSP (FMD/ LUSTAT)_RCV X'08' BC_RCV X'09' EC_RCV_RQD X'0A' EC_RCV_RQE, CD X'0B' EC_RCV_RQD, CD X'0C' EC_RCV_RQE, CEB X'0D' +RSP (FMD/ LUSTAT)_SEND X'0E' -RSP (FMD/ LUSTAT)_SEND X'0F' BB_SEND_COMPLETE X'10' BB_ACCEPTED X'11' BRACKET_ENDED	
		Byte 10 Current state	
		X'01' BETWEEN_CHAINS_ SEND X'02' IN_CHAIN_SEND X'03' AWAIT_RESPONSE_ SEND X'04' PEND_RESPONSE_SEND X'05' NEGATIVE_RSP_RCVD X'06' BETWEEN_CHAINS_RCV X'07' IN_CHAIN_RCV X'08' PEND_RESPONSE_RCV X'09' AWAIT_RESPONSE_RCV X'0A' NEGATIVE_RSP_SEND	

Figure 17 (Part 52 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FC' DFHZCP	**Byte 1** (continued) X'56' DFHZUSR, LU6.2 conversation state machine	The following states correspond to byte TCTEUSRS	

Byte 8 Addr(TCTTE)
X'00' SET
X'80' CHECK
Byte 9 Request type
X'01' ALLOCATE_RESOURCE
X'02' SEND_PROCESS
X'03' SEND_DATA
X'04' SEND_DATA_INVITE
X'05' SEND_DATA_LAST
X'06' WAIT
X'07' ATTACH_INBOUND
X'08' RECEIVE_DATA
X'09' INVITE_RECEIVED
X'0A' FREE_RECEIVED
X'0B' FREE_RESOURCE
X'0C' SYSTEM_FREE
X'0D' SEND_SIGNAL
X'0E' ISSUE_ERROR
X'0F' ISSUE_ABEND
X'10' ERROR_RECEIVED
X'11' ABEND_RECEIVED
X'12' SEND_CONFIRM
X'13 SEND_CONFIRMATION
X'14 RECEIVE_CONFIRM
X'15' RECEIVE_CONFIRMATION
X'16' SEND_PREPARE
X'17' SEND_SPR
X'18' SEND_COMMIT
X'19' SEND_FORGET
X'1A' RECEIVE_PREPARE
X'1B' RECEIVE_SPR
X'1C' RECEIVE_COMMIT
X'1D' RECEIVE_FORGET
X'1E' SEND_BACKOUT
X'1F' RECEIVE_BACKOUT
X'20' CONVERSATIONAL_
FAILURE

Figure 17 (Part 53 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FC' DFHZCP	**Byte 1** X'56' DFHZUSR (continued)	**Byte 10** Current state X'01' NOT_ALLOCATED X'02' ALLOCATE_IN_PROGRESS X'03' ALLOCATED_SEND X'04' ALLOCATED_RECEIVE_PENDING X'05' ALLOCATED_RECEIVE X'06' FREE_PENDING_SEND X'07' FREE_REQUIRED X'08' IN_SYNCPT_SENDER_ONE_PHASE X'09' IN_SYNCPT_RCVER_ONE_PHASE X'0A' IN_SYNCPT_SENDER_TWO_PHASE X'0B' IN_SYNCPT_RCVER_TWO_PHASE X'0C' IN_SYNCPT_BACKOUT_SENDER X'0D' IN_SYNCPT_BACKOUT_RECEIVER X'0E' ALLOCATED_CONFIRM_SENDER X'0F' ALLOCATED_CONFIRM_RECEIVER	
	X'57' SNA-ASCII translation		
	X'58' DFHZEV1 encryption validation 1	Reserved	Format B1
	X'59' DFHZEV2 encryption validation 2	Reserved	Format B1
	X'5A' DFHZATD install entry	Reserved	Reserved
	X'5B' DFHZATD install exit	TERMIDNT	Addr(TCTTE) (0 if failure)
	X'5C' DFHZATD delete entry	TERMIDNT	Addr(TCTTE)
|	X'5D' DFHZATD delete exit	Reserved	0 (Addr(TCTTE) if failure)
|	X'80' Response logger	Format A1	Format B1
	X'82' NEP	Format A1	Format B1
	X'83' Second NACP	Format A4	Format B1
	X'84' Resync system task	Format A1	Format B1
	X'85' Reserved		
	X'86' Good morning	Format A1	Format B1

Figure 17 (Part 54 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FC' DFHZCP	**Byte 1** (continued)		
	X'C0' DFHZCQ00 request router	Reserved	Reserved
	X'C1' DFHZCQIN initialize	Reserved	Reserved
	X'C2' DFHZBAN bind analysis	Reserved	Reserved
	X'C3' DFHZCQCH change	Reserved	Reserved
	X'C4' DFHZCQDL delete	Reserved	Reserved
	X'C5' DFHZCQIT install TCTTE	Reserved	Reserved
	X'C6' DFHZCQRC recover	Reserved	Reserved
	X'C7' DFHZCQRS restore	Reserved	Reserved
	X'C8' DFHZCQIQ inquire	Reserved	Reserved
	X'C9' DFHZCQIS install	Reserved	Reserved
	X'CA' DFHTRZCP build terminal BPS	Reserved	Reserved
	X'CB' DFHTRZXP build connection BPS	Reserved	Reserved
	X'CC' DFHTRZIP build sessions BPS	Reserved	Reserved
	X'CD' DFHTRZYP build type BPS	Reserved	Reserved
	X'CE' DFHTRZPP build pool BPS	Reserved	Reserved
	X'CF' DFHTRZZP merge terminal and type BPS	Reserved	Reserved
	X'D7' DFHZXRE0 Valid on the return/exit trace	**Byte 8** Pass-number **Byte 9** Number of unbound entries **Byte 10** Number of standby-bound entries **Byte 11** Number with TCTEXRE set in error	Address of last TCTTE processed if error
	X'D8' DFHZXQO exit trace	**Byte 8** XQO-REQCODE 'I' Initialize 'A' Add action 'P' Post 'D' Drain **Bytes 10 to 11** XQO-RESPONSE 0 Complete 1 Scheduled 3 Queued 4 GMU 8 Invreq	A(action)

Figure 17 (Part 55 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
	X'D9' DFHXST exit trace	**Byte 8** X'00' BIND complete in active	TCTTE address
		X'01' Logon data free in active	TCTTE address
		X'02' UNBIND complete in active	TCTTE address
		X'03' BOUND catch-up in active	TCTTE address
		X'05' UNBOUND catch-up in active	TCTTE address
		X'FC' Set logon start and modify USERVAR	
		X'FD' Drain session tracking and modify USERVAR	Reserved
		X'FE' REQCODE session tracking in alternate	XTR-XT-SESS-NAME netname
		Byte 9 RESPCODE X'0' Complete X'1' Scheduled X'4' Error	
		Byte 10 XTR-ST-REQUEST X'1' BIND X'2' Free LOGON data X'3' UNBIND	
		Byte 11 XTR-ST-FLAGS-1 1... .... XRF-capable	
		X'FF' Action complete in alternate	TCTTE address
	X'DA' DFHZXTR Put/get header	**Bytes 8-9** X'0000' Normal X'0401' XRF inactive X'0402' End of data X'0403' Backup signed off X'0801' Invalid request X'0802' Service closed X'0803' Task canceled X'0804' Length error X'0805' Overlap X'0806' No destination X'0807' Queue busy X'0808' Program check X'0809' Abend X'080A' I/O error X'080B' Format error X'080C' Sequence number error X'080D' System not active **Bytes 10-11** Data length	**Bytes 12-13** Record ID **Byte 14** Type X'C3' Contents X'E2' Status **Byte 15** Request ID X'01' Put message X'02' Get message X'03' Put request X'04' Put response

Figure 17 (Part 56 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
	X'DB' DFHZXTR Put/get key	**Byte 8** Key length **Bytes 9-11** First 3 bytes of key	**Bytes 12-15** Next 4 bytes of key
	X'DC' DFHZXTR Put/get data	**Bytes 8-9** Data length **Bytes 10-11** First 2 bytes of data	**Bytes 12-15** Next 4 bytes of data
	X'DE' DFHZXQO diagnostic	**Byte 8** 'A' **Byte 9** Action element address	0 Action outstanding 1 Action processed
	X'E1' DFHTBSB(P) build (See note 5 on page 110)	Reserved	Reserved
	X'E2' DFHTBSSP sync point process	Reserved	Reserved
	X'E3' DFHTBSD(P) destroy (See note 5 on page 110)	Reserved	Reserved
	X'E4' DFHTBSR(P) recover (See note 5 on page 110)	Reserved	Reserved
	X'E5' DFHTBSC(P) change (See note 5 on page 110)	Reserved	Reserved
	X'E6' DFHTBSM add entry to msg-set	Reserved	Reserved
	X'E7' DFHTBSQ(P) inquire (See note 5 on page 110)	Reserved	Reserved
	X'E8' DFHTBSL(P) catalog (See note 5 on page 110)	Reserved	Reserved
	X'EE' Exit trace entry	Format A7	Format B1

Figure 17 (Part 57 of 58). Trace table entries

ID (Byte 0)	REQD (Bytes 1 and 2)	FIELD A (Bytes 8 through 11)	FIELD B (Bytes 12 through 15)
X'FD' DFHTRP internal	**Byte 2, bits 4-7** X'1' FE type trace **Byte 1** X'99' Entry made on behalf of global trap/trace exit	Data supplied by exit	Data supplied by exit
	Byte 2, bits 4-7 X'4' System trace **Byte 1** X'01' Repeat entry; bytes 24-27 contain the no. of times the preceding entry is repeated (packed decimal)	Time at which first repeat entry made (CSACSCC)	Time at which last entry made
X'FE' (Trace turned on)	**Byte 2, bits 4-7** X'D' On/off entry **Byte 1** Reserved **Byte 2, bits 0-3**		
	X'0'	) Images before trace request) Byte 8: CSATRMF1) Byte 9: CSATRMF2) Byte 10: CSATRMF3) Byte 11: CSATRMF4	Images before trace request Byte 12: TCATRTR Byte 13: TCATRID Byte 14: TCATRMF Byte 15: TCATRID1
	X'1'	) Images before trace request) Byte 8: CSATRMF1) Byte 9: CSATRMF2) Byte 10: CSATRMF3) Byte 11: CSATRMF4	Images after trace request Byte 12: CSATRMF1 Byte 13: CSATRMF2 Byte 14: CSATRMF3 Byte 15: CSATRMF4
X'FF' (Trace turned off)	Bytes 1 through 15 as for X'FE'		

Notes:

1. *The X'CF' trace occurs when an application program issues an EXEC CICS SEND PAGE with RELEASE before the BMS module issues a DFHPC TYPE=RETURN macro to exit from the calling program.*

2. *When byte 10 contains X'01', bytes 12 and 13 contain the current page number, TCAMSPGN, and bytes 14 and 15 contain the overflow control number, TCAMSOCN.*

3. *The X'CD' trace occurs when BMS receives an IDERROR from temporary storage.*

4. *Trace entries for byte 14 greater than X'80' are CICS internal trace entries.*

5. *These trace entries are each provided by two modules, for example DFHTBSB and DFHTBSBP.*

Figure 17 (Part 58 of 58). Trace table entries

ID	Program	Request Type(s)	Resource
X'C4'	DFHWMS	Entry	Bytes 16-19: instance number (WMSINSTN) Bytes 20-23: version number (WMSVERN)/ queue name (WMSQNAME)
X'D5'	User exit interface	All requests	Exit program name
X'D6'	Allocation program	X'17' allocate	Modename
X'E3'	System spooling interface	Where appropriate	Data set ID token
X'E5'	Security program	X'04',X'08' sign-on X'0C' check auth	User ID Resource name
X'EA'	Table management program	Where appropriate	Resource name
X'EE'	VTAM I/O trace	All requests	VTAM communications ID
X'F0'	Task control	X'0E',X'0F' CANCEL	Facility ID (if any) || abend code
		X'28' LOCATE X'29' LOCATE-REGION	Transid Sysid || transid
X'F2'	Program control	Where appropriate	Program name
X'F3'	Interval control	Where appropriate X'90' RETRY X'F0' CANCEL	Transid Request ID Request ID
X'F5'	File control	Where appropriate	Data set name
X'F6'	Transient data control	Where appropriate	Queue name
X'F8'	CICS-DL/I interface	'PCB' function Data-base access call	PSB name DBD name
X'FC'	VTAM terminal control	X'51' LU6.2 services X'5A' install entry	Modename Netname of LU requesting logon

Figure 18. Contents of the trace entry resource field

FORMAT A1

	BYTE 8	BYTE 9	BYTE 10	BYTE 11
....1	Task created by avail (ATI)	Resynchronization required	Definite response send in progress	Between brackets
.... ..1.	Input journal required	Previous session ABEND	QRI type response	Begin bracket receive pending
.... .1..	Chain assembly in progress	Emergency restart	Log first input after sync	Begin bracket received
.... 1...	Resynch/recovery in progress	Bracket protocol required	Pending resp. exception	End bracket sent
.... 0...			Pending resp. definite	
...1	NACP in progress	Overlength data	Awaiting positive response	Begin bracket sent
..1.	ATI bid in progress	Mode (CS=X'20', CA=not X'20')		BB EB sent state
.1..	Data in progress	CICS quiesced by node	Pending FME response	Begin bracket pending state
1...	Command in progress	Node quiesced by CICS	Pending RRN response	In bracket state

FORMAT B1

BYTE 12 (Hex)	BYTE 13	BYTE 14	BYTE 15
00 No exit since last trace entry	)		
17 Receive specific exit	)		
19 Send DFSYN data exit	)		
21 DFASY exit	)		
23 CLSDST exit	)		
25 LERAD exit	)		
26 LOGON exit	)		
28 LOSTERM exit	)		
2A OPNDST exit	)		
2E Release request exit	)----------Bytes 13 through 15 contain the address of the TCTTE----------		
2F Network service error exit	)		
31 Send DFASY exit	)		
32 SESSIONC input exit	)		
36 SESSIONC exit	)		
3A Send DFSYN com'd exit	)		
3B SYNAD exit	)		
3C Turnaround exit	)		
3D TPEND exit	)		
38 SIMLOGON exit	)		
4D LU6.2 receive exit	)		
4F LU6.2 send exit	)		

FORMAT A2

BYTE 8 (Hex)	BYTE 9	BYTE 10	BYTE 11
00 I/O request & Format B2	....1 Write	Deferred READ	Pseudobinary mode
01 SIGNAL	.... ..1. Converse	INVITE	Notranslate
40 Program	.... .1.. Synchronize	DEFER	Print
80 EODS	.... 1... Disconnect	Transparent TIOA	Copy
	...1 .... Read	STRFIELD	Read lock
	..1. .... Line addressing req.	CONDITIONAL	Write lock
	.1.. .... Save terminal storage	TCTTE addr	Erase all unprotected
	1... .... Erase	CCOMPL=NO	Read buffer

Figure 19 (Part 1 of 5). Formats of DFHZCP trace entry fields A1-A11, B1, and B2

FORMAT B2 (used only if byte 8 = X'00')

BYTE 12 (Hex)	BYTE 13	BYTE 14	BYTE 15
01 Wait request			
02 Override sync request			
04 Override async request			
08 Last write from task	---------- Bytes 13 through 15 contain the address of the TCTTE ----------		
10 FMH provided with data			
20 LDC mnemonic present			
40 Wait on inbound SIGNAL			
80 FORCE=YES			

FORMAT A3

BYTE 8, 9, 10, OR 11 (Hex)	BYTE 8, 9, 10, OR 11 (Hex)	BYTE 8, 9, 10, OR 11 (Hex)
10 NODE not activated	3A Hierarchical reset occurred	shutdown
11 Session bind failure	3C Inquire USERVAR error	63 VTAM orderly shutdown
13 VTAM halting	3D XRF recovery message	64 VTAM abend shutdown
14 Logic error detected	3E XRF session state error	65 Invalid response detected
15 Permanent channel error NCP shutdown	3F XRF session restarted	66 SIGNAL command received
18 Logic error, no feedback	40 Invalid copy request	67 VTAM immediate shutdown
19 Terminate self from NODE	41 TOLTEP request	68 Error in DFHZEMW
1A Apparent VTAM error	42 Printer unavailable	69 Out of service after SDT
1D VTAM inactive to application or TCB abend	43 No copy support	70 Clear received
20 VTAM inactive	44 Exception response received	73 Read only terminal
21 Attach FMH error	45 Max chain size exceeded	74 Unsupported command
22 Attach FMH not found	46 Incompl outbound chain w/read	75 Unsupported command
23 Bracket FSM error	47 Good morning message reqd	78 Response requested error
24 Chain FSM error	48 Session just opened	79 Device does not support ATI
25 Contention FSM error	49 Session just closed	80 Temp VTAM storage problem
26 Invalid request to DFHZSDL	4A VTAM ACB opened dynamically	81 Exception response rec'd
27 No BUFFLST for DFHZSDL	4B Print request failed	82 Unknown command received
28 Invalid request to DFHZRVL	4C Data stream protocol error	83 ATI no longer requested
29 DFHZRVL buffer too small	4D VTAM release level check	84 Invalid normal response to bid
2A Non-process level exception response received	4E Unbind - protocol error	88 Invalid CID detected
2B Bid received with data, no OIC	4F Close all LU6.2 links	89 Unknown symbolic name
2C Bid received with data, cannot reject	50 No action after commit RECV	90 ZCP logic error detected
2D Record length > buffer length	51 BIND name not matching TCTTE	91 Invalid RTYPE specified
2E EOC received and invalid DFC	52 BIND parameter unacceptable	92 Send DFSYN req incomplete
2F +ve response rejected by VTAM	53 BIND out of service	93 CA mode and task attached
30 BIS received with invalid DFC	54 Unbind recd without BIS/BIR	94 Input status error
31 Inexplicable response recvd	55 EMW cannot send msg in BETB	95 TIOA length incorrect
32 Session closing via BIS	56 XRF-VTAM shutdown	96 RPL missing (receive specific)
33 Invalid indicators received	57 Release issued by MT op	97 TIOA missing (receive specific)
34 Invalid data received	58 No action when commit RECV	98 No task on terminal to resume
35 Read timeout occurred	59 Record outstanding at shutdown	99 No TIOA available for send
36 Standby BIND too late	5A SBI received causing CLSDST	9B Cancel received cancel mode
37 Active session got UNBIND(13)	5B Network error one	9C Outbound chain not completed
38 XRF R(switch) area too small	5C Network error two	9D Unexpected response received
39 Standby OPNDST not issued	5D Connect not terminal or REJ	9E Device end from 3270
	5E BIND response unacceptable	9F Access method insufficient for LU6.2
	5F Bad session qualifier pair	
	60 Unsupported command detected	
	61 LU status condition	
	62 Task cancel due to immed	

Figure 19 (Part 2 of 5). Formats of DFHZCP trace entry fields A1-A11, B1, and B2

FORMAT A3 (CONTINUED)

BYTE 8, 9, 10, OR 11 (Hex)	BYTE 8, 9, 10, OR 11 (Hex)	BYTE 8, 9, 10, OR 11 (Hex)
A0 ISC modules not loaded	C1 Unknown error code from VTAM	a definite response send
A1 Invalid read request	C4 Dummy TCTTE ID	DD Negative response received to an exceptional response send
A3 Bracket state error	C5 NACP restarted	DE Failed to get into send mode
A7 Node bracket protocol error	C6 LU successfully passed	DF Autoinstall O/S getmain failed
A8 FMH lngth exceeds data lngth	C7 CLSDST pass procedure error	E0 Unknown symbol name
A9 Receive specific in receive any	C8 LU inhibited for sessions	E1 Receive any problem
AA Outboard chaining not supported	C9 CLSDST pass not authorized	E2 CLSDST failed in LGX
AB LU error	CB Terminal released	E3 CNCL received in 'CS' mode
AC VR deactivated	CC Clear was issued	E4 Segmenting error
AD Unrecoverable LU fail	CD Exception in chain	E5 Maximum RU size exceeded
AE Recoverable LU fail	CE Hold	E6 Logic error dummy TCTTE
AF Cleanup received	CF VR_INOP	E7 Inbound chain purged
B1 RPL is active	D0 VTAM recovered node	E8 Negative response to BIND
B2 Invalid command setting	D1 Node unrecoverable	E9 STSN expected, not recd
B3 No RPL exists for operation	D2 Node recovery in progress	EA STSN logic error
B5 Unknown command	D3 TCTTE pending delete error	EB STSN resync mismatch
B6 No seed received	D4 Request recovery received	EC STSN resync mismatch
B7 Inconsistent attach security	D5 Unsupported command received	ED STSN test negative
B8 Encryption validation failed	D6 CICS released by node	EF STSN test negative
B9 No FMH12 received	D7 CICS quiesced by node	F0 User error
BA Negative response	D8 Multiple errors	F1 Unknown modename
BB Unknown command in RPL	D9 Exception req received (not VTAM 3270)	F2 Exception req received (VTAM 3270)
BC Invalid inp after inp req	DA Req. shutdown during a task	
BD Getmain for DES failed	DB Autoinstall max WE reached	
	DC Negative response received to	

FORMAT A4

BYTE 8	BYTE 9	BYTE 10	BYTE 11
....1		Terminate session	No level 1 message
.... ..1.	SIMLOGON required	Keep node out of service	No sense code message
.... .1..		Send negative response	No messages
.... 1... Print BIND area	Send EM message	Normal CLSDST	No security message
...1 .... Print TIOA			No action message
..1. .... Print TCTTE	ABEND task	Normal CLSDST (no reset)	
.1.. .... Print RPL	ABORT VTAM receive	NOINTLOG req	
1... .... Print action flags	ABORT VTAM send	INTLOG req	

Figure 19 (Part 3 of 5). Formats of DFHZCP trace entry fields A1-A11, B1, and B2

FORMAT A5

	BYTE 8	BYTE 9	BYTE 10	BYTE 11
....1	Task created by avail (ATI)	Chain purged indicator	Definite response send in progress	Between brackets
.... ..1.	Input journal required	Msg writer GETMAIN	Bid to be retried	Begin bracket receive pending
.... .1..		Original mode was 'CA'	Log first input after sync	Begin bracket received
.... 1...	Resynch/recovery in progress	First RU of inbound chain purged	Pending response exception	End bracket sent
...1	NACP in progress	Send bid request	Awaiting positive response	Begin bracket sent
..1.	ATI bid in progress	Send exception response request	Deferred Write pending	RTR pending state
.1..	Data in progress	Send message request	Pending FME response	Begin bracket pending state
1...	Command in progress	Purge request for inbound chain	Pending RRN response	In bracket state

FORMAT A6

	BYTE 8 (TCTEVTPS)	BYTE 9 (TCTEVST2)	BYTE 10 (TCTEVIPS)	BYTE 11 (TCTEVBPS)
....1	Task created by avail (ATI)		Definite response send in progress	
.... ..1.	Input journal required		Bid to be retried	
.... .1..		EODS indication received	Log first input after sync	
.... 1...	Resynch/recovery in progress	EOC indication received	Pending response exception	End bracket sent
...1	NACP in progress	GETMAIN request for TIOA	Awaiting positive response	Begin bracket sent
..1.	ATI bid in progress		Deferred Write pending	RTR pending state
.1..	Data in progress	Deblock in progress	Pending FME response	Begin bracket pending state
1...	Command in progress		Pending RRN response	In bracket state

FORMAT A7

BYTE 8	BYTE 9	BYTE 10	BYTE 11
Same as byte 12 of format B1 (Earliest exit)	Same as byte 12 of format B1	Same as byte 12 of format B1	Same as byte 12 of format B1 (Latest exit)

FORMAT A8

BYTES 8 & 9	BYTE 10	BYTE 11
Current/predicted sequence number For ZRAC & ZRVS, field TCTESQIP+1 For ZSDR, field TCTESQIP For ZSDS, field TCTESQOP+1	Same as byte 10 of format A1	Same as byte 11 of format A1

Figure 19 (Part 4 of 5). Formats of DFHZCP trace entry fields A1-A11, B1, and B2

FORMAT A9

BYTE 8	BYTE 9	BYTES 10 & 11
....1 VTAM orderly close	Non-VTAM quiesce	Reserved
.... ..1. VTAM immediate close	ZSHU first time	
.... .1.. VTAM abended	Use RRN outbound	
.... 1... VTAM quiesced	Use FME outbound	
...1 .... ACB opened		
..1. .... VTAM generated		
.1.. .... Dynamic OPEN active		
1... .... TPEND exit invoked		

FORMAT A10

BYTE 8	BYTE 9	BYTE 10	BYTE 11
....1 Log response	ZDET detach	ZSKR command response	
.... ..1. Commit queue	Exit added	ZATD autodelete	
.... .1.. ZRVS receive	Delay	ZNAC	
.... 1... Resume	ZRST resetsr	ZRSY resync	
...1 .... Asynchronous exit	ZSDR send response	ZSIM simlogon	
..1. .... ZATT attach	ZSES sessionc	ZATI ATI	ZRVL receive
.1.. .... ZFRE freemain	ZSDA send command	ZCLS clsdst	ZSDL send
1... .... ZGET getmain	ZSDS send	ZOPN opndst	ZTRA trace

FORMAT A11

BYTE 8	BYTE 9	BYTE 10	BYTE 11
01 Locate request	0000 0000 Address request	Not used	Not used
	....1 ID request		
	.... ..1. Next request		
	.... ..11 Unique request		
	.... .1.. First request		
	.... .1.1 NETNAME request		
02 ATI request	Not used	Not used	Not used
04 Status request	....1 Not used	Out of service	Acquire
	.... ..1.	In service	Resync override
	.... .1..	Transaction	Release
	.... 1... System entries	Auto initiate	
	...1 .... ALL	No poll	
	..1.	Input	
	.1.. .... Global	Auto page	
	1... .... Remote	Page	
08 LDC request	Contains the LDC	Not used	Not used
10 Detach request	Not used	Not used	Not used
20 Sync point request	Not used	Not used	Not used
80 Remote	)... ...1 Out of service	Acquire	
C0 Global	)... ..1. In service	Cold	
	.... .1.. Transaction	Release	
	.... 1... Transceive		
	...1 .... Receive nopoll		
	..1. .... Input autopage		
	.1.. .... Autopage		
	1... .... Page		

Figure 19 (Part 5 of 5). Formats of DFHZCP trace entry fields A1-A11, B1, and B2

Command Codes (Entry & Exit Traces)

Field EIBFN	Command Function
EI Requests	
02 02	ADDRESS
02 04	HANDLE CONDITION
02 06	HANDLE AID
02 08	ASSIGN
02 0A	IGNORE CONDITION
02 0C	PUSH
02 0E	POP
Terminal Control Requests	
04 02	RECEIVE
04 04	SEND
04 06	CONVERSE
04 08	ISSUE EODS
04 0A	ISSUE COPY
04 0C	WAIT TERMINAL
04 0E	ISSUE LOAD
04 10	WAIT SIGNAL
04 12	ISSUE RESET
04 14	ISSUE DISCONNECT
04 16	ISSUE ENDOUTPUT
04 18	ISSUE ERASEAUP
04 1A	ISSUE ENDFILE
04 1C	ISSUE PRINT
04 1E	ISSUE SIGNAL
04 20	ALLOCATE
04 22	FREE
04 24	POINT
04 26	BUILD ATTACH
04 28	EXTRACT ATTACH
04 2A	EXTRACT TCT
04 2C	WAIT CONVID
04 2E	EXTRACT PROCESS
04 30	ISSUE ABEND
04 32	CONNECT PROCESS
04 34	ISSUE CONFIRMATION
04 36	ISSUE ERROR
04 38	ISSUE PREPARE
04 3A	ISSUE PASS
04 3C	EXTRACT LOGONMSG

Response Codes (Exit Trace Only)

Field EIBFN	Field EIBRCODE Bytes 8	9	10	11	Meaning of response code
EI Responses					
02	..	E0	..	..	.. INVREQ
Terminal Control Responses					
04	..	04	..	..	.. EOF
04	..	10	..	..	.. EODS
04	..	C1	..	..	.. EOF
04	..	C2	..	..	.. ENDINPT
04	..	D0	..	..	.. SYSIDERR
04	..	D0	08	..	.. SYSID out of service
04	..	D0	0C	..	.. SYSID definition error
04	..	D0	0C	00	.. SYSID name not found
04	..	D0	0C	04	.. SYSID name not that of TCTSE
04	..	D0	0C	08	.. MODENAME not found
04	..	D0	0C	0C	.. Profile not found
04	..	D2	..	..	.. SESSIONERR
04	..	D3	..	..	.. SYSBUSY
04	..	D4	..	..	.. SESSBUSY
04	..	D5	..	..	.. NOTALLOC
04	..	E0	..	..	.. INVREQ
04	..	E0	00	..	04 ALLOCATE-TCTTE already allocated
04	..	E0	00	..	08 FREE-TCTTE in wrong state
04	..	E0	00	..	10 EXTRACT ATTACH invalid data
04	..	E0	00	..	18 EXTRACT TCT invalid NETNAME
04	..	E0	00	..	1C Command/TERMTYPE conflict
04	..	E0	00	..	20 Command/CONVTYPE conflict
04	..	E0	00	..	28 ISSUE PASS GETMAIN failure

Figure 20 (Part 1 of 7). EXEC interface command and response codes

Command Codes
(Entry & Exit Traces)

Field EIBFN	Command Function
File Control Requests	
06 02	READ
06 04	WRITE
06 06	REWRITE
06 08	DELETE
06 0A	UNLOCK
06 0C	STARTBR
06 0E	READNEXT
06 10	READPREV
06 12	ENDBR
06 14	RESETBR

Response Codes
(Exit Trace Only)

Field EIBFN	Field EIBRCODE Bytes 8	9	10	11	Meaning of response code
Terminal Control Responses (continued)					
04	..	E1	..	..	.. LENGERR
04	..	E1	00	..	.. Input data too long
04	..	E1	04	..	.. Output length error
04	..	E1	08	..	.. Input length error
04	..	E1	0C	..	.. ISSUE PASS length error
04	..	E3	..	..	.. WRBRK
04	..	E4	..	..	.. RDATT
04	..	E5	..	..	.. SIGNAL
04	..	E6	..	..	.. TERMIDERR
04	..	E7	..	..	.. NOPASSBKRD
04	..	E8	..	..	.. NOPASSBKWR
04	..	EA	..	..	.. IGREQCD
04	..	EB	..	..	.. CBIDERR
04	..	F1	..	..	.. TERMERR
04	..	..	20	..	.. EOC
04	..	..	40	..	.. INBFMH
04	..	..	..	..	F6 NOSTART
04	..	..	..	..	F7 NONVAL
File Control Responses					
06	..	01	..	..	.. DSIDERR
06	..	02	xx	xx	.. ILLOGIC (See note 1 on page 123)
06	..	08	..	..	.. INVREQ
06	..	0C	..	..	.. NOTOPEN
06	..	0D	..	..	.. DISABLED
06	..	0F	..	..	.. ENDFILE
06	..	80	xx	xx	xx IOERR (See note 1 on page 123)
06	..	81	..	..	.. NOTFND
06	..	82	..	..	.. DUPREC
06	..	83	..	..	.. NOSPACE
06	..	84	..	..	.. DUPKEY
06	..	D0	..	..	.. SYSIDERR
06	..	D1	..	..	.. ISCINVREQ
06	..	D6	..	..	.. NOTAUTH
06	..	E1	..	..	.. LENGERR

Figure 20 (Part 2 of 7). EXEC interface command and response codes

Command Codes (Entry & Exit Traces)

Field EIBFN	Command Function
Transient Data Requests	
08 02	WRITEQ TD
08 04	READQ TD
08 06	DELETEQ TD
Temporary Storage Requests	
0A 02	WRITEQ TS
0A 04	READQ TS
0A 06	DELETEQ TS
Storage Control Requests	
0C 02	GETMAIN
0C 04	FREEMAIN
Program Control Requests	
0E 02	LINK
0E 04	XCTL
0E 06	LOAD
0E 08	RETURN
0E 0A	RELEASE
0E 0C	ABEND
0E 0E	HANDLE ABEND
Interval Control Requests	
10 02	ASKTIME
10 04	DELAY
10 06	POST
10 08	START
10 0A	RETRIEVE
10 0C	CANCEL

Response Codes (Exit Trace Only)

Field EIBFN	Field EIBRCODE Bytes 8	9	10	11	Meaning of response code
Transient Data Responses					
08	..	01	..	..	QZERO
08	..	02	..	..	QIDERR
08	..	04	..	..	IOERR
08	..	08	..	..	NOTOPEN
08	..	10	..	..	NOSPACE
08	..	C0	..	..	QBUSY
08	..	D0	..	..	SYSIDERR
08	..	D1	..	..	ISCINVREQ
08	..	D6	..	..	NOTAUTH
08	..	E1	..	..	LENGERR
Temporary Storage Responses					
0A	..	01	..	..	ITEMERR
0A	..	02	..	..	QIDERR
0A	..	04	..	..	IOERR
0A	..	08	..	..	NOSPACE
0A	..	20	..	..	INVREQ
0A	..	D0	..	..	SYSIDERR
0A	..	D1	..	..	ISCINVREQ
0A	..	D6	..	..	NOTAUTH
0A	..	E1	..	..	LENGERR
Storage Control Responses					
0C	..	E1	..	..	LENGERR
0C	..	E2	..	..	NOSTG
Program Control Responses					
0E	..	01	..	..	PGMIDERR
0E	..	D6	..	..	NOTAUTH
0E	..	E0	..	..	INVREQ
Interval Control Responses					
10	..	01	..	..	ENDDATA
10	..	04	..	..	IOERR
10	..	11	..	..	TRANSIDERR
10	..	12	..	..	TERMIDERR
10	..	14	..	..	INVTSREQ
10	..	20	..	..	EXPIRED
10	..	81	..	..	NOTFND
10	..	D0	..	..	SYSIDERR
10	..	D1	..	..	ISCINVREQ
10	..	D6	..	..	NOTAUTH
10	..	E1	..	..	LENGERR
10	..	E9	..	..	ENVDEFERR
10	..	FF	..	..	INVREQ

Figure 20 (Part 3 of 7). EXEC interface command and response codes

Command Codes (Entry & Exit Traces)		Response Codes (Exit Trace Only)					
Field EIBFN	Command Function	Field EIBFN	Field EIBRCODE Bytes 8	9	10	11	Meaning of response code
Task Control Requests		**Task Control Responses**					
12 02	WAIT EVENT	12 ..	32	..	..	..	ENQBUSY
12 04	ENQ	12 ..	E0	..	..	..	INVREQ
12 06	DEQ						
12 08	SUSPEND						
Journal Control Requests		**Journal Control Responses**					
14 02	JOURNAL	14 ..	01	..	..	..	JIDERR
14 04	WAIT JOURNAL	14 ..	02	..	..	..	INVREQ
		14 ..	05	..	..	..	NOTOPEN
		14 ..	06	..	..	..	LENGERR
		14 ..	07	..	..	..	IOERR
		14 ..	09	..	..	..	NOJBUFSP
		14 ..	D6	..	..	..	NOTAUTH
Sync Point Requests		**Sync Point Responses**					
16 02	SYNCPOINT	16 ..	01	..	..	..	ROLLEDBACK
16 04	RESYNC						
BMS Requests		**BMS Responses**					
18 02	RECEIVE MAP	18 ..	01	..	..	..	INVREQ
18 04	SEND MAP	18 ..	02	..	..	..	RETPAGE
18 06	SEND TEXT	18 ..	04	..	..	..	MAPFAIL
18 08	SEND PAGE	18 ..	08	..	..	xx	INVMPSZ
18 0A	PURGE MESSAGE						(See note 2 on page 123)
18 0C	ROUTE	18 ..	20	..	..	..	INVERRTERM
18 0E	RECEIVE PARTN	18 ..	40	..	..	..	RTESOME
18 10	SEND PARTNSET	18 ..	80	..	..	..	RTEFAIL
18 12	SEND CONTROL	18 ..	E1	..	..	..	LENGERR
		18 ..	E3	..	..	..	WRBRK
		18 ..	E4	..	..	..	RDATT
		18 ..	..	02	..	..	PARTNFAIL
		18 ..	..	04	..	..	INVPARTN
		18 ..	..	08	..	..	INVPARTNSET
		18 ..	..	10	..	..	INVLDC
		18 ..	..	20	..	..	UNEXPIN
		18 ..	..	40	..	..	IGREQCD
		18 ..	..	80	..	..	TSIOERR
		18 ..	..	..	01	..	OVERFLOW
		18 ..	..	..	04	..	EODS
		18 ..	..	..	08	..	EOC
		18 ..	..	..	10	..	IGREQID

Figure 20 (Part 4 of 7). EXEC interface command and response codes

Command Codes (Entry & Exit Traces)		Response Codes (Exit Trace Only)					
Field EIBFN	Command Function	Field EIBFN	Field EIBRCODE Bytes 8	9	10	11	Meaning of response code
Trace Requests		**Trace Responses**					
1A 02	TRACE ON/OFF	1A ..	E0	..	..	..	INVREQ
1A 04	ENTER						
Dump Control Requests							
1C 02	DUMP						
Data Interchange Requests		**Data Interchange Responses**					
1E 02	ISSUE ADD	1E ..	04	..	..	..	DSSTAT
1E 04	ISSUE ERASE	1E ..	08	..	..	..	FUNCERR
1E 06	ISSUE REPLACE	1E ..	0C	..	..	..	SELNERR
1E 08	ISSUE ABORT	1E ..	10	..	..	..	UNEXPIN
1E 0A	ISSUE QUERY	1E ..	E1	..	..	..	LENGERR
1E 0C	ISSUE END	1E ..	..	11	..	..	EODS
1E 0E	ISSUE RECEIVE	1E ..	..	2B	..	..	IGREQCD
1E 10	ISSUE NOTE	1E ..	..	..	20	..	EOC
1E 12	ISSUE WAIT						
1E 14	ISSUE SEND						
BIF Requests							
20 02	DEEDIT						
User Exit Management Requests		**User Exit Management Responses**					
22 02	ENABLE	22 ..	80	xx	xx	..	INVEXITREQ
22 04	DISABLE						(See note 3 on page 123)
22 06	EXTRACT EXIT						
ASKTIME ABSTIME and FORMATTIME Requests		**ASKTIME ABSTIME and FORMATTIME Responses**					
4A 02	ASKTIME ABSTIME	4A ..	..	..	..	01	ERROR
4A 04	FORMATTIME						
INQUIRE/SET DATASET Requests		**INQUIRE/SET DATASET Responses**					
4C 02	INQUIRE DATASET	4C ..	..	..	..	0C	DSIDERR
4C 04	SET DATASET	4C ..	..	..	..	10	INVREQ
		4C ..	..	..	..	11	IOERR
		4C ..	..	..	..	15	ILLOGIC
		4C ..	..	..	..	46	NOTAUTH
		4C ..	..	..	..	53	END
INQUIRE/SET PROGRAM Requests		**INQUIRE/SET PROGRAM Responses**					
4E 02	INQUIRE PROGRAM	4E ..	..	..	..	01	ERROR
4E 04	SET PROGRAM	4E ..	..	..	..	10	INVREQ
		4E ..	..	..	..	15	ILLOGIC
		4E ..	..	..	..	1B	PGMIDERR
		4E ..	..	..	..	46	NOTAUTH
		4E ..	..	..	..	53	END

Figure 20 (Part 5 of 7). EXEC interface command and response codes

Command Codes (Entry & Exit Traces)

Field EIBFN	Command Function
INQUIRE/SET TRANSACTION Requests	
50 02	INQUIRE TRANSACTION
50 04	SET TRANSACTION
INQUIRE/SET TERMINAL Requests	
52 02	INQUIRE TERMINAL
52 04	SET TERMINAL
52 06	INQUIRE NETNAME
INQUIRE/SET SYSTEM Requests	
54 02	INQUIRE SYSTEM
54 04	SET SYSTEM
System Spooling Interface Requests	
56 02	SPOOLOPEN
56 04	SPOOLREAD
56 06	SPOOLWRITE
56 10	SPOOLCLOSE
INQUIRE/SET CONNECTION Requests	
58 02	INQUIRE CONNECTION
58 04	SET CONNECTION
INQUIRE/SET MODENAME Requests	
5A 02	INQUIRE MODENAME
5A 04	SET MODENAME

Response Codes (Exit Trace Only)

Field EIBFN	Field EIBRCODE Bytes 8 9 10 11	Meaning of response code
INQUIRE/SET TRANSACTION Responses		
50	 10	INVREQ
50	 15	ILLOGIC
50	.. .. 1C	TRANSIDERR
50	 46	NOTAUTH
50	 53	END
INQUIRE/SET TERMINAL Responses		
52	 01	ERROR
52	.. .. 0B	TERMIDERR
52	 10	INVREQ
52	 15	ILLOGIC
52	 53	END
INQUIRE/SET SYSTEM Responses		
54	 10	INVREQ
System Spooling Interface Responses		
56	.. .. 0D	NOTFND
56	 10	INVREQ
56	 13	NOTOPEN
56	 14	ENDFILE
56	 15	ILLOGIC
56	 16	LENGERR
56	.. .. 2A	NOSTG
56	 46	NOTAUTH
56	 50	NOSPOOL
56	 55	ALLOCERR
56	 56	STRELERR
56	 57	OPENERR
56	 58	SPOLBUSY
56	 59	SPOLERR
56	.. .. 5A	NODEIDERR
INQUIRE/SET CONNECTION Responses		
58	 10	INVREQ
58	 15	ILLOGIC
58	 35	SYSIDERR
58	 53	END
INQUIRE/SET MODENAME Responses		
5A	 10	INVREQ
5A	 15	ILLOGIC
5A	 35	SYSIDERR
5A	 53	END

Figure 20 (Part 6 of 7). EXEC interface command and response codes

For the following commands, EIBFN is not set and the response is placed in the RETCODE area supplied by the application program.
For GDS IDs, see the *Format and Protocols Reference Manual: Architecture Logic for LUType 6.2*.

Command Codes			Response Codes	
GDS Requests			**GDS Responses**	
24	02	ALLOCATE	01	SYSIDERR error
24	04	ASSIGN	01 04 04	No session available and NOQUEUE specified
24	06	EXTRACT PROCESS		
24	08	FREE	01 04 08	MODENAME not known
24	0A	ISSUE ABEND	01 04 0C	MODENAME invalid
24	0C	CONNECT PROCESS	01 04 10	Task canceled or timed out during wait for allocation
24	0E	ISSUE CONFIRMATION		
24	10	ISSUE ERROR		
24	12	ISSUE SIGNAL	01 04 14	Mode group is out of service
24	14	RECEIVE		
24	16	SEND	01 08 ..	SYSID is out of service
24	18	WAIT	01 08 00	Local queuing was not attempted
24	1A	PREPARE		
			01 08 04	Local queuing did not succeed
			01 0C ..	SYSID is not known in TCT
			01 0C 00	SYSID name is not known
			01 0C 04	SYSID name is not that of an LU6.2 TCTSE
			01 0C 08	SYSID.MODENAME is not known
			02	SYSBUSY error
			03	INVREQ error
			03 00 ..	Session is not defined as LU6.2
			03 04 ..	Conversation level is wrong
			03 08 xx	State error xx = state
			03 0C ..	SYNCLEVEL cannot be supported
			03 10 xx	LLCOUNT error xx = remaining LL value
			03 14 ..	Invalid request
			03 18 ..	TPN SEND check failed
			04	NOTALLOC error
			05	LENGERR error

Notes:

1. *When either the IOERR or ILLOGIC condition exists during file control operations, further information is provided in bytes 9-12 of the trace entry (EIBRCODE bytes 1-4) as follows:*

   ```
   For BDAM (IOERR):                  Bytes 9-12  = BDAM response
   For VSAM (ILLOGIC and IOERR):      Byte  9     = VSAM return code and
                                      Byte  10    = VSAM error code
   ```

   Details of the response codes are given in the Data Management Macro Instructions *manual for BDAM, and in the* OS/VS VSAM Programmer's Guide *for VSAM.*

2. *When the INVMPSZ condition exists during BMS operation, byte 11 of the trace entry (EIBRCODE byte 3) contains the terminal code character.*

3. *Details of the contents of bytes 1-2 of the EIBRCODE (bytes 9-10 of the trace entry) are given in the description of the user exit interface, in the* Customization Guide.

Figure 20 (Part 7 of 7). EXEC interface command and response codes

Global Trap/Trace exit

The global trap/trace exit (DFHTRAP) is intended to be used only under the guidance of IBM Service personnel. It is designed so that a detailed diagnosis of a problem can be made without having to stop and restart CICS.

Typically, the global trap/trace exit is used to detect error situations that cannot be diagnosed by other methods. These could be intermittent problems that are difficult to reproduce.

DFHTRAP is an assembler language program, which can be invoked by the trace program (DFHTRP) when the trace facility is active. When the exit is activated, it is invoked each time the trace program makes an entry in the trace table. The trace entry can be system, user or FE type.

A skeleton version of DFHTRAP is supplied in both source and load-module forms. The source of DFHTRAP is cataloged in the CICS.SOURCE library. The comments in the supplied source contain much useful information, and you should read them carefully.

The code in DFHTRAP must **not**:

- Make use of any CICS services.
- Cause the current task to lose control.
- Change the status of the CICS system.

Establishing the exit

DFHTRAP is a PPT module, allowing the use of the CEMT NEWCOPY command.

An entry for DFHTRAP will be in the PPT if the RDO group DFHFE is installed in the CICS system either from the GRPLIST parameter of the SIT or dynamically using the CEDA transaction, or the FN=FE group is included in the assembled PPT. This group also includes the entries required for the CSFE transaction.

Once CICS is running, activation and deactivation of the trap exit routine can be requested using the TRAP operand of the CSFE DEBUG command:

```
CSFE DEBUG,TRAP={ON|OFF}
```

The trap exit can also be activated at CICS initialization by specifying TRAP=ON either in DFHSIT or as a startup override. In this case, the exit is available from the start of the third (final) stage of initialization, before any user PLT programs are executed.

If you want to **replace** a trap exit routine while CICS is running, use the CSFE DEBUG commands in conjunction with the CEMT NEWCOPY command. The following sequence of commands causes the currently active version of DFHTRAP to be refreshed:

```
CSFE DEBUG,TRAP=OFF
CEMT SET PROGRAM(DFHTRAP) NEWCOPY
CSFE DEBUG,TRAP=ON
```

Information passed to the exit

To assist in the diagnosis of faults in a CICS system, DFHTRP passes information to the exit in a parameter list addressed by register 1. A DSECT (DFHTRADS) is supplied for this list, which contains the addresses of:

- The return-action flag byte.
- A copy of the trace entry that has just been added to the table.
- An area for building a further trace entry.
- An 80-byte static work area for sole use of the trap exit.

The DSECT also contains EQU statements for use in setting the return-action flag byte.

The exit can look at the current trace entry to determine whether or not the problem under investigation has appeared. It can also look at the TCA of the current task, and the CSA. The DSECTs for these areas are included in the skeleton source, and their addresses are passed to the exit in registers 12 and 13 respectively.

Actions the exit can take

The return-action flag byte can be set by the exit to tell DFHTRP what action is required on return from the exit. The following list gives the possible choices:

- Do nothing.
- Make a further trace entry.
- Take a dump (the type depends on the SIT option).
- Abend the current task (with abend code ATRA).
- Abend CICS with a dump.
- Disable the trap exit, so that it will not be invoked again until reactivated by the CSFE transaction.

Any combination of these actions can be chosen, and, if possible, all actions are honored on return to DFHTRP. The exceptions to this are:

- A request to abend the current task is ignored in the following situations:
  - An abend of CICS is also requested.
  - The current task is already abending.
  - The current task is one of the following:
    - Journal control
    - Terminal control
    - Task dispatcher.
- A request to abend CICS is ignored if the current task is the task dispatcher.

The exit is automatically disabled if a request to abend either the current task or CICS is honored.

To reactivate the trap exit when it has been disabled, use CSFE DEBUG,TRAP=ON, unless the exit routine is to be replaced. In this case the sequence of commands given above applies.

The skeleton program shows how to make a further trace entry. When DFHTRAP detects the creation of a new task, it requests that a further trace entry be made by

entering the data required in the area supplied for this purpose, and by setting the appropriate bit in the return-action flag byte.

DFHTRP then makes an FE-type trace entry with trace ID = X'FD', incorporating the information supplied by the exit. FE trace does not need to be on for the creation of such a trace entry.

Trace entries created in this way are also written to the trace data set if the auxiliary trace facility is active.

Program check handling

The occurrence of a program check in the trap exit is detected by the program-check exit routine in the system recovery program (DFHSRP). Control is then passed to a special recovery routine in DFHTRP which:

- Restores DFHTRP's environment.
- Marks the exit as unusable.
- Issues the message DFH1409 to the system console.
- Takes a formatted dump, showing the PSW and registers at the time of the interrupt.
- Continues (ignoring the exit on future invocations of DFHTRP).

To recover from this situation, execute the commands given above for replacing the current version of the exit routine.

Coding the exit

When using the trap exit, note the following points:

- A skeleton version of DFHTRAP is supplied in both source and load-module forms. Make sure that the library search sequence in the CICS startup JCL finds the correct version of the load module.

- DFHTRP saves its registers before invoking the exit, and restores them on return from the exit. There is no need for DFHTRAP to save and restore the registers, as demonstrated by the skeleton version.

- On entry to DFHTRAP, register 14 contains the return address to DFHTRP. If this register is modified by the exit, care must be taken to return to the correct address.

- The 80-byte static storage area is provided as a work area for the sole use of the exit. It is set to binary zeros before the first use of the exit. Deactivation of the exit causes this area to be re-initialized the first time the exit is used following reactivation. In a formatted dump, this area can be found at the end of the static storage area owned by DFHTRP.

- Use the DSECTs supplied; corruption of storage by DFHTRAP could have disastrous results!

- MVS/XA users can code and link-edit their version of DFHTRAP so that it is able to address areas of storage above the 16 megabyte line. DFHTRAP **cannot** reside above the 16 megabyte line. Be careful when using addresses in CICS control blocks. These can be in either 24-bit or 31-bit format.

Chapter 2.2. XRF trace

Introduction

The XRF process trace is a wraparound trace table in main storage. It is a fixed size of 64K bytes, contains 32 byte trace entries, and tracing is always active when you are running with XRF.

Finding them

The trace table can be located from:

- *CICS environment*

  CSAOPFL -> SSA -> WS static -> WS global -> XRF trace table

- *XRF environment*

  Register 12 -> XPB -> WS global -> XRF trace table

Trace table format

The trace table starts with the following control information:

Bytes	Contents
0-15	'*** XRF TRACE **'
16-19	Address of start of trace entries
20-23	Address of end of trace entries
24-27	Address of end of most recent entry

Each entry has the following format:

Bytes	Contents
0	Type code
1	Subtype
2- 3	Process ID of XRF process which made the entry
4-27	Trace data - the format depends on the type/subtype
28-31	Clock value when entry was made, same format as CICS trace entries

Process IDs are assigned in order of process ATTACH starting from 1. Certain special values are used for processes which are not known to the dispatcher, but which cause trace entries to be made. They are:

Process ID	Function
X'0000'	Initial attach
X'FFFE'	ESPIE/ESTAE error handling
X'FFFF'	Dispatcher activities.

Entry types

The entries are as follows:

Module	Type	Subtype	Description
DFHWLGET	1	1	Module entry 4-11 Module name 12-15 LIFO allocation address
DFHWLFRE	1	2	Module return 4-11 Module name 12-15 LIFO allocation address 16-27 0
DFHWDATT	2	1	XRF process attach 4- 7 Process entry point 8-11 Initial data parameter 12-15 Address of ESPIE routine 16-19 Address of ESTAE routine 20-23 Address of attached process XPB 24-27 Process ID attached process XPB
DFHWDISP	2	2	XRF process detach 4-27 0
DFHWDISP	2	3	XRF process dispatch 4- 7 Address of external ECB waited for (if any) 8-11 Address of internal ECB waited for (if any) 12-15 Awaited broadcast events which were posted 16-19 Broadcast events still posted for this process 20-23 Address of process XPB 24-27 Locks held by this process

```
DFHWDWAT  2   4     XRF process wait (event data)
                      4- 7  Address of external ECB to wait for
                            (if any)
                      8-11  Address of internal ECB to wait for
                            (if any)
                     12-15  Broadcast events to wait for
                            (if any)
                     16-19  Events to be broadcast to all
                            processes
                     20-23  Events to be reset for this process
                     24-27  0

DFHWDAT   2   5     XRF process wait (lock data)
                      4- 7  Locks to be freed
                      8-11  Locks to be acquired
                     12-19  0
                     20-23  Locks held by all other processes
                            at time of call
                     24-27  Locks held by this process
                            at time of call

DFHWDISP  2   6     Dispatcher termination
                      4-27  0

DFHDISP   2   7     Dispatcher issuing OS WAIT
                      4-19  0
                     20-23  Address of WAIT list
                     24-27  Number of ECBs in WAIT list

DFHWDISP  2   8     Dispatcher resume after OS WAIT
                      4-27  0

DFHWMMT   3   1     Message manager issuing VSAM GET
                      4- 7  RPL address
                      8-11  RBA of CI to be read
                     12-27  0

DFHWMMT   3   2     Message manager issuing VSAM PUT
                      4- 7  RPL address
                      8-11  RBA of CI to be written
                     12-27  0
```

DFHWMMT	3	3	Message manager I/O complete
			4- 7 RPL address
			8-11 RBA of CI to be read
			12 0
			13-15 VSAM feedback information
			16-27 0
DFHWMQH	4	1	Message manager message received
			4- 7 0
			8-11 Queue name
			12-15 Message sequence number
			16-19 Address of message block (contains message copy)
			20-27 0
DFHWMWR	4	2	Message manager message sent
			4- 7 0
			8-11 Queue name
			12-15 Message sequence number
			16-19 Message file cycle number
			20-23 RBA of this message
			24-25 0
			26-27 Response to PUTMSG request (WMSRESP)

Chapter 2.3. Formatted dump

Introduction

The CICS formatted dump program DFHFDP is intended to make it easier to diagnose errors by performing automatically the mechanical process of finding the control blocks in storage. Each control block is printed separately in the dump, and is preceded by a heading; for some of the blocks the important fields are printed by name. In addition the dump may contain error messages where certain easily detectable errors have been found.

Formatted dumps are written to the CICS dump data set, DFHDMPA or DFHDMPB. They can be printed using the dump utility program, DFHDUP.

Alternatively, or in addition, the formatted dump program can produce one of the following types of partition dump:

- An unformatted dump of the CICS region to the CICS dump data set.
- A dump of the CICS region as produced by an MVS SNAP macro. This is written to a data set defined by the DDNAME DFHSNAP in the JCL, usually the SYSOUT = A data set.
- A complete MVS dump, as produced by the MVS SDUMP macro. This dump includes the MVS nucleus and common areas, as well as the CICS private areas, and is written to a SYS1.DUMP data set. See the *MVS/XA System Programming Library: System Macros and Facilities* manual for more information. Because of the large amount of output produced, this type of dump should be requested only if the problem is suspected to be caused by interaction of CICS with another component of the system.

You are advised to use SDUMP, paricularly when using XRF. The SDUMP can be processed using the MVS/XA AMDPRDMP service aid; see the *MVS/XA System Programming Library: Service Aids* manual for more details. There is a PRINTDUMP exit (DFHPDX1) that formats the dump for CICS areas. If you want to look at the output of AMDPRDMP online, you can place it in a VSAM data set, and then examine it under TSO using the Interactive Problem Control System (IPCS). IPCS is described in the *MVS/XA Interactive Problem Control System: User Guide and Reference.*

In all cases, the short symptom string and the trace table are written to the CICS dump data set. This data set can be printed using the dump utility program, DFHDUP. An interpreted form of the trace table is always produced.

Invocation of the formatted dump program

The formatted dump program is usually invoked each time CICS terminates abnormally, either because of a program check or abend, or because the operator requests a dump at system shutdown time. The program may also be invoked by use of the master terminal transaction CEMT, or as a result of a storage violation. In addition it may be invoked optionally for ASRA or ASRB transaction abends by use of the FDUMP operand on the DFHPCT macro. Finally, a DFHFD macro can be coded in a user program (assembler language only), though this facility should be used with caution, because all other activity in the CICS system stops for the duration of the dump.

The formatted dump program uses no CICS facilities and does not have access to information on the state of the CICS address space. In particular, it uses the operating system's facilities for its I/O operation so that once it is invoked it does not relinquish control until it has completed dumping. It issues DFHIC TYPE=STOPTIME to ensure that it is not abended by runaway task control and it issues its own SPIE macro so that it can handle any program interrupt that can occur. Hence, once it has been invoked no other CICS transaction can execute until the formatted dump program has completed. Use of the CEMT command or the coding of a DFHFD macro in an application program therefore locks out all other terminals and temporarily prevents all other transactions from running.

The DFHSIT specification

The DUMP operand of the DFHSIT macro is used to control the output from DFHFDP. The format is:

```
DUMP ={NO|([FORMAT|PARTN|FULL][,{SNAP|SDUMP}])}
```

Alternatively, at CICS startup time, DUMP can be specified by the operator. In either case the options specify the following:

FORMAT (the default) specifies that only a formatted dump should be produced.

PARTN specifies a partition dump only.

FULL specifies that both a partition dump and a formatted dump are required.

NO specifies an ABEND or CANCEL dump and abnormally terminates CICS.

SNAP|SDUMP specifies that, when a partition dump is taken (by FULL or PARTN being specified), then a SNAP or SDUMP should be issued instead of writing the dump to the CICS dump data set.

SDUMP dumps the entire address space on to the SYS1.DUMP data set. If SDUMP fails, a CICS partition dump is taken. SDUMP includes the link pack area, the MVS common service area (CSA), the MVS nucleus, and the CICS private areas. Consequently, you can see the task control block (TCB), service request blocks (SRBs), address space control block (ASCB), and address space extension block (ASXB).

CEMT transaction

The master terminal operator can request, at any time, that the formatted dump program be invoked by using the CEMT PERFORM SNAP command.

If, however, DUMP = NO was specified in the system initialization table or at startup time then, instead of issuing ABEND or CANCEL, the following message is sent to the operator:

```
FORMATTED DUMP INOPERATIVE
```

Assuming that DUMP = NO was not specified, and that no options are specified with the CEMT PERFORM SNAP command, the SIT options determine whether a partition or a formatted dump or both will be produced. On the other hand, if one or both of the options PARTITION and FORMAT are specified, then the SIT option is ignored, apart from any SNAP qualifier for PARTITION, and these options determine the output produced.

Storage violation dump

The formatted dump program may be invoked on the occurrence of a storage violation. This option may be selected by specifying SVD = YES or SVD = n, where $0 < n < 100$, either on the DFHSIT macro or at system startup. In the case of a storage violation dump, DFHFDP also writes the entire CICS dynamic storage area to the CICS dump data set, unless a partition dump is to be produced. DFHFDP is invoked before any storage recovery is attempted so it may be expected to find several errors. A storage violation dump is also produced if SVD = NO, but, in this case, the system is abended with code DFH0501.

ASRA and ASRB transaction abends

The formatted dump program can be invoked for ASRA and ASRB transaction abends. You can control the production of the dump both for individual transactions, or for all transactions on a system-wide basis.

If you use RDO to define your transactions, you may specify DUMP(NO) on the TRANSACTION definition. If you do this, you will *not* get a formatted dump if the transaction abends with ASRA or ASRB. The default is DUMP(YES).

If you use the DFHPCT macro to define your transactions, the default is no formatted dump. You have to code FDUMP = ASRA, or FDUMP = ASRB, or FDUMP = (ASRA,ASRB), depending on which abends you want the dumps for.

You can suppress the formatted dump, on a system-wide basis using the DFHSIT operands ABDUMP and PCDUMP. If you code ABDUMP = NO, no formatted dumps will be produced for ASRB abends. If you code PCDUMP = NO, no formatted dumps will be produced for ASRA abends. YES is the default for both these operands

To summarize, DFHFDP is invoked for ASRA abends if PCDUMP = YES, and the transaction is defined with FDUMP = ASRA or DUMP(YES). DFHFDP is invoked for ASRB abends if ABDUMP = YES, and the transaction is defined with FDUMP = ASRB or DUMP(YES).

The SIT options can be overridden during CICS execution by the command:

```
CEMT SET DUMPOPTIONS {ABDUMP|NOABDUMP} {PCDUMP|NOPCDUMP}
```

The DFHFD macro

The DFHFD macro (available only to CICS assembler programs) may be used to invoke DFHFDP although it is intended primarily for use by CICS service programs. The format is:

```
DFHFD   TYPE=(option[,option]...)
```

where **option** is one of:

PARTN specifies that a partition dump is required.

FORMAT specifies that a formatted dump should be produced.

CONDNL specifies that execution is to continue after this call, so that DFHFDP does not abend if DUMP = NO was specified in the SIT.

Notes:

1. *Using this macro destroys the contents of registers 14 and 15 before they are printed.*
2. *If neither PARTN nor FORMAT is specified, then the DFHSIT option is used to determine the output.*

The output of the formatted dump program

As mentioned above, the output from DFHFDP may be a partition dump and/or a formatted dump. The rest of this chapter is concerned only with the **formatted** dump.

The reason for the dump

At the front of the formatted dump there is a brief message indicating the reason for the dump together with associated information, and the short symptom string. In the case of a program check or an abend, the program status word (PSW) and the registers at the time of the program check or abend are printed. In the case of a program check when recovering from a previous program check (abend 602), the PSW and registers corresponding to both program checks are printed. For each program check, the area of storage in which the program check occurred is also printed.

For more detailed information on diagnosing some common abends, see the chapter on failure analysis structure tables, in the *Messages and Codes* manual.

Program status word (PSW)

In a formatted dump, produced after a program check or abend has occurred, the program status word (PSW) is printed as 16 bytes, in groups of 4 bytes. The first 8 bytes are the PSW at the time of interrupt, printed in the EC-mode format. CICS received the PSW in BC mode, and converted it to EC-mode format. The next 2 bytes are the instruction length code (ILC), multiplied by 2 to give the number of bytes in the instruction in error. CICS obtains the ILC from the BC-mode PSW. The next 2 bytes are the interrupt code, also obtained from the BC-mode PSW. See the *IBM System/370 Principles of Operation* for information on these PSW formats.

Short symptom string

The short symptom string is a set of words that give information on the state of the CICS system and is designed to be used as a search argument for the software support facility (SSF). The short symptom string is printed at the beginning of a formatted dump or a transaction dump.

The SSF is a data base that holds descriptions of previously reported CICS problems. Each problem is identified by one or more keywords that are entered at the same time as the description is entered. The usefulness of the SSF system is dependent on having a standard set of keywords, and the short symptom string helps this objective.

The short symptom string is of the form:

```
SYMPTOMS=AB/S1111  PIDS/222222222   FLDS/33333333  RIDS/444444
              U

where

   S or U stands for system or user respectively
     1111 is the abend code
222222222 is the name of the CICS component
   333333 is the module identifier of the owner of the lowest level
           of LIFO storage
 44444444 is the module name from the PPT entry
```

An example of a short symptom string is:

```
SYMPTOMS=AB/UASRA  PIDS/5665-4030  FLDS/F000KC  RIDS/ZUBGHOD
```

This was caused by a program interrupt (ASRA) in an application program called ZUBGHOD. The system used was CICS/MVS and the CICS module that last obtained LIFO storage was DFHKCP.

The control blocks

The major part of the dump then follows. It consists of a hexadecimal dump of most of the control blocks and tables in the CICS address space. Each block is preceded by a heading indicating what it is and, for some of the more important blocks, a display of some of the major fields of the block. For each field highlighted in this way, the hexadecimal offset, the name and the contents are printed. Depending on the nature of the field, the contents may be printed in either character, hexadecimal or bit notation or as an interpretation of the contents of the field.

The control blocks are not printed in address order but instead an attempt is made to group them logically. Thus the CSA is always printed first and is followed by the optional features list; and each DCA is followed by the corresponding TCA which is in turn followed by the control blocks associated with that task.

Trace table

Each entry in the trace table is printed in the format shown in Figure 21 for convenient interpretation.

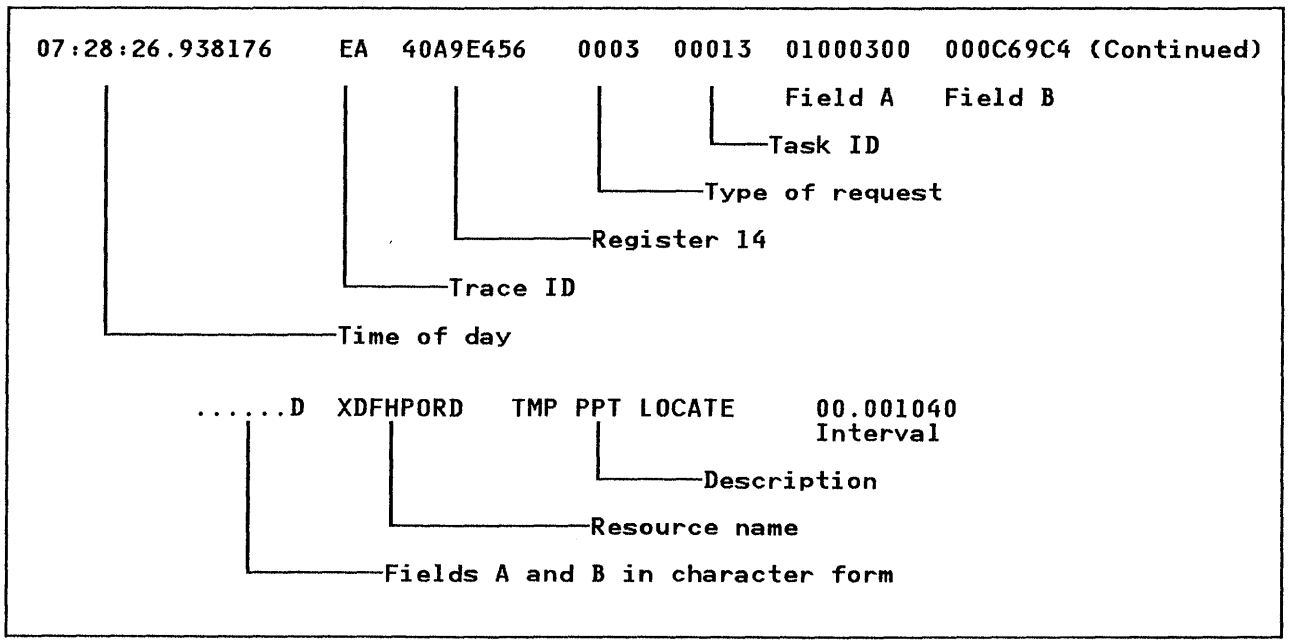


Figure 21. Trace table entry

Program check and abend trace table

The format of the program check and abend trace table header (SRP's trace table) is as follows:

```
Byte          Contents
X'00'-X'1F'   C'PROGRAM CHECK/ABEND TRACE TABLE:'
X'20'         Address of next table entry to be used
X'24'         Address of beginning of table
X'28'         Address of end of table
X'2C'         The number of program checks that have occurred
               in the system since it was last started
X'30'         The number of abends that have occurred in the
               system since it was last started
X'34'         Length of this table (including header)
X'38'         Reserved
X'40'         Start of table entries
```

The format of a table entry (128 bytes) is as follows:

```
X'00'-X'01'    C'AB', C'PC', or C'WT' to say whether the entry is
                for (respectively) an abend, a program check,
                or a DFHWTO with dump request.
X'02'-X'03'    The number of this program check or abend.
X'04'-X'07'    TCA address
X'08'-X'0B'    The address of the current LIFO stack entry.
X'0C'-X'0F'    The address of the current entry in the
               (in-storage) trace table  at the time of the
                program check or abend.
X'10'-X'17'    Trace ID and task ID
X'18'-X'1F'    Register 14 and time stamp
X'20'-X'27'    Reserved
X'28'-X'2F'    C'REGS.PSW'
C'30'-X'6F'    Registers 0-15
X'70'-X'77'    EC-mode PSW
X'78'-X'7F'    Interruption information (first halfword is
                instruction length, second halfword is
                interruption code) The remainder of this
                field depends on the operating system.
```

The address of the current trace table entry helps you to debug when using a partition dump, but you should be aware that the trace table may have cycled, and overwritten the entry.

The program check and abend trace table can be located in a dump by looking at CSAOPFLA in the CSA which points to the CSA optional features list. Then CSASTRTA in the optional features list points to the program check and abend trace table.

The indexes

At the end of the dump two indexes are printed. The first index shows in address order what control blocks have been printed. For each control block the number of the page on which it was printed is given together with an indication of the type of control block and a count of the number of errors that were encountered while processing it.

The second index is an index of most of the programs and tables in the CICS address space, again in address order. For each item the entry point, the load point and the name are printed. For user application programs that have been loaded, this is all that is printed, but for CICS programs or tables the fields resulting from the DFHVM macro at the start of each program or table are also printed. These are the version, the time and date of assembly, and the option bytes that are generated at the start of each program.

This (second) index should be useful in the case of a program check. Knowing the address where the program check occurred and that the index is in address order, the module and the offset within that module can be found very quickly. The area of storage printed at the start of the dump can be used to check that this is the correct module and offset. The likeliest error if this is not so is that the wrong assembly listing is being used.

Formatted dump program error messages

The formatted dump program has its own program interrupt handler and several levels of recovery routines. Whenever it encounters an error, it prints a message in the dump and tries to take the appropriate recovery action. Errors fall into two main categories, those for which an explicit test is made and those that are encountered as a result of some other action. Not every possible error that could be detected by an explicit test is in fact detected. For example, not every pointer is checked for validity and not every packed decimal field is checked to ensure that it is indexed in packed decimal format.

Main messages

There are five main error messages:

DFH5011 POINTER TO xxxxx AT OFFSET yyyy IS INVALID
– the pointer at offset yyyy in the current control block contains an address which lies outside the CICS address space. The pointer should address a control block of type xxxxx.

DFH5012 CONSTANT AT OFFSET xxxxx SHOULD BE X'yyyy'
– the value of offset xxxxx in the current control block is not yyyy as it should be. The first byte of most storage areas is checked to ensure that it contains the expected storage class byte; see "Storage classes" on page 265.

DFH5013 PACKED DECIMAL FIELD AT OFFSET xxxxx IS INVALID
– the field at offset xxxxx is not in packed decimal format.

DFH5017 ADDRESSING EXCEPTION OCCURRED AT LOCATION xxxxx
– this message may occasionally occur when a control block or a highlighted field is being formatted, and is probably caused by a control block appearing to overrun the CICS address space. xxxxx is the address of the data that caused the interrupt and not of the instruction where it occurred.

DFH5018 INVALID STORAGE ACCOUNTING AREA
– the storage accounting area of the current control block is invalid either because its length field is either 0 or not a multiple of 8 or because it differs from the duplicate storage accounting area. This probably indicates that either a pointer was incorrect or that storage has been overwritten.

Other messages

In addition, messages may be produced by the recovery routines:

DFH5002 PROGRAM CHECK LIMIT EXCEEDED
– the number of program checks has exceeded the number predefined in DFHFDP. The formatted dump program is probably in a catastrophic situation and so the dump is terminated immediately.

DFH5003 PROGRAM CHECK IN FORMATTED DUMP
– a program check occurred on an instruction for which no explicit recovery routine has been provided. This message will be followed by a dump of the PSW and the registers at the time of the interrupt. If DFHFDB, the interpreter, was in control at the time these will be followed by message DFH5014.

DFH5005 AREA FROM xxxxx TO yyyyy IS NOT AVAILABLE
– when attempting to dump a control block or the partition, the specified area was found to be not addressable.

DFH5010 INSUFFICIENT STORAGE SPACE – FUTURE POINTERS WILL BE IGNORED
– the formatted dump program has run out of working storage space and cannot obtain any more from the operating system. All control blocks that have been found so far will be processed and printed but no new pointers can be saved and followed. Twelve bytes of working storage are required for each control block printed and after an initial allocation of 4096 bytes is used up, operating system requests are made to obtain more storage. See "Storage for use by the operating system" on page 266.

DFH5014 FORMATTED DUMP ERROR DURING xxxxx PROCESSING: ERROR CODE = y
– an undefined error occurred while processing a control block of type xxxxx. If the error code is 2 or 3, then there is a logical error in the descriptor tables in DFHFDC. If the error code is 1 then there was a program check and message DFH5003 and a dump of the PSW and registers should precede this message.

DFH5015 INVALID CONTROL BLOCK POINTER – xxxxx
– after message DFH5003, it was discovered that register 10 (which should contain the address of the current control block) actually contained an address of a location outside the CICS address space.

DFH5016 INVALID CONTROL BLOCK LENGTH – xxxxx
– after message DFH5003, it was discovered that the computed length of the correct control block is either negative, greater than 65536, or such that the end of the control block is outside the CICS address space.

DFH5019 PROGRAM CHECK DURING ERROR RECOVERY
– a potentially recursive situation exists, and so no further recovery is attempted for the current control block.

How to interpret a formatted dump

AFTER A PROGRAM CHECK
From the PSW at the start of the dump and the program index at the end of the dump, determine which program was executing and the offset within that program. Find the instruction that caused the interrupt in the area of storage printed immediately after the register. If necessary, find any storage areas referenced by using the control block index at the end of the dump. Using this information, the immediate cause of the program check should be clear and further investigation will depend on the circumstances.

A frequently occurring case is when the PSW is equal to X'00000004'. This usually happens when a BALR 14,14 or BALR 14,15 has been executed and the address loaded was 0. For the latter, register 15 is 0. In these cases, register 14 should be noted and the program index examined to locate the program that issued the BALR.

AFTER A CICS ABEND

The abend code only is printed and it is necessary to look at the console log to see the message itself. The *Messages and Codes* manual should provide more information. After the message and abend code, the PSW, and the registers are printed. Using the PSW and the program index at the end of the dump, it is possible to determine which program issued the abend.

AFTER A STORAGE VIOLATION

If the error occurred on a FAQE chain, then DFHFDP may not detect any errors in storage. However, in many cases, a storage accounting area will have been overwritten, and this will be detected. Check the control block index at the end of the dump and check those control blocks for which errors have been found. If a block is found whose storage accounting areas are invalid, determine if it is the initial or the duplicate area that is in error. If it is the initial area, use the control block index to find the block immediately in front of the one in error. The likelihood is that one of the users of these blocks has overwritten it. The commonest cause of a storage violation is for a TWA, as specified in the PCT, being insufficient, or for a TIOA to be too small.

More information on how to interpret storage violation dumps can be found under Abend U0501 in failure analysis structure tables in the *Messages and Codes* manual.

Use of control block index

The control block index at the end of the dump contains, for each block printed, a count of the errors detected. Sometimes these errors may be caused by errors in DFHFDP, but they should always be checked. When an error does occur, for example, a bad TCA pointer in a DCA, there will frequently be a cascade of errors as DFHFDP attempts to interpret a piece of storage as something it is not. In cases like this, try to determine where the errors started. If an area described as a TCA is obviously not a TCA (the first byte is not X'8A', for example), try to find the control block which pointed to it – the control block should normally be a DCA immediately in front of it.

Use of program index

In the program index at the end of the dump, check that different programs from the same group have been assembled with the same options and that the correct version has been used, check the suffix and the level, and the date and time of assembly.

Chapter 2.4. Control block linkages

The figures in this chapter indicate how the main control blocks of CICS are linked. Each figure starts either from the CSA or from a TCA. Not all control blocks used by CICS are included and some control blocks appear in more than one figure. Similarly, not all possible linkages between control blocks are included.

Finding the common system area (CSA)

Before using the figures in this chapter, it is necessary to find the CSA in the dump under consideration. If the formatted dump program has been used, then there is no problem because the CSA is the first control block to be printed. If a partition dump (only) is available, then (unless the code being executed is in an access method) register 13 should address either the CSA or a save area that contains the address of the CSA at offset X'4'.

If register 13 is not available, or the CSA cannot be found from it, then the CSA may be located either by inspecting the nucleus load tables (NLTs) used during initialization or by visually scanning the dump. The CSA is contained in the module DFHCSA.

During initialization, loading of the CICS nucleus is controlled by means of an NLT. Modules are loaded from high address space to low address space in the order specified by the NLT. If the user specifies an NLT, it is used first, and then the default NLT (provided by CICS in module DFHSIB1) is used to control the loading of any modules not specified in the user's NLT. The ordering provided by the default NLT is shown in the *Resource Definition (Macro)* manual.

Another method of locating the CSA is to examine the authorized facility control block (AFCB) field which contains the address of the CSA. The word TCBCAUF, in the second extension of the TCB, points to AFCB. The address of the CSA is contained in field AFCSA at offset X'8' in the AFCB.

The CSA can be recognized in a partition dump by the presence of a DFHVM expansion beginning '*DFHCSAxx*' and a copyright notice, followed by a short piece of code to handle AICA transaction abends and the character string 'STORAGE '. The CSA itself then begins at the next doubleword boundary as shown in Figure 22 on page 142.

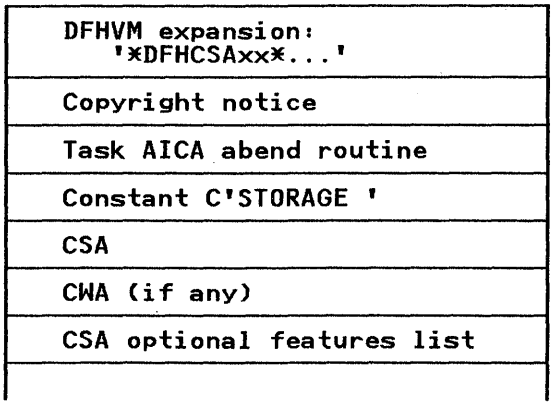


Figure 22. CSA

It may also be possible to locate the CSA by using the register 13 save area chain starting from the OS/VS TCB.

Finding DCT, FCT, PCT, PPT, or TCT entries in a partition dump

A general procedure for finding DCT, FCT, PCT, PPT, or TCT entries in a partition dump is as follows:

1. Find the CSA (see Figure 22).
2. Field CSAOPFLA (offset X'C8') in the CSA holds the address of the CSA optional features list (CSAOPFL). Find this area.
3. Field CSASSA (offset X'1C0') in the CSAOPFL holds the address of the static storage area address list (SSA). Find this area.
4. Field SSATMP (offset X'14') in the SSA holds the address of the table management static storage area (TMSSA). Find this area.
5. Look at TMSSA in the *Data Areas* manual. The fields TMASKT1-TMASKT9 hold the addresses of the scatter tables for various control blocks. Find the scatter table for the control block you are interested in:

```
TMASKT1 = addr of local PCT entries
TMASKT2 = addr of remote PCT entries
TMASKT3 = addr of PPT entries
TMASKT4 = addr of profile table (PFT) entries
TMASKT5 = addr of FCT entries
TMASKT6 = addr of DCT entries
TMASKT7 = addr of TCT entries
TMASKT8 = addr of TCTSK entries
TMASKT9 = addr of TCTSE entries
```

6. Field SKTFDEA (offset X'10') in the scatter table holds the address of the first directory element. Find this area.
7. Directory elements are chained together in alphabetic order. The address of the next element is in field DIRGNCHN (offset X'10').

8. Look at each directory element until you find the name of the control block you are looking for. The name is in field DIRKEY (offset X'18'). Then field DIRTEA (offset X'0') holds the address of the desired control block. See Figure 23 for the relationship of the table management control blocks.

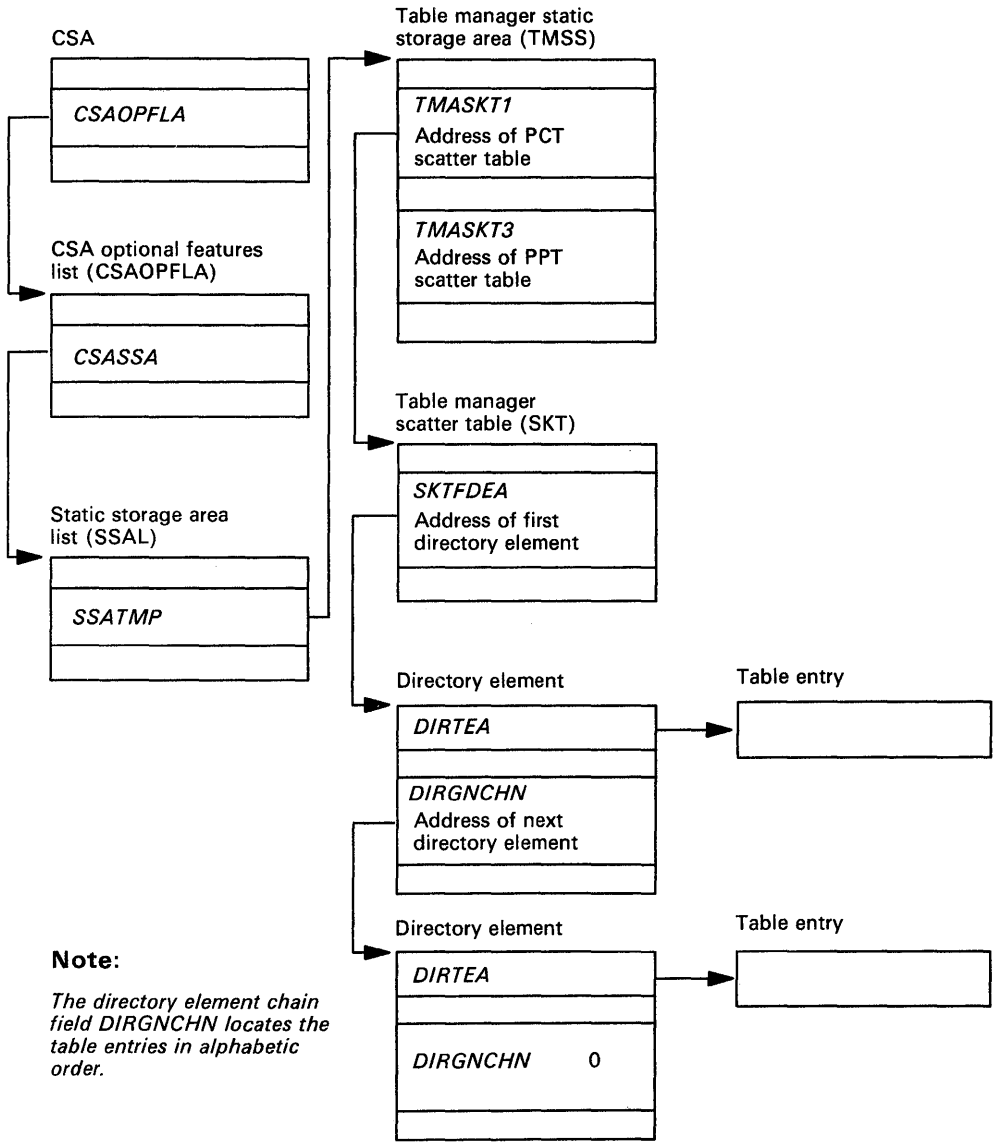


Figure 23. Control blocks associated with table management

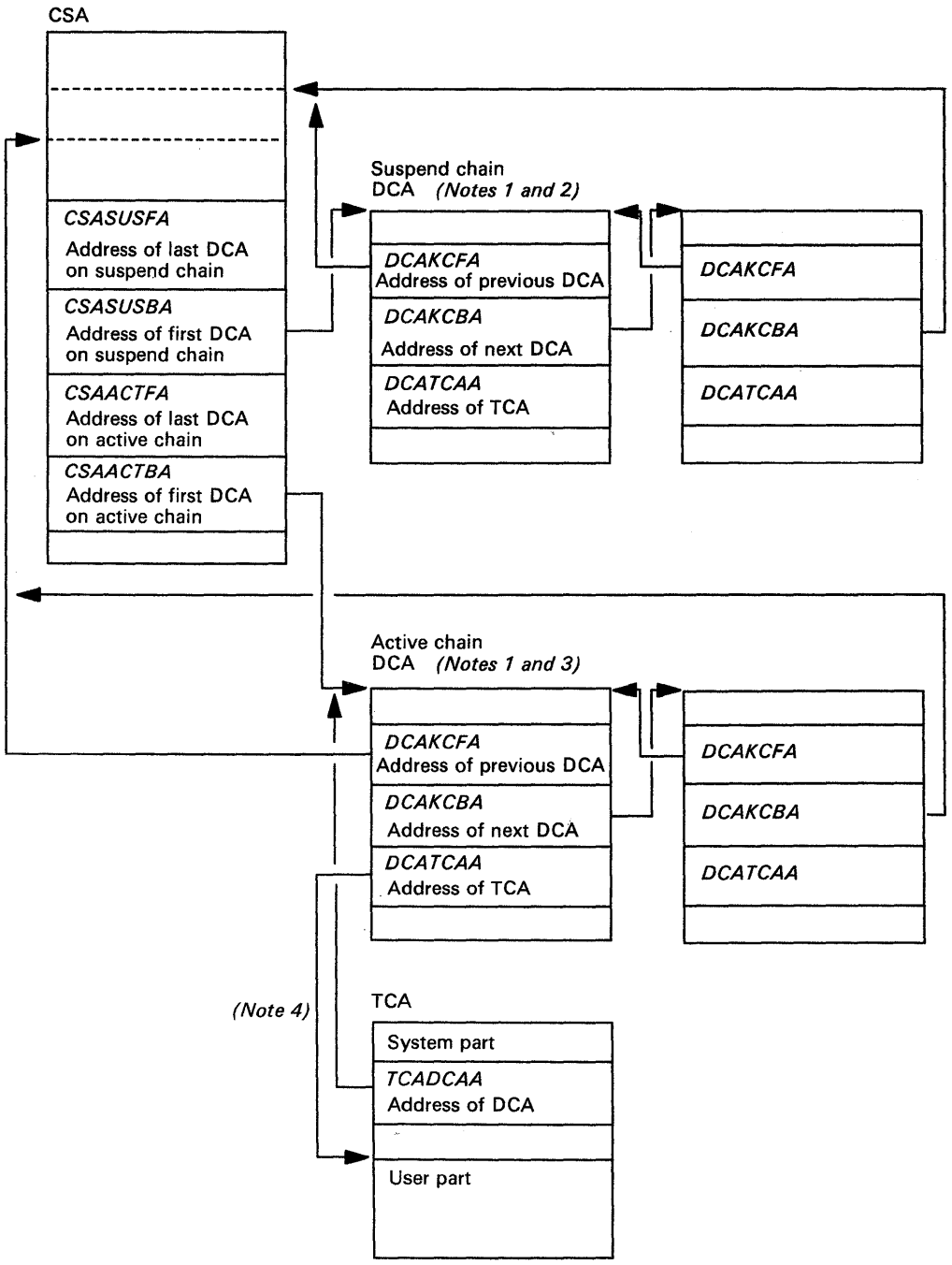


Figure 24 (Part 1 of 2). Task dispatching

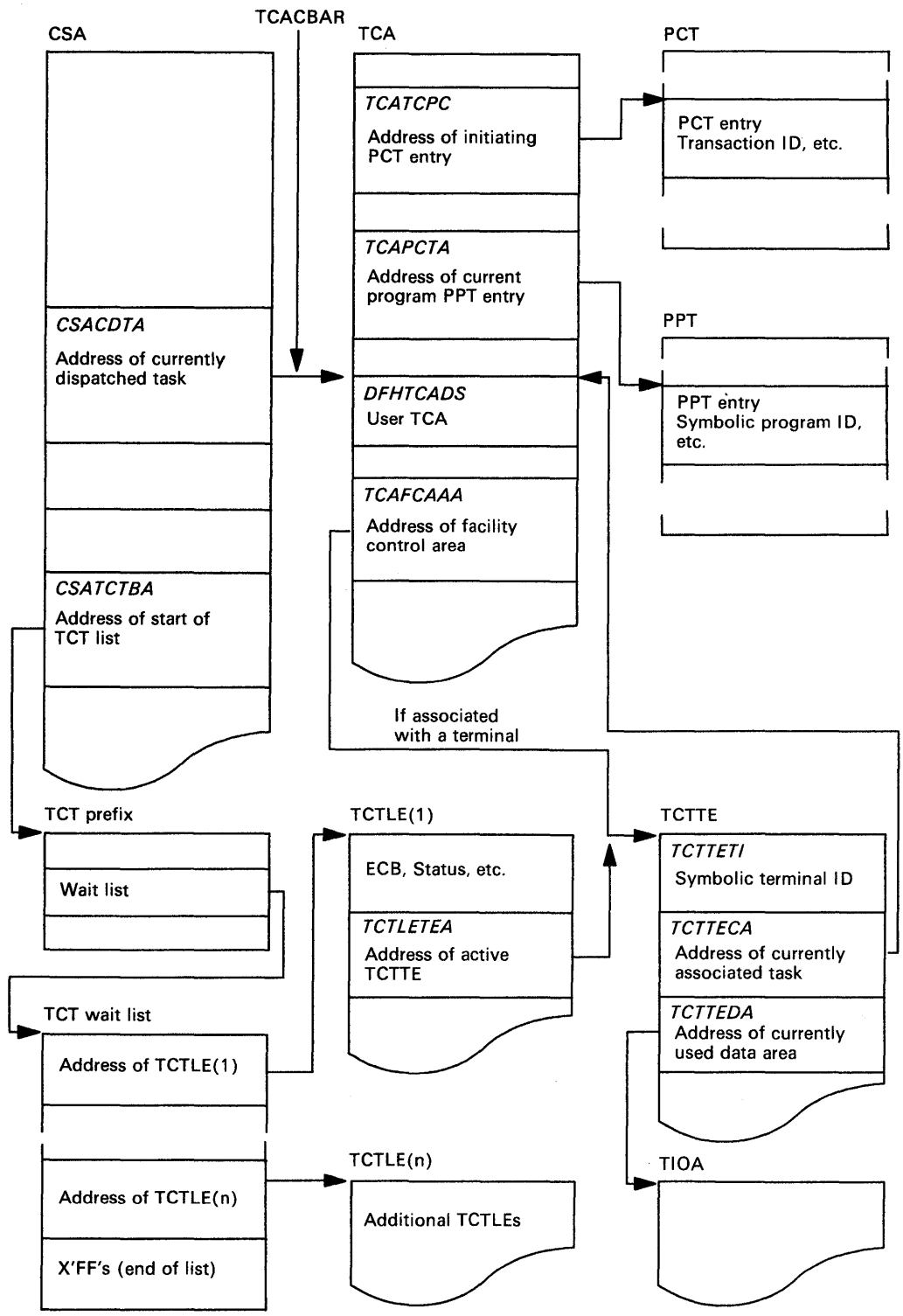


Figure 24 (Part 2 of 2). Task dispatching

Task dispatching

Notes:

1. *There are two chains of DCAs anchored in the CSA. The active chain contains those DCAs that are scanned by the CICS dispatcher when it is looking for a task to dispatch. The other tasks are on the suspend chain. The CSA contains a dummy DCA for each chain.*

2. *Tasks subject to deadlock or read timeout are in ascending timeout order at the end of the chain. The other tasks are inserted at the head of the chain, which is addressed by CSASUSBA.*

3. *The active chain is in priority order with CSAACTBA addressing the highest priority task. A task entering the active chain gets inserted after any tasks of the same priority. The first task on the active chain is always the terminal control task. The CICS dispatcher usually starts its scan with the second task on the active chain. If it finds no task to dispatch, it tests to see if the terminal control task is dispatchable.*

4. *Most new tasks do not possess a TCA until they are first dispatched. Flag DCATCA in field DCATSKST is set to 1 when the TCA is acquired.*

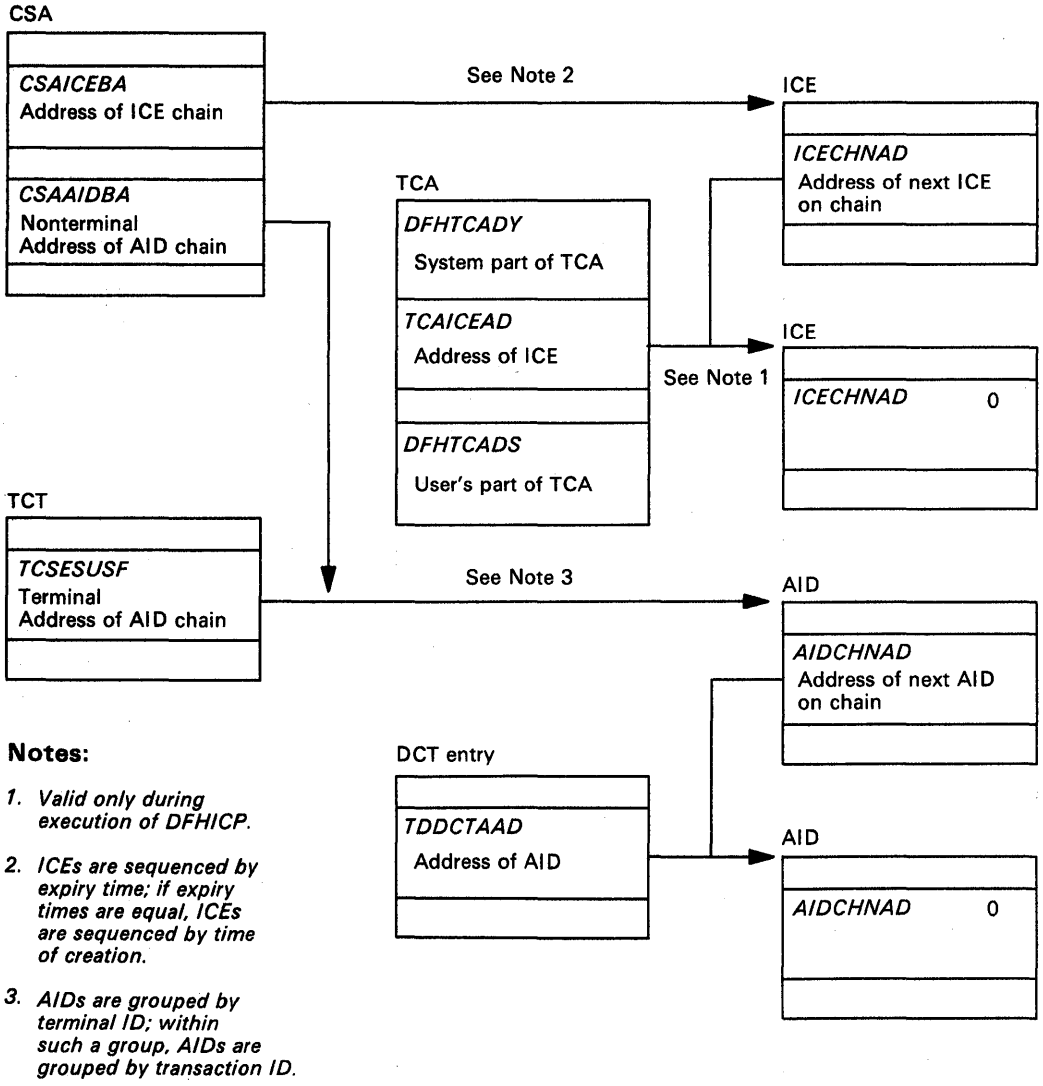


Figure 25. AID and ICE chains

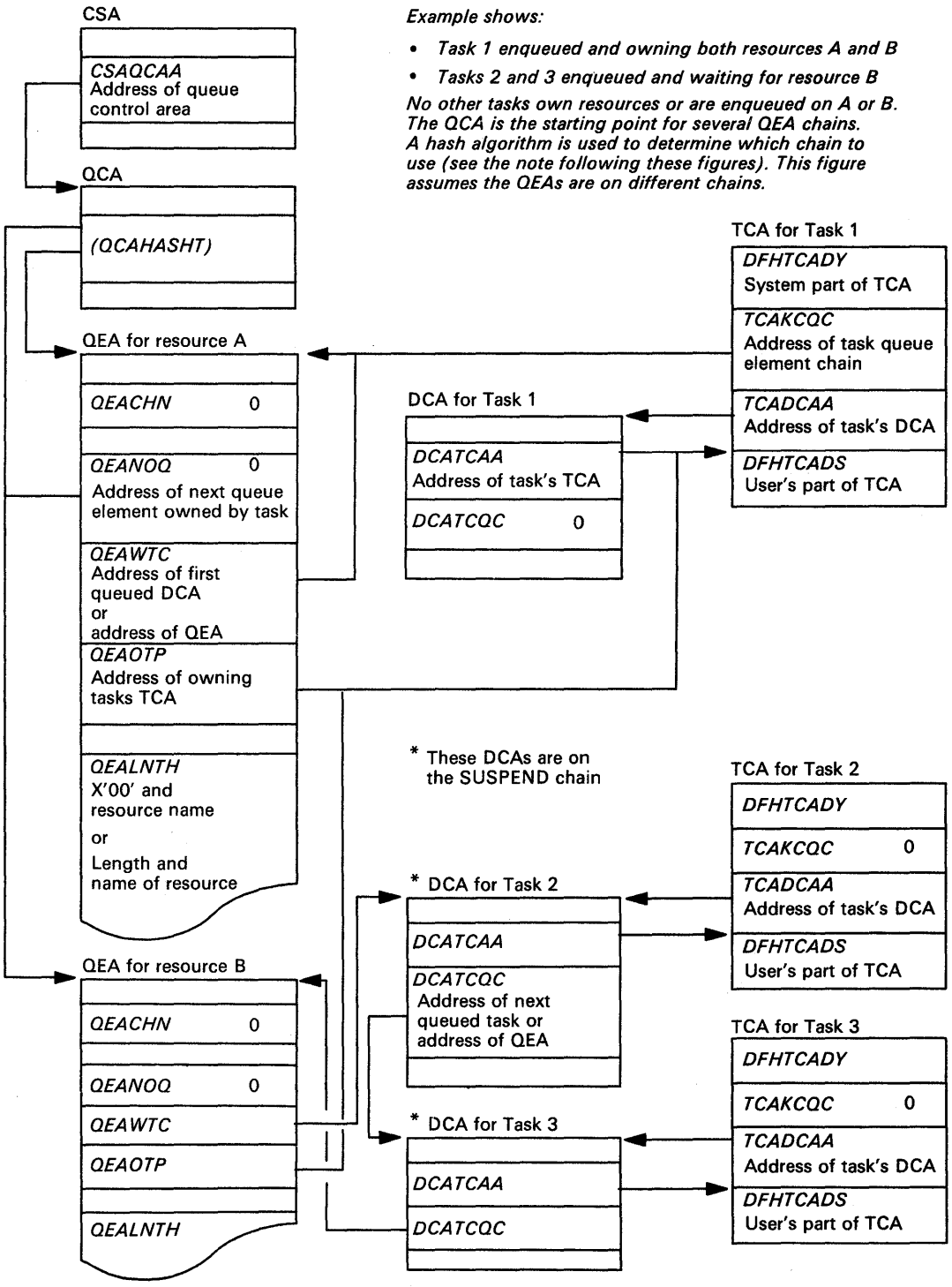


Figure 26. Queue element area chains

Queue element area

The queue element area (QEA) chain hash algorithm works as follows (see Figure 26 on page 148):

1. Take eight characters from the resource identifier. If the identifier is less than eight characters long, it is padded on the right with binary zeros. If it is more than eight characters long, the last eight characters are used.

2. **Exclusive or** all characters into the last one.

3. Multiply the last character by four and use it as an offset in the table of addresses of chains of QEAs.

QEAs representing resources owned by a given task are chained from the field QEANOQ in last-in-first-out (LIFO) order. QEAs representing resources owned by the CICS system are chained from the field QEACHN in first-in-first-out (FIFO) order. The DCAs of tasks waiting to become owners of resources are chained from the field DCATCQC in FIFO order. The address of the first DCA is in field QEAWTC of the QEA representing the resource.

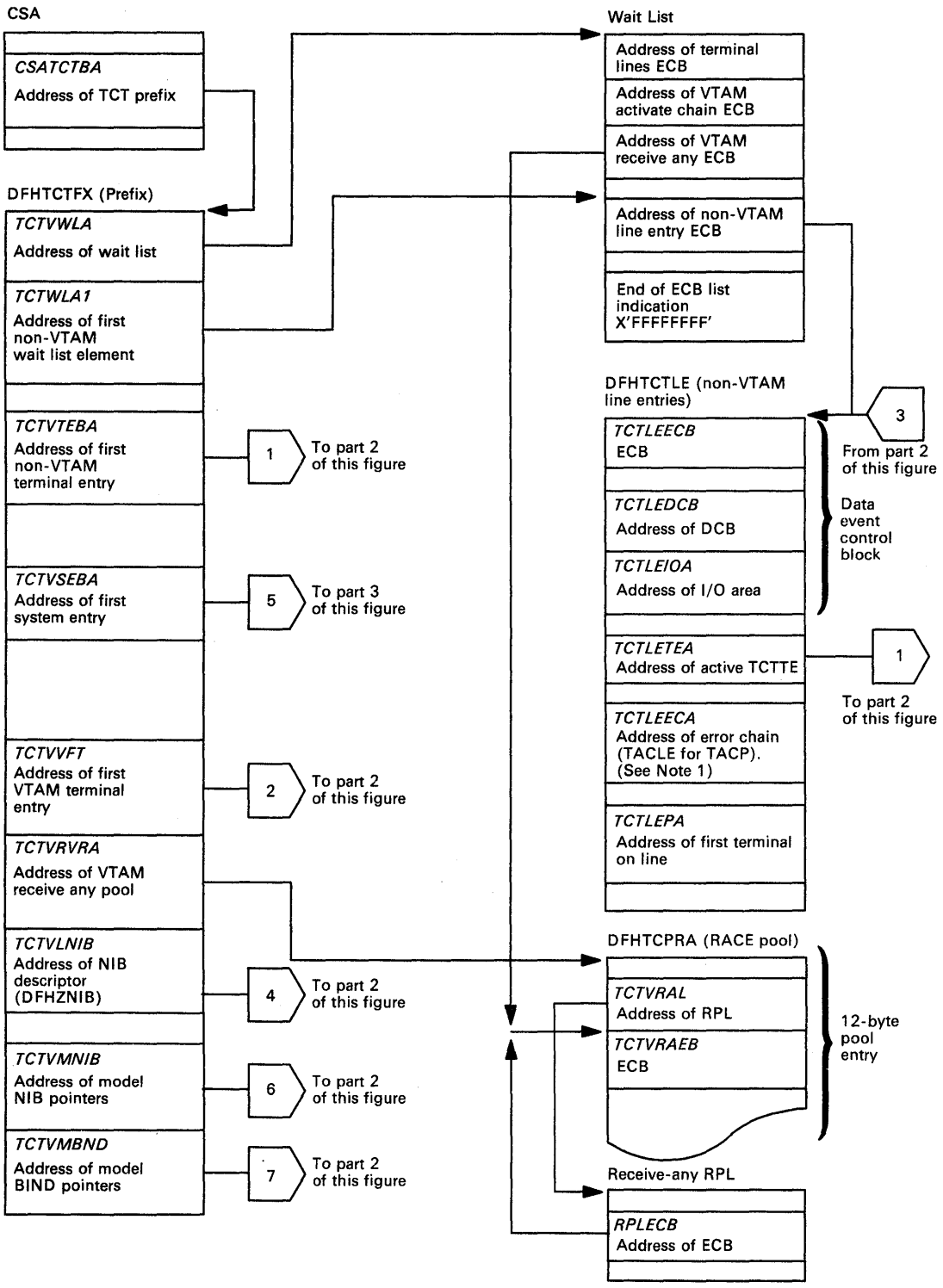


Figure 27 (Part 1 of 3). Control blocks associated with terminal control

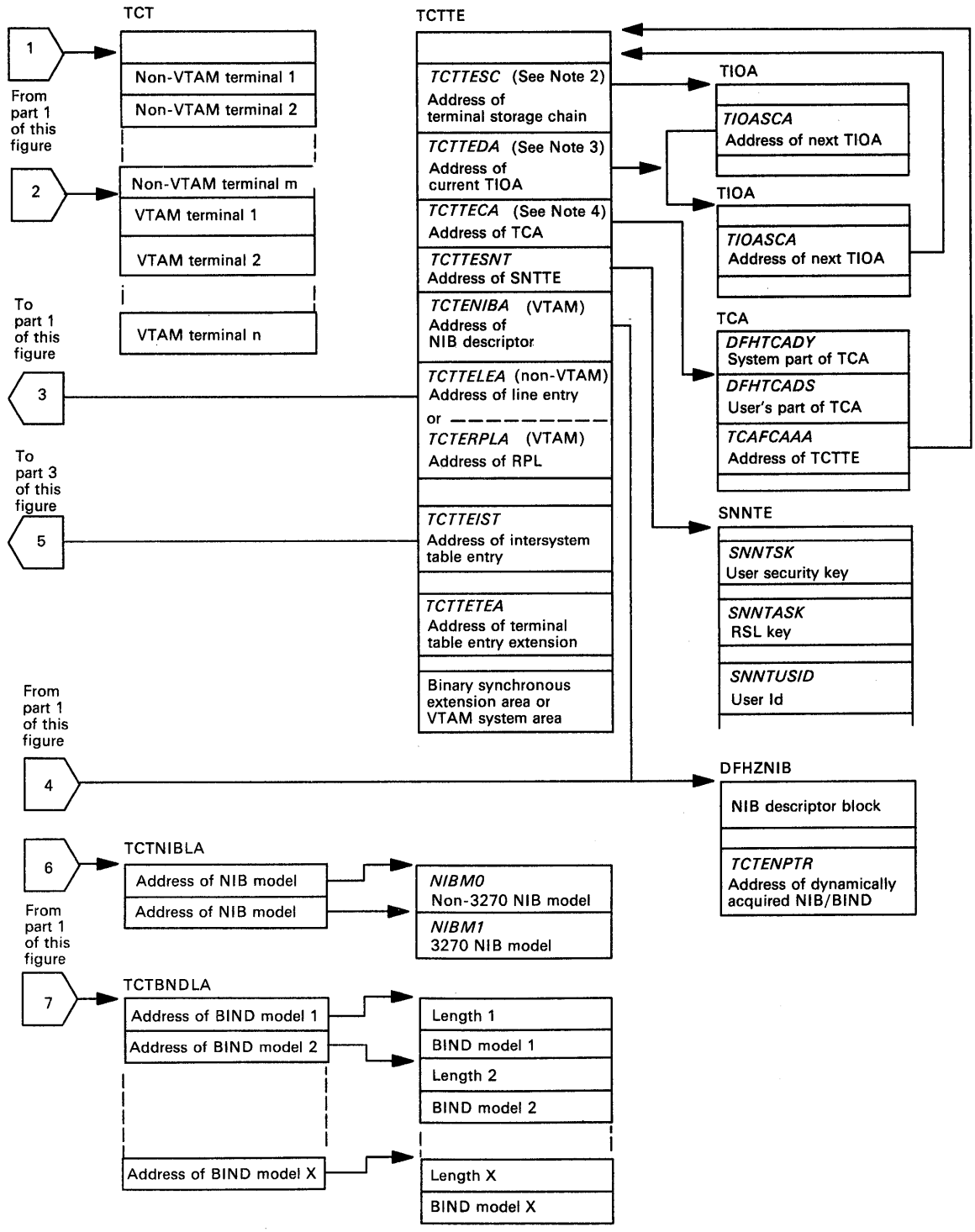


Figure 27 (Part 2 of 3). Control blocks associated with terminal control

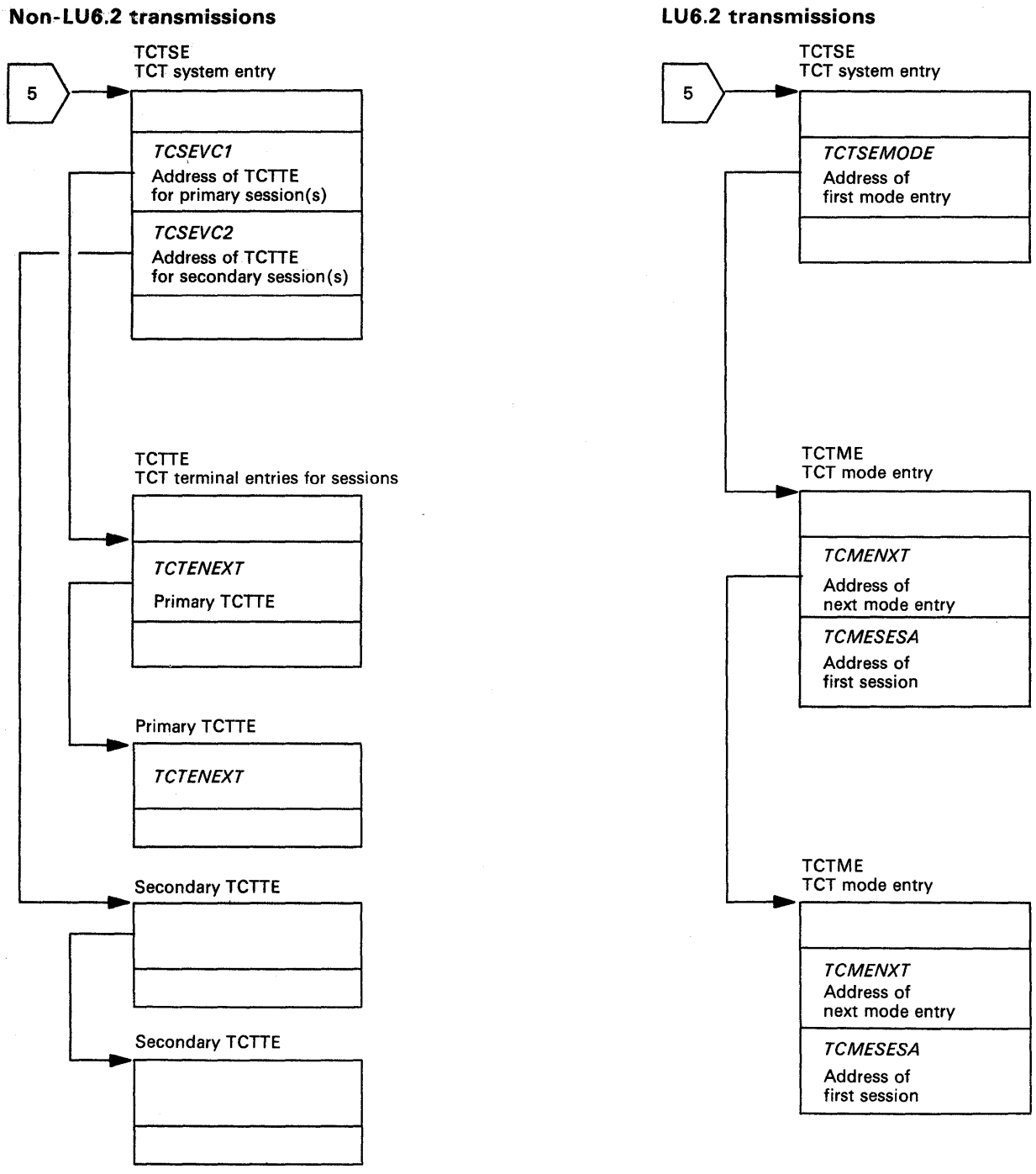


Figure 27 (Part 3 of 3). Control blocks associated with terminal control

Terminal control

Notes:

1. *A TACLE is created only when a line or terminal error has occurred.*

2. *The chain field TIOASCA of the last TIOA in the chain addresses label TCTTESCF in the TCTTE. The offset between TCTTESF and TCTTESC is the same as the offset of TIOASCA in the TIOA.*

3. *TCTTEDA addresses the TIOA being used for the current input/output operation. This TIOA can be anywhere in the TIOA chain.*

4. *TCTTECA addresses the DCA between ATTACH and first dispatch, and the TCA from then on. If TCTTECA is nonzero, flag TCTEDCAO = 1 means that TCTTECA addresses a DCA, and TCTEDCAO = 0 means that TCTTECA addresses a TCA. TCTEDCAO is bit 6 at offset X'2F' in the TCTTE.*

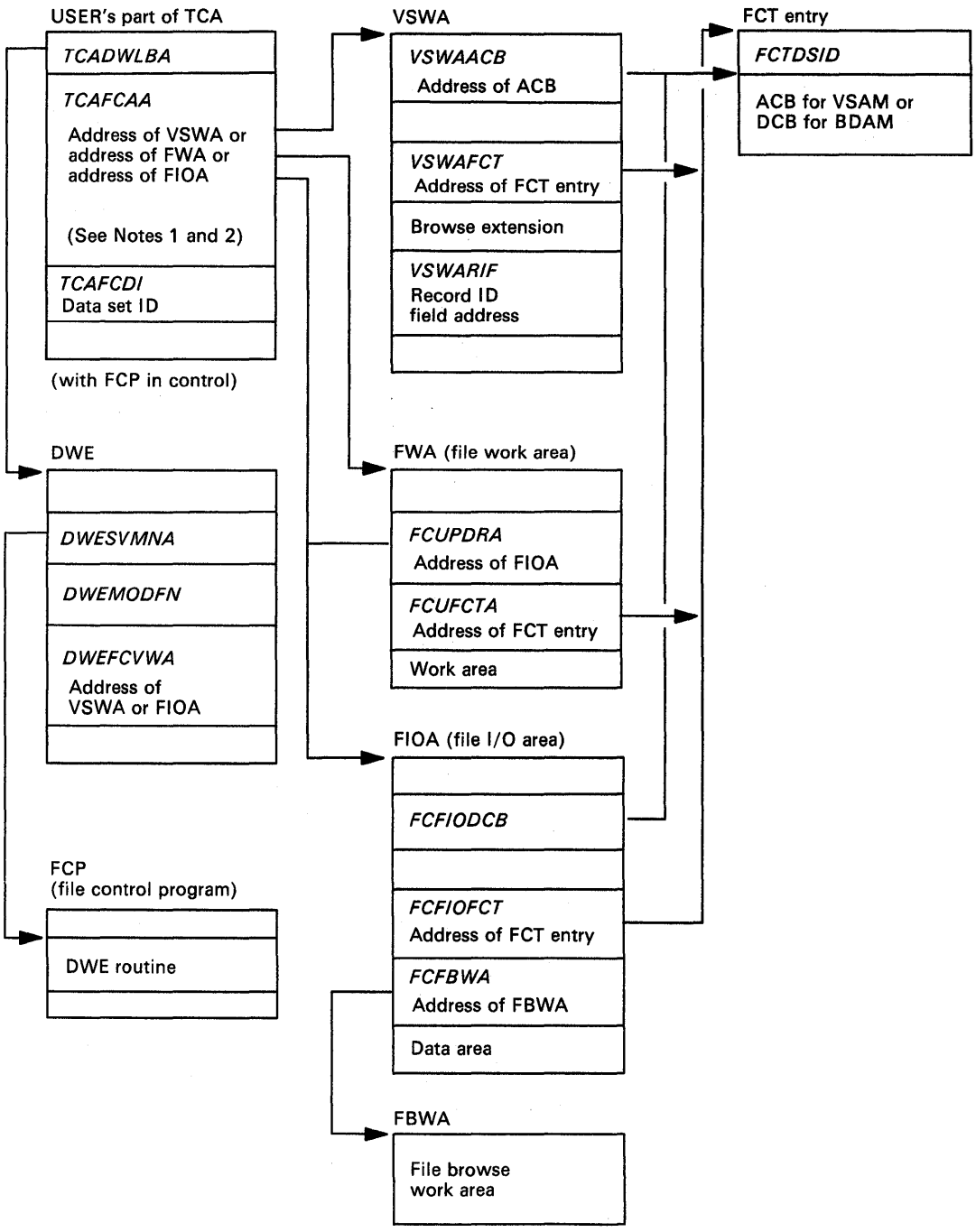


Figure 28 (Part 1 of 2). Control blocks associated with file control

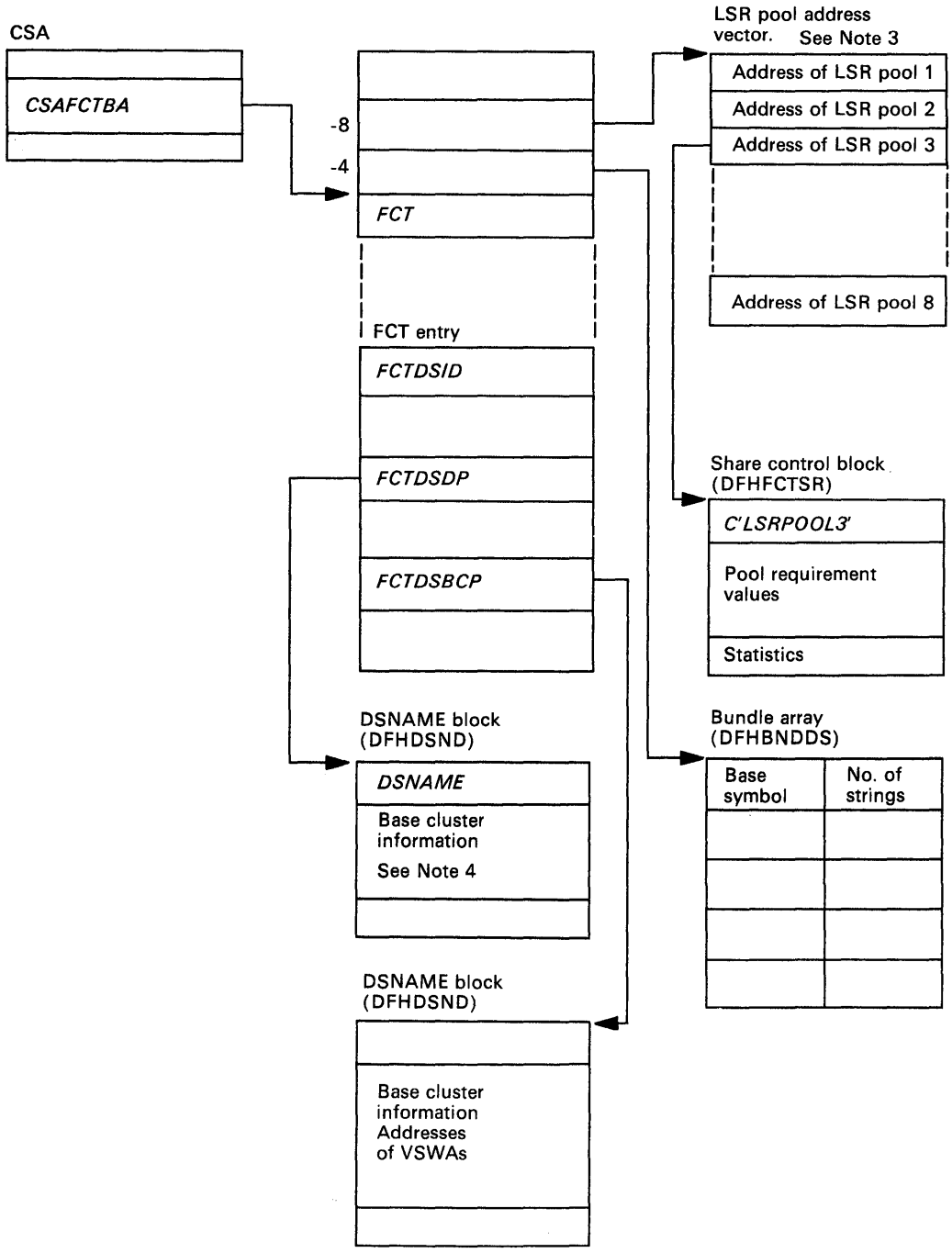


Figure 28 (Part 2 of 2). Control blocks associated with file control

File control

Notes:

1. *For VSAM file control, TCAFCAA contains the address of:*
   - *VSWA when in locate mode, or error situation.*
   - *FWA otherwise.*
2. *For BDAM file control, TCAFCAA contains the address of:*
   - *FIOA for read-only and unblocked files.*
   - *FWA otherwise.*
3. *There is one share control block for each LSR pool ID. If VSAM does not provide multiple LSR pool support, share control block 1 is used for all LSR pools specified in the FCT. If VSAM does provide multiple LSR pool support, share control block 1 exists even if it is not specified in the FCT.*
4. *The pointer to the DSNAME block, FCTDSDP, is different from the pointer to the base cluster DSNAME block, FCTDSBCP, only when the FCT entry does not represent a base. DSNAME blocks that do not correspond to bases do not have the base cluster information, although the space is allocated.*

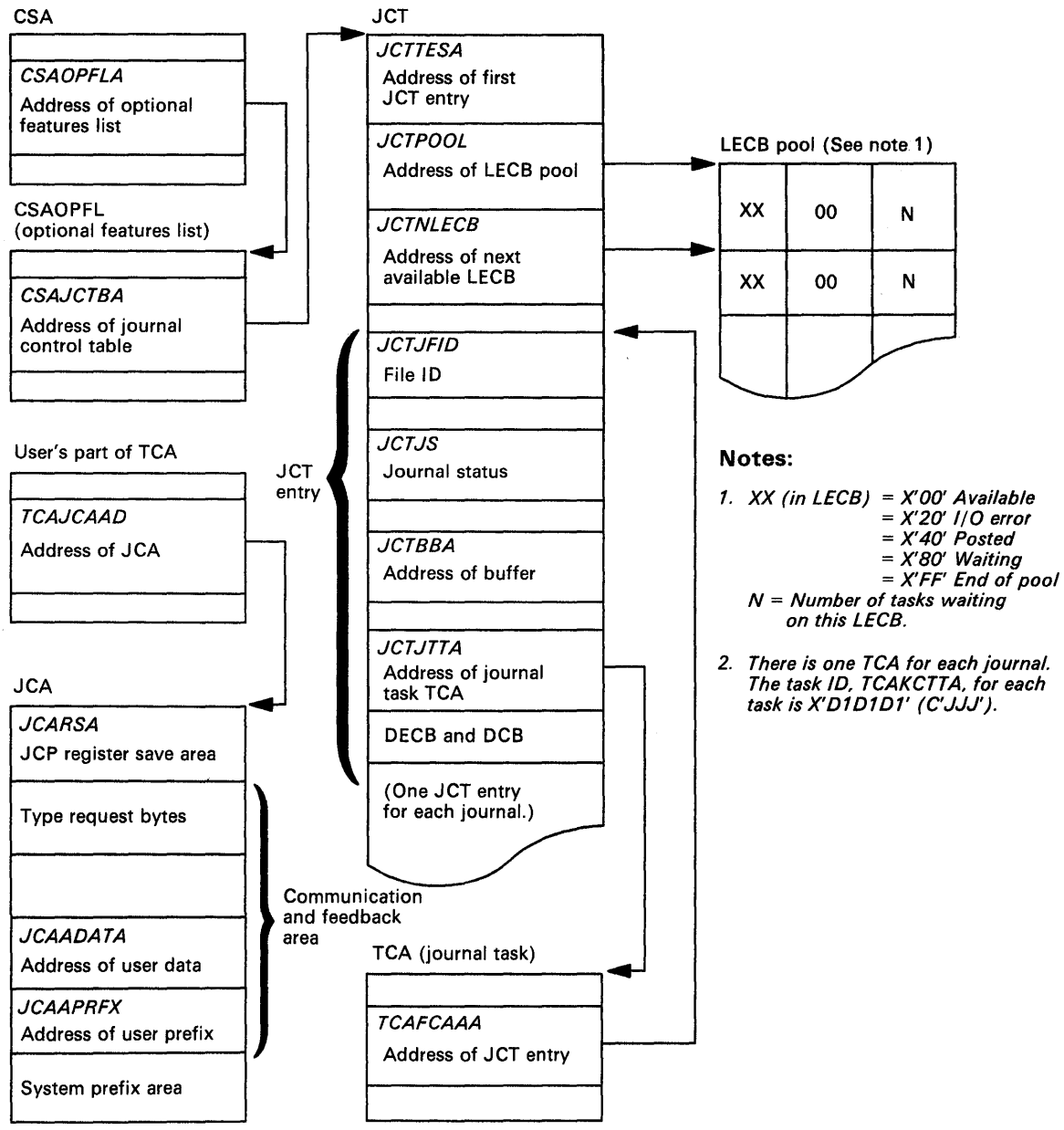


Figure 29. Control blocks associated with journal control

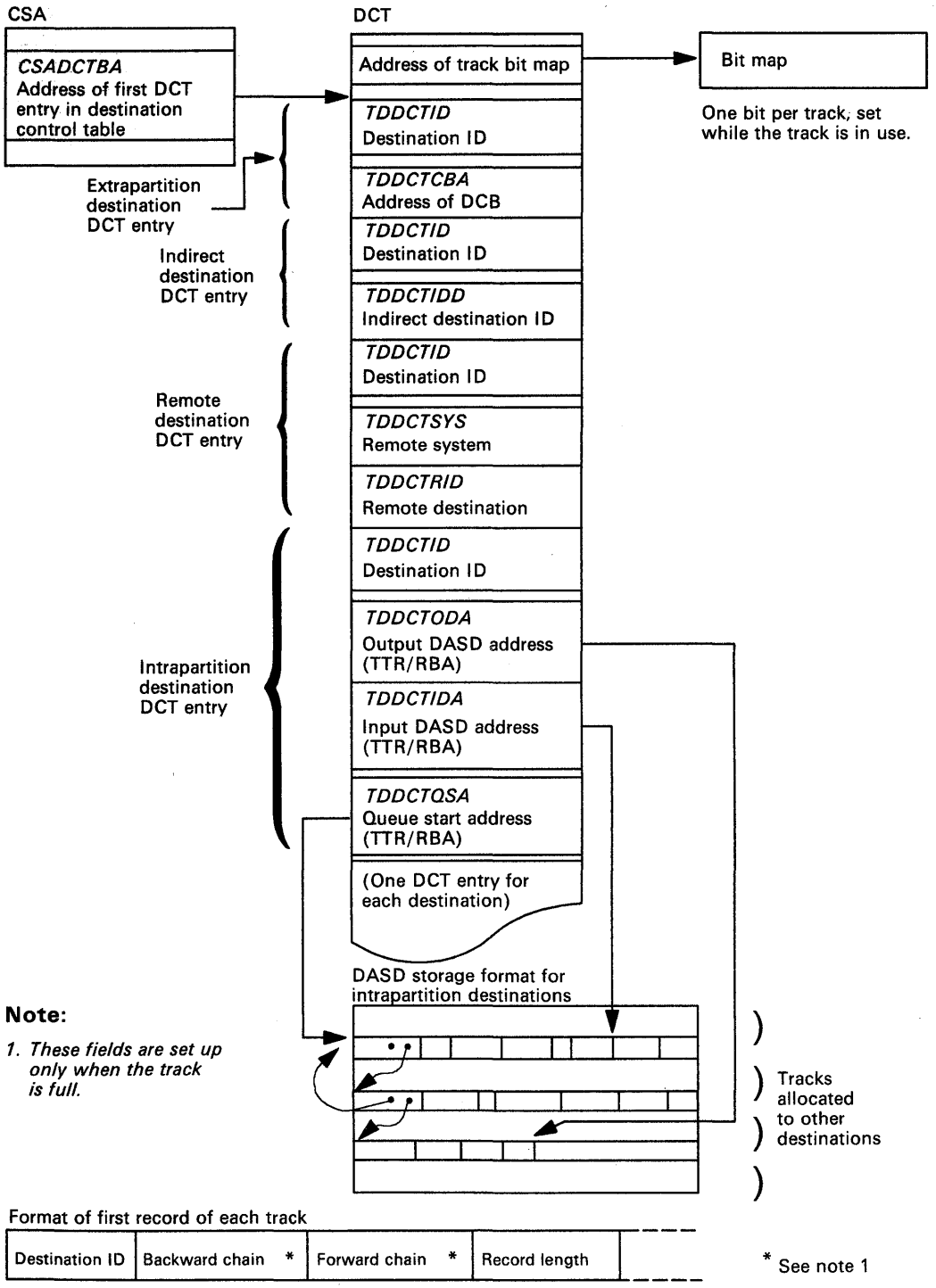


Figure 30 (Part 1 of 2). Control blocks associated with transient data (destination control table)

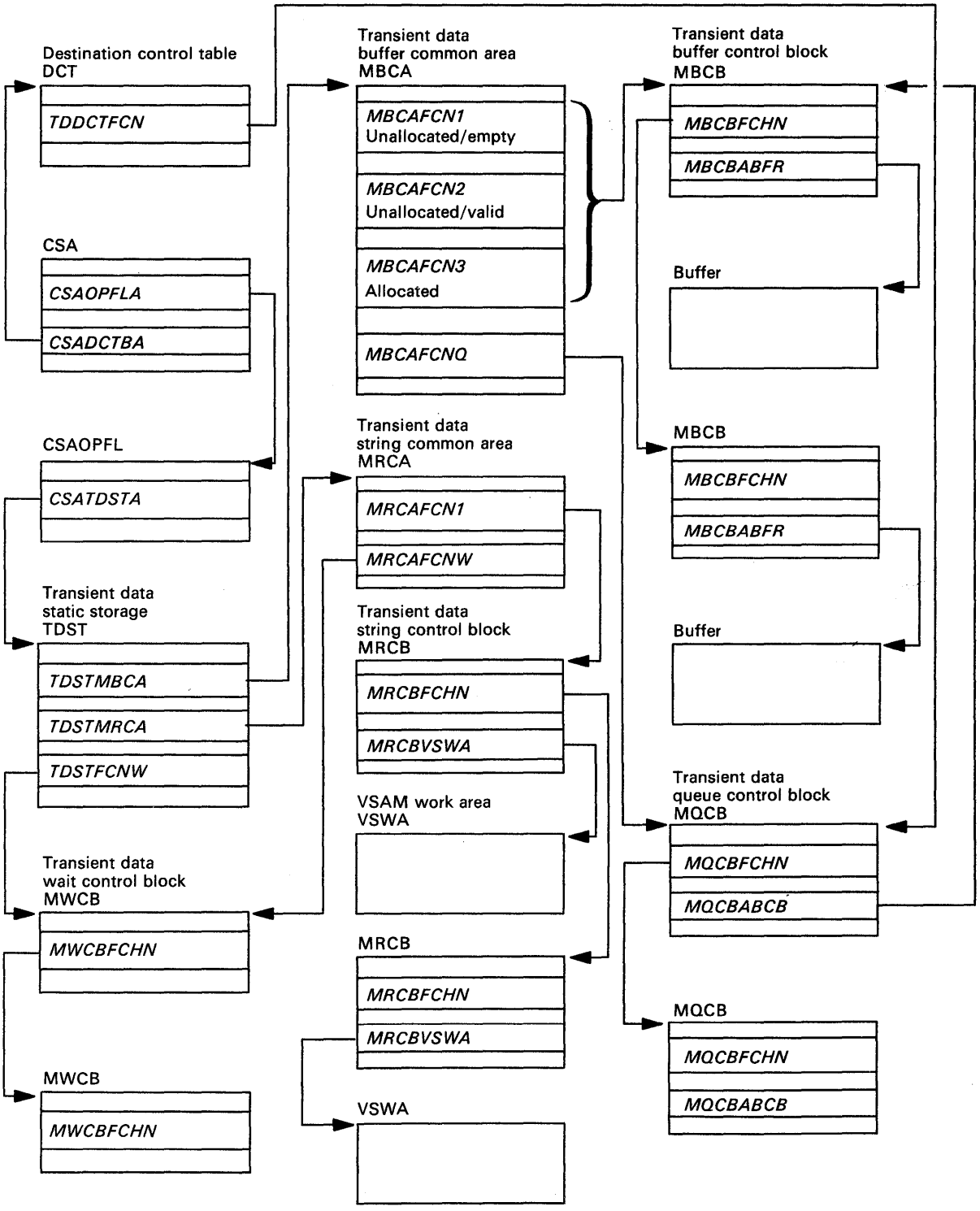


Figure 30 (Part 2 of 2). Control blocks associated with transient data (destination control table)

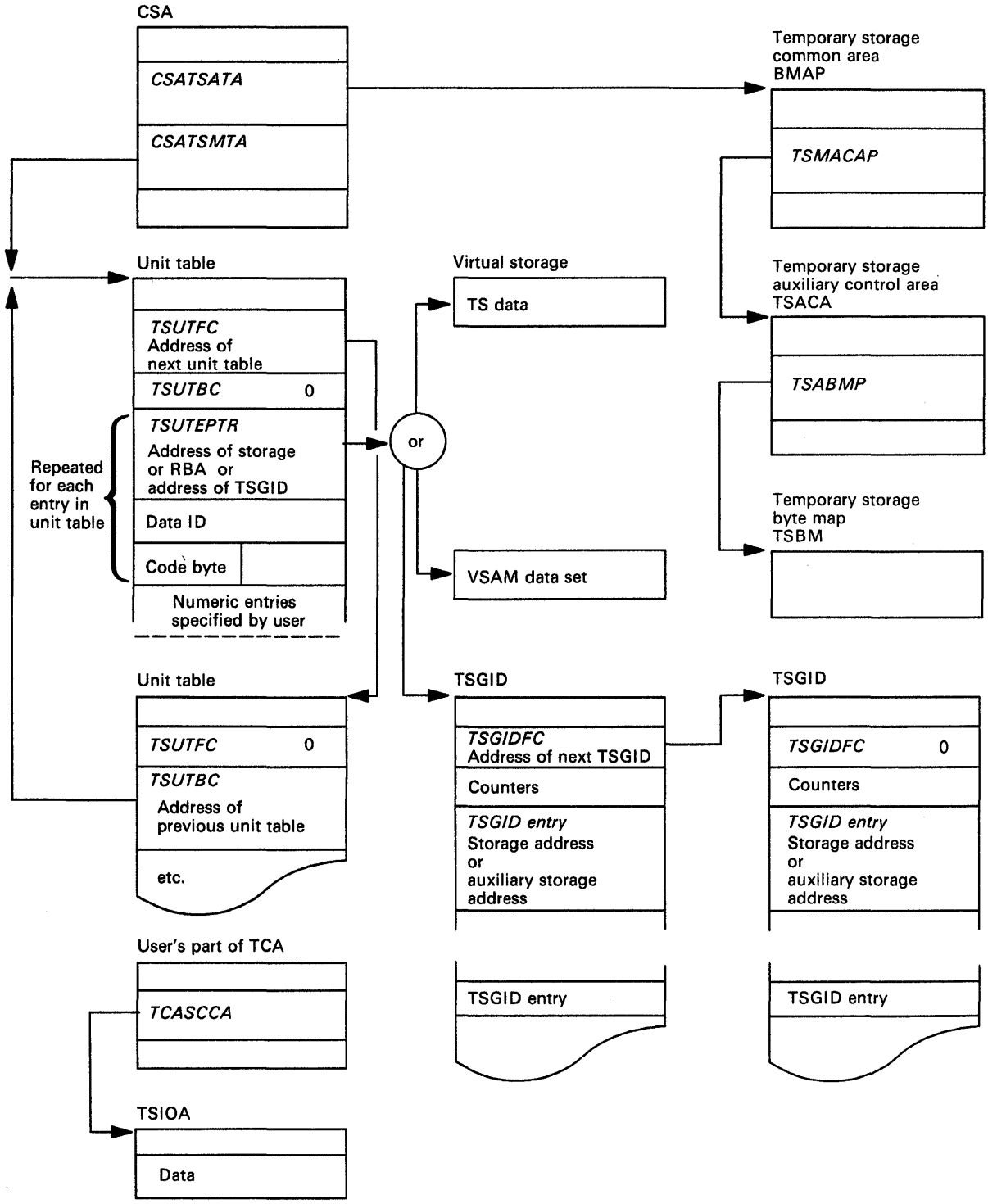


Figure 31 (Part 1 of 2). Control blocks associated with temporary storage control

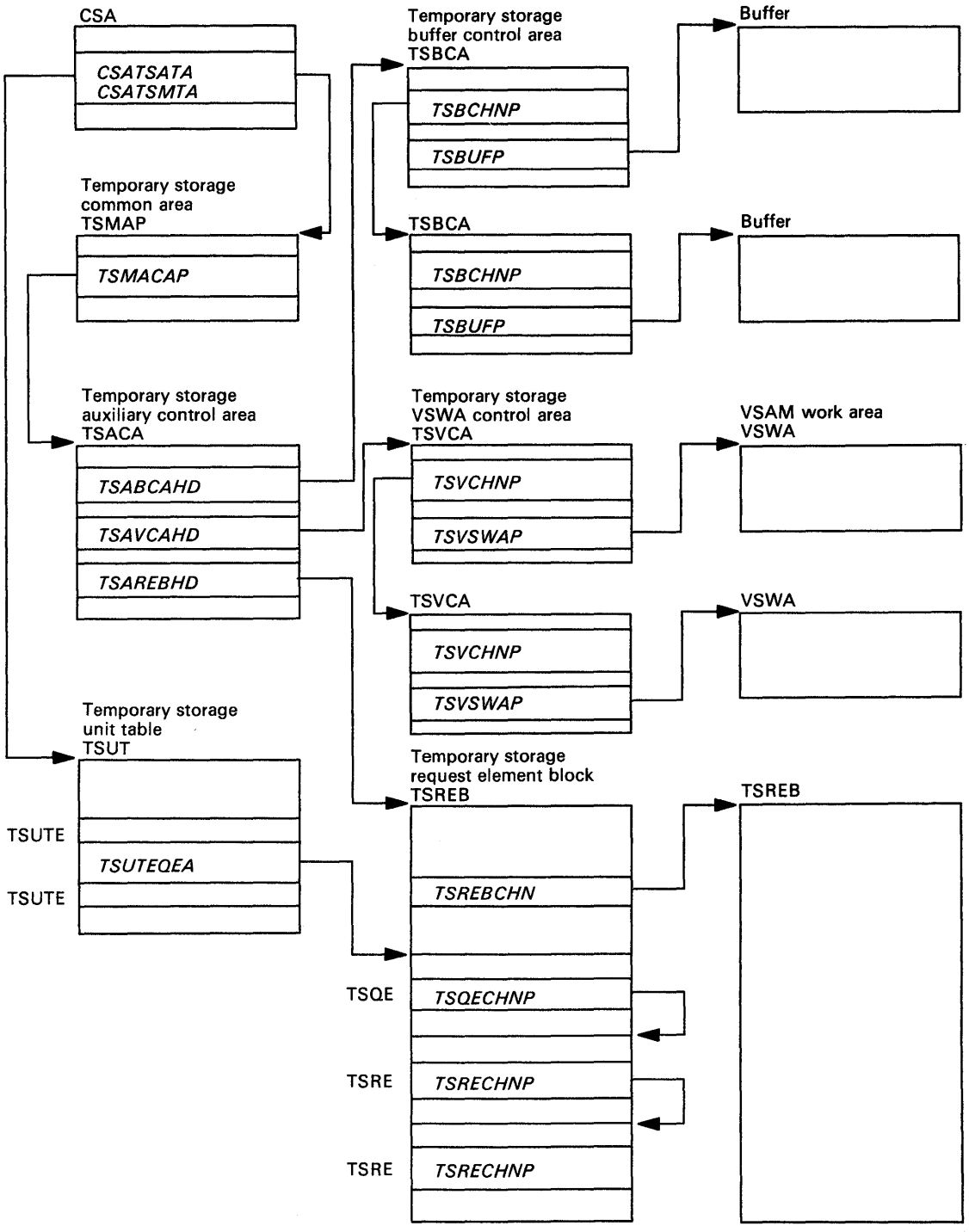


Figure 31 (Part 2 of 2). Control blocks associated with temporary storage control

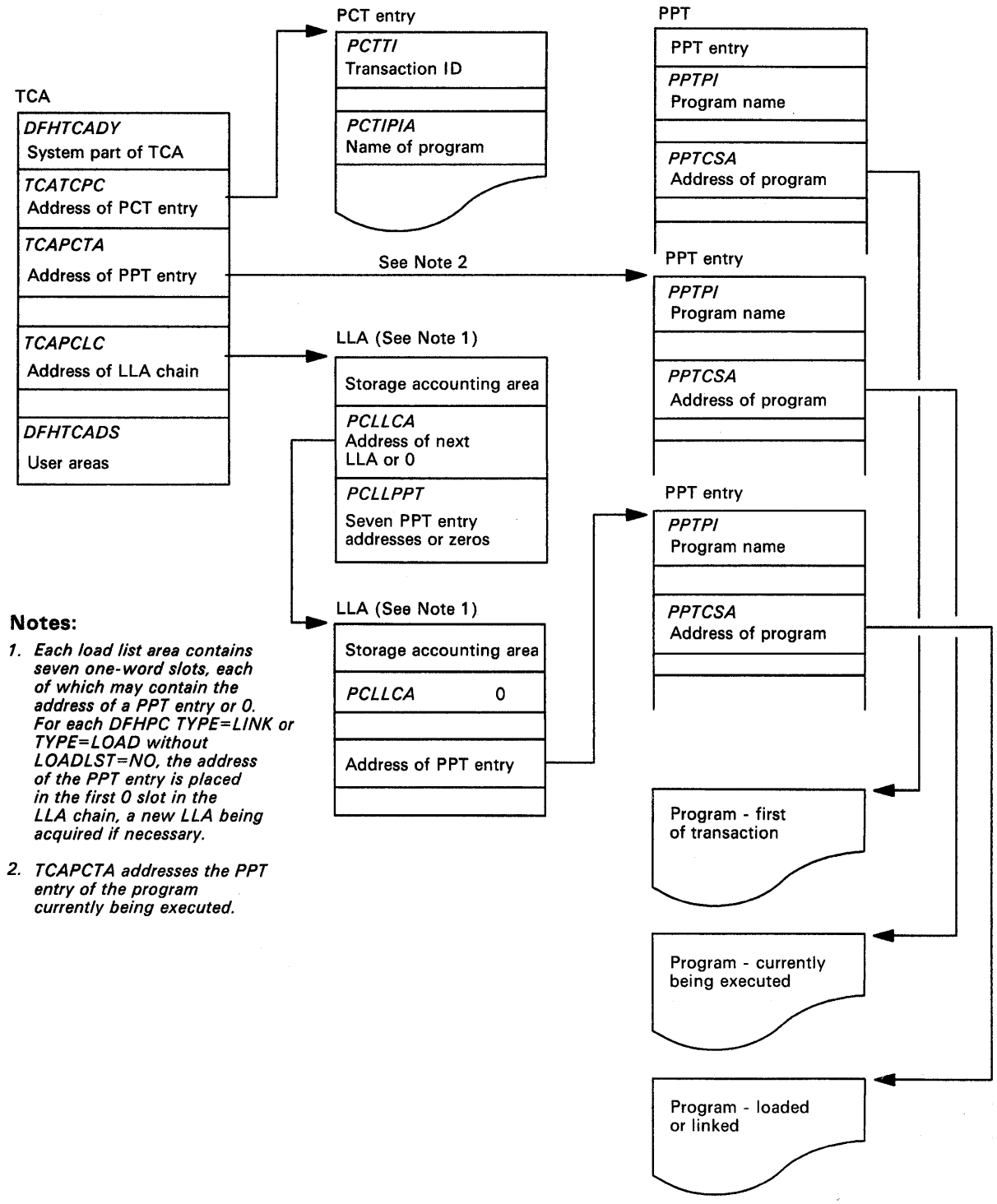


Figure 32. Control blocks associated with processing program table

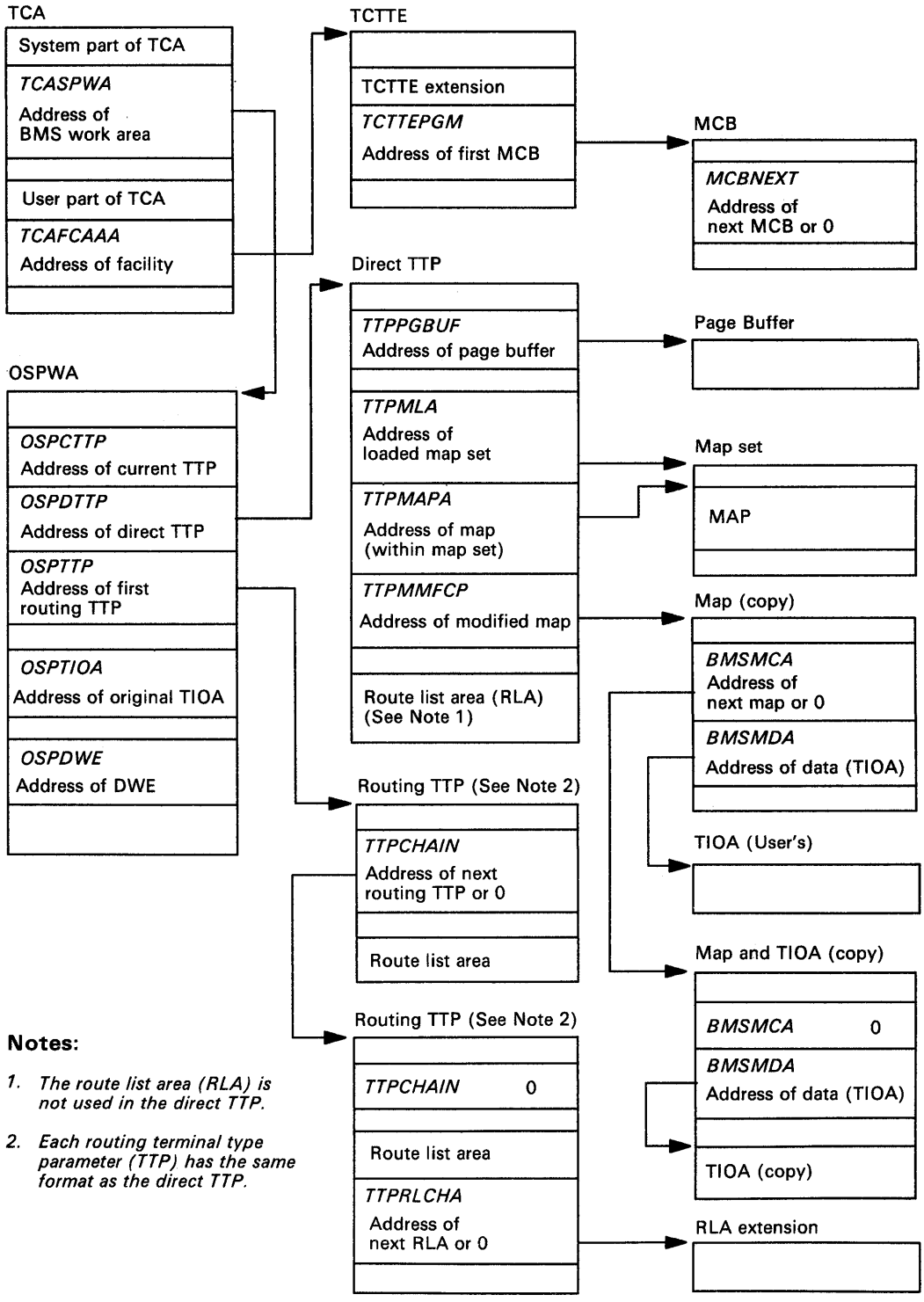


Figure 33. BMS control blocks

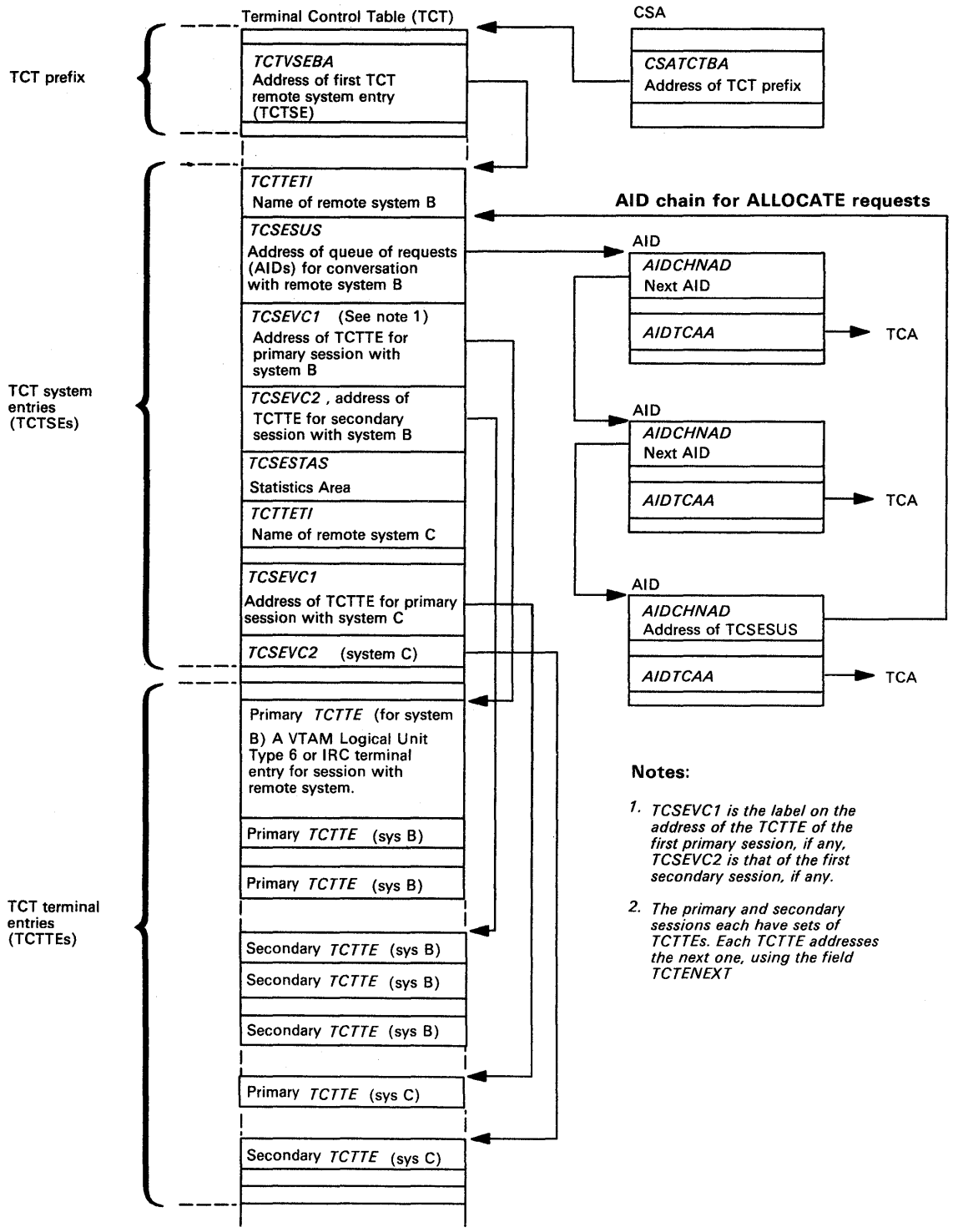


Figure 34 (Part 1 of 2). Control blocks associated with CICS intercommunication facility (non-LU6.2)

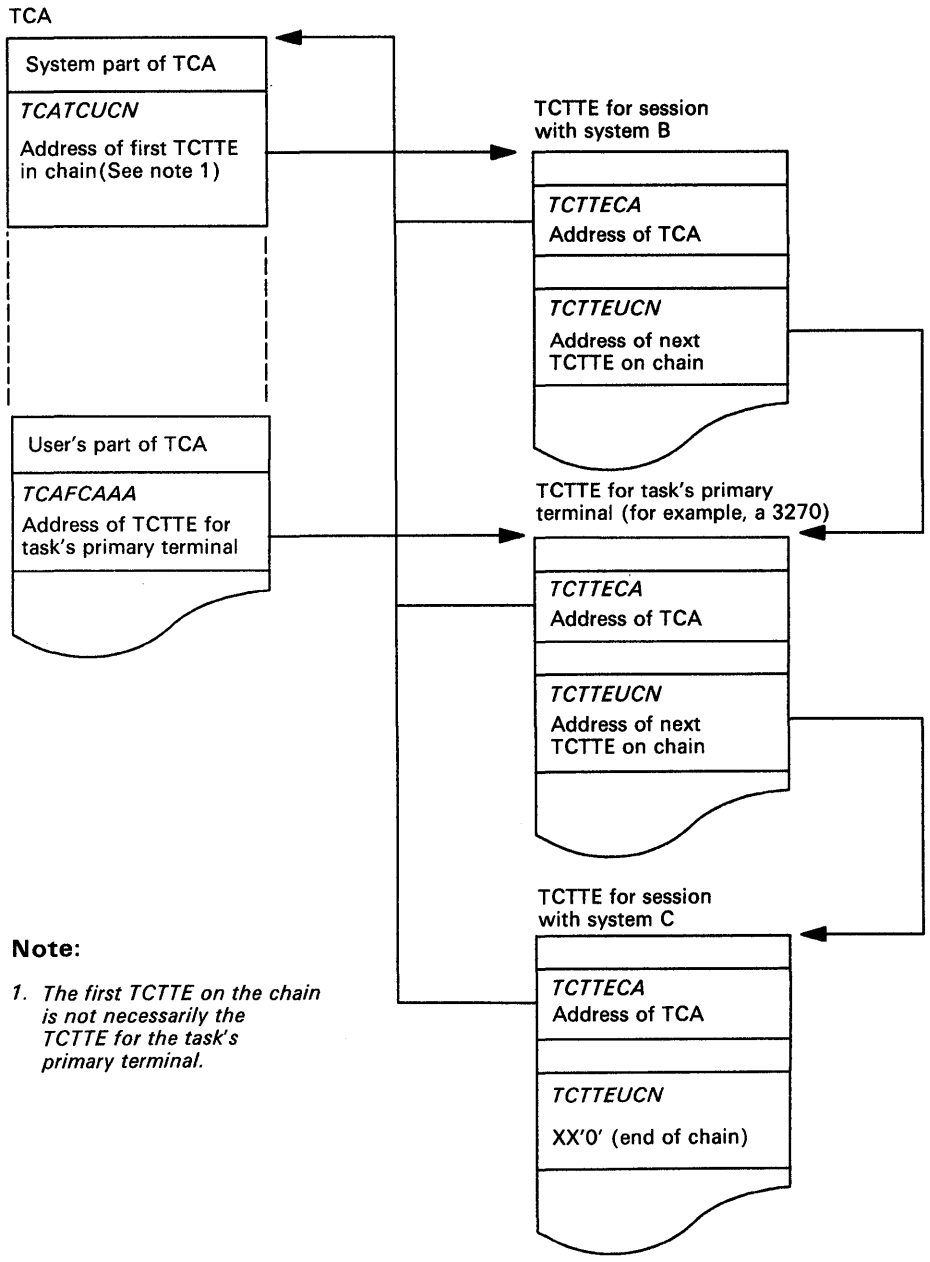


Figure 34 (Part 2 of 2). Control blocks associated with CICS intercommunication facility (non-LU6.2)

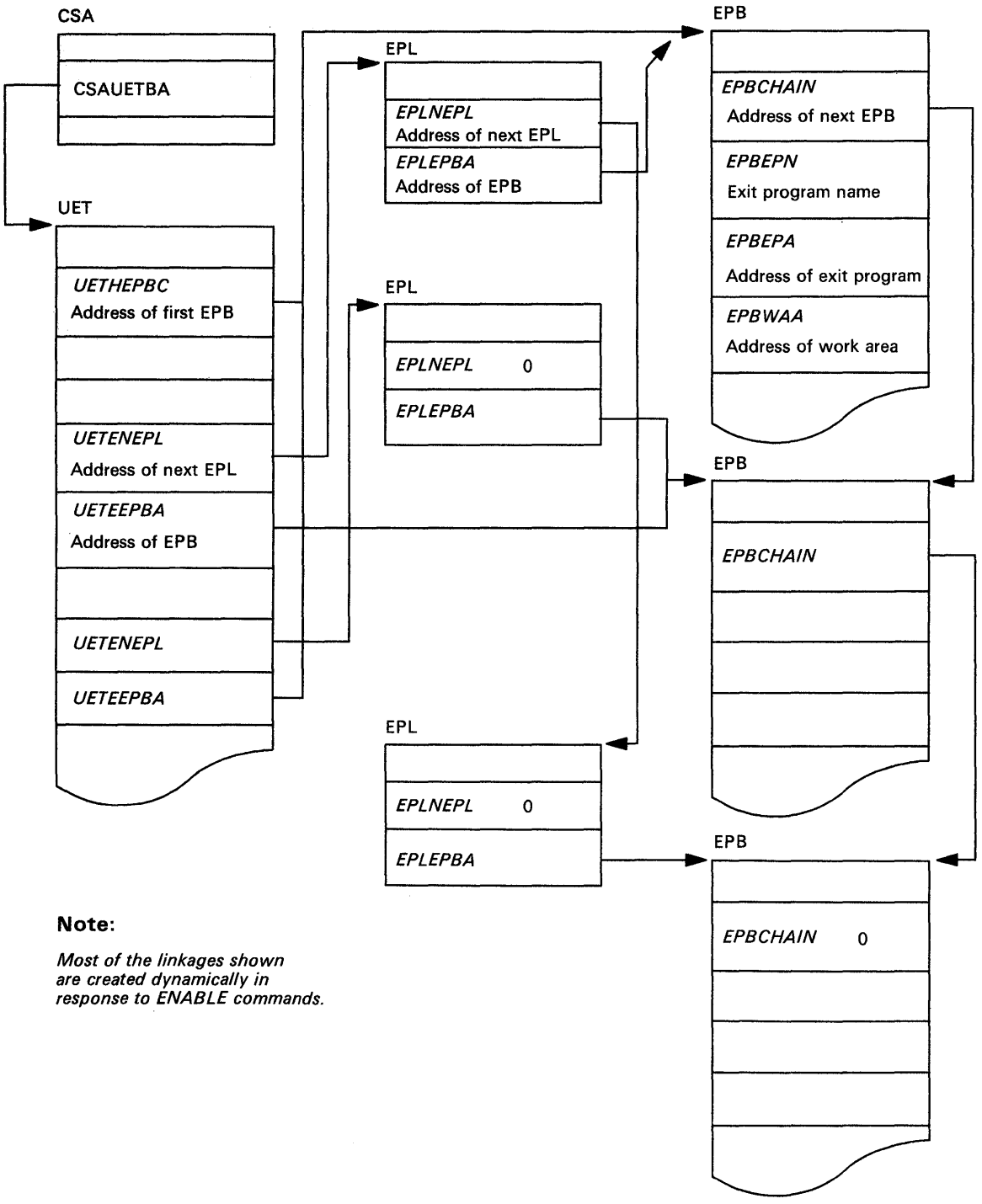


Figure 35. Control blocks associated with the user exit interface

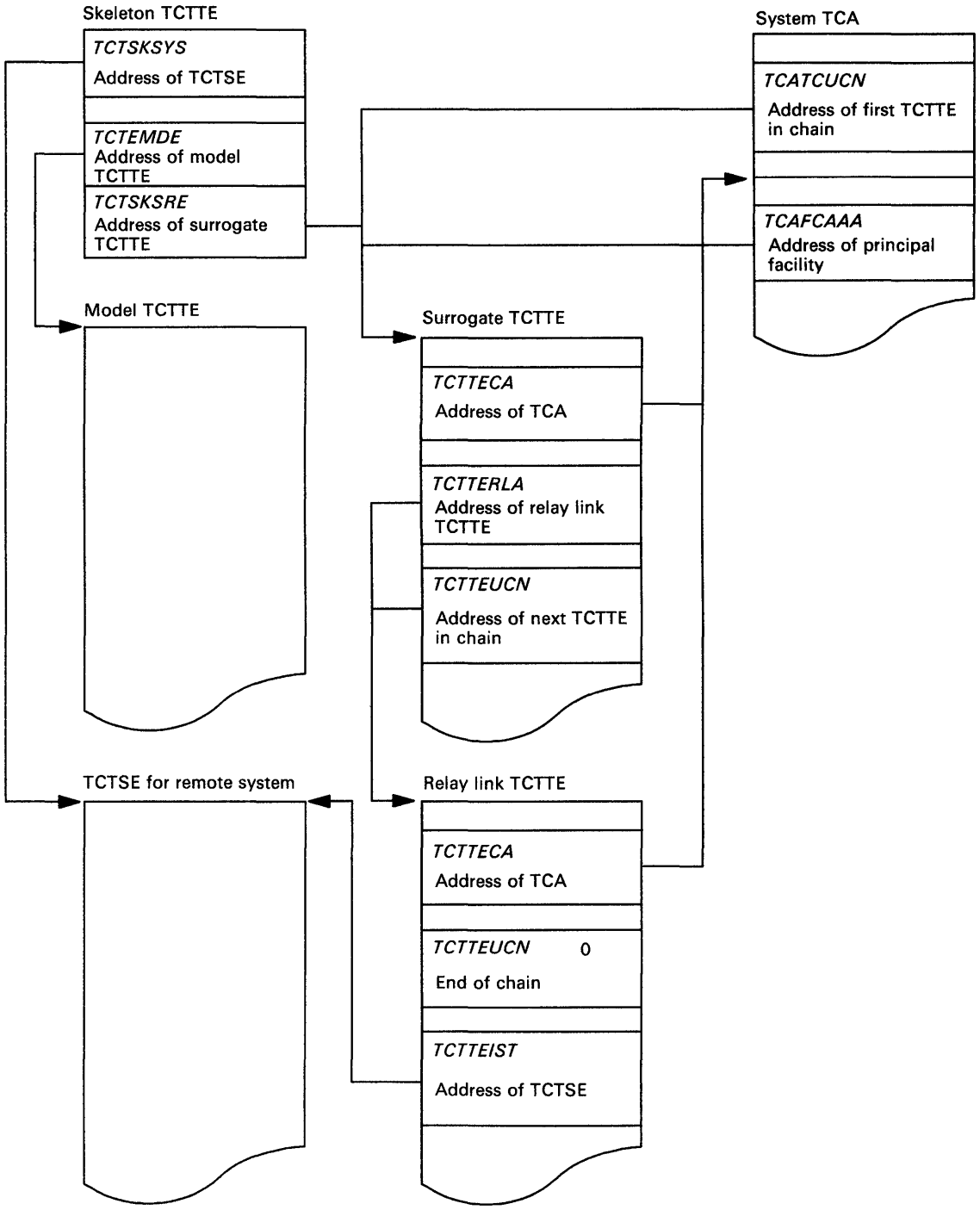


Figure 36. Control blocks associated with transaction routing

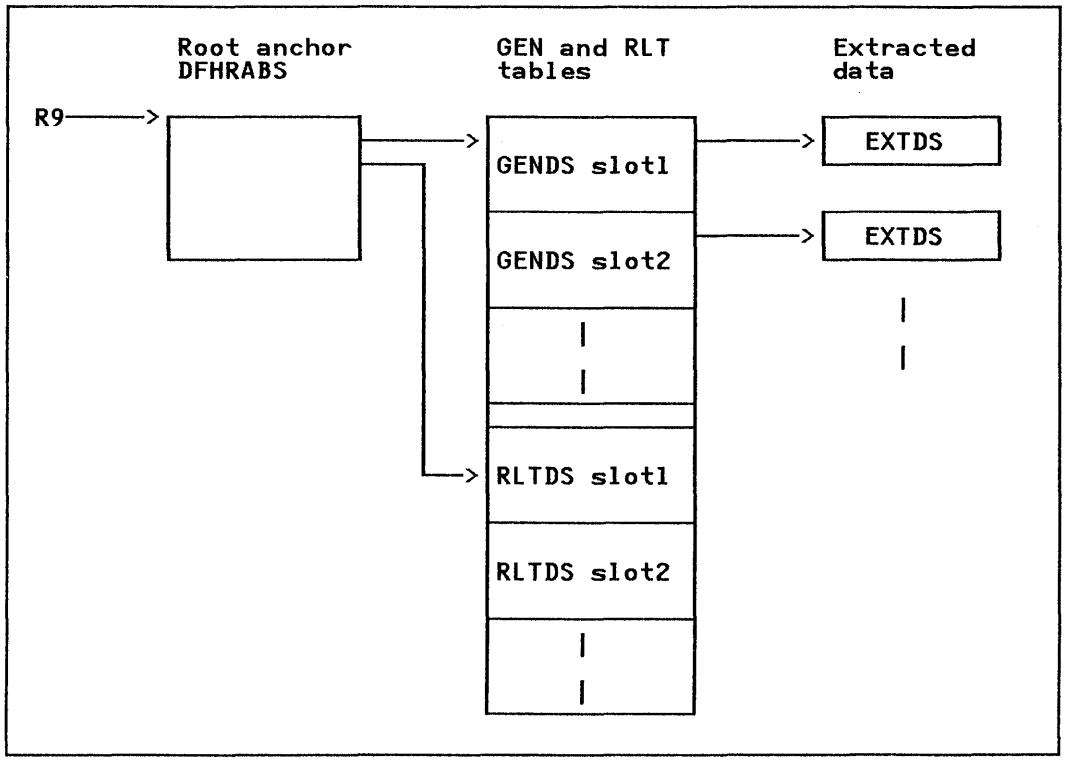


Figure 37. Overseer control blocks

Debugging the overseer sample program

This section helps you diagnose problems in the overseer sample program (DFH$AXRO). If the overseer detects an error, it will either put out a diagnostic message, or take a snap dump. The operator can request a snap dump by using the S command.

Figure 37 on page 168 shows the basic control blocks of the overseer with DSECT names.

If the overseer program fails, a message may be written to the console, and a dump produced. Certain failure return and reason codes are saved within the RABDS control block, and individual GENDS control blocks (if the error is connected with a particular GEN).

The following points may help you avoid overseer problems:

- Some overseer requests are asynchronous, and post a user ECB when they complete. You should therefore be careful not to reuse user areas before the events complete.
- There is one system key area, and one user key area, managed by the overseer authorized services. See DFHWOSB for the assembler language DSECTs and the logic.

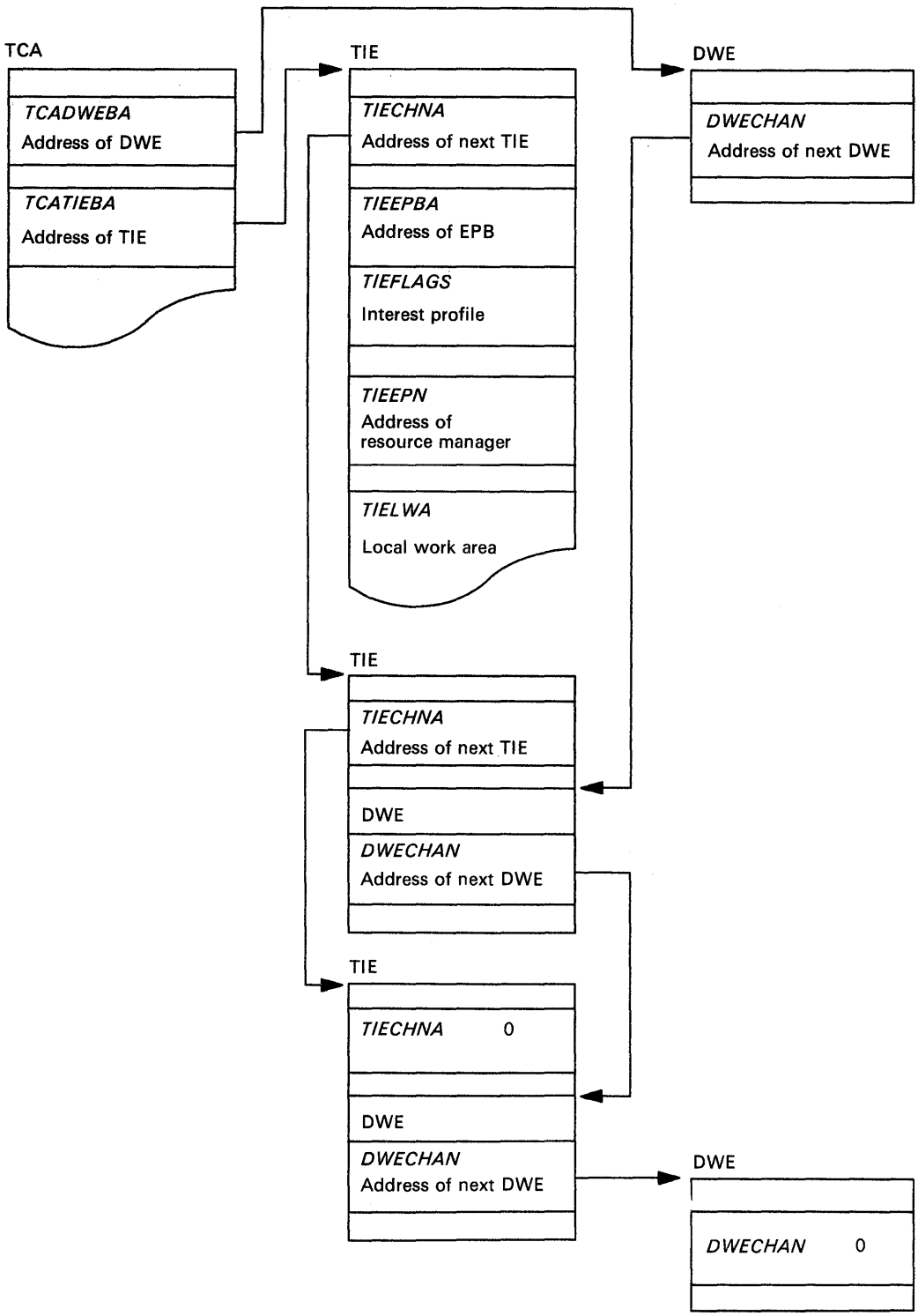


Figure 38. Control blocks associated with the task-related user exit interface

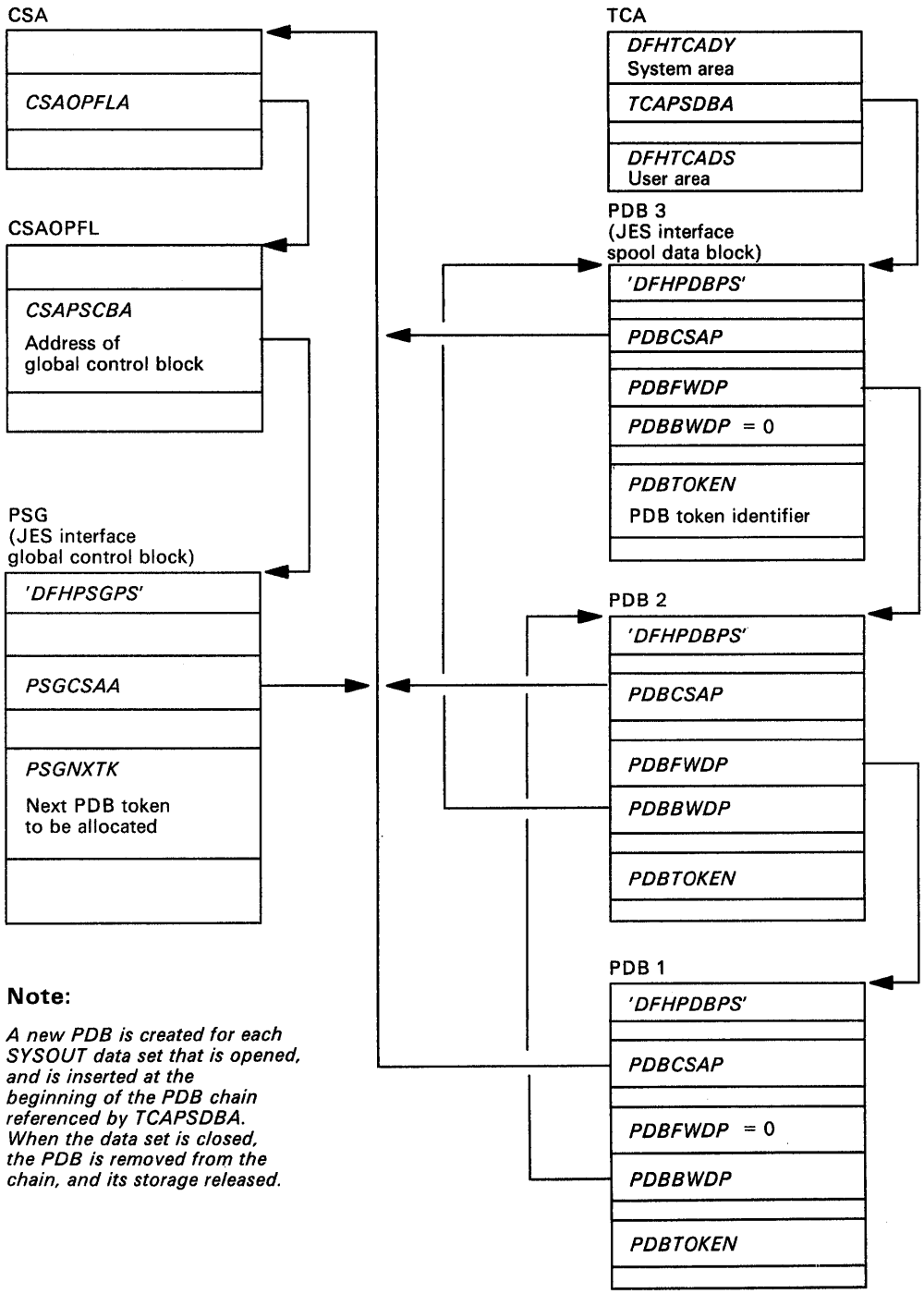


Figure 39. Control blocks associated with the JES interface

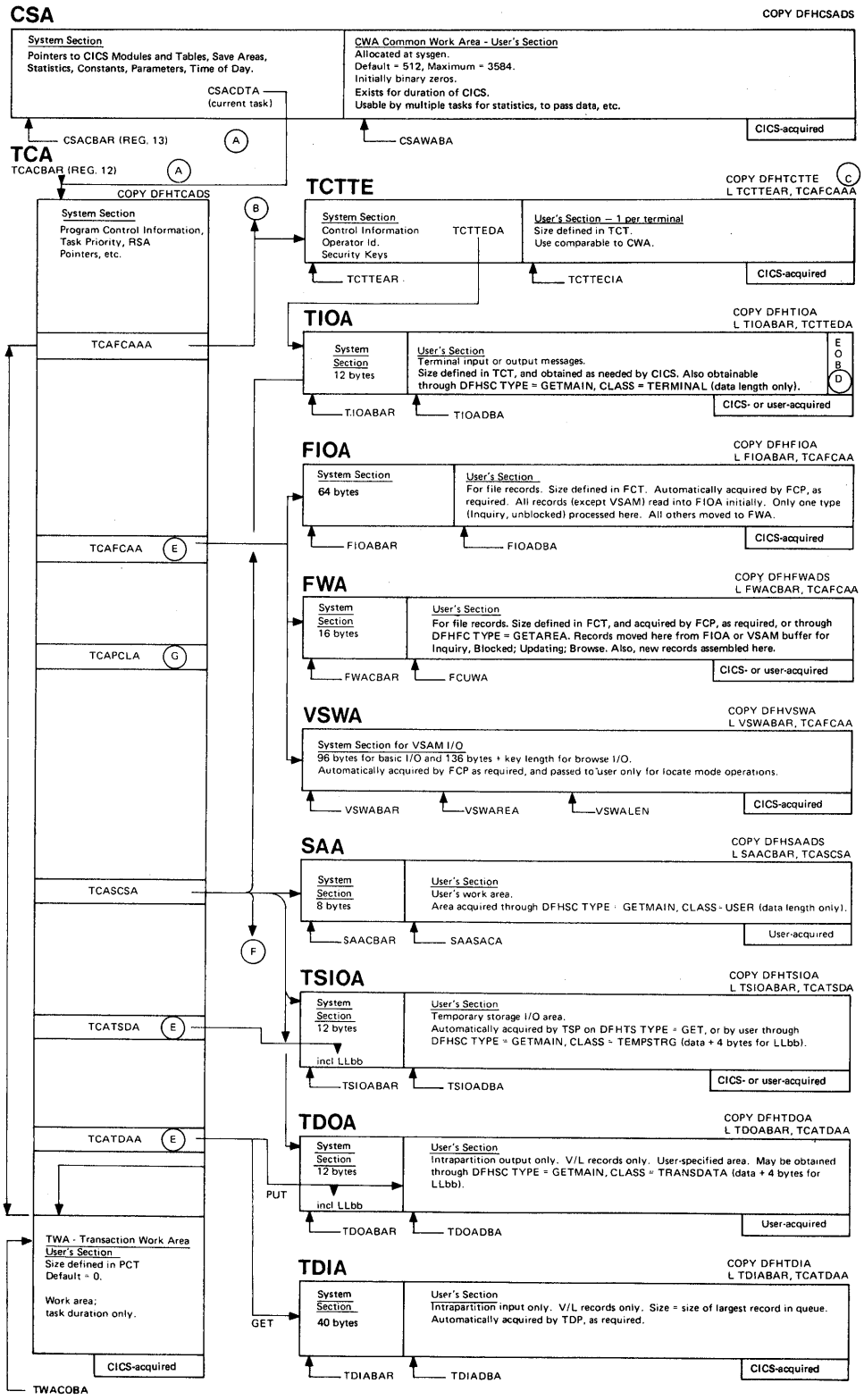


Figure 40. CICS areas

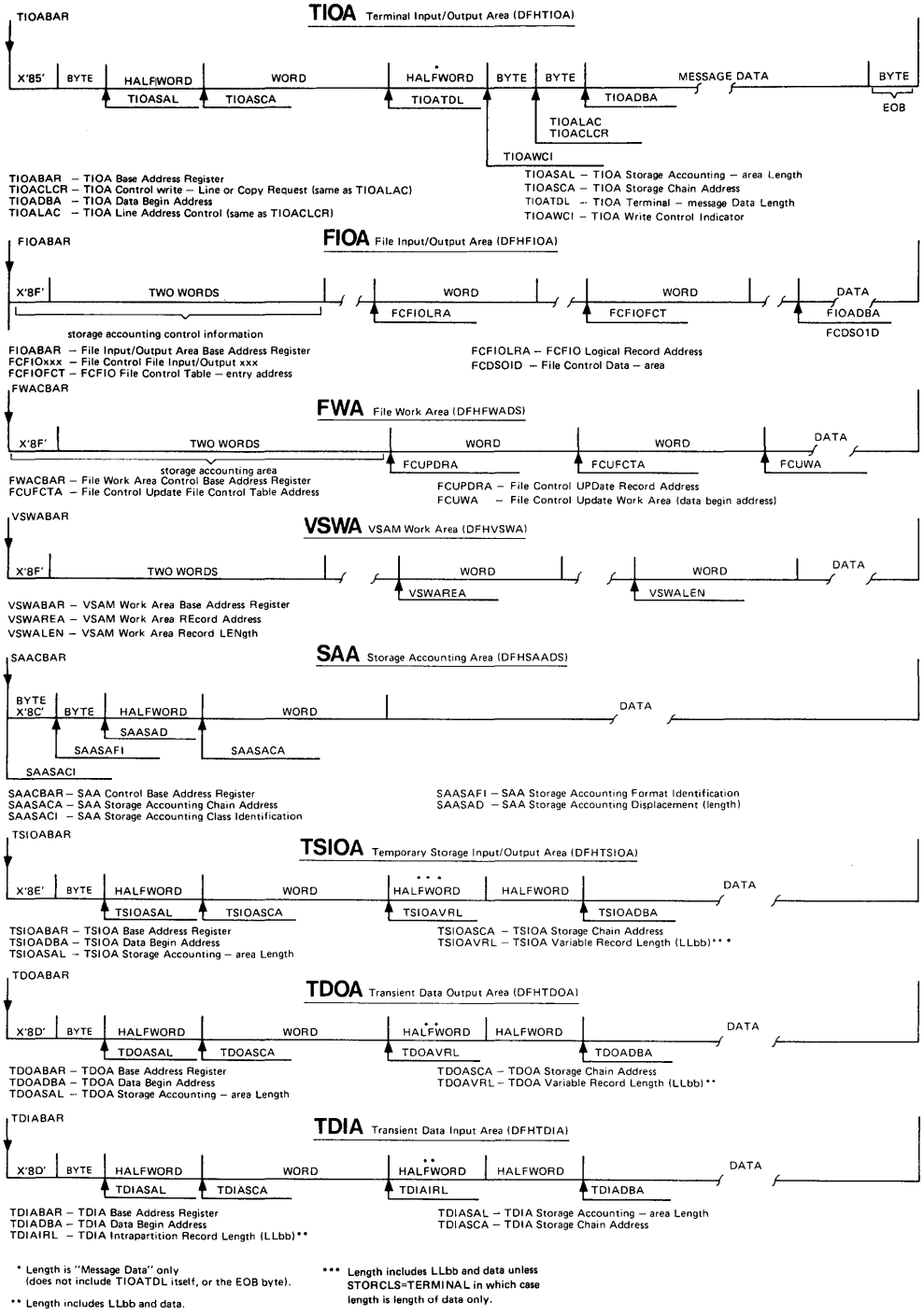
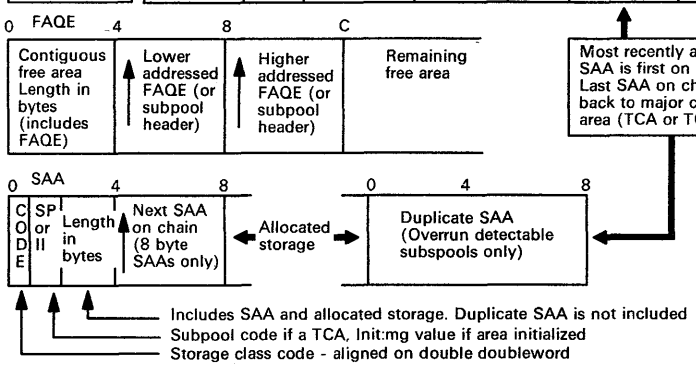
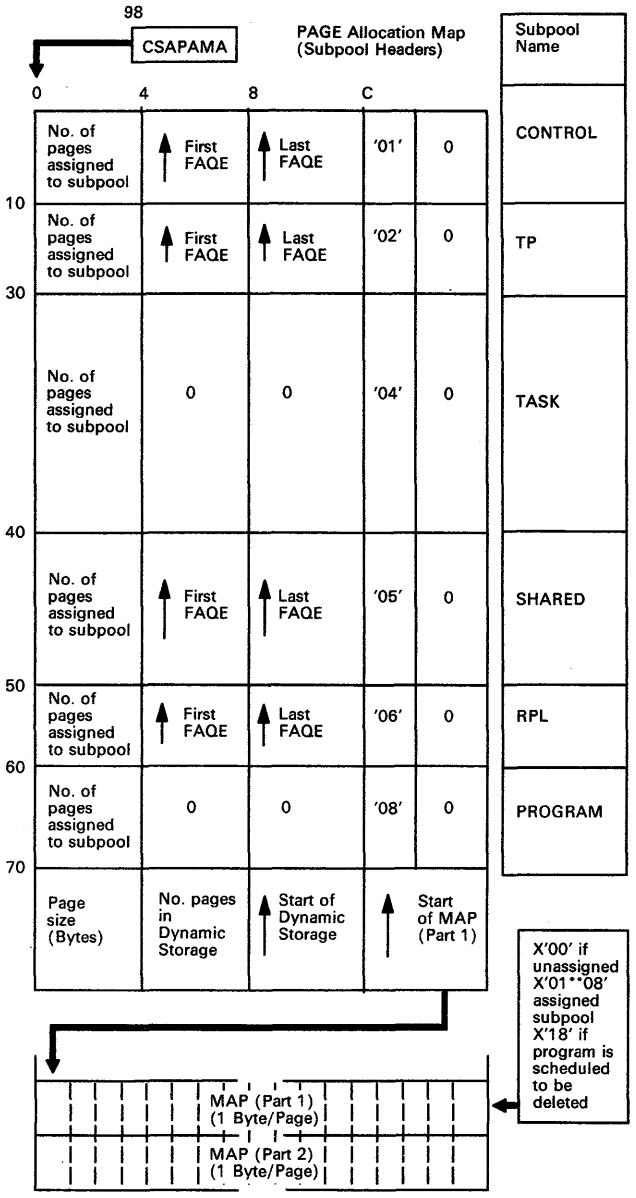


Figure 41. CICS areas — control sections

Figure 42. Dynamic storage accounting reference chart

Subpool Name	Storage Class	Storage Code	Subpool Page Assignment Direction	Subpool Storage Allocation Algorithm	Storage Accounting Area (SAA) Bytes	Storage Accounting Area (SAA) Chained off	Overrun Detection & Correction (Dupl SAA)	Free Area Queue Element Chained off
CONTROL	AID CONTROL DCA ICE QEA	87 94 81 86 82	Lowest addressed available page	First Fit	4	None	No	Subpool Header
TP	LINE TERM or TERMINAL	84 85	Lowest addressed available page	First Fit	8	LINE SELF TERM TCTTE (Push down)	Yes	Subpool Header
TASK	DWE FILE JCA LLA MAPCOPY RSA TCA TEMPSTRG or TS TRANS-DATA or TD USER	9D 8F 9B 8B 9E 89 8A 8E 8D 8C	Task initially assigned 1 page lowest addressed available page	First Fit TCA from low address Others from low address of available	8	TCA (Push-down)	Yes	Each Task's TCA (Syst Area) 08 ↑ First FAQE 0C ↑ Last FAQE
SHARED	MAP SHARED TACLE TSMAIN TSTABLE DL/I	99 93 96 97 98 9F	Lowest addressed available page	First Fit	4	None	No	Subpool Header
RPL	RPL	90	Lowest addressed available page	First Fit	4	None	No	Subpool Header
PROGRAM	PROGRAM	88	Highest addressed available page(s)	Full Pages	8	None	No	None





Chapter 2.5. The CAVM data sets

The control and message data sets, used by the CICS availability manager (CAVM), are both VSAM ESDS data sets which are accessed using control interval processing with user buffering. The control intervals (CIs) have the standard VSAM format with CI definition fields (CIDFs) and record definition fields (RDFs). All data in a given control or message data set relates to a single generic APPLID.

The **control data set** contains a CI for recording CAVM's view of the overall state of the generic APPLID (the state management record), and a status CI for each potential member system (the active and alternate systems).

The **message data set** contains secondary status CIs, to be used when a system is unable to write its status CI in the control data set, and enough message CIs to hold the maximum anticipated backlog of messages from the active system to the alternate system.

At most times, when the active and alternate systems are running normally, serialization of access to the data sets is unnecessary, because the state management record remains unchanged, and all other CIs in the two data sets are owned (updated) by only one of the member systems. However, when the state changes, at SIGNON, TAKEOVER or SIGNOFF, the state management record is changed, and ownership of other CIs may also change. RESERVE is used to serialize updates of the state management record, and any changes of ownership of other CIs are effected under this RESERVE. DEQ is used to free the data set, so that any system can update it again.

The contents of the data sets are described in more detail in the following sections. The contents may be useful when the problem cannot be easily ascribed to the active system or the alternate system. For example, a terminal might be lost (not switched), or the system waiting indefinitely.

Contents of the control data set

The control data set contains the following sequence of CIs:

1. Control CI
2. State management record CI
3. Primary active system status CI
4. Primary alternate system status CI.

The control CI is used for validation. It contains:

```
Bytes      Contents
 0- 4      CAVM data set format number
 8-15      DD name (DFHXRCTL for control data set,
                    DFHXRMSG for message data set)
16-23      Generic APPLID
24-43      Unique identification comprising:
            24-31    TOD clock at time of formatting by CAVM
            32-43    MVS ID of formatting system
                      (SMF ID and IPL time and date)
44-        Zeros
```

A control CI with the same format is also present in the message data set. During the CAVM SIGNON procedure, CAVM writes both CIs as part of its formatting operation if both data sets were previously unused. Otherwise, it validates the two CIs by checking that:

- The CAVM code level and data-set format are compatible.
- They have the same unique identification data.
- The DD name value is appropriate for the intended use of the data set.
- The generic APPLID is the same as that of the system attempting to sign on.

The state management record contains information about the overall state of the generic APPLID. It consists of a fixed part, followed by a number of job description parts, one for each potential member system. The format of the fixed part is defined by the DSECT SMDESCR in DFHWSMDS. The format of a job description part is defined by the DSECT WSJDESC in DFHWSMDS.

The status CIs contain information about a particular member system. Unlike the state management record, they are continually updated. The status data may be recorded in a corresponding CI in the message data set if a system is for some reason unable to update its control data set status CI. The preferred place for status information is the control data set.

Each status CI consists of a fixed part which describes that system, and may be of interest to any partner system. This is followed by parts containing information directed to a specific partner system. Since the implementation allows only one alternate system, there is in fact only one instance of these specific parts in each CI.

The fixed part of the status CI consists of a 12-byte status record header, followed by data described by the DSECT WSAS in DFHWSADS. The format of the status record header is:

```
Bytes      Contents
 0- 7      TOD clock value at time of issuing write request
 8-11      Write sequence number
```

The sequence numbers of the status CIs in the control data set are independent. However corresponding status CIs in the control and message data sets share a sequence

number. Except for the special case when both contain their initial value (zero), corresponding status CIs should always contain different sequence numbers with the higher value in the CI containing the more up-to-date date.

The following part of the status CI contains in order:

1. Data described by DSECT WSAR in DFHWSADS.
2. Either invalidation data for the alternate system (DSECT WSARIV in DFHWSADS) if it is an active system status CI, or a takeover message for the active system (DSECT WSARTM in DFHWSADS) if it is an alternate system status CI.
3. Expedited message data from PUTREQ and PUTRSP requests (DSECT WSARQR in DFHWSADS).

Contents of the message data set

The message data set contains the following sequence of CIs:

1. Control CI
2. Unused CI (set to zeros)
3. Secondary active system status CI
4. Secondary alternate system status CI
5. Message data CIs up to the end of the space allocated.

The control CI and status CIs have already been described in the section on the control data set.

The message data CIs form a wraparound store of message records generated as the result of PUTMSG requests issued by the active system to the CAVM message manager. The active system's write cursor and the alternate system's initial-read cursor are both located in the active system's status CI (see the DSECTs WSAR and WSAS in DFHWSADS.) The alternate system's current-read cursor is located in the alternate system's status CI (see the DSECT WSAR in DFHWSADS.)

The CIs are in standard VSAM format, and each message is represented by a single non-spanned ESDS record. CIDF and RDF control fields are maintained accordingly. The message manager does not attempt to combine RDFs for adjacent records of equal length, consequently there is one RDF for each record in the CI.

Each message CI contains:

1. A control record
2. Zero or more message records.

The control record format is defined by the DSECT WMRCR in DFHWMRPS.

The message record format is defined by the DSECT DFHWMRPS.

Chapter 2.6. Debugging application programs

This chapter discusses the debugging of application programs with the aid of a dump. Before using the information in this chapter for command-level programs, the reader should first have attempted to discover faults in the application program with the aid of the execution diagnostic facility (EDF).

When debugging a CICS application program which shares an IMS/VS DL/I data base with an IMS/VS batch program, see "Shared data base problems" on page 227.

The source language of a failed program can be determined from the settings of bits 2, and 3 in the PPTTLR byte of the PPT:

```
PPTTLR bit 2 on means a PL/I program
PPTTLR bit 3 on means a COBOL program
These bits turned off means an assembler-language program
```

The dynamic storage area associated with a particular task's execution of an application program (in any source language) is addressed by field TCAPCDSA in the system part of the TCA. (For PL/I programs, TCAPCDSA is also labeled TCAPCPA; and for COBOL programs is also labeled TCAPCCA.) The way in which the dynamic storage area is used depends on the source language and whether the command-level or macro-level CICS interface is being used.

The standard save area consists of 72 bytes as follows:

```
Bytes  1-4     Flags
Bytes  5-8     Backward chain address
Bytes  9-12    Forward chain address
Bytes 13-72    Registers 14 to 12
```

The following sections describe how to debug COBOL, PL/I, and assembler programs with CICS. VS COBOL II is always referred to by its full name and other versions of COBOL simply called COBOL.

COBOL application programs

VS COBOL II application programs

If the FDUMP compiler option is used for a VS COBOL II program, VS COBOL II diagnostics, including a formatted dump, are written to a temporary storage queue called CEBRname, where "name" is the name of the terminal. You can look at this queue using the CEBR transaction either by entering CEBR directly or invoking it by the EXEC diagnostic facility (EDF). The diagnostics include the abend code, an abend information message, the PSW and registers at the moment of abend, the program ID(s), the date and time of the compilation of the program(s), the TGT and the contents of all the base locator and index cells (or indications that there are none). The formatted dump refers to data by the names used in the PIC and USAGE clauses in the program. How to execute the CEBR transaction is described in the *Application Programmer's Reference.* To find a description of the contents of the temporary storage queue, see the *VS COBOL II General Information* and *VS COBOL II Debugging* manuals.

Using EDF, you can press the PF5 key to display the WORKING STORAGE and offsets from the beginning of WORKING STORAGE. If you used the VS COBOL II compiler option MAP, you can locate any data in the display. In the VS COBOL II map, find the BLW number (B) and the displacement (D) of the data. The displacement of the data from the start of the WORKING STORAGE is (B*X'1000') + D.

In a CICS dump, the characters C2THDCOM, C2RUNCOM and C2TGT+48 are at the beginning of the corresponding areas. THDCOM points to chains of RUNCOMs and RUNCOMs point to TGTs. A TGT always starts to with the word X'0010xy01', where the first bit of 'x' is '1'. The base locators for WORKING STORAGE (BLWs) in the TGT address the WORKING STORAGE.

All COBOL application programs

COBOL load modules contain at least the following fields:

- Command-level programs (only) begin with the EXEC interface processor DFHECI. See the link-edit listing to find the length of this processor.
- INIT1 – executable code to initialize registers and address constants to other fields and instructions, and invoke INIT2.
- DATA AREA – working storage, various constants and control blocks.
- TGT – the task global table contains many variables, including the register save area and the BLL cells, which point to areas defined in the linkage section.
- PGT – the program global table contains address constants and literals that are constant throughout program execution.
- PROCEDURE – code compiled from the Procedure Division.
- INIT2 – chains the TGT save areas to the caller's save area.

- INIT3 – on the first invocation, relocates each address in the TGT and PGT by adding the address of the beginning of the program as loaded, loads base registers, and then branches to the first instruction of the PROCEDURE.

Dump areas to be considered in COBOL program diagnosis

In the dump of a COBOL program, the main areas to be considered are:

- The program itself, of which there is one copy
- The COBOL area – a CICS dynamic storage area that is allocated by task.

Location of COBOL dumped areas

The dumped program

The dumped COBOL program is addressed by PPTCSA in the PPT, or by the appropriate SCP trace entry for PROGRAM class. The register save area INIT1 + X'48' (covering registers 0 through 14) should have register 12 pointing to the PGT, register 13 pointing to the TGT, and some others to locations in the data area and compiled code of the program storage. If not, a CICS error is indicated.

For each invocation of the COBOL program, CICS copies the **static** TGT from program storage into CICS **dynamic** storage (the COBOL area) and uses the dynamic copy instead of the static one. (For command-level COBOL programs only, CICS also copies working storage from program storage to the COBOL area, above the TGT.) Task-related COBOL areas thus enable the single copy of a COBOL program to multithread as if it were truly reentrant.

The dumped COBOL area

The COBOL area is addressed by TCAPCDSA (TCAPCCA) in the system part of the TCA (and forms part of the transaction storage chain). For command-level interface programs, the COBOL area contains (1) the COBOL working storage for the task and (2) a copy of the task global table (TGT); for macro-level programs, the COBOL area contains only a copy of the TGT. In either case, at any time after the program has issued a CICS request, the TGT is always addressed by TCAPCHS in the system part of the TCA.

The TGT is used to hold intermediate values set at various stages during program execution. The first 18 words of the TGT constitute a standard save area, in which the program's current registers are stored on any request for CICS service.

The TGT also holds the base locator for linkage (BLL) cells, which contain the addresses of all areas defined in the linkage section of the COBOL program. The BLL cells are located at or about offset X'1F4'.

The BLL cells should be recognizable in the dump thus:

- In a command-level program, the **third** BLL cell points to itself, and the **fourth** to the CSA. (This is true only when the COMMAREA is 4096 bytes or less; see "BLL cells in a command-level program" on page 182.)

- In a macro-level program, the **first** BLL cell points to itself, and the **second** to the CSA.

BLL cells in a command-level program

The **first** BLL cell points to the EXEC interface block (an area that follows the literal string 'DFHEIB').

The **second** BLL cell points to the **first** (or only) 4096-byte block of COMMAREA. If the COMMAREA is greater than 4096 bytes, the **third** BLL cell points to the next 4096-byte block of COMMAREA; further 4096-byte COMMAREAs are pointed to by succeeding BLL cells.

The next BLL cell always points to itself; the remaining BLL cells are set by the application program.

Thus, for example, if the COMMAREA size is 10K bytes, it is the **fifth** BLL cell that points to itself.

Initialization of COBOL programs within CICS (command-level)

Note: For a generalized description of the way in which command-level CICS requests are handled, see "Invocation of CICS services by command" on page 188.

COBOL programs, when compiled, include the task global table embedded in the object code. The areas addressed by the BLL cells include the EXEC interface block (EIB).

In order that concurrently executing tasks may execute the same COBOL program, a separate copy of the TGT must be kept for each task using the program. For programs using the command-level interface, or a mixture of command-level and macro-level, a separate copy of the WORKING STORAGE section is also made, so that data areas in WORKING STORAGE can be modified. If the command-level interface is not used, WORKING STORAGE is read-only. When a task requests a LINK or XCTL to a COBOL program, DFHPCP builds, for that task, a separate copy of WORKING STORAGE and the TGT in transaction storage. It is listed as part of a CICS transaction dump. A single area of CICS transaction storage is used to contain the copy of WORKING STORAGE and the copy of the TGT. This area contains a save area for the registers after the COBOL INIT routines have been executed, followed by the copy of WORKING STORAGE, followed by the copy of the TGT.

When a COBOL program is first fetched, DFHPCP determines the TGT size and stores it in PPTCCR in the PPT COBOL extension. It also determines the total size of the register save area, the WORKING STORAGE, and the TGT, and stores it in PPTCOTP in the PPT COBOL extension. It also sets PPTCOTGT to address the static TGT in program storage (from which copies will be made) and sets the bit PPTCINIT in PPTTLR2 to show that the program has been initialized. The offset of the first BLL cell in the TGT is stored in PPTCOBLL. If the program is later reloaded from disk, these fields are reinitialized.

For each transaction issuing a LINK or XCTL to the COBOL program, DFHPCP acquires a new COBOL area. The first four BLL cells in the TGT copy are initialized; the first cell points to EIB, the second to DFHCOMMAREA, the third to itself to address the rest of the BLL cells, and the fourth to the CSA. All other cells must be set

by user code as required. (This includes the CALL 'DFHEI1' inserted by the translator at the start of the PROCEDURE DIVISION for COMMAREAs that may be greater than 4096 bytes.) The field TCAPCDSA (TCAPCCA) in the TCA system prefix points to the acquired COBOL area for that task. The original WORKING STORAGE section and TGT are only used as a base for creating new COBOL areas.

Note: The register save area starts 16 (X'10') bytes in from the COBOL area start, the length of the register save area is 60 bytes.

1. Most COBOL programs also use a program global table (PGT). When a COBOL program is first fetched, address constants in this have to be relocated. This is done once only. Thereafter, all tasks using the program use the PGT for reference only.

2. At the point in DFHPCP where a high-level language application is invoked, programs using the EXEC interface invoke DFHEIP at the initialization entry point DFHEIPIN. This sets up the EXEC environment by (1) obtaining EXEC interface storage (EIS) unless it has already been acquired by task control; and (2) initializing it. EIS (which includes the EIB) is addressed by TCAEISA in the system part of the TCA. DFHEIP then invokes the application.

3. The COBOL area (in CICS dynamic storage) is freed when the program issues an XCTL or a RETURN.

Initialization of COBOL programs within CICS (macro-level)

CICS conventions for macro-level COBOL programs require the first four BLL cells to hold the addresses of the BLL cells, the CSA, the CSA optional features list and the TCA. Pointers to the TIOA, TCTTE and other areas may follow in any order, depending on what is in the linkage section.

Since some of these are task-dependent, a separate copy of the TGT is kept for each task using the program.

When a COBOL program is fetched due to a LINK or XCTL request, DFHPCP determines the TGT's size, its address, and the offset in it of the first BLL cell, and stores them in the 12-byte PPT entry extension, at PPTCCR, PPTCOTGT, and PPTCOBLL respectively.

It also finds the unresolved ADCON in the PGT (generated for DFHCBLI), and inserts there the address of PCCBLIN, the point in DFHPCP which the program, when running, will invoke for CICS services. DFHPCP then allows the compiled code INIT1, INIT2, and INIT3 to run, but causes control to return to DFHPCP rather than the procedure. Thus, the COBOL addresses that require it are relocated, and COBOL base registers are evaluated, and stored at offset X'48' in the compiled program.

For each task that uses the COBOL program, DFHPCP obtains a new **COBOL area** in user storage, 8 bytes longer than the original TGT. DFHPCP copies the original TGT into it, after changing the TGT pointers in the program itself, and sets the first BLL cell to address itself, and the second to address the CSA. Finally, the COBOL program is invoked by a BALR 14,15 instruction near label PCPEHLL. Due to the preceding manipulations, it uses the set of base registers which remain at offset X'48' as if from a previous execution, including the base of the new TGT.

PL/I application programs

Optimizer-compiled PL/I load modules contain several CSECTs, typically including:

- PLISTART, to carry the load module entry point and send control to the library initialization routines which then invoke the compiled code. Under CICS, this CSECT is replaced by DFHPL1I or DFHPL1OI; see "Initialization of PL/I programs within CICS (macro-level)" on page 186. Whichever is present, it is always at offset 0.

- PLIMAIN, of 8 bytes only, containing the entry point of the MAIN PROCEDURE.

- A static internal control section, and the compiled code itself. The external (main) procedure appears before any internal blocks. Each block entry point is preceded by the block name.

- PLISHRE, the shared library addressing module, present if the shared library option is in use. These CSECTs have dummy entry points that duplicate the names of all entry points in the shared library, and code to send control to the corresponding working entry point.

- Various resident PL/I library routines, which are automatically link-edited for operations required by compiled code but not provided by the installation's shared library.

An initial storage area (ISA) is acquired during PL/I initialization, and secondary segments of storage may be obtained during execution as logical extensions to it. The low-address part of the ISA is the Program Management Area, which consists of the (PL/I-type) TCA, its TCA appendage, other housekeeping fields, and a dummy DSA (dynamic storage area). These are created by initialization routines, and are detailed in the *OS PL/I Optimizing Compiler: Execution Logic* manual. A PL/I DSA is created for each level of PL/I code invoked (by a compiled CALL, BEGIN, or function reference), either in the ISA, or, when that overflows, in a secondary storage segment.

Each DSA starts with a standard save area and contains various housekeeping fields, the AUTOMATIC variables belonging to its block, and temporary workspace. CONTROLLED and BASED variables, which are created by the program, may occur in the ISA near the high-address end, or in segments obtained specifically.

Dump areas to be considered in PL/I program diagnosis

In a dump involving PL/I, identify and examine the following:

1. The current CICS DSA. This can be found from TCAPCDSA (TCAPCPA) normally or from PL/I's register 13 or its value kept in register 11 during a CICS macro request. The registers saved in the standard save area are those current when the last call occurred to a lower-level block, a library routine, or a CICS service. Hence register 14 gives the address where that call occurred in the compiled code, and register 15 the address called. (For a CICS service, the latter is always PCPL1IN in DFHPCP.) Any parameter list passed will be found from the saved register 1.

2. TCAPCTA in the CICS system TCA. If, in the macro interface, the X'01' flag is not on, the call to set CSACBAR has been omitted or not yet reached.

Location of PL/I dumped areas

The transient data destination CPLI holds debugging messages, and destination CPLD holds the dump for PL/I programs.

The dumped PL/I program is addressed by PPTCSA in the PPT, or by the appropriate SCP trace entry for PROGRAM class.

Initialization of PL/I programs within CICS (command-level)

Note: For a generalized description of the way in which command-level CICS requests are handled, see "Invocation of CICS services by command" on page 188.

1. PL/I programs use static and dynamic storage.

2. STATIC storage is embedded in the object code. If programs use this for anything but read-only data, there is a danger of data corruption when multithreading.

3. PL/I dynamic storage (AUTOMATIC, CONTROLLED and BASED) is allocated from a PL/I area called the initial storage area (ISA). A separate block is allocated from the ISA for each PL/I procedure entered, and released when control returns from the procedure. These blocks are called PL/I dynamic storage areas (DSAs). Each PL/I DSA begins with a register save area, to be used for all subroutine calls from that procedure. CONTROLLED and BASED storage are allocated in separate blocks from the ISA.

4. When a task requests a LINK or XCTL to a PL/I program, control is passed (after any necessary program fetch) to the CICS module DFHSAP. This performs the PL/I initialization instead of the subroutines that would do this in batch PL/I. An area of transaction storage is acquired to be used as the ISA for that task. Thus each task using a PL/I program has its own area for dynamic storage.

5. The ISA allocated per task is not large. If insufficient space is available for acquiring another area from the ISA (for example, a PL/I DSA for an inner procedure), DFHSAP is entered again to acquire a further area of transaction storage. This may be done repeatedly, provided CICS dynamic storage is available.

6. The successive areas acquired by DFHSAP are chained in a push down stack. Each area has a 16-byte prefix (including the 8-byte CICS SAA) of which offset 8 points to the previously acquired area. The latest area acquired is addressed by TCAPCDSA (TCAPCPA) in the TCA system prefix.

   Refer to the *OS PL/I Optimizing Compiler: Execution Logic* manual for a detailed layout of the ISA. The ISA begins 16 bytes on from the PL/I area at the far end of the push-down chain.

7. At the point in DFHPCP where a high-level language application is invoked, programs using the EXEC interface invoke DFHEIP at the initialization entry point DFHEIPIN. This sets up the EXEC environment by obtaining EXEC interface storage (EIS) which includes EIB and initializing it. EIS is addressed by TCAEISA in the system part of the TCA. DFHEIP then invokes the application.

Initialization of PL/I programs within CICS (macro-level)

CICS-PL/I load modules differ from those that run in batch in the following respects:

1. The first CSECT is an interface routine, DFHPL1I or DFHPL1OI. During execution, this routine redirects control to the CICS nucleus on each request for certain PL/I library functions (including creation of the PL/I environment) and all CICS services.

2. The shared library resides in the link pack area, and is usable concurrently by batch and CICS-run programs. However, the PLISHRE that CICS programs have as the addressing module has the entry points IBMBPIRA, IBMBPIRB, and IBMBPIRC deleted to prevent clashes with DFHPL1OI.

The nucleus module DFHSAP takes over the entry points and functions not only of PL/I initialization, but also of storage management. It assigns the ISA and secondary segments in blocks of CICS transaction storage instead of using operating system macros, but does not leave any free space for PL/I to suballocate within these blocks; whereas in batch, the ISA may take most of the address space. CICS uses an extra chain through these blocks, headed by TCAPCPA in the system part of the TCA, and linked through the word at offset X'8' in successively older segments, ending with the ISA. The PL/I storage begins at offset X'10' in each CICS allocation.

Besides the initialization done by DFHSAP, CICS conventions require the PL/I program to execute the

```
CALL DFHPL1C (CSACBAR);
```

statement before invoking any CICS macro-level services. This call is passed by DFHPL1I (or DFHPL1OI) to DFHPCP which stores the current DSA address (usually the main procedure DSA) in the CICS TCA at TCAPCHS, turns on the X'01' bit in TCAPCTA, and returns the address of the CSA, so that the PL/I program can address the major CICS fields. For more information on the CICS-PL/I interface and debugging corresponding problems, see the *OS PL/I Optimizing Compiler: Programmer's Guide.*

Assembler-language application programs (command-level only)

Assembler-language load modules contain:

- The EXEC interface processor DFHEAI. See the link-edit listing to find the length of this processor.

- The code and constant data areas generated by the assembler.

- The EXEC interface processor DFHEAI0. See the link-edit listing to find the length of this processor.

Dump areas to be considered in assembler-language program diagnosis

In the dump of an assembler-language program, the main areas to be considered are:

- The program itself, of which there is one copy.
- The dynamic storage area which is allocated by task. This is the storage for the variables in the DFHEISTG DSECT.

Location of assembler dumped areas

The dumped program

The dumped assembler program is addressed by PPTCSA in the PPT, or by the appropriate SCP trace entry for PROGRAM class.

The dumped DFHEISTG

This dynamic storage is addressed by TCAEISTG, which points to the currently active dynamic storage area on the chain of dynamic storage areas pointed to by TCAPCDSA.

The DFHEISTG DSECT begins with a standard save area; this is followed by the parameter list storage in which the argument list to be passed to DFHEIP is built, together with other variables including the address of EIB and the address of COMMAREA. The user's own variables in dynamic storage begin at offset X'B0' in DFHEISTG.

Initialization of assembler-language programs within CICS (command-level)

Note: For a generalized description of the way in which command-level CICS requests are handled, see "Invocation of CICS services by command" on page 188.

Assembler-language programs using the command-level interface are invoked in a system-standard way:

```
Register 13 points at save area
Register 14 points at return address
Register 15 points at entry point
Register 0 (undefined)
Register 1 points at argument list
```

The entry point is the code generated by the prolog macro DFHEIENT, which saves the caller's registers, invokes DFHEIP to obtain the dynamic storage required by the application, loads the registers to address the application code, its data (in DFHEISTG), and EIB.

At the end of an application program it can return to CICS by invoking the DFHEIRET macro, which restores the caller's registers (saved by DFHEIENT) and returns to the address in register 14.

Invocation of CICS services by command

During execution of command-level programs, registers 12 and 13 do not address the TCA and CSA. A direct branch into CICS modules is therefore not possible.

Each EXEC CICS command is translated to include a CALL statement (but in an assembler-language program, it includes an invocation of the DFHEICAL macro). This passes control to DFHEIP at the DFHEIPCN entry point. The AICB contains the addresses of the subsystems that can be used by the application. This includes EIP entry points and the DL/I entry point.

The registers of the application program at the time of the CALL are stored in a register save area. For COBOL, this area is held at the start of the TGT; for PL/I, it is at the start of the current DSA; and for assembler, it is at the start of the DFHEISTG DSECT. In all cases, register 13 points to the save area, and registers 14 through 12 are stored from offset X'C' on. DFHEIPIN saves register 13 (the pointer to the saved register values) in TCAPCHS and in EISARSA. DFHEIP makes two trace entries for each EXEC CICS command; one on entering DFHEIP and the other just before returning to the application. On entry to DFHEIP, the trace entry gives the return address in the application, an encoding of the command to be executed (EIBFN) and the address of the application's dynamic storage: WORKING STORAGE for COBOL; DSA for PL/I; and DFHEISTG for assembler. On exit from DFHEIP, the trace entry gives the return address in the application, the encoding of the command, and the response from the command (EIBRCODE). The return address on exit will be the same as on entry unless an exceptional condition has occurred; in this case, the return address will be the point in the program where the exception is to be handled. (This is specified by EXEC CICS HANDLE.) These pairs of trace entries (which may have intervening trace entries made by the rest of CICS), together with the response from each command and the return address, provide a record of the flow of control through the application. For further details, refer toFigure 20 on page 117 . If an exception occurs that the application has not mentioned in an EXEC CICS HANDLE, then DFHEIP will take a system action, which is usually to abend the task with an abend code AEIx or AEYx where x is a suffix that uniquely identifies the exceptional condition. For further information, see the *Messages and Codes* manual.

A dump contains the transaction storage EIS which contains EIB with the character string 'DFHEIB' preceding EIB to assist in locating it. This EIB contains various fields of interest, for example, the task number to identify the trace entries for this task, the last EXEC CICS command and its response, the last DATASET used. For further information, see *Application Programmer's Reference.*

If a program interrupt occurs in a command-level program, registers 12 and 13 may or may not address the TCA and CSA, depending on whether a CICS service is being executed or not.

The command interpreter is an assembler command-level application, and uses the trace entries and storage area dumps available to such an application.

Invocation of CICS services by macro

During execution of COBOL and PL/I programs, registers 12 and 13 are used for special purposes. A direct branch into CICS modules (which expect these registers to address the TCA and CSA) is therefore not possible.

Each CICS macro in an application program is expanded to include a CALL statement. This passes control to DFHPCP at a special entry point for that source language (through an interface module, in the case of PL/I). The appropriate entry point is stored in each HLL program after it is fetched into storage, as explained earlier.

The registers of the COBOL or PL/I program at the time of the CALL are stored in a register save area, which is reserved by the program compiler. For COBOL programs, this area is held at the start of the TGT; for PL/I, it is at the start of the current DSA. In either case, register 13 points to the save area, and registers 14 through 12 are stored from offset X'C' on.

The COBOL or PL/I interface section in PCP saves register 13 (the pointer to the saved register values) in register 11, then loads registers 12 and 13 to address the TCA and CSA, and invokes the appropriate CICS service request. Thus the key to the HLL registers saved for any CICS service request is the value of register 11, as saved in the appropriate CICS save area in the TCA. This will point to one of the transaction storage areas, addressed directly or indirectly through TCAPCCA/TCAPCPA. This value is also stored at TCAPCHS in the system prefix for COBOL programs; for PL/I, TCAPCHS points to the save area for the program block which was active on the last invocation of CICS.

If a program interrupt occurs in an COBOL or PL/I program, registers 12 and 13 may or may not address the TCA and CSA, depending on whether a CICS service is being executed or not.

Chapter 2.7. Debugging terminal errors

Two routines (DFHTACP for non-VTAM terminals, and DFHZNAC for VTAM terminals) are provided for error handling. These routines perform only basic error processing; for further error processing, control is passed to a user-written routine – DFHTEP (for non-VTAM) or DFHZNEP (for VTAM).

Non-VTAM terminal errors

For non-VTAM errors associated with a line or terminal, DFHTCP places the terminal out of service and creates a TACLE, which contains error information. The TACLE is chained from TCTLEECA in the relevant TCTLE and is a copy of the event control block (ECB) with a 16-byte prefix.

Control is passed to DFHTACP, with TCAFCAAA addressing the TACLE. The field TCTLEPFL at offset X'8' in the TACLE contains the error code. DFHTACP inspects the error code and sets bytes TCTLEECB + 1 and TCTLEECB + 2 of the TACLE to indicate the default actions to be taken. The error codes and the corresponding default actions are shown in Figure 44 on page 192. DFHTACP then links to DFHTEP which takes one of the following forms:

- A default DFHTEP, which immediately returns control to DFHTACP
- A sample DFHTEP provided with the system
- A user-written DFHTEP to provide further error processing.

On return to DFHTACP, the default actions are performed, unless overridden by the DFHTEP.

Trace entries are made before and after DFHTACP's link to DFHTEP. See page 70 for details of these DFHTACP (X'E6') system trace entries.

Figure 43 on page 192 provides a quick cross-reference chart for finding the appropriate message number (and associated error condition) for each error detected by DFHTACP. Figure 44 on page 192 shows the error messages, error codes, and default actions for abnormal actions detected by BTAM and TCAM.

Error Code	Message Number	Error Code	Message Number	Error Code	Message Number	Error Code	Message Number
X'80'	none	X'89'	DFH2526	X'90'	DFH2532	X'99'	DFH2521
X'81'	DFH2501	X'89'	DFH2527	X'91'	DFH2531	X'9A'	none
X'82'	none	X'89'	DFH2528	X'92'	none	X'9B'	DFH2523
X'83'	DFH2503	X'8A'	DFH2510	X'93'	none	X'9C'	DFH2524
X'84'	DFH2502	X'8B'	DFH2512	X'94'	DFH2516	X'9D'	DFH2525
X'85'	DFH2511	X'8C'	DFH2506	X'95'	none	X'9E'	DFH2508
X'86'	DFH2505	X'8D'	DFH2513	X'96'	DFH2518	X'9F'	DFH2534
X'87'	DFH2569	X'8E'	DFH2514	X'97'	DFH2519	X'A0'	DFH2530
X'88'	DFH2507	X'8F'	DFH2515	X'98'	DFH2520	X'A1'	DFH2509

Figure 43. Non-VTAM terminal error message numbers

Error Message	Error Code (Symbolic Label)	Condition	Action Set By DFHTACP (See notes at end of figure)
DFH2501	X'81' (TCEMCMTL)	Input message exceeds read length, or lost data signaled on read text for remote device (follows unit check error).	3
DFH2502	X'84' (TCEMCTCT)	TCT search error:	
		- Switched line - real terminal	3
		- Switched line - dummy terminal	none
		- Nonswitched line - real terminal	2,3
		- Nonswitched line - dummy terminal.	1
DFH2503	X'83' (TCEMCAAR)	2740-2 Communication Terminal auto output request. (Used by DFHTACP, not passed to DFHTEP.)	none
DFH2505	X'86' (TCEMCPL)	Polling list error.	1
DFH2506	X'8C' (TCEMCOER)	BTAM return code on write:	
		- Local 3270 Information Display System open failure, invalid relative line number (RLN), device under OLTEP	2,3
		- Local 2260 Display Station	2,3
		- All other conditions.	1,3
DFH2507	X'88' (TCEMCIER)	BTAM return code on read:	
		- Local 3270 Information Display System open failure, invalid relative line number (RLN), device under OLTEP	2,3
		- Local 2260 Display Station	2,3
		- All other conditions.	1,3

Figure 44 (Part 1 of 5). Error messages, error codes, and default actions generated by DFHTACP for DFHTEP

Error Message	Error Code (Symbolic Label)	Condition	Action Set By DFHTACP (See notes at end of figure)
DFH2508	X'9E' (TCEMCUP)	A 3270 Information Display System print request was made, but no printer was available to print the data. DFH2531 or DFH2532 may subsequently appear - see code X'91' and "3270 Unavailable Printer" in the chapter "The Terminal Error Program" in the *Customization Guide*.	none
DFH2509	X'A1' (TCEMCIDR)	A transaction has requested a DFHTC TYPE=(RESET,DISCONNECT) on a switched binary synchronous line, and no end-of-transmission (EOT) has been received from the terminal; this indicates that more data is to follow. Terminal control issues a read to the terminal. If the EOT is not received on that read, the error code is set and passed to DFHTACP.	5
DFH2510	X'8A' (TCEMCTO)	7770 Audio Response Unit 32-second time-out	3,5
DFH2511	X'85' (TCEMCROT)	Invalid write request: - A write request was made to a terminal in input status.	3
		- A write request was made to a 3735 Programmable Buffered Terminal before EOT (indicated by the EOF condition) was received from the 3735 during batch transmission.	3
DFH2512	X'8B' (TCEMCOBE)	Hardware buffer exceeded (shift character not properly accounted for).	3
DFH2513	X'8D' (TCEMCOLZ)	Output length zero.	3
DFH2514	X'8E' (TCEMCNOA)	No output area provided.	3
DFH2515	X'8F' (TCEMCOAE)	Output area exceeded (TIOATDL value larger than output area).	3

Figure 44 (Part 2 of 5). Error messages, error codes, and default actions generated by DFHTACP for DFHTEP

Error Message	Error Code (Symbolic Label)	Condition	Action Set By DFHTACP (See notes at end of figure)
DFH2516	X'94' (TCEMCUC)	Unit check:	
		- Local 2260 Display Station, local 3270 Information Display System, or SAM:	
		1. SAM line	1,3
		2. Local 2260 or local 3270 operation check only	3
		3. Local 2260 or local 3270 other sense code (including local 3270 unit error).	2,3
		- Remote lines	none
		- Switched line disabled	4
		- Intervention sense:	
		1. Switched line	3,4
		2. Nonswitched, real terminal	2,3
		3. Nonswitched, dummy terminal.	1
		Unit check (actions same as TCEMCUCS):	
		- Lost data on read text (message DFH2501 also issued, see code X'81')	3
		- Data check sense:	
		1. Real terminal	2,3
		2. Dummy terminal.	1
		- Time-out sense:	
		1. READ or WRITE TEXT command on start/stop device	none
		2. Other time-out, real terminal	2,3
		3. Other time-out, dummy terminal.	1
		- All other sense codes.	1
DFH2518	X'96' (TCEMCUE)	Unit exception (actions same as TCEMCUES).	none
DFH2519	X'97' (TCEMCUES)	Unit exception:	
		- Switched line	3,4
		- Real terminal	2,3
		- Dummy terminal.	1
DFH2520	X'98' (TCEMCNR)	Negative response:	
		- Negative response to addressing	2,3
		- Negative response to 'DFH2503 AUTO OUTPUT' message.	none
DFH2521	X'99' (TCEMCUDT)	Undetermined unit error:	
		- Local 2260 Display Station	2,3
		- Local 3270 Information Display System (see code X'94')	
		- Write connect on autocall line	1,2,3
		- All other devices.	1,3

Figure 44 (Part 3 of 5). Error messages, error codes, and default actions generated by DFHTACP for DFHTEP

Error Message	Error Code (Symbolic Label)	Condition	Action Set By DFHTACP (See notes at end of figure)
DFH2523	X'9B' (TCEMCICR)	The terminal entries for the sending and receiving devices did not specify the COPY feature (3270 Information Display System).	3
		The device address specified for the receiving device does not exist on the control unit.	3
		The length of the COPY command was not specified as one.	3
DFH2524	X'9C' (TCEMCIMB)	Invalid message block received: - An unidentified message block was received from a local or remote 3270 Information Display System.	2,3
		- The type of input block received from a 3735 Programmable Buffered Terminal did not agree with the mode of the active transaction inquiry batch.	2,3
DFH2525	X'9D' (TCEMCICM)	An incomplete message was received from a remote 3270 Information Display System. The device terminated transmission prior to message completion (that is, end-of-transmission (EOT) was received prior to end-of-text (ETX)).	2,3
DFH2526	X'89' (TCEMCSM)	Error status received from remote BSC device - 3270 Information Display System intervention required (printer).	none
DFH2527	X'89' (TCEMCSM)	Error status received from remote BSC device - 3270 Information Display System intervention required (screen).	none
DFH2528	X'89' (TCEMCSM)	Error status received from remote BSC device: - 3735 Programmable Buffered Terminal (all conditions). - Operation check (other devices) - All other conditions/devices.	 none 3 2,3
DFH2529	X'87' (TCEMCUI)	Unsolicited input:	
		- Input has occurred on an out-of-service 3735 Programmable Buffered Terminal.	none
		- Terminal **receive only** (TCAM) (TCTLEECB+3 = X'03') - Terminal **out of service** (TCAM) (TCTLEECB+3 = X'04') - Task has not issued a read (TCAM) (TCTLEECB+3 = X'01') - No available TCTTE from the pool (TCAM) (TCTLEECB+3 = X'02').	6 2,6 none none

Figure 44 (Part 4 of 5). Error messages, error codes, and default actions generated by DFHTACP for DFHTEP

Error Message	Error Code (Symbolic Label)	Condition	Action Set By DFHTACP (See notes at end of figure)
DFH2530	X'A0' (TCEMCWOT)	Invalid read request:	
		- A read request was issued to a terminal in RECEIVE status.	3
		- A read was issued to a 3735 Programmable Buffered Terminal after an end-of-transmission (EOT) (indicated by the end-of-file (EOF) condition) was received from the terminal during batch transmission.	3
DFH2531	X'91'	IC error on unavailable printer entry (see "3270 Unavailable Printer" in the chapter "The Terminal Error Program" in the *Customization Guide*).	none
DFH2532	X'90'	Print request queued for IC PUT after unavailable printer (see "3270 Unavailable Printer" in the chapter "The Terminal Error Program" in the *Customization Guide*).	none
DFH2534	X'9F' (TCEMIDR)	TCAM has issued an invalid destination return code to CICS.	3

Notes:

Default Action	Description	Bit Setting Mask
1	Line out of service	X'80' at TCTLEECB+1
2	Terminal out of service	X'08' at TCTLEECB+1
3	ABEND transaction	X'04' at TCTLEECB+1
4	Switched line disabled	X'20' at TCTLEECB+1
5	Disconnect switched line	X'10' at TCTLEECB+1
6	Release TCAM incoming message	X'80' at TCTLEECB+2

Unless otherwise stated, the bits at TCTLEECB+1, which are left unset by default, have the values given in the action bits table under "TACLE Action and Information Bits" in the chapter "The Terminal Error Program" in the *Customization Guide*.

Figure 44 (Part 5 of 5). Error messages, error codes, and default actions generated by DFHTACP for DFHTEP

VTAM-associated errors

For VTAM-associated errors, the processing is similar. DFHZCP passes control to DFHZNAC, which in turn links to DFHZNEP (the node error processing program) for further error processing. In this case, the TCTTE associated with the error is put onto the NACP queue that is pointed to by the TCT prefix.

Figure 45 on page 197 provides a quick cross-reference chart for finding the appropriate message number (and associated error condition) for each error detected by DFHZNAC. Figure 46 on page 198 shows the error codes and action flag settings for abnormal conditions detected during sessions involving VTAM-supported logical units. The code containing a particular error condition is passed to DFHZNAC in a 1-byte field of DFHZNAC's TWA at label TWAEC. DFHZNAC passes control to the appropriate node error program for resolution of the error. See the *Diagnosis Reference* manual for a description of the CICS-VTAM interface.

Error Code	Message Number	Error Code	Message Number	Error Code	Message Number	Error Code	Message Number
X'10'	DFH2405	X'4A'	DFH3463	X'90'	DFH2422	X'C7'	DFH3485
X'11'	DFH2403	X'4B'	DFH2498	X'91'	DFH2429	X'C8'	DFH3486
X'13'	DFH2416	X'4C'	DFH3481	X'92'	DFH2428	X'C9'	DFH3487
X'14'	DFH2404	X'4D'	DFH3473	X'93'	DFH2455	X'CB'	DFH2431
X'15'	DFH2407	X'4E'	DFH3479	X'94'	DFH2426	X'CC'	DFH2451
X'18'	DFH2404	X'4F'	none	X'95'	DFH2445	X'CD'	DFH2454
X'19'	DFH2406	X'50'	DFH3417	X'96'	DFH2435	X'CE'	DFH3469
X'1A'	DFH2408	X'51'	DFH3418	X'97'	DFH2436	X'CF'	DFH3471
X'1D'	DFH2417	X'52'	DFH3419	X'98'	DFH2439	X'D0'	DFH2409
X'20'	DFH2417	X'53'	DFH3420	X'99'	DFH2459	X'D1'	DFH2410
X'21'	DFH4902	X'54'	DFH3434	X'9B'	DFH2486	X'D2'	none
X'22'	DFH4903	X'55'	DFH3440	X'9C'	DFH2487	X'D3'	DFH2463
X'23'	DFH4904	X'57'	DFH3464	X'9D'	DFH3465	X'D4'	DFH2453
X'24'	DFH4905	X'58'	DFH3433	X'9E'	DFH3472	X'D5'	DFH2452
X'25'	DFH4906	X'59'	DFH2443	X'9F'	DFH3478	X'D6'	DFH2441
X'26'	DFH4907	X'5A'	DFH3421	X'A0'	DFH3480	X'D7'	DFH2440
X'27'	DFH4908	X'5B'	DFH3422	X'A1'	DFH2438	X'D8'	DFH2457
X'28'	DFH4909	X'5C'	DFH3424	X'A3'	DFH2444	X'D9'	DFH2469
X'29'	DFH4910	X'5E'	DFH3454	X'A7'	DFH2449	X'DA'	DFH2470
X'2A'	DFH4911	X'5F'	DFH3455	X'A8'	DFH2471	X'DB'	DFH3483
X'2B'	DFH4912	X'60'	DFH2421	X'A9'	DFH2472	X'DC'	DFH2442
X'2C'	DFH4913	X'61'	none	X'AA'	DFH2473	X'DD'	DFH2458
X'2D'	DFH4914	X'62'	none	X'AB'	DFH3470	X'DE'	DFH3403
X'2E'	DFH4915	X'63'	DFH3441	X'AC'	DFH3474	X'DF'	DFH3482
X'2F'	DFH4916	X'64'	DFH3443	X'AD'	DFH3475	X'E0'	DFH2411
X'30'	DFH4917	X'65'	DFH2448	X'AE'	DFH3476	X'E1'	DFH2412
X'31'	DFH4918	X'66'	DFH3452	X'AF'	DFH3477	X'E2'	DFH2413
X'32'	DFH4922	X'67'	DFH3442	X'B1'	DFH2401	X'E3'	DFH2485
X'33'	DFH4919	X'69'	DFH3466	X'B2'	DFH2425	X'E4'	DFH2491
X'34'	DFH4920	X'70'	DFH3468	X'B3'	DFH2402	X'E5'	DFH2499
X'35'	DFH4930	X'73'	DFH2437	X'B5'	DFH2420	X'E6'	DFH2404
X'3C'	DFH3488	X'74'	DFH2423	X'B6'	DFH4924	X'E7'	DFH2488
X'40'	DFH2489	X'75'	DFH2424	X'B7'	DFH4925	X'E8'	DFH3416
X'41'	DFH2490	X'78'	DFH2430	X'B8'	DFH4926	X'E9'	DFH2102
X'42'	DFH2497	X'79'	DFH2474	X'B9'	DFH4927	X'EA'	DFH3432
X'43'	DFH2434	X'80'	DFH2414	X'BA'	DFH2433	X'EB'	DFH3428
X'44'	DFH2456	X'81'	DFH2432	X'BB'	DFH2418	X'EC'	DFH3429
X'45'	DFH3400	X'82'	DFH2419	X'BC'	DFH3410	X'ED'	DFH3430
X'46'	DFH3402	X'83'	DFH2450	X'BD'	DFH4928	X'EF'	DFH3431
X'47'	none	X'84'	DFH2446	X'C1'	DFH2400	X'F0'	none
X'48'	DFH3461	X'88'	DFH2467	X'C5'	DFH2427	X'F1'	DFH4931
X'49'	DFH3462	X'89'	DFH2468	X'C6'	DFH3484	X'F2'	DFH2469

Figure 45. VTAM terminal error message numbers

Error Message	Error Code	(Symbolic Label)	Condition	Action Flags (See notes at end of figure)
	X'47'	(TCZGMMS)	Good morning message to be sent.	13
	X'61'	(TCZLUSTA)	LUSTATUS received after intervention required type of condition.	32
	X'62'	(TCZVTAMQ)	Cancel task, and close VTAM session due to quick close or abend.	24,32
	X'D2'	(TCZCRP)	Node recovery in progress.	9,10,11,32
DFH2102	X'E9'	(TCZSTIND)	Data base changes found to be synchronized. Details are given of failure.	3
DFH2400	X'C1'	(TCZSRCAT)	Error not supported: • An unanticipated error code was passed to CICS by VTAM. • The SYNAD exit was unable to identify the error received from VTAM.	2,3,9,10, 11,23,24
DFH2401	X'B1'	(TCZRPLAC)	Request parameter list (RPL) active: • A logic error has occurred such that a VTAM request is being set up using an RPL which already has an active request. • The VTAM CHECK macro instruction was not issued by the appropriate exit.	2,3,9,10, 11,23,24
DFH2402	X'B3'	(TCZNORPL)	No RPL available when one was expected: • The RPL pointer field in the TCTTE (TCTERPLA) was inadvertently cleared. • A logic error has occurred within CICS that caused the RPL to be freed.	3

Figure 46 (Part 1 of 17). Error codes, error messages, and default actions generated by DFHZNAC for DFHZNEP

Error Message	Error Code (Symbolic Label)	Condition	Action Flags (See notes at end of figure)
DFH2403	X'11' (TCZSRCBF)	Session bind failure: • A session cannot be established because no physical path can be found to the logical unit. • The logical unit does not exist. • The logical unit does not agree with the BIND parameters. Possible system generation mismatch. For an intersystem communication session, refer to the section on sense codes sent by CICS.	2,5,18,24
DFH2404	X'14' (TCZLRCER)	VTAM detected a logic error associated with a request: • VTAM request was either not complete or not executable. • Conflicting parameters in the request parameter list (RPL).	2,3,9,10, 11,23,24
	X'18' (TCZLRCNR)	• LERAD exit entered.	2,3,9,10, 11,23,24
	X'E6' (TCZDMLG)	• Not on known TCTTE.	none
DFH2405	X'10' (TCZSRCTU)	Node not activated. The node either was not activated, or was deactivated by the network operator.	18
DFH2406	X'19' (TCZSRCTS)	Terminate-self command received. The logical unit has requested to be disconnected.	9,10,11,18
DFH2407	X'15' (TCZSRCPF)	Permanent failure with channel or NCP/VS: • Either the NCP/VS has abnormally terminated, or was shut down by the operator. • A channel failure has occurred.	2,3,9, 10,11,24
DFH2408	X'1A' (TCZSRCVE)	An error has occurred in VTAM processing (apparent VTAM error). VTAM has encountered an error in its own processing of the request to or from the logical unit.	2,3,9,10, 11,18,24
DFH2409	X'D0' (TCZTXCS)	VTAM recovered node. VTAM has successfully reestablished communications with the node. CICS will reinitiate a session with the node.	2,3,9,10, 11,15,24

Figure 46 (Part 2 of 17). Error codes, error messages, and default actions generated by DFHZNAC for DFHZNEP

Error Message	Error Code (Symbolic Label)	Condition	Action Flags (See notes at end of figure)
DFH2410	X'D1' (TCZTXCU)	Node unrecoverable. VTAM cannot reestablish communications with the node.	2,3,9,10, 11,18,24
DFH2411	X'E0' (TCZDMSN)	Node attempted invalid logon. An unknown node has attempted a logon. The symbolic node name is contained in the first 8 bytes of the message.	none
DFH2412	X'E1' (TCZDMRA)	Receive-any problem. Receive-any initiation failed. VTAM may be in termination.	none
DFH2413	X'E2' (TCZDMCL)	Node CLSDST failed. The CLSDST of a node in logon exit has failed:	2
		• VTAM storage problem.	
		• Apparent VTAM error.	
DFH2414	X'80' (TCZSRCSP)	Temporary VTAM storage problem. VTAM has run short of available working storage. This situation should eventually terminate. Not defining enough VTAM buffer storage at VTAM generation is a common way of creating such a problem.	none
DFH2415	-	Node out of service. The node has been taken out of service by CICS because of an earlier node error condition.	none
DFH2416	X'13' (TCZSRCVH)	VTAM is halting. The HALT QUICK command was entered by the network operator while a SIMLOGON or OPNDST request was in progress.	18
DFH2417	X'1D' (TCZSRCVI)	VTAM inactive to TCB:	
		• CICS has failed to open its VTAM ACB.	2,24
	X'20' (TCZVTAMI)	• VTAM was halted.	none
DFH2418	X'BB' (TCZSEXUC)	Unknown command in RPL. SESSIONC exit, while validating the RPL at completion for a SESSIONC request, was unable to determine which SESSIONC command was sent:	2,3,9,10, 11,23,24
		• Invalid RPL address.	
		• RPL was altered.	

Figure 46 (Part 3 of 17). Error codes, error messages, and default actions generated by DFHZNAC for DFHZNEP

Error Message	Error Code (Symbolic Label)	Condition	Action Flags (See notes at end of figure)
DFH2419	X'82' (TCZSSXUC)	Unknown command in RPL. Send-data-flow synchronous exit was unable to validate the command sent: • Invalid RPL address. • RPL was altered.	2,3,9,10, 11,23,24
DFH2420	X'B5' (TCZSAXUC)	Unknown command in RPL. Send-data-flow asynchronous exit was unable to validate the indicator sent: • Invalid RPL address. • RPL was altered.	2,3,9,10, 11,23,24
DFH2421	X'60' (TCZUNCMD)	Unsupported command received. Receive-specific exit or receive-any has received an indicator it cannot handle: • If DFHZNAC finds that the message is not an LU status message, the indicator is unsupported. If the indicator proves to be LU status, see the system sense table below. • RPL was altered.	2,3,9, 10,11,24
DFH2422	X'90' (TCZLGCER)	DFHZCP logic error. OPNDST, SIMLOGON or CLSDST has detected one of the following: • OPNDST - node has already been opened. • SIMLOGON - node has already been logged on. • CLSDST - CLSDST-activate-request bit is not on.	1,2,3,9, 10,11,23
DFH2423	X'74' (TCZSDSE5)	Incomplete command request. A request to send-data-flow synchronous command was made with incomplete bit settings: • TCTTE has been altered. • Requesting module has a logic error. • TCTTE was queued inadvertently to send synchronous.	3,9,10,11,24

Figure 46 (Part 4 of 17). Error codes, error messages, and default actions generated by DFHZNAC for DFHZNEP

Error Message	Error Code (Symbolic Label)	Condition	Action Flags (See notes at end of figure)
DFH2424	X'75' (TCZSESE1)	Command request invalid. SESSIONC command request bits are invalid or incomplete: • TCTTE has been altered. • Command request bits are incomplete. • Queued inadvertently to SESSIONC.	3,9,10, 11,15,24
DFH2425	X'B2' (TCZSDAUC)	Command request invalid. Send-data-flow asynchronous command request bits are invalid or incomplete: • TCTTE has been altered. • Command request bits are incomplete.	3,9,10, 11,15,24
DFH2426	X'94' (TCZRACES)	Input status error. The receive-any module received data from a node in one of the following conditions: • The node is permanently out of service. • TCT specifies the node as an output only device.	2,3,9, 10,11,22
DFH2427	X'C5' (TCZSRCNA)	NCP/VS restarted. NCP/VS has been restarted after failing while an OPNDST was in progress.	2
DFH2428	X'92' (TCZSDSE6)	Send-synchronous request incomplete. A send-synchronous request failed to indicate whether a command or data was to be sent: • TCTTE has been altered. • Incomplete request queued to DFHZSDS.	3,9,11
DFH2429	X'91' (TCZRSTLE)	Invalid RTYPE. An incorrect RESETSR request was made: • The requester failed to specify, or incorrectly specified, the RTYPE. • TCTTE was inadvertently altered.	3,10,11

Figure 46 (Part 5 of 17). Error codes, error messages, and default actions generated by DFHZNAC for DFHZNEP

Error Message	Error Code (Symbolic Label)	Condition	Action Flags (See notes at end of figure)
DFH2430	X'78' (TCZSDRE2)	Command request invalid. A send-response request was made in error: • Request failed to specify whether response was to be RQD1 or RQD2. • TCTTE has been altered.	3,9,11,22
DFH2431	X'CB' (TCZSRCTC)	Request to a released node. If the access method control block (ACB) is open, this is a DFHZCP logic error. If the ACB is closed, CICS is halting and CLOSE ACB has been issued.	2,3,9,10,11
DFH2432	X'81' (TCZSSXNR)	Exception response received. A SEND DFSYN exit received an exception response. An exception response to a bid was received, containing sense information that does not relate to the expected ready-to-receive (RTR) sense.	none
DFH2433	X'BA' (TCZSEXNR)	Exception response received. SESSIONC exit received an exception response. An exception response to a SESSIONC command was received from the logical unit. CICS requires a normal response.	none
DFH2434	X'43' (TCZCPYNS)	Source not valid for copy.	3,11
DFH2435	X'96' (TCZRVSZ1)	RPL missing. Receive-specific was activated without an RPL: • CICS error such that an RPL is expected to be present on entry into the receive-specific module but has been freed or was never allocated. • TCTERPLA was altered.	3,10,11,24
DFH2436	X'97' (TCZRVSZ3)	TIOA missing. Receive-specific found the original TIOA in an overlength data condition missing: • CICS error such that the original TIOA was freed. • TCTTEDA was altered.	3,10,11
DFH2437	X'73' (TCZSDSE4)	Read-only node. A DFSYN SEND was scheduled for an input-only device: • TCTTETS was altered. • Task was attached that does a SEND.	3,9,11

Figure 46 (Part 6 of 17). Error codes, error messages, and default actions generated by DFHZNAC for DFHZNEP

Error Message	Error Code (Symbolic Label)	Condition	Action Flags (See notes at end of figure)
DFH2438	X'A1' (TCZRVSZ2)	Invalid read request. A read (RECEIVE) request was attempted for a receive-output only device: • Task was attached that issued a read. • TCTTETS was altered.	3,10,11
DFH2439	X'98' (TCZACT01)	Invalid resume request. Activate scan found a TCTTE to resume (but a task was not attached): • TCTTECA was altered. • CICS encountered a logic error such that the task was detached, yet the resume flag was left on in the TCTTE.	2,18
DFH2440	X'D7' (TCZSXC1)	CICS quiesced by node. The node has indicated that all data flow to it should stop. Temporarily cannot store any more data.	none
DFH2441	X'D6' (TCZSXC2)	CICS released by node. The node is now ready or capable of receiving data from CICS.	none
DFH2442	X'DC' (TCZPXE1)	Exception response received to a definite response send. Response exit received an exception response.	none
DFH2443	X'59' (TCZROCT)	Request outstanding: • A receive request was outstanding on a node at system shutdown time. • The high order bit in the node initialization block (NIB) address-field was turned on by DFHZNAC, indicating no further communication should be attempted with this node.	2,3,9, 10,11,24
DFH2444	X'A3' (TCZBKTSE)	CICS bracket state error. Application program violated CICS bracket protocol. Application issued a DFHTC read after a WRITE LAST request.	2,3,9, 10,11,24

Figure 46 (Part 7 of 17). Error codes, error messages, and default actions generated by DFHZNAC for DFHZNEP

Error Message	Error Code (Symbolic Label)	Condition	Action Flags (See notes at end of figure)
DFH2445	X'95' (TCZSDSE8)	Output area exceeded. TIOA length error: • Application set up TIOATDL incorrectly. • Application overran the TIOA.	3,9,11
DFH2446	X'84' (TCZSSXIB)	Invalid response to bid. A normal response was received to a bid while in bracket state. The 3601 Communication System application program is in error. It has lost track of the bracket status of the node.	2,3,9,10, 11,23,24
DFH2448	X'65' (TCZINVRR)	Invalid response requested. Receive-specific, receive-any or receive-specific exit received data from the logical unit without a response requested (RQD1-RQD2). RPL altered.	2,3,10,11, 22,23,24
DFH2449	X'A7' (TCZBOEB)	Bracket error. An application program sent either a begin-bracket when the transaction was in the bracket state, or an end-bracket, or data not marked with a begin-bracket, following an end-bracket.	2,3,11, 18,22,24
DFH2450	X'83' (TCZSSXAR)	Bid issued but ATI canceled. The ATI which caused a bid to be sent was canceled. • The ATI was time-initiated dependent. • CICS error.	none
DFH2451	X'CC' (TCZSRCCI)	Outstanding request when clear was issued. A request (receive-specific) was outstanding when clear was issued by CICS or by VTAM. A clear-unbind was issued by VTAM when any one of the following occurred: • Communication with the node was lost (LOSTERM exit node unrecoverable). • CICS terminated the session (CLSDST). • CICS issued the clear in an effort to tidy up or resynchronize the session.	2,3,9,10,11

Figure 46 (Part 8 of 17). Error codes, error messages, and default actions generated by DFHZNAC for DFHZNEP

Error Message	Error Code (Symbolic Label)	Condition	Action Flags (See notes at end of figure)
DFH2452	X'D5' (TCZCXE2)	Invalid command received. The session control input exit received a command other than request recovery from a logical unit.	3,9,10, 11,18,24
DFH2453	X'D4' (TCZCXRR)	Request recovery received. The logical unit has indicated that recovery processing is needed.	1,2,3
DFH2454	X'CD' (TCZSRCCX)	Exception in chain. Exception response returned on a POST=RESP chained data send: • Error within CICS. CICS does not send chained data with POST=RESP. • VTAM error.	2,3,9,10,11
DFH2455	X'93' (TCZRACET)	Continue-any mode with task attached. The node was in continue-any mode, yet a task was still present. The task currently on the node should have been abnormally terminated, but was not. The node was closed, and logon simulated, which put it into continue-any mode.	2,3,9,10, 11,15,22,24
DFH2456	X'44' (TCZSRCDE)	Received exception response to a command. Logical unit sent an exception response to a command. CICS does not support an exception response from the logical unit to a command other than bid.	2,3,9,10, 11,18,24
DFH2457	X'D8' (TCZRNCH)	Multiple catastrophic errors encountered. More than one consecutive error has been encountered by a node without the first error being processed. That is, while DFHZNAC is processing the first error encountered with a node, another synchronous error occurs that overlays the previous error code.	2,3,9, 10,11,24
DFH2458	X'DD' (TCZPXE2)	Received exception response to exception response SEND.	none
DFH2459	X'99' (TCZSDSE7)	No TIOA available for SEND. TIOA for SEND was not available. TCTTEDA was not loaded prior to issuing the DFHTC TYPE=WRITE macro or was inadvertently cleared.	3,9,11

Figure 46 (Part 9 of 17). Error codes, error messages, and default actions generated by DFHZNAC for DFHZNEP

Error Message	Error Code (Symbolic Label)	Condition	Action Flags (See notes at end of figure)
DFH2467	X'88' (TCZLEXCI)	Invalid communications identifier (CID) detected. A VTAM request was made with an invalid CID. TCTECID was altered.	2,3,9, 10,11,23,24
DFH2468	X'89' (TCZLEXUS)	Unknown symbolic name. A request was made to VTAM with an invalid symbolic node name: • Symbolic name altered in the NIB. • VTAM definition and TCT entries do not agree.	2,3,9, 10,11,23,24
DFH2469	X'D9' (TCZYX43)	Exception response received. An exception condition exists for an inbound message. Invalid sequence numbers.	2,3,9,10,11
DFH2470	X'DA' (TCZSXC3)	Request shutdown received while task active: • The controller application program sent RSHUTD (request shutdown) on behalf of a node while a task was still attached. • During VTAM shutdown, a shutdown complete indicator was received from the controller application program on behalf of a node while a task was still attached. • During VTAM shutdown, a task was still attached to a VTAM 3270 Information Display System (which cannot send request shutdown or shutdown complete).	9,10,11,24
DFH2471	X'A8' (TCZFMHLE)	FMH length error. The function management header length was greater than that of the data received from the logical unit: • The logical unit built the FMH incorrectly. • Transmission error.	2,3,4,10, 11,22,24
DFH2472	X'A9' (TCZRACRF)	Unable to retrieve overlength data. The receive request for the remainder of the data that was in excess of the receive-any input area was not accepted by VTAM.	11

Figure 46 (Part 10 of 17). Error codes, error messages, and default actions generated by DFHZNAC for DFHZNEP

Error Message	Error Code (Symbolic Label)	Condition	Action Flags (See notes at end of figure)
DFH2473	X'AA' (TCZSDSE9)	Outbound chaining not supported. The CICS application program has attempted to send more data to the logical unit than its generated maximum permitted length, and the logical unit does not support outbound chaining.	3,9,11
DFH2474	X'79' (TCZATINS)	ATI not supported. A task was automatically initiated for a logical unit which was defined at table generation time as not supporting ATI.	3
DFH2485	X'E3' (TCZCNCL)	Cancel received in continue-specific (CS) mode. A CANCEL indicator was received while a task was active.	3,9,10,11
DFH2486	X'9B' (TCZRACNL)	Cancel received in continue-any (CA) mode. A CANCEL indicator was received while no task was active.	3
DFH2487	X'9C' (TCZOCNL)	Outbound chain canceled. An outbound chain was not completed at task detach time.	3,9,10,11
DFH2488	X'E7' (TCZIPGC)	Inbound chain purged. Unprocessed inbound data remained at task detach time.	none
DFH2489	X'40' (TCZINCPY)	3270 Information Display System - invalid copy request. The TCTTE of the sending device did not specify the COPY feature, or is not defined in the TCT, or is not a 3270, or is not connected to CICS through VTAM.	3,9,11
DFH2490	X'41' (TCZTOLRQ)	Request for TOLTEP. On a request for TOLTEP, a receive request completes in error.	2,3,9,10, 11,15,24
DFH2491	X'E4' (TCZSXC4)	Segmenting error. A segmenting error was detected by the LOSTERM exit.	2,3,9, 10,11,24
DFH2497	X'42' (TCZUNPRT)	Unavailable printer. A print function was requested on a 3270 Information Display System, and neither the PRINTTO nor the ALTPRT printer was available to receive the information.	none
DFH2498	X'4B' (TCZICPUT)	Interval control PUT request to printer failed.	none

Figure 46 (Part 11 of 17). Error codes, error messages, and default actions generated by DFHZNAC for DFHZNEP

Error Message	Error Code	(Symbolic Label)	Condition	Action Flags (See notes at end of figure)
DFH2499	X'E5'	(TCZRUER)	RU exceeds RUSIZE at maximum chain size. If chain assembly has been specified in the TCTTE, the request unit (RU) read in is bigger than the remaining space in the TIOA, and bigger than the maximum RUSIZE.	3,10,11
DFH3400	X'45'	(TCZCHMX)	Chain exceeds maximum chain size. If chain assembly has been specified in the TCTTE, the chain being assembled does not fit into the TIOA for a maximum chain. The remaining space in the TIOA is smaller than the maximum RUSIZE.	3,10,11,22
DFH3402	X'46'	(TCZOCIR)	Invalid read. A DFHTC TYPE=READ request is being processed although the previously issued DFHTC TYPE=WRITE request did not complete a chain.	3,9,10,11
DFH3403	X'DE'	(TCZFSMD)	Failed to get in send mode. CICS could not break the inbound data flow in order to send a message to the logical unit.	3,9,10,11
DFH3410	X'BC'	(TCZINIIR)	Invalid input when LUSTATUS expected. Input other than LU status message received after system sense X'0802' (intervention required) or X'0807' (resource temporarily unavailable).	2,3,9,10,11
DFH3416	X'E8'	(TCZDMSLE)	Negative response to bind parameters failed.	2,3
DFH3417	X'50'	(TCZSDRE3)	A sync point request has been ignored. Neither commit nor abend has been issued.	3,9,10,11,24
DFH3418	X'51'	(TCZBDPRI)	The name in the bind area matches a primary TCTTE. The TCTTE generation should specify a secondary.	3,9,10, 11,24
DFH3419	X'52'	(TCZBDUAC)	The requesting system has passed unacceptable bind parameters.	2,3,5
DFH3420	X'53'	(TCZBDTOS)	Logon was requested for a terminal that had not been placed in service.	20
DFH3421	X'5A'	(TCZSBIRV)	A shutdown request was received for the system, and an orderly termination procedure was begun.	20

Figure 46 (Part 12 of 17). Error codes, error messages, and default actions generated by DFHZNAC for DFHZNEP

Error Message	Error Code	(Symbolic Label)	Condition	Action Flags (See notes at end of figure)
DFH3422	X'5B'	(TCZNSP01)	An error occurred while trying to establish an ISC session. The request was terminated before the session was established.	2,3,9,10, 11,18,24
DFH3424	X'5C'	(TCZNSP02)	Session failure. Session terminated immediately.	9,10,11, 15,24
DFH3428	X'EB'	(TCZSTRMH)	CICS expected a resynchronization process to occur during session initiation, but the LU did not resynchronize.	3
DFH3429	X'EC'	(TCZSTRMM)	Resynchronization error. CICS did not resynchronize, and the other LU was expecting resynchronization.	2,3
DFH3430	X'ED'	(TCZSTON)	Resynchronization error. Outbound flow sequence numbers do not agree with those of the other LU.	2,3
DFH3431	X'EF'	(TCZSTIN)	Resynchronization error. Flow sequence numbers do not agree.	2,3
DFH3432	X'EA'	(TCZSTLER)	Resynchronization error. Unexpected code received in response to STSN.	2,3
DFH3433	X'58'	(TCZERMGR)	Error message received. One side of the intersystem link sent a negative response or LUSTAT with code X'0846', implying that an error message would be the next message to follow. The sense code obtained from within the message is used to drive any appropriate actions.	none
DFH3434	X'54'	(TCZUNBIS)	UNBIND received while the session was active. One side of the intersystem link (the secondary LU) received an UNBIND command without the normal termination protocol being observed. An abnormal termination of the session was performed, probably caused because the other side of the link abnormally terminated.	2,3,9, 10,11,24
DFH3437			After an error has been processed by DFHZNAC, certain actions may be taken to correct the error. This message lists the actions taken.	
DFH3440	X'55'	(TCZEMWBK)	Unable to send error message.	none
DFH3441	X'63'	(TCZVTAMO)	Orderly termination of VTAM sessions requested - either by the CICS master terminal command or by the VTAM network closing down.	none

Figure 46 (Part 13 of 17). Error codes, error messages, and default actions generated by DFHZNAC for DFHZNEP

Error Message	Error Code	(Symbolic Label)	Condition	Action Flags (See notes at end of figure)
DFH3442	X'67'	(TCZVTAMK)	Immediate termination of VTAM sessions requested.	none
DFH3443	X'64'	(TCZVTAMA)	VTAM has been canceled.	none
DFH3452	X'66'	(TCZSIGR)	Signal received - code xxxx xxxx. SIGNAL command received from an LUTYPE4 logical unit. Signal code is available in TCTESIDI.	none
DFH3454	X'5E'	(TCZBRUAC)	Negotiable BIND response session parameter unacceptable.	2,3,5,18,24
DFH3455	X'5F'	(TCZBDSQP)	Bad session qualifier in BIND response.	2,3,5,18,24
DFH3461	X'48'	(TCZOPSIN)	Session started.	none
DFH3462	X'49'	(TCZCLSIN)	Session terminated.	none
DFH3463	X'4A'	(TCZPACB)	VTAM ACB opened. If the return code is nonzero, refer to the *ACF/VTAM Messages and Codes* manual.	none
DFH3464	X'57'	(TCZRELIS)	Release command issued by master terminal operator.	20
DFH3465	X'9D'	(TCZRSPER)	Unexpected response received.	1,2,3,9, 10,11,23
DFH3466	X'69'	(TCZSEXOS)	Terminal out of service after SDT sent.	20,23
DFH3468	X'70'	(TCZCLRRV)	Clear command received. The session is canceled immediately.	9,10,11, 15,24
DFH3469	X'CE'	(TCZVHOLD)	Session reestablishment is awaited. The secondary LU is passed to a new application program by CLSDST.	9,10,11,24
DFH3470	X'AB'	(TCZLUERR)	LU session failure.	9,10,11,24
DFH3471	X'CF'	(TCZVRNOP)	The session has been broken because the virtual route it was using has failed.	9,10,11,24
DFH3472	X'9E'	(TCZDEVND)	Device end received.	
			• A task is attached.	28,29,30,31,32
			• No task is attached, and the **good morning** message is supported.	13
			• No task is attached, and the **good morning** message is not supported.	11
DFH3473	X'4D'	(TCZSLSRL)	The CICS modules listed an earlier version of VTAM than that used for execution.	none

Figure 46 (Part 14 of 17). Error codes, error messages, and default actions generated by DFHZNAC for DFHZNEP

Error Message	Error Code	(Symbolic Label)	Condition	Action Flags (See notes at end of figure)
DFH3474	X'AC'	(TCZVRDAC)	The virtual route was deactivated.	9,10,11,24
DFH3475	X'AD'	(TCZNRLUF)	Unrecoverable LU failure.	9,10,11,24
DFH3476	X'AE'	(TCZRCLUF)	Recoverable LU failure.	9,10,11,24
DFH3477	X'AF'	(TCZCLEAN)	Cleanup received.	9,10,11,24
DFH3478	X'9F'	(TCZNOLUN)	Session could not be started.	23,24
DFH3479	X'4E'	(TCZUNBFE)	Unbind received.	2,3,9,10,11,24
DFH3480	X'A0'	(TCZNOISC)	CICS cannot support an LU6.2 system. This system entry will be prevented from being acquired in future.	23,24
DFH3481	X'4C'	(TCZDSPCL)	Zero length received from the 3270 Information Display System.	2,3,9, 10,11,24
DFH3482	X'DF'	(TCZGMDF)	OS GETMAIN failed during automatic installation.	none
DFH3483	X'DB'	(TCZDMAX)	Maximum active work elements reached during automatic installation.	none
DFH3484	X'C6'	(TCZPASSD)	CLSDST-PASS successful.	none
DFH3485	X'C7'	(TCZPSPRE)	CLSDST-PASS procedure error.	24
DFH3486	X'C8'	(TCZLUINH)	LU inhibited for sessions.	24
DFH3487	X'C9'	(TCZNPSAU)	CLSDST-PASS not authorized.	24
DFH3488	X'3C'	(TCZXUVAR)	Error on INQUIRE USERVAR	2,3,9,10,11,24
DFH4902	X'21'	(TCZLUCF1)	Attach FMH or subfield length error.	3,9,10,11,24
DFH4903	X'22'	(TCZLUCF2)	Attach FMH not found.	3,9,10,11,24
DFH4904	X'23'	(TCZFSMBE)	The bracket finite state machine found an error in the use of LU6.2 bracket protocols.	3,9,10,11,24
DFH4905	X'24'	(TCZFSMCS)	The chain finite state machine found an error in the use of LU6.2 chaining protocols.	3,9,10,11,24
DFH4906	X'25'	(TCZFSMCR)	The contention finite state machine found an error in the use of LU6.2 contention protocols.	3,9,10,11,24
DFH4907	X'26'	(TCZSDLER)	Invalid request to send data routine (DFHZSDL).	3,9,10,11,24
DFH4908	X'27'	(TCZSDLBF)	No buffer list passed to send data routine (DFHZSDL).	3,9,10,11,24

Figure 46 (Part 15 of 17). Error codes, error messages, and default actions generated by DFHZNAC for DFHZNEP

Error Message	Error Code	(Symbolic Label)	Condition	Action Flags (See notes at end of figure)
DFH4909	X'28'	(TCZRVLER)	Invalid request to receive data routine (DFHZRVL).	3,9,10,11,24
DFH4910	X'29'	(TCZRVLRB)	Receive buffer passed to receive data routine (DFHZRVL) too small.	3,9,10,11,24
DFH4911	X'2A'	(TCZRLXEX)	LU6.2 exception response received.	2,3,9, 10,11,24
DFH4912	X'2B'	(TCZRLXBD)	BID received with invalid data flow control (DFC) indicators.	2,3,9, 10,11,24
DFH4913	X'2C'	(TCZRLXBR)	BID command with data received with invalid data flow control (DFC) indicators.	2,3,9, 10,11,24
DFH4914	X'2D'	(TCZRLXIL)	Data length exceeds maximum RU size.	2,3,9, 10,11,24
DFH4915	X'2E'	(TCZRLXEC)	End-of-chain received with invalid data flow control (DFC) indicators.	2,3,9, 10,11,24
DFH4916	X'2F'	(TCZRLXRR)	Send response failed.	2,3,9, 10,11,24
DFH4917	X'30'	(TCZRLXIF)	BIS received with invalid data flow control (DFC) indicators.	2,3,9, 10,11,24
DFH4918	X'31'	(TCZRLXIR)	Unexpected response received.	2,3,9, 10,11,24
DFH4919	X'33'	(TCZIVIND)	Invalid data flow control (DFC) indicator received.	2,3,9, 10,11,24
DFH4920	X'34'	(TCZIVDAT)	Invalid data received. The data is not in the correct generalized data stream (GDS) format.	2,3,9, 10,11,24
DFH4922	X'32'	(TCZRLXCL)	088B received, DRAIN=CLOSE. on a single session.	20
DFH4924	X'B6'	(TCZNSEED)	No bind seed received.	2,3,5,24
DFH4925	X'B7'	(TCZASINC)	Inconsistent attach security requested.	2,3,5,24
DFH4926	X'B8'	(TCZEVBAD)	Invalid bind encryption.	2,3,5,24
DFH4927	X'B9'	(TCZFMH12)	No FMH12 received.	2,3,5,24
DFH4928	X'BD'	(TCZDESGM)	Encryption GETMAIN failed.	24
DFH4930	X'35'	(TCZRTMT)	RTIMEOUT on LU6.2 session.	2,3,9,10, 11,24
DFH4931	X'F1'	(TCZBDMOD)	VTAM detected an unknown MODENAME.	18,24

Figure 46 (Part 16 of 17). Error codes, error messages, and default actions generated by DFHZNAC for DFHZNEP

Notes:

	Action	Bit Setting Mask
1	Print action flags	X'80' at TWAOPT1
2	Print RPL	X'40' at TWAOPT1
3	Print TCTTE	X'20' at TWAOPT1
4	Print TIOA	X'10' at TWAOPT1
5	Print bind area	X'08' at TWAOPT1
9	Abend VTAM SEND	X'80' at TWAOPT2
10	Abend VTAM RECEIVE	X'40' at TWAOPT2
11	Abend task	X'20' at TWAOPT2
13	Good morning message required	X'08' at TWAOPT2
15	SIMLOGON required	X'02' at TWAOPT2
17	INTLOG can be set	X'80' at TWAOPT3
18	No internal LOGONs can be generated	X'40' at TWAOPT3
20	Close session (no user reset)	X'10' at TWAOPT3
21	Close session (user reset allowed)	X'08' at TWAOPT3
22	Negative response	X'04' at TWAOPT3
23	Keep node out of service	X'02' at TWAOPT3
24	Cancel session abnormally	X'01' at TWAOPT3
28	No action message	X'10' at TWAOPT4
29	No security message	X'08' at TWAOPT4
30	No message three	X'04' at TWAOPT4
31	No sense code message	X'02' at TWAOPT4
32	No level one message	X'01' at TWAOPT4

The default actions can vary from the actual actions set, depending on the state of the node at the time of the error.

Figure 46 (Part 17 of 17). Error codes, error messages, and default actions generated by DFHZNAC for DFHZNEP

System sense codes received

Figure 47 on page 215 lists the actions taken by DFHZNAC on receipt of inbound system sense codes. If no system sense code is received, no action is taken by DFHZNAC. However, the user sense code is available for analysis by the user's node error program. Refer to the chapter "The Node Error Program" in the *Customization Guide*.

If sense/status information is received from a 3270 Information Display System, one of the messages in Figure 48 on page 219 will be issued in addition to DFH2442, DFH2458, or DFH2469.

Detailed information on 3270 Information Display System sense/status is given in the *3274 Control Unit Description and Programmer's Guide*.

System Sense Received	Action Flags (See Figure 46)	Error Message	Condition
X'0801'	3,9,10,11	DFH2476	A component of the logical unit is no longer available. Probable cause is a device error or loss of contact. This sense can also be received with LUSTATUS.
X'0802'	none	DFH2461	Intervention required. Forms are required at an output device. See note at the end of this figure. This sense can also be received with LUSTATUS.
X'0806'	none	DFH3426	No matching TCTTE found.
X'0807'	none	DFH3411	Resource temporarily not available. See note at the end of this figure.
X'080B'	2,3,9,10,11, 15,24	DFH2462	Bracket error. Task initiation attempted by both the logical unit and CICS.
X'080E'	23	DFH3448	Security identification error. LU not authorized.
X'080F'	9,10,11	DFH2462/ DFH3436	End user not authorized.
X'0811'	9,10,11	DFH2464	Terminate chain. Chain rejected by the logical unit when purging incoming messages following error recovery.
X'0812'	2,3	DFH2465	Insufficient resources: • Diskette data set is full. • Data segment not large enough to handle data set. • Component temporarily not available.
X'081B'	2,3	DFH2483	Receiver in transmit mode. Action flags 2,3 are set for negative response received to a send that requested a definite response.
	2,3,9,10,11		2,3,9,10,11 are set for negative response to an exception response send, or if end bracket has been sent.
	2,3,9,10,11, 24		2,3,9,10,11,24 force close destination to be issued This action is set if CICS is not in send mode.
X'081C'	2,3,9,10,11	DFH2466	Function not executable. The logical unit cannot deliver the message to the node because of a data check condition or because the node is not available. This sense can also be received with LUSTATUS.

Figure 47 (Part 1 of 5). Actions taken by DFHZNAC upon receipt of inbound sense codes

System Sense Received	Action Flags (See Figure 46)	Error Message	Condition
X'0824'	3,9,10,11	DFH2475	The logical unit has abended all current processing with one of its components because of failure or loss of contact with that unit. This sense can also be received with LUSTATUS.
X'0825'	2,3,9,10,11	DFH2484	Component not available. An application request could not be satisfied.
X'0827'	3	DFH2480	The logical unit has requested retransmission of data from the host. This applies only to those transactions with message integrity or to requests using definite response protocol.
X'0829'	1,2,3,10,11, 24	DFH3407	Read command not marked change direction. The request requires the change direction indicator to be set. This was not done.
X'082A'	9,32	-	Presentation space error.
X'082B'	2,3,10,11,13	DFH3408	Presentation space integrity lost. Error on display or in buffer due to hardware error; for example, regenerated buffer parity error. This sense can also be received with LUSTATUS.
X'082D'	none	DFH3413	Logical unit busy - unable to process request. See note at the end of this figure.
X'082E'	none	DFH3412	Intervention required on secondary resource; resource is currently not available. See note at the end of figure.
X'082F'	2,3,9,10,11	DFH3414/ DFH3436	Request not executable. Secondary resource unavailable.
X'0831'	none	DFH3438	Device powered off. This sense can also be received with LUSTATUS.
X'0833'	none	DFH3427	Error in bind format.
X'0847'	none	DFH3439	Negative response received to start-data-traffic (SDT).
X'084A'	32	-	Presentation error on read. Display buffer alteration, due to operator intervention, detected on a READ command to a compatibility mode logical unit. X'082A' received in response to a send request is not treated as an error condition.

Figure 47 (Part 2 of 5). Actions taken by DFHZNAC upon receipt of inbound sense codes

System Sense Received	Action Flags (See Figure 46)	Error Message	Condition
X'084C'	9,10,11	DFH3467	Permanent insufficient resource. The programmed symbols (PS) buffer addressed by the LOAD PS statement is not available at this terminal.
X'0860'	none	DFH3459	The data stream sent to the logical unit contains control data for unsupported functions.
X'0863'	9,10,11	DFH3460	The requested programmed symbols (PS) set has not been loaded.
X'0864'	3,9,10,11	DFH2475	The logical unit has abended all processing connected with one of its components and no retry is possible.
X'0865'	3,9,10,11	DFH2465	The subsystem controller application program has insufficient resources to handle the request.
X'0866'	3,9,10,11	DFH2475	The logical unit has abended all processing connected with one of its components and retry is possible.
X'0867'	9,10,11		Function abend received from a device. A negative response to a chain was sent, but purged.
X'0868'	2,9,10,11	DFH3456	An outboard format is referenced, but no outboard formats are loaded on this logical unit.
X'0869'	2,9,10,11	DFH3457	The requested outboard format is not loaded on this logical unit.
X'08FF'	2,3,9,10,11	DFH3447	Request reject error.
X'1001' or X'1002'	2,3,9,10,11	DFH2481	The request-response (RR) unit transmitted to the logical unit was either untranslatable or it was too long or too short.
X'1003'	2,3,9,10,11	DFH2479	The request-response (RR) unit passed to the logical unit is not a supported function. Either a transmission error or data overlaying the RU caused the problem, or SCS parameters are not supported.
X'1005'	2,3,9,10,11	DFH3406	Parameter error. The RU received by the logical unit contains a control function with invalid parameters.

Figure 47 (Part 3 of 5). Actions taken by DFHZNAC upon receipt of inbound sense codes

System Sense Received	Action Flags (See Figure 46)	Error Message	Condition
X'1008'		DFH2478	Invalid function management header (FMH). Invalid length/type field in TIOA received with FMH, or invalid FMH parameters.
	2,3,9,10,11		If the request was not made by means of the batch data interchange program (DFHDIP), the action flags are set to 2,3,9,10,11.
	none		If DFHDIP was used, no action flag is set if DFHDIP determines that it can handle the situation, and
	9,10		9,10 are set if the error occurs on an exception response made by DFHDIP. In both cases, control returns to DFHDIP, which passes control to the application program with an appropriate DFHDIP return code.
X'1009'	2,9,10,11	DFH3458	Format group not selected.
X'10xx'	2,3,9,10,11	DFH3446	Request error.
X'2003'	2,3,9,10,11, 15,24	DFH3405	Catastrophic bracket error. An attempt to start a bracket conflicted with the receiver's understanding of the current bracket state.
X'20xx'	2,3,9,10,11, 24	DFH3445	State error.
X'400B'	1,3,11	DFH2477	The logical unit does not support chained data from the host. Consideration must be given to the amount of data to be transmitted to the logical unit.
X'40xx'	2,3,9,10,11, 24	DFH3453	Request header (RH) usage error.
X'8000'	2,3,9,10,11, 18,24	DFH3435	Path error.
X'80xx'	2,3,9,10,11, 18,24	DFH3435	Path error.
Other codes	2,3,9,10,11, 24	DFH2460	Sense received not supported. Sense codes not supported by CICS were received from the logical unit. If the system sense is zero, the user sense may indicate the source of the error. Codes X'0813' and X'0814' are handled entirely by CICS without DFHZNEP intervention.

Figure 47 (Part 4 of 5). Actions taken by DFHZNAC upon receipt of inbound sense codes

System Sense Received	Action Flags (See Figure 46)	Error Message	Condition
The following codes may be in the form of LUSTATUS command codes only. In addition, some codes of the X'08xx' type, as indicated above, may also be sent by LUSTATUS and are handled in the same way as when sent by negative response system sense.			
X'0001'	2	DFH3401	Component now available.
X'0002'	2,3,10,11	DFH3415	No data available. Logical unit has no data to send.
X'0003'	none	DFH3449	Leaving unattended mode.
X'0004'	none	DFH3450	Entering unattended mode.
X'0007'	none	DFH3451	Currently no data to send. No action flags are set if a task is attached or if outstanding operations are to complete;
	21		otherwise, 21 is set.

Note: For logical unit types 1, 2, 3, and 4 (that is, other than 3600, 3650, or 3790 inquiry logical units) CICS will not retry the failing request until it receives an LUSTATUS command with the system sense code of X'0001'. If the LUSTATUS system sense received is not X'0001', the resultant error action code will apply to the original request.

Figure 47 (Part 5 of 5). Actions taken by DFHZNAC upon receipt of inbound sense codes

3270 Sense/Status Received	Action Flags (See Figure 46)	Error Message	Condition
X'0210'	2,3	DFH2492	Intervention required on 3270 Information Display System printer: • Printer out of paper, cover open, or offline. • Transaction request to start printer, but no printer present. • Printer adapter feature not present.
X'0010'	none	DFH2493	Intervention required on a 3270 Information Display System.
X'xxxx'	2,3,9,10,11,	DFH2494	xxxx is any other status received from a 3270 Information Display System. The message contains the sense/status received.

Figure 48. Sense/Status information

Logon-reject sense codes

If the run-time level of VTAM is Version 3 Release 1, or later, CICS can send sense code information to VTAM. When a logon rejection occurs, this information can be used by VTAM to produce a message at the terminal (using the standard VTAM USSTAB mechanism). This message indicates whether the rejection was due to some transient problem, or a permanent condition requiring intervention to make the logon succeed.

Also, CICS sends messages to the CSMT message log, detailing the reasons for the logon failure. These messages should be used to diagnose problems, and aid problem resolution. The sense codes and their meanings are shown in Figure 49.

Sense	Meaning
X'0801'	The terminal is not available at this time, for example, CICS or VTAM is quiescing. You cannot repeat this.
X'080E'	The terminal (NETNAME) is undefined to CICS. The terminal should be defined, or allowed to autoinstall. You cannot repeat this.
X'0812'	A temporary condition prevents logon completion. You can repeat this.
X'0821'	Either invalid session parameters were supplied (for example, the terminal was specified as secondary), or an attempt to autoinstall this terminal failed (CICS messages explain why). You cannot repeat this.

Figure 49. Logon-reject sense codes

Session outage notification (SON)

CICS sends one of the following session outage notification (SON) codes when it issues CLSDST for a LU6.2 session:

```
X'01'  - Normal UNBIND (default)
X'0F'  - UNBIND after cleanup
X'FE'  - UNBIND after protocol violation. Message
           DFH3479 and actions 2,3,9,10,11,24.
```

In, addition, the following sense codes are set by CICS in the request parameter list (RPL) when appropriate (these sense codes qualify the SON codes):

```
X'1001'      - RU data error
X'1002'      - RU length error
X'10086000'  - FMH length incorrect
X'2002'      - Chaining error
X'2003'      - Bracket error
X'4003'      - BB not allowed
X'4019'      - Incorrect indicators with LIC request
```

Chapter 2.8. Debugging intercommunication problems

In a transaction that uses remote resources through CICS function request shipping, two transactions are involved – the CICS application transaction in the **local** system and the mirror transaction in the **remote** system. Information about either or both systems may be needed in debugging these problems; it is therefore suggested that, on each system, there should be **one** terminal defined to VTAM as a cross-domain resource that can be logged on to either of the systems and used to examine the status of the systems.

Offline debugging

Determine the system in which the **first** problem appears, by inspecting the time stamps on the CSMT logs of the two systems. If EXEC CICS START NOCHECK is used, then errors are generally not transmitted to the other system because the two systems are not holding a synchronized conversation.

Note: In general, most error indications will appear in both local and remote systems: an error indication in one system can be caused by an error recovery action taken after an error in the other system.

If the transaction (application or mirror) in the system that showed the first problem is found to be waiting, determine why the wait has occurred. (See "Chapter 1.3. Approach to wait problems" on page 33.)

Valid points in CICS function request shipping processing at which a transaction may wait are:

- In the application or mirror system when a terminal control read has been issued.
- In the application system only, when DFHZISP is attempting to allocate a session (TCTTE) to the application transaction.
- In the mirror system when a request is reissued.

Note: In normal operation, only one of the interconnected systems can be waiting (except during the period in which data is actually being transmitted from one system to the other).

If (from inspection of a trace) a CICS function request shipping transaction (application only) seems to be suspended indefinitely in DFHZISP, the problem can be approached thus:

1. Check first that the task is represented by an AID chained off the correct terminal control table system entry (TCTSE).

2. Assuming that it is represented in this way, see if the required session TCTTE is available for use (as indicated by TCTTECA being zero).

3. If the required session TCTTE **is** available, investigate why the freeing task's DFHKC TYPE=AVAIL has not worked.

4. If the required session TCTTE is **not** available, investigate the task that currently owns the session. If there is no task, investigate why DFHKC TYPE=UNAVAIL did not work.

Online debugging

Assuming that a terminal is available that has been defined to VTAM as a **cross-domain resource**, it can be used to log on to the remote CICS system and invoke the remote system's EDF (execution diagnostic facility) specifying the session TCTTE as the terminal that is to be put into EDF mode. It is then possible to monitor the actions of the **mirror** transaction in response to the actions of the local application transaction.

Transaction routing can be used to access the remote system if no terminals are available in that system. However, EDF cannot be used to monitor the actions of the mirror transaction because use of EDF in a two-terminal mode is not supported unless both terminals are connected to the same system.

Intersystem communication (LU6.2 transmission)

Organization of LU6.2 transmission modules

The modules that manage LU6.2 sessions are contained in the composite modules DFHZCC and DFHZCW. An LU6.2 command is passed to DFHETC and then to DFHZARL through DFHETL, or through DFHZARQ and DFHZARM. For SEND requests, DFHZARL assembles data in buffers (not TIOAs), and calls DFHZSDL to send the data to VTAM. For RECEIVE requests, DFHZARL places data from VTAM into buffers. The modules between the application request and DFHZARL handle the data in TIOAs, in the same way as the corresponding modules for non-LU6.2 transmission.

TCTTE LU6.2 control block

The most useful control block to look at is the LU6.2 extension to the TCTTE (TCTTELUC), which contains addresses and flags used by LU6.2. The SEND buffer is pointed to by TCTESBA, and the RECEIVE buffer by TCTERBA. The SEND/RECEIVE buffer is printed in a transaction dump.

LU6.2 data stream

Headers

X'LL0502FF....' The function management header (FMH) type 5 is transmitted at the beginning of every LU6.2 conversation. It is built automatically by CICS or as a result of the CONNECT PROCESS command.

X'LL07....' The FMH type 7 conveys error information; for example, as a consequence of an ISSUE ERROR command.

X'0001....' The SPS header conveys sync point information.

Data fields

A 4-byte header precedes data fields. The first two bytes are the length, the second two bytes are an identifier. The length includes the LLID bytes of the data. Common headers are:

X'LLLL12F2' This indicates that a non-LU6.2 FMH is included in the data which follows.

X'LLLL12F4' The data following is generated by CICS for error situations.

X'LLLL12FF' This indicates that general data follows.

Interpretation of traces

The following examples illustrate the history of a CONNECT PROCESS command (going through DFHETL) and a SEND command (going through DFHZARQ and DFHZARM).

CONNECT PROCESS command

The part of the trace table (or auxiliary trace) of interest is shown in Figure 50.

Trace ID	Type of Request	Trace Module	Description	Note
E1	0004	EIP	CONNECT-PROC ENTRY	1
FC	0203	ZCP	ZLOC LOC REQ ID LOCAL	2
FC	0215	ZCP	RETN ZLOC	
FC	4A03	ZCP	ZARL APPL REQ ISSUE ATTACH	3
FC	5604	ZCP	ZUSR CHECK CONNECT ALLOC-IN-PROG	4
F1	CC04	SCP	GETMAIN INITIMG	
C8	CC04	SCP	ACQUIRED USER STORAGE	
FC	5604	ZCP	ZUSR SET CONNECT ALLOC-IN-PROG	5
FC	0035	ZCP	RETN ZARL	
E1	00F4	EIP	CONNECT-PROC RESPONSE	6

Notes:

1. *The command is received.*
2. *Find the TCTTE address.*
3. *Pass control to DFHZARL.*
4. *DFHZARL checks the state of the session to see if the call is valid.*
5. *DFHZARL sets the new state of the session.*
6. *Return to DFHEIP with appropriate response.*

Figure 50. Trace table entries for LU6.2 CONNECT PROCESS command

In this example, DFHZARL has placed data into the buffer, but the buffer has not been sent. It is transmitted later.

SEND command

The part of the trace table (or auxiliary trace) of interest is shown in Figure 51 on page 225.

Trace ID	Type of Request	Trace Module	Description	Note
E1	0004	EIP	SEND_TC ENTRY	1
FC	0203	ZCP	ZLOC LOC REQ ID LOCAL	2
FC	0215	ZCP	RETN ZLOC	
F1	C304	SCP	GETMAIN INITIMG	
C8	C304	SCP	ACQUIRED TERMINAL STORAGE	3
FC	0103	ZCP	ZARQ APPL REQ WRITE WAIT	4
FC	4B03	ZCP	ZARM MIGR REQ SEND	5
FC	4A03	ZCP	ZARL APPL REQ SEND FROM	6
FC	5604	ZCP	ZUSR CHECK SEND ALLOC_SEND	7
FC	5604	ZCP	ZUSR SET SEND ALLOC_SEND	7
FC	0005	ZCP	RETN ZARL	
FC	4A03	ZCP	ZARL APPL REQ SEND FROM WAIT	8
FC	5604	ZCP	ZUSR SET SEND ALLOC_SEND	9
FC	5604	ZCP	ZUSR SET WAIT ALLOC_SEND	9
FC	4E04	ZCP	ZSDL SEND	10
F0	4004	KCP	WAIT DCI=DISP	
EE	2214	VIO	SEND BB FMH BC DATA RQE1	11
EE	0024	VIO	DATA	12
F0	4024	KCP	WAIT DCI=LIST	
FC	0005	ZCP	RETN ZARL	
F1	4404	SCP	FREEMAIN	
C9	4404	SCP	RELEASED TERMINAL STORAGE	13
FC	0005	ZCP	RETN ZARM	
FC	0005	ZCP	RETN ZARQ	
E1	00F4	EIP	SEND_TC RESPONSE	14

Notes:

1. *The command is received.*
2. *Find the TCTTE address.*
3. *A TIOA is acquired.*
4. *DFHZARQ receives the request.*
5. *Control is passed to DFHZARM.*
6. *The first call is made to DFHZARL, passing the length and data ID.*
7. *DFHZARL checks the request and sets the state of the session.*
8. *The second call to DFHZARL is made, passing the data and indicating WAIT.*
9. *DFHZARL checks the request and sets the state of the session.*
10. *The WAIT option makes DFHZARL call DFHZSDL to send the data to VTAM.*
11. *The SEND SNA indicators are shown.*
12. *Fields A and B of the trace (not shown in the figure) hold the beginning of the data sent to VTAM. The first half of field A is X'0031', the total length of data passed. The second half of field A and the whole of field B are X'240502FF0003', which is the start of the FMH type 5 for ATTACH.*
13. *The TIOA is released.*
14. *Return to DFHEIP with appropriate response.*

Figure 51. Trace table entries for LU6.2 SEND command

Multiregion operation

Suppose an error is suspected in the communication between System A and System B. The problem can be determined by looking at either system. The following procedure applies to System A, but could equally well apply to System B. The first step is to look at the field CSACRBA in the CSA optional features list in System A. If CSACRBA is zero, then the interregion communication routine is not active in that system.

If CSACRBA is not zero, then examine the TCTSEs in System A, and find the TCTSE for System B. In this TCTSE, TCSEIRCF is the flag byte that indicates the state of communication between the two systems. If bit TCSEIRNC in this byte is on, then there is no communication between System A and System B. This is either because System B has not started interregion services, or because System A is out of service with respect to System B, or because System B is out of service with respect to System A.

If bit TCSEIRNC is off, then a session should exist. The next step is to inspect the primary and secondary session(s) between the systems. The first primary session is pointed to by field TCSEVC1 in the TCTSE, and the first secondary session is pointed to by TCSEVC2. If System A initiated the session, then look at the secondary session(s), otherwise look at the primary session(s).

Each session is defined by a TCTTE. The field TCTESCCB in TCTTE is zero if the session is not connected to the other system, otherwise it contains the address of the subsystem connection control block (SCCB) that the interregion SVC routine uses to represent that end of the connection.

Assuming the session is connected, TCTTECA is zero if no task is using the session. Otherwise, TCTTECA points to the TCA of the task that uses the session.

The protocol for interregion SVC transfer is similar to that for VTAM SNA data flow control. Field TCTEIRF1 contains information on the state of the session, field TCTESBRS gives the bracket status, field TCTESRHI is the inbound request header, and field TCTESRHO is the outbound request header.

The field TCTENIBA points to the TCTTE extension for the NIB descriptor. Within this TCTTE extension, TCTEPSQ contains the primary name, and TCTESSQ contains the secondary name. Thus a session in System A can be related to a session in System B.

Shared data base problems

The *Diagnosis Reference* manual describes how the interregion communication (IRC) facility enables IMS/VS batch programs to share an IMS/VS DL/I data base with CICS application programs. Briefly, each DL/I call from an IMS/VS batch program is handled by batch region controller modules that invoke an interregion SVC routine to pass the request across to the CICS address space. In CICS, a mirror task handles the request and invokes the same SVC routine to return the response to the batch region. The batch region controller modules handle the response and pass it on to the batch program. The interface between IMS/VS batch programs and IMS/VS DL/I data bases is shown in Figure 52.

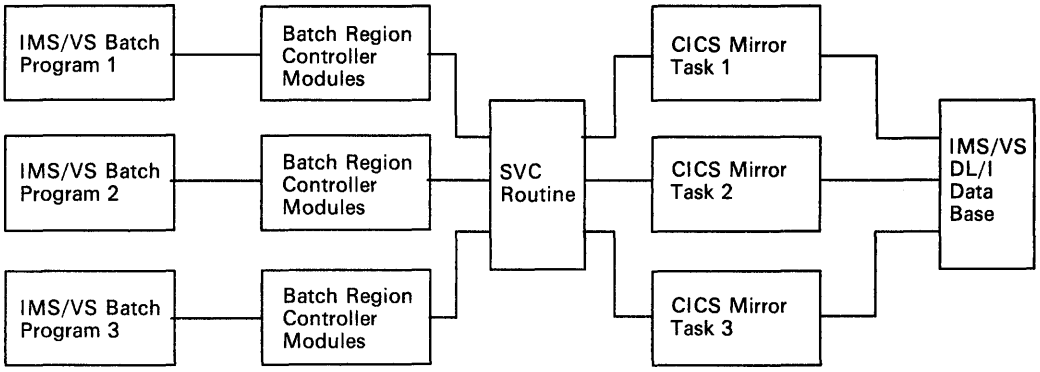


Figure 52. Interfaces between IMS/VS batch programs and IMS/VS DL/I data base

This section outlines suggested approaches to the investigation of problems in (1) the batch region, and (2) the CICS address space.

Investigation of problems in the batch region

This section describes how to obtain the following information from a batch region dump:

- The contents of the application program's registers at the time of abnormal termination or at the time of the last-issued DL/I call
- Information relating to messages caused by unsuccessful invocation of the SVC routine
- Input buffer contents.

IRC information in the batch region dump is contained in the **batch region work area**, which is identified with the following character string:

```
'*****DFHDRWA  -  WORK AREA'
```

The entire area is described in the DSECT named DFHDRWA (copy book DFHDRCA). The main diagnostic areas within DFHDRWA are shown in Figure 53.

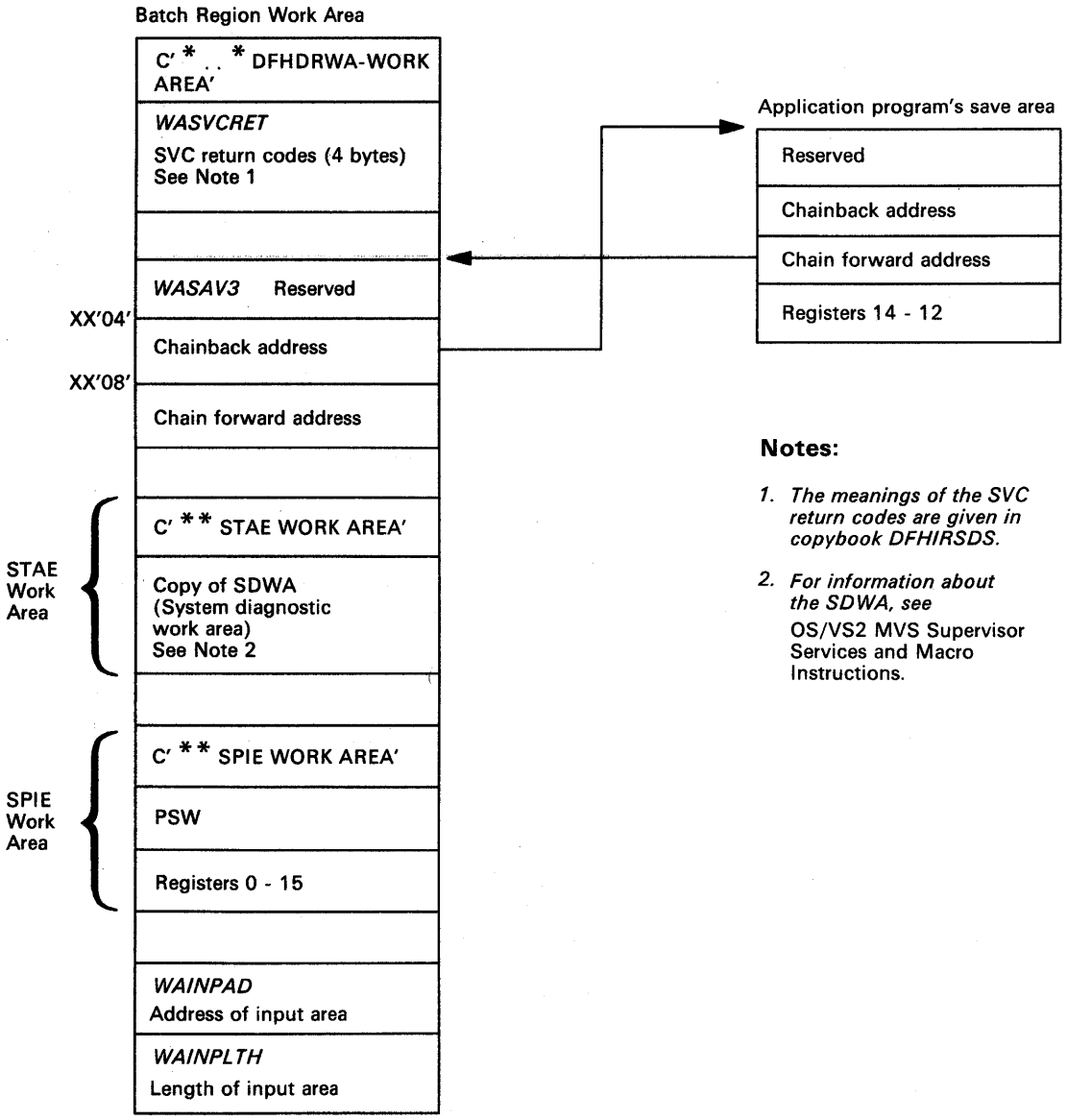


Figure 53. Diagnostic areas in IRC batch region dump

Dealing with a program check (message DFH3710)

To determine whether the program check occurred in the application program code or in the batch region controller code, inspect the PSW in the SPIE work area. The SPIE work area is identified by the characters C'**SPIE WORK AREA'; see Figure 53.

If the PSW shows that the program check occurred in the application program code, then the contents of the application's registers at that time can be found from the SPIE work area.

If the program check occurred in the controller code (DFHDRP...), then the application's registers at the time of the most recent DL/I call can be found from the application's save area (addressed by WASAV3 + X'04'). In particular, register 14 contains the address from which the most recent DL/I call was made.

Dealing with an abnormal termination (message DFH3701)

To determine whether the abend occurred in the application program code or in the batch region controller code, inspect the PSW in the STAE work area. The STAE work area is identified by the characters C'**STAE WORK AREA'; see Figure 53 on page 228. (Immediately after these characters is the system diagnostic work area (SDWA) as provided by MVS/XA on entry to the STAE exit routine. In the event of MVS/XA failing to provide an SDWA, work area (SDWA) as provided by OS/VS on entry to the STAE exit routine. In the event of OS/VS failing to provide an SDWA, the WANSDWA bit in WAFLG1 of DFHDRWA is set to 1. For information about the SDWA, see the *MVS Supervisor Services and Macro Instructions* manual.)

If the PSW shows that the abend occurred in the application program code, then the contents of the application's registers at that time can be found from the STAE work area.

If the abend occurred in the controller code (DFHDRP...), then the application's registers at the time of the most recent DL/I call can be found from the application's save area (addressed by WASAV3 + X'04'). In particular, register 14 contains the address from which the most recent DL/I call was made.

Dealing with failed SVC calls (messages DFH3706, DFH3709, and DFH3713)

Each of the messages DFH3706, DFH3709, and DFH3713 is the result of an unsuccessful invocation of the interregion SVC routine. The 4-byte return code returned by the SVC is stored in WASVCRET. The meanings of the SVC return codes are described in the interregion communication subsystem block copybook DFHIRSDS.

Reading contents of Application's IRC input buffer

If the problem is suspected to be the result of incorrect data received from CICS, then the contents of the application's IRC input buffer can be inspected. In the batch region work area, WAINPAD contains the address and WAINPLTH contains the length of the input area.

Online method of finding mirror task identification

Use the CEMT INQ IRBATCH request. The response to this request lists the task identifications of the mirror transactions for all batch jobs that are currently in communication with CICS.

If the faulty batch region and its mirror transaction have already terminated, then they will not appear in the inquiry list. In this event, follow the procedure described below under "Using trace table when mirror task identification is not known" on page 232.

Offline method of finding mirror task identification

Use the CICS dump as follows:

1. Find the TCT system entry that is identified with the characters C'=@BCH'; see Figure 54 on page 231.

2. From the address in the field TCSEMM, locate the first batch-connection TCTTE, which is identified with the characters C'@B1'. (Subsequent batch-connection TCTTEs are chained together, using the field TCTENEXT as a chain address, and are identified with the characters C'@B2', C'@B3', and so on. All TCTTEs have the same length, which is given in field TCTTETEL.)

3. In the batch-connection TCTTE whose TCTESLWD field contains the identification of the faulty batch job, use the TCTTECA field to locate the TCA for the mirror task. (If no TCTTE contains the required batch job identification, or if the TCTTECA field contains all zeros, then the mirror task was not active at the time of the dump; in this event, follow the procedure described below under "Using trace table when mirror task identification is not known" on page 232.)

4. Assuming the mirror task TCA has been located, obtain the mirror task identification from the TCAKCTTA field.

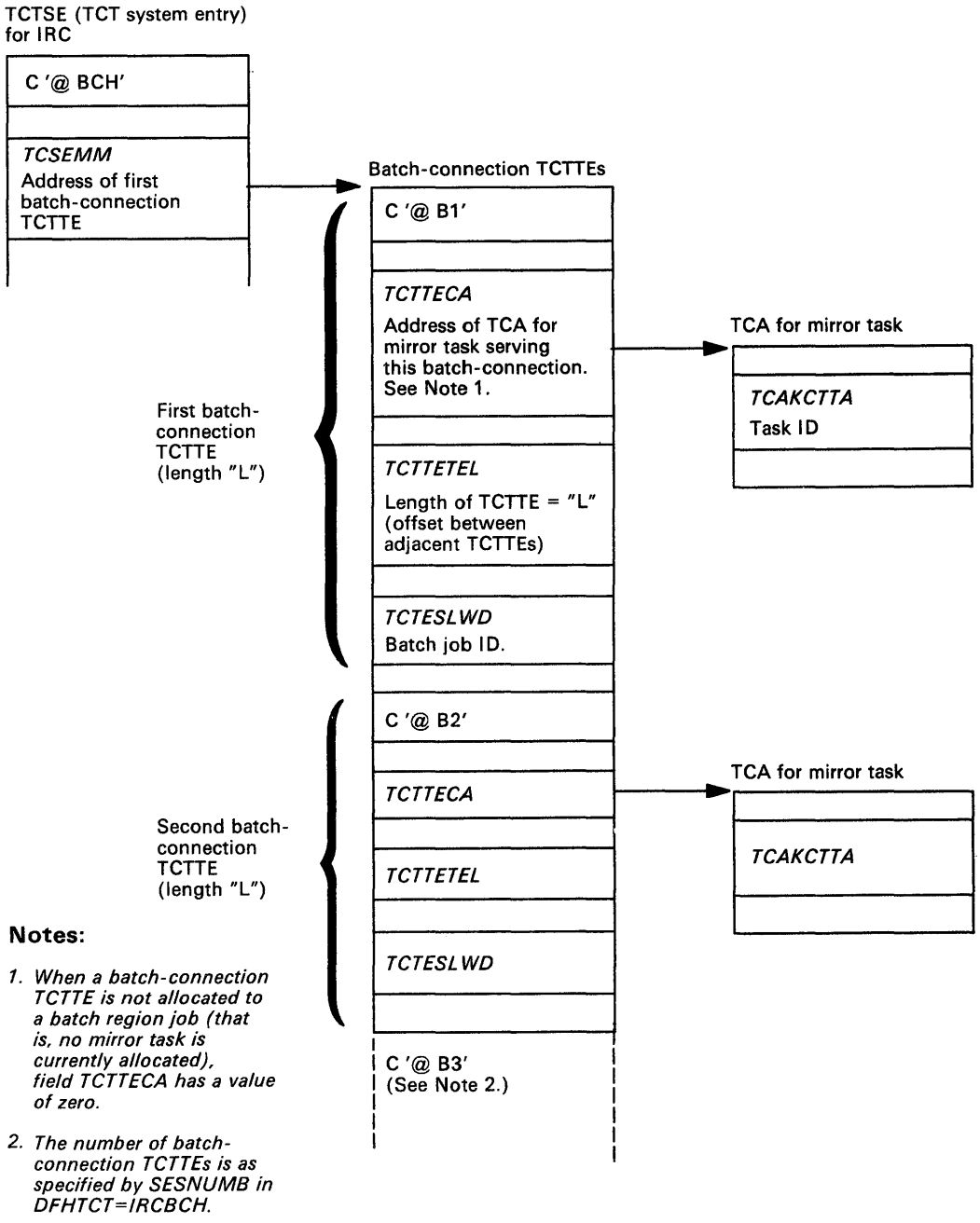


Figure 54. TCTTEs for batch job connections

Using trace table when mirror task identification is not known

The request may have been satisfied by the resumption of a reusable mirror task. To identify this resumption, look for a DFHKCP CREATE entry for a mirror task at about the same time as the batch job (or checkpoint) started. Analyze the subsequent trace entries that have the same task identification.

Also, DFHCRNP makes entries in the trace table that contains the batch LUWID.

Part 3. Review of CICS operation

Chapter 3.1. Normal operation

This chapter describes and illustrates:

- CICS initialization
- Normal transaction flow
- CICS termination.

CICS initialization

CICS can be initialized with one of the following:

- A complete cold start, preserving or deleting previously created series of journal tapes
- A complete warm start
- A partial warm start
- An emergency restart
- A standby restart.

We give here a brief description of these starts. For more details, see the *Recovery and Restart Guide.*

Complete cold start

A complete cold start results in complete reinitialization of CICS and system data sets to their status as specified at system generation, except that volume management data is maintained. The system initialization table (SIT) is used to specify the particular versions of the different CICS system programs and tables that are to be used for CICS initialization.

Resource definition information is obtained from the CICS system definition (CSD) file for those resources that the user has defined using the resource definition facility (transaction CEDA). The GRPLIST system initialization parameter specifies the particular groups to be used.

Note: A cold start does not use any system log or warm keypoint information from a prior execution of CICS.

Complete warm start

A complete warm start reinitializes CICS to the status that existed at the previous controlled shutdown – all system activity having been quiesced normally prior to shutdown. All CICS system programs and tables are first cold started using the SIT as described above. All CICS system tables are then reestablished to their status as at controlled shutdown, using the warm keypoint information written to the restart data set at that time.

A warm start is performed if START=AUTO is specified and CICS determines from the restart data set that the last shutdown was controlled; otherwise an emergency restart is performed.

Partial warm start

A partial warm start is similar to a complete warm start, except that only selected CICS system tables are warm started, as specified in the SIT. Information is obtained from the warm keypoint written at controlled shutdown, only for those tables specified to be warm started. The remaining tables are cold started. An example requiring a partial warm start is an application that requires a warm start of the DCT so that data queued to intrapartition destinations prior to a controlled shutdown may be retrieved on restart. The FCT may also need to be warm started to reestablish file status as at controlled shutdown. The application may, however, require a cold start of temporary storage.

The PCT, PPT, and TCT cannot be individually specified for warm or cold start.

After all items have been initialized and control is about to be given to CICS, the group of user-written programs specified in the program list table is sequentially executed. This is referred to as the **postinitialization** phase. These programs perform application dependent functions, for the recovery of application dependent information recorded by the user on termination, prior to complete restart of CICS. All CICS facilities are available except for direct terminal communication. Following postinitialization execution of programs in the program list table, the terminal control program is activated to enable terminal transactions to be received and processed.

Emergency restart

An emergency restart restores certain CICS facilities to a predefined point that existed prior to an uncontrolled shutdown. Information describing all changes, modifications and updates made to various system tables and to user data sets during previous CICS execution is recorded on the system log data set.

During emergency restart, the recovery control recovery program (DFHRCRP) calls the recovery utility program (DFHRUP) to scan the system log backward, and the records relating to **in-flight** tasks (those that had not completed at the time of the uncontrolled shutdown) are copied to the restart data set by DFHRUP. The backout programs then use the information on the restart data set to restore the recoverable resources to the state they were in at the start of the task or when the last sync point was executed.

Standby restart

This start is used by the alternate system in an XRF environment. The alternate system is only partially initialized, because certain resources can only be used by one system at a time. Initialization is completed when the alternate system takes over from the active system.

Initialization sequence

Figure 55 on page 238 shows the sequence of initialization of CICS. The messages referred to in this figure are not complete. The *Messages and Codes* manual refers to the module issuing the message and is useful in debugging initialization problems, most of which are likely to be errors in setting up the CICS system.

MODULE	ACTION	NOTE/MESSAGE
DFHSIP		DFHSIP is loaded into storage by the host operating system
	Acquire storage in address space	DFH1500 CICS startup in progress
	Accept SIT override parameters Check timer operation Load SIT	DFH1501 SIT is being loaded
	Allocate area for temporary CSA	
DFHSIA1	Check and merge SIT parameters and overrides	
DFHSIB1	Build list of nucleus modules Load nucleus load table if required Build CICS nucleus Initialize CSA addresses Build trace table Issue ESTAE macro for IRC error-handling exit Determine whether COLD start, WARM start, or EMERGENCY restart is to be carried out If VTAM=YES is specified, build a receive-any RPL pool (RACE)	DFH1500 loading CICS nucleus
DFHSIC1	Build user exit control blocks Open the restart data set Determine the type of start Attach DFHTEOF if required Initialize the user exit table For XRF=YES, sign on to the CAVM	
DFHSID1	Allocate transient data control blocks Warm/cold start DCT Initialize intrapartition storage if required Initialize multiple buffers for transient data.	
DFHSIE1	Warm/cold start FCT Build resource pool if using DL/I and VSAM shared resources, and if multiple resource pool capability is not available.	
DFHSIF1	Open BTAM terminal control data set Open VTAM ACB Initialize the OS console program Construct TCT hash table Build ZCP module list in the TCT Initialize TCT attach tables For START=STANDBY, open the VSAM ACB	DFH1500 terminal data sets are being opened

Figure 55 (Part 1 of 3). CICS initialization modules — sequence of execution

MODULE	ACTION	NOTE/MESSAGE
DFHSIG1	Open dump data set	DFH1500 dump data set A
	Initialize dump control	or B is being opened
DFHSIH1	Attach journal open/close subtask	
	Initialize journal control	
	Calculate OSCOR	
	For START=STANDBY, call phase 1A of DFHDLQ	
	For other values of START, call phase 1 of DFHDLQ	
	Format page allocation map	DFH1500 subpool size for
	Initialize multiple buffers for temporary storage.	this startup is nnnK
	Allocate temporary storage control blocks	
DFHSII1	Initialize system console support	
	Abend exit initialization (SPIE or ESTAE)	See note 1
	Open auxiliary trace data set	
	Attach the III task	
	Warm start CSA	
	Warm start TS	
	Warm/cold start ICE, AID	
	Take activity keypoint	
	Ask operator whether to continue EMER restart	
	Pass control DFHZDSP	
DFHSIJ1	Start IRC session if IRCSTRT=YES	DFH1500 interregion communication started
	Load PLT, execute programs	
	Initiate CRSQ transaction if intercommunication is in use	
	Attach CSFU to open files	
	Branch to DFHTCP to begin soliciting terminal input	DFH1500 control is being given to CICS
	If using CICS VSAM subtasking, the OS/VS subtask program DFHVSP is attached.	
	For systems with DL/I, call DFHDLX to perform **end-of-initialization** processing.	
	Return, using program control	

Figure 55 (Part 2 of 3). CICS initialization modules — sequence of execution

MODULE	ACTION	NOTE/MESSAGE
III task	For XRF=YES: Attach the surveillance task (DFHXRSP) For XRF=YES,START=STANDBY: Attach the console communication task (DFHXRCP) Attach the transient data resource manager task Attach the temporary storage resource manager task Wait for takeover complete For XRF=NO, or XRF=YES,START=STANDBY: Open the restart data set (RSD) Wait for open check For all cases: Attach the remaining resource manager tasks Wait for recovery complete Process resident programs Pass control to DFHSIJ1	

Notes:

1. *Branch to DFHSRP, issue abend exit macro giving abend exit and save areas within DFHSRP (see "Chapter 3.3. Abnormal condition handling" on page 275 in this manual), return to DFHSIP.*

Figure 55 (Part 3 of 3). CICS initialization modules – sequence of execution

Normal transaction flow

The flow of control in executing a transaction is illustrated by an annotated auxiliary trace output in Figure 58 on page 243. See "Chapter 2.1. Trace" on page 45 for a description of the trace entries. Commands are numbered according to their sequence in the program, rather than the order of execution; they are referenced by these numbers in the annotated trace.

The trace is produced by executing the supplied sample application programs, DFH$PREN and DFH$CREN. The PL/I source of DFH$PREN (in outline only) is shown in Figure 56. The COBOL source of DFH$CREN (in outline only) is shown in Figure 57 on page 242.

```
/*****************************************************************/
/*      DFH$PREN - CICS/VS SAMPLE FILEA ORDER ENTRY - PL/I       */
/*****************************************************************/
ORDER:PROC  OPTIONS(MAIN);
    .
    .
    EXEC CICS HANDLE AID CLEAR(ENDPORD);                          Command 1.
    EXEC CICS HANDLE CONDITION MAPFAIL(MAPFAIL)                   Command 2.
                               ERROR  (ERRORS)
                               NOTFND (NOTFOUND);
    .
    .
    EXEC CICS SEND MAP('DFH$PGK') ERASE;                          Command 3.
RECEIVE:
    EXEC CICS RECEIVE MAP('DFH$PGK');                             Command 4.
    .
    .
    EXEC CICS READ DATASET('FILEA') INTO(FILEA) RIDFLD(CUSTNOI);  Command 6.
    .
    .
    EXEC CICS WRITEQ TD QUEUE ('L860') FROM (L860) LENGTH(22);    Command 7.
    EXEC CICS SEND MAP('DFH$PGK') MAPONLY ERASEAUP;               Command 8.
GOTO RECEIVE;
    .
    .
ENDPORD:
    EXEC CICS SEND TEXT FROM(PRESMSG) ERASE;                      Command 13.
    EXEC CICS SEND CONTROL FREEKB;                                Command 14.
    EXEC CICS RETURN;                                             Command 15.
    END;
```

Figure 56. Sample transaction PL/I source

```
*****************************************************************
*    DFH$CREN - CICS/VS SAMPLE FILEA ORDER ENTRY - COBOL        *
*****************************************************************
 PROCEDURE DIVISION.
      .
      .
      EXEC CICS HANDLE AID CLEAR(ENDA) END-EXEC.                    Command 1.
      EXEC CICS HANDLE CONDITION MAPFAIL(MAPFAIL)                   Command 2.
                                 NOTFND(NOTFOUND)
                                 ERROR(ERRORS) END-EXEC.
      .
      .
      EXEC CICS SEND MAP('ORDER') MAPSET('DFH$CGK')                 Command 3.
                ERASE END-EXEC.
 RECEIVM.
      EXEC CICS RECEIVE MAP('ORDER') MAPSET('DFH$CGK') END-EXEC.    Command 4.
      .
      .
      EXEC CICS READ DATASET('FILEA') INTO(FILEA) RIDFLD(CUSTNOI)   Command 6.
                                                  END-EXEC.
      .
      .
      EXEC CICS WRITEQ TD QUEUE('L860') FROM(L860) LENGTH(22)       Command 7.
                                                   END-EXEC.
      EXEC CICS SEND MAP('ORDER') MAPSET('DFH$CGK')                 Command 8.
                MAPONLY ERASEAUP END-EXEC.
      GO TO RECEIVM.
      .
      .
 ENDA.
      EXEC CICS SEND TEXT FROM(PRESMSG) LENGTH(20) ERASE END-EXEC.  Command 13.
      EXEC CICS SEND CONTROL FREEKB END-EXEC.                       Command 14.
      EXEC CICS RETURN END-EXEC.                                    Command 15.
      GOBACK.
```

Figure 57. Sample transaction COBOL source

The trace output is a result of executing DFH$PREN. For space reasons, the interval between consecutive trace entries is not included in the illustration. The trace output for DFH$CREN is the same, except for the program names, maps, and transaction identifiers.

Note: Reference is made, in the following annotated trace, to CICS nucleus modules, as follows:

```
EIP - DFHEIP - exec interface program
KCP - DFHKCP - task control program
ZCP - DFHZCP - terminal control program (BTAM and VTAM)
SCP - DFHSCP - storage control program
TDP - DFHTDP - transient data program
FCP - DFHFCP - file control program
TMP - DFHTMP - table management program
BMS - DFHMCP - basic mapping support
PCP - DFHPCP - program control program
SPP - DFHSPP - sync point program
XSP - DFHXSP - external security program
```

```
   TIME OF DAY    ID  REG 14  REQD TASK   FIELD A  FIELD B  CHARS   RESOURCE  TRACE TYPE

08:35:17.502784  D0 502F1E6A 0504 TCP   00000000 00000000 ........           KCP DISPATCH
08:35:17.504256  F0 402DE1E8 4004 TCP   40000000 002572E0  .......           KCP WAIT DCI=LIST
08:35:18.465056  D0 502F1E6A 0904 KCP   00010000 D2250FC3 ....K..C           KCP SYSTEM RESUME
08:35:18.466560  D0 502F1E6A 0504 TCP   00000000 00000000 ........           KCP DISPATCH
08:35:18.466912  FC 402DE14E 1404 TCP   00020001 0011F784 ......7.           ZCP ZRAC RECEIVE ANY
08:35:18.466976  FC 502D9052 1304 TCP   01000001 0011F784 ......7.           ZCP ZGET GETMAIN
                              Terminal control program detects incoming data from VTAM.
08:35:18.467456  EE 702D80EC 2314 TCP   000201A0 0111F784 ......7. 16331624  VIO RECEIVE OIC DATA
08:35:18.467488  EE 702D80EC 0024 TCP   00077D40 C4979699 ..' D... 16331624  VIO DATA
                              The resource is the VTAM communications ID, consisting of
                              the origin address field and the destination address field.
                              This can be used to refer to a VTAM trace.
08:35:18.467680  F1 502D846C A504 TCP   001103E8 0111F784 ...Y..7.           SCP GETMAIN CONDITIONAL
08:35:18.467904  C8 502ABA30 0004 TCP   0011A240 851103F8 ... ...8           SCP ACQUIRED TERMINAL
                                                                                 STORAGE
                              Terminal control asks storage control
                              for a terminal input/output area (TIOA) to hold the
                              incoming data.
08:35:18.469088  FC 402DD3D4 1103 TCP   01200001 0011F784 ......7.           ZCP ZATT ATTACH
                              Terminal control starts to attach a CICS task.
08:35:18.469152  F0 502E03A6 9304 TCP   D3F7F7C3 D7D6D9C4 L77CPORD           KCP ATTACH-CONDITIONAL
08:35:18.469216  EA 502EF786 0003 TCP   01040100 002F62B0 ........ PORD      TMP PCT LOCATE
08:35:18.469504  EA 402C0E34 0025 TCP   01040100 002AA750 .......&           TMP RETN NORMAL
                              TMP is invoked to look up the PCT entry for the transaction PORD.
08:35:18.504128  EA 402EFEAA 0003 TCP   0C000100 002AA750 .......& PORD      TMP PCT TRANSFER
08:35:18.504192  EA 402C0E34 0025 TCP   0C000100 00000000 ........           TMP RETN NORMAL
                              TMP is invoked to transfer the lock-on PCT entry for PORD
                              to the CICS task about to be created.
08:35:18.504224  FC 402E0496 1125 TCP   00000000 00000000 ........           ZCP RETN ZATT ATTACH
08:35:18.504320  F0 402DE1E8 4004 TCP   40000000 002572E0  .......           KCP WAIT DCI=LIST
08:35:18.504352  D0 502F1E6A 0504 TCP   00000000 00000000 ........           KCP DISPATCH
08:35:18.504448  F0 402DE1E8 4004 TCP   40000000 002572E0  .......           KCP WAIT DCI=LIST
08:35:18.504512  F1 402F111C EA04 KCP   00000B10 00000000 ........           SCP GETMAIN CONDITIONAL
                                                                                 INITIMG
08:35:18.505248  C8 502ABA30 0004 KCP   0011B000 8A040B18 ........           SCP ACQUIRED TCA STORAGE
                              KCP obtains storage for the new task's TCA.
08:35:18.505280  D0 502F1400 0604 00022 D3F7F7C3 D7D6D9C4 L77CPORD           KCP CREATE
                              Record the task creation using the task number
                              of the new task.
08:35:18.505312  D0 502F1E6A 0504 00022 00000000 00000000 ........           KCP DISPATCH
08:35:18.505344  FC 052E0C38 0503 00022 00200001 0011F784 ......7.           ZCP ZSUP START UP TASK
                              ZSUP receives control to perform security checking,
                              TCTTE logging, PCT option checking, and
                              transfers control to the user code.
08:35:18.505472  E5 502E1272 0C03 00022 0011B428 00000000 ........ PORD      XSP SECURITY CHECK
08:35:18.505504  E5 402E78AA 0005 00022 00000000 00000000 ........           XSP SECURITY RETN
                              The security program confirms that this transaction
                              can be used by this operator.
08:35:18.505536  F2 402E1E30 8804 00022 00000000 00000000 ........ DFH$PREN  PCP XCTL-CONDITIONAL
                              PCP is invoked to transfer control to the program DFH$PREN.
08:35:18.505600  EA 402A6CE4 0003 00022 01000300 0011B214 ........ DFH$PREN  TMP PPT LOCATE
08:35:18.505856  EA 402C0E34 0005 00022 01000300 002A46C4 .......D           TMP RETN NORMAL
                              TMP checks the PPT entry for DFH$PREN.
08:35:18.703680  F1 402A63BC 8804 00022 000001C5 0111F784 ...E..7.           SCP GETMAIN
08:35:18.705088  C8 502ABA30 0004 00022 00205000 88001000 ..&.....           SCP ACQUIRED PGM STORAGE
                              Task 22's TCA is now the controlling TCA.
                              PCP detects that the program is not in storage, so it obtains
                              storage into which the program is loaded.
                              PCP then relinquishes control until the load is complete.
```

Figure 58 (Part 1 of 11). Trace table entries for sample transaction

```
   TIME OF DAY   ID  REG 14  REQD TASK   FIELD A  FIELD B  CHARS   RESOURCE  TRACE TYPE

08:35:18.705376  F0 502A649A 2004 00022 FF000000 00000000 ........          KCP CHAP
08:35:18.705536  D0 502F1E6A 0504 00022 00000000 00000000 ........          KCP DISPATCH
                           The program is loaded under the user's task on behalf of the
                           rest of CICS. Task 22 is given the highest priority
                           until the load is complete.
08:35:18.709568  F0 70105E10 4004 00022 80000000 00105440 .......          KCP WAIT DCI=SINGLE
08:35:18.727968  D0 502F1E6A 0904 KCP   00010001 D22510E9 ....K..Z          KCP SYSTEM RESUME
08:35:18.729184  D0 502F1E6A 0504 00022 00000000 00000000 ........          KCP DISPATCH
                           TCP is given control to check for terminal activity.
                           There is none, so KCP relinquishes control by an
                           operating system WAIT macro.
                           After a certain elapsed time (ICV value), an operating system
                           interval timer interrupt causes control to return to KCP.
                           KCP passes control to TCP to scan for terminal activity.
                           This cycle continues until .............
08:35:18.899424  D0 502F1E6A 0504 00022 00000000 00000000 ........          KCP DISPATCH
08:35:18.900960  F0 70105E10 4004 00022 80000000 00105440 .......          KCP WAIT DCI=SINGLE
08:35:18.916000  D0 502F1E6A 0904 KCP   00010001 D2251118 ....K...          KCP SYSTEM RESUME
                           Above three entries repeated several times.
08:35:18.916512  D0 502F1E6A 0504 00022 00000000 00000000 ........          KCP DISPATCH
08:35:18.916608  F0 402A7306 2004 00022 01000000 00000000 ........          KCP CHAP
                           ........ the load is complete, when KCP restores the task's
                                      original priority.
08:35:18.916672  D0 502F1E6A 0504 00022 00000000 00000000 ........          KCP DISPATCH
08:35:18.916736  F1 502F2FFC CC04 00022 0000024C 0111F784 ......7.          SCP GETMAIN INITIMG
08:35:18.916800  C8 502ABA30 0004 00022 0011BB20 8C000258 ........          SCP ACQUIRED USER
                                                                              STORAGE
                           PCP obtains storage for the PL/I ISA. The area is for PL/I
                           automatic storage, and is for the use of this invocation of
                           DFH$PREN only.
08:35:18.951424  F1 50280CD0 CC04 00022 00000670 0111F784 ......7.          SCP GETMAIN INITIMG
08:35:18.952352  C8 502ABA30 0004 00022 0011BD80 8C000678 ........          SCP ACQUIRED USER
                                                                              STORAGE
08:35:18.952480  E1 5020589E 0004 00022 0011C200 00000206 ..B.....          EIP HANDLE-AID ENTRY
                           Command 1 in the sample program DFH$PREN is started.
08:35:18.952512  F1 602F39F0 CC04 00022 0000007F 0111F784 ......7.          SCP GETMAIN INITIMG
08:35:18.952544  C8 502ABA30 0004 00022 0011C400 8C000088 ..D.....          SCP ACQUIRED USER
                                                                              STORAGE
                           EIP obtains storage for a HANDLE AID label table.
                           More areas would be required in the case of a COBOL program.
08:35:18.952576  E1 5020589E 00F4 00022 00000000 00000206 ........          EIP HANDLE-AID RESPONSE
                           EIP traces the completion of the command, and returns
                           to the application program.
08:35:18.952608  E1 50205904 0004 00022 0011C200 00000204 ..B.....          EIP HANDLE-CONDITION
                                                                              ENTRY
                           Command 2 in DFH$PREN is started by EIP.
08:35:18.952640  F1 602F39F0 CC04 00022 000000B5 0111F784 ......7.          SCP GETMAIN INITIMG
08:35:18.952672  C8 502ABA30 0004 00022 0011C490 8C0000C8 ..D....H          SCP ACQUIRED USER
                                                                              STORAGE
                           EIP obtains storage for a HANDLE CONDITION label table.
08:35:18.952736  E1 50205904 00F4 00022 00000000 00000204 ........          EIP HANDLE-CONDITION
                                                                              RESPONSE
                           EIP traces completion, and returns to the application program.
08:35:18.952768  E1 50205988 0004 00022 0011C200 00001804 ..B.....          EIP SEND-MAP ENTRY
                           Command 3 in DFH$PREN is started by EIP.
08:35:19.002912  F1 502F39F0 4004 00022 0011A240 0111F784 ... ..7.          SCP FREEMAIN
08:35:19.003008  C9 502ABA6C 0004 00022 0011A240 851103F8 ... ...8          SCP RELEASED TERMINAL
                                                                              STORAGE
                           The original TIOA passed to the task by TCP is released.
08:35:19.044800  FA 50290726 0003 00022 000004E2 04000020 ...S....          BMS SEND-OUT MAP SAVE
                                                                              ERASE
                           EIP requests the services of basic mapping support (BMS).
```

Figure 58 (Part 2 of 11). Trace table entries for sample transaction

```
  TIME OF DAY   ID  REG 14  REQD TASK   FIELD A  FIELD B  CHARS   RESOURCE  TRACE TYPE

08:35:19.044832  F1 4028E150 CC04 00022 00000310 0111F784 ......7.          SCP GETMAIN INITIMG
08:35:19.044896  C8 502ABA30 0004 00022 0011C560 8C000318 ..E-....          SCP ACQUIRED USER
                                                                               STORAGE
                           BMS obtains an output services processor work area (OSPWA)
                           the first time it is invoked for a task.
08:35:19.064608  F1 602896CA CC04 00022 000000CC 0111F784 ......7.          SCP GETMAIN INITIMG
08:35:19.064672  C8 502ABA30 0004 00022 0011C880 8C0000D8 ..H....Q          SCP ACQUIRED USER
                                                                               STORAGE
                           BMS obtains storage for a direct terminal type parameter (TTP)
                           and places its address in the OSPWA.
                           A TTP is obtained for each device type to receive this message.
08:35:19.105952  F2 4028F3EC 0404 00022 00000000 00000000 ........ DFH$PGK  PCP LOAD
                           BMS starts to load the user's map.
08:35:19.108672  EA 402A6CE4 0003 00022 01000300 0011B214 ........ DFH$PGK  TMP PPT LOCATE
08:35:19.108928  EA 402C0E34 0005 00022 01000300 002A47DC ........          TMP RETN NORMAL
                           TMP is invoked to look up the PPT entry for map DFH$PGK.
08:35:19.283872  F1 402A63BC 8804 00022 00110031 0111F784 ......7.          SCP GETMAIN
08:35:19.285152  C8 502ABA30 0004 00022 00204000 88001000 .. .....          SCP ACQUIRED PGM
                                                                               STORAGE
                           BMS requests PCP to load the map. PCP obtains
                           storage in which the map will reside.
08:35:19.285312  F0 502A649A 2004 00022 FF000000 00000000 ........          KCP CHAP
08:35:19.285440  D0 502F1E6A 0504 00022 00000000 00000000 ........          KCP DISPATCH
                           The map is loaded under the user's task on behalf of the
                           rest of CICS. Task 22 is given the highest priority
                           until the load is complete.
08:35:19.287872  F0 70105E10 4004 00022 80000000 00105440 .......          KCP WAIT DCI=SINGLE
                           PCP relinquishes control while the map is loaded.
08:35:19.362080  D0 502F1E6A 0504 00022 00000000 00000000 ........          KCP DISPATCH
08:35:19.363488  F0 70105E10 4004 00022 80000000 00105440 .......          KCP WAIT DCI=SINGLE
08:35:19.378688  D0 502F1E6A 0904 KCP   00010001 D2251188 ....K...          KCP SYSTEM RESUME
                           Above three entries repeated several times.
08:35:19.379168  D0 502F1E6A 0504 00022 00000000 00000000 ........          KCP DISPATCH
08:35:19.379232  F0 402A7306 2004 00022 01000000 00000000 ........          KCP CHAP
                           The load is complete, so KCP restores task 22's priority.
08:35:19.379296  D0 502F1E6A 0504 00022 00000000 00000000 ........          KCP DISPATCH
                           KCP returns control to BMS after I/O completion.
08:35:19.410624  F1 6028B7B6 9E04 00022 002001D1 0111F784 ...J..7.          SCP GETMAIN
08:35:19.410656  C8 502ABA30 0004 00022 0011C960 9E2001E8 ..I-...Y          SCP ACQUIRED MAPCOPY
                                                                               STORAGE
                           BMS obtains storage for a MAPCOPY, as the loaded map is
                           logically read-only. Many such areas may be obtained
                           during page-building.
08:35:19.479040  F1 702924E6 8504 00022 00110170 0111F784 ......7.          SCP GETMAIN
08:35:19.479264  C8 502ABA30 0004 00022 0011A240 85110188 ... ....          SCP ACQUIRED TERMINAL
                                                                               STORAGE
                           BMS obtains storage for a TIOA, which is used to output
                           the map to the screen.
08:35:19.479264  F1 502925A8 CC04 00022 00000074 0111F784 ......7.          SCP GETMAIN INITIMG
08:35:19.479328  C8 502ABA30 0004 00022 0011CB50 8C000088 ...&....          SCP ACQUIRED USER
                                                                               STORAGE
                           BMS obtains a merged data area to continue building the TIOA.
08:35:19.497536  F1 60292EFE 4004 00022 0011C960 0111F784 ..I-..7.          SCP FREEMAIN
08:35:19.497568  C9 502ABA6C 0004 00022 0011C960 9E2001E8 ..I-...Y          SCP RELEASED MAPCOPY
                                                                               STORAGE
08:35:19.497600  F1 60292F58 4004 00022 0011CB50 0111F784 ...&..7.          SCP FREEMAIN
08:35:19.497632  C9 502ABA6C 0004 00022 0011CB50 8C000088 ...&....          SCP RELEASED USER
                                                                               STORAGE
                           The TIOA is complete, so BMS releases its work areas.
```

Figure 58 (Part 3 of 11). Trace table entries for sample transaction

```
   TIME OF DAY   ID  REG 14  REQD TASK   FIELD A  FIELD B  CHARS   RESOURCE  TRACE TYPE

08:35:19.497856  FC 4028DD4E 0103 00022 00810000 0011F784 ......7.          ZCP ZARQ APPL REQ ERASE
                                                                               WRITE
                           BMS requests ZCP to write the map from the TIOA.
                           The I/O request to the access method may
                           be deferred until ZCP is called for a later request,
                           to allow SNA indicators to be sent together with
                           message traffic.
08:35:19.497920  FC 402DE97C 0105 00022 00810400 0011A240 .......           ZCP RETN ZARQ APPL REQ
                                                                               ERASE WRITE DEFER
                           ZCP returns control to BMS.
08:35:19.497952  FA 4028EF3E 0005 00022 00000000 00000000 ........          BMS RETN
                           BMS returns control to EIP.
08:35:19.498016  E1 5C205988 00F4 00022 00000000 00001804 ........          EIP SEND-MAP RESPONSE
                           EIP returns control to the sample user program DFH$PREN.
08:35:19.498080  E1 502059C8 0004 00022 0011C200 00001802 ..B.....          EIP RECEIVE-MAP ENTRY
                           Command 4 is started.
08:35:19.498112  FA 50290726 0003 00022 00020409 00000020 ........          BMS MAP WAIT IN
08:35:19.515168  FA 5028E0EC 0003 00022 00020409 00000020 ........          BMS MAP WAIT IN
                           EIP requests BMS to map in the data from the terminal.
                           The duplicate trace entry indicates that MCP has detected
                           that the BMS fast-path option is suitable on this occasion,
                           and calls MCX to process the request.
08:35:19.515232  F2 50291176 0404 00022 00000000 00000000 ........ DFH$PGK   PCP LOAD
08:35:19.515424  EA 402A6CE4 0003 00022 01000300 0011B214 ........ DFH$PGK   TMP PPT LOCATE
08:35:19.515648  EA 402C0E34 0005 00022 01000300 002A47DC ........          TMP RETN NORMAL
                           BMS ensures that the map is in storage; TMP returns its address.
08:35:19.515712  FC 50291456 0103 00022 00140000 0011F784 ......7.          ZCP ZARQ APPL REQ READ
                                                                               WAIT
                           BMS requests ZCP to read from the terminal,
                           giving the address of the TCTTE.
08:35:19.515808  FC 702DB570 1804 00022 00040001 0011F784 ......7.          ZCP ZSDS SEND
                           ZCP requests the services of VTAM to read the data.
08:35:19.530176  F0 702DEA9C 4004 00022 13000000 0011F784 ......7.          KCP WAIT DCI=TERMINAL
08:35:19.530208  D0 502F0838 0A04 00022 00000000 00000000 ........          KCP SUSPEND
                           Task 22 is suspended until I/O completion.
08:35:19.530272  D0 502F1E6A 0504 TCP   00000000 00000000 ........          KCP DISPATCH
08:35:19.530432  EE 702DD140 2214 TCP   00040126 0111F784 ......7. 16331624  VIO SEND OIC DATA RQE1
08:35:19.530464  EE 702DD140 0024 TCP   00BAF5C3 11C54E13 ..5C.E+. 16331624  VIO DATA
                           TCP traces the VTAM activity.
08:35:19.530496  FC 702DD3D4 1204 TCP   00200001 1911F784 ......7.          ZCP ZFRE FREEMAIN
08:35:19.530528  F1 602DD646 4104 TCP   0011A240 0111F784 ... ..7.          SCP FREEMAIN
08:35:19.530560  C9 502ABA6C 0004 TCP   0011A240 85110188 ... ....          SCP RELEASED TERMINAL
                                                                               STORAGE
                           TCP releases the TIOA used to send the map to the screen.
08:35:19.530592  FC 702DD3D4 1604 TCP   00030001 0011F784 ......7.          ZCP ZRVS RECEIVE SPECIFIC
                           ZCP sets up a RECEIVE SPECIFIC request parameter list for VTAM
08:35:19.530656  F1 602DA1C6 E304 TCP   000003E8 0011F784 ...Y..7.          SCP GETMAIN CONDITIONAL
                                                                               INITIMG
08:35:19.530688  C8 502ABA30 0004 TCP   0011A240 850003F8 ... ...8          SCP ACQUIRED TERMINAL
                                                                               STORAGE
                           TCP obtains storage for a new TIOA to hold the unprocessed
                           input from the terminal via VTAM.
                           Because task 22 was the only task running at the time,
                           SCP has allocated the same area from the storage subpool
                           reserved for terminals.
```

Figure 58 (Part 4 of 11). Trace table entries for sample transaction

```
   TIME OF DAY   ID  REG 14  REQD TASK   FIELD A  FIELD B  CHARS   RESOURCE  TRACE TYPE

08:35:19.532032  F0 402DE1E8 4004 TCP    40000000 002572E0  .......          KCP WAIT DCI=LIST
                            TCP checks for terminal activity.
                            There is none, so KCP relinquishes control by an
                            operating system WAIT macro.
                            After a certain elapsed time (ICV value), an operating system
                            interval timer interrupt causes control to return to KCP.
                            KCP passes control to TCP to scan for terminal activity.
                            This cycle continues until .............
08:35:26.745760  D0 502F1E6A 0504 TCP    00000000 00000000 ........          KCP DISPATCH
08:35:26.747552  F0 402DE1E8 4004 TCP    40000000 002572E0  .......          KCP WAIT DCI=LIST
08:35:27.502016  D0 502F1E6A 0904 KCP    00010000 D2251893 ....K...          KCP SYSTEM RESUME
                            Above three entries repeated several times.
08:35:27.503264  D0 502F1E6A 0504 TCP    00000000 00000000 ........          KCP DISPATCH
                            ....... The terminal control program detects incoming data from VTAM
08:35:27.503680  EE 702DD140 2314 TCP    000301A0 0111F784 ......7. 16331624  VIO RECEIVE OIC DATA
08:35:27.503712  EE 702DD140 0024 TCP    001311C5 4EF1F0F0 ...E+100 16331624  VIO DATA
                            The resource is the VTAM communications ID, consisting of
                            the origin address field and the destination address field.
                            This can be used to refer to a VTAM Trace.
08:35:27.503904  F0 602DD29A 0804 TCP    0011B190 0100017C ........          KCP RESUME
08:35:27.504032  F0 402DE1E8 4004 TCP    40000000 002572E0  .......          KCP WAIT DCI=LIST
08:35:27.504064  D0 502F1E6A 0504 00022  00000000 00000000 ........          KCP DISPATCH
08:35:27.504096  FC 402DE97C 0105 00022  00000000 0011A240 .......           ZCP RETN ZARQ APPL REQ
                            ZCP returns control to BMS.
08:35:27.504448  F1 6029149A C504 00022  00000058 0111F784 ......7.          SCP GETMAIN INITIMG
08:35:27.504704  C8 502ABA30 0004 00022  0011A640 85000068 ... ....          SCP ACQUIRED TERMINAL
                                                                               STORAGE
                            BMS makes a storage control request for a TIOA.
                            BMS processes the input via the mapset into this TIOA.
08:35:27.504800  F1 402918AC 4004 00022  0011A240 0111F784 ... ..7.          SCP FREEMAIN
08:35:27.504832  C9 502ABA6C 0004 00022  0011A240 850003F8 ... ...8          SCP RELEASED TERMINAL
                                                                               STORAGE
                            BMS can free the original TIOA containing the unprocessed input.
08:35:27.504864  FA 402918E8 0005 00022  00000000 00000000 ........          BMS RETN
08:35:27.505024  FA 4028E10A 0005 00022  00000000 00000000 ........          BMS RETN
                            BMS traces the end of request processing
                            and the end of fast-path processing.
08:35:27.505088  F1 502F39F0 4004 00022  0011A640 0111F784 ... ..7.          SCP FREEMAIN
08:35:27.505120  C9 502ABA6C 0004 00022  0011A640 85000068 ... ....          SCP RELEASED TERMINAL
                                                                               STORAGE
                            EIP copies the processed input map (from the TIOA just
                            created by BMS) into DFH$PREN's storage in the PL/I DSA,
                            and releases the TIOA.
08:35:27.505152  E1 502059C8 00F4 00022  00000000 00001802 ........          EIP RECEIVE-MAP
                                                                               RESPONSE
                            EIP traces completion of command 4.
08:35:27.505216  E1 50205B28 0004 00022  0011C200 00000602 ..B.....          EIP READ ENTRY
                            Command 6 in DFH$PREN is started.
08:35:27.505312  F5 402A1F7A F103 00022  00000000 00000000 ........ FILEA     FCP CTYPE LOCATE
                            EIP invokes the file control program to locate FILEA.
08:35:27.505376  EA 402A1092 0003 00022  01000500 0011B214 ........ FILEA     TMP FCT LOCATE
08:35:27.505632  EA 402C0E34 0005 00022  01000500 0029D63C ......O.          TMP RETN NORMAL
                            FCP invokes TMP to look up the FCT entry for FILEA.
08:35:27.505664  F5 4029EDF6 0025 00022  00000000 00000000 ........          FCP RETN NORMAL
                            FCP returns control to EIP.
08:35:27.505664  F1 602F39F0 CC04 00022  00000090 0111F784 ......7.          SCP GETMAIN INITIMG
08:35:27.505728  C8 502ABA30 0004 00022  0011C960 8C000098 ..I-....          SCP ACQUIRED USER
                                                                               STORAGE
                            EIP requests storage for a file control data-set table
                            for this task, and makes the entry for this data set
                            in the table.
08:35:27.505760  F1 602F39F0 CC04 00022  0011000C 0111F784 ......7.          SCP GETMAIN INITIMG
```

Figure 58 (Part 5 of 11). Trace table entries for sample transaction

```
   TIME OF DAY   ID  REG 14  REQD TASK   FIELD A  FIELD B  CHARS   RESOURCE  TRACE TYPE

08:35:27.505792  C8 502ABA30 0004 00022 0011CA00 8C110018 ........          SCP ACQUIRED USER
                                                                             STORAGE
                           EIP requests an area of storage to hold two copies of the
                           key of the record about to be read from FILEA.
08:35:27.505824  F5 502A22DC 8003 00022 00000000 00000000 ........ FILEA     FCP GET
                           EIP requests file control to perform a GET MOVE.
08:35:27.525472  F1 5029F5FC 8F04 00022 001100EC 0111F784 ......7.          SCP GETMAIN
08:35:27.525536  C8 502ABA30 0004 00022 0011CA20 8F1100F8 .......8          SCP ACQUIRED FILE
                                                                             STORAGE
                           FCP acquires a VSAM work area (VSWA) to hold the VSAM RPL and
                           CICS request storage.
08:35:27.525568  F1 402A0086 D304 00022 00000028 0111F784 ......7.          SCP GETMAIN INITIMG
08:35:27.525600  C8 502ABA30 0004 00022 001174C0 93000030 ........          SCP ACQUIRED SHARED
                                                                             STORAGE
                           FCP acquires a deferred work element (DWE).
08:35:27.525632  F1 6029F8D6 8F04 00022 00110058 0111F784 ......7.          SCP GETMAIN
08:35:27.525664  C8 502ABA30 0004 00022 0011CB20 8F110068 ........          SCP ACQUIRED FILE
                                                                             STORAGE
                           FCP acquires a file work area (FWA) to hold the record after
                           its retrieval by VSAM.
08:35:27.607680  F0 4029DC58 4004 00022 43000000 0011CA30 ........          KCP WAIT DCI=IOEVENT
08:35:27.639456  D0 502F1E6A 0904 KCP   00010001 D2251966 ....K...          KCP SYSTEM RESUME
08:35:27.640320  D0 502F1E6A 0504 00022 00000000 00000000 ........          KCP DISPATCH
08:35:27.645312  F0 4029DC58 4004 00022 43000000 0011CA30 ........          KCP WAIT DCI=IOEVENT
08:35:27.688928  D0 502F1E6A 0904 KCP   00010001 D225196E ....K...          KCP SYSTEM RESUME
08:35:27.689504  D0 502F1E6A 0504 00022 00000000 00000000 ........          KCP DISPATCH
                           The I/O is complete; control returns to FCP.
08:35:27.689952  F1 4029F900 4004 00022 0011CA20 0111F784 ......7.          SCP FREEMAIN
                           FCP releases the VSWA used for the RPL.
08:35:27.690016  C9 502ABA6C 0004 00022 0011CA20 8F1100F8 .......8          SCP RELEASED FILE
                                                                             STORAGE
08:35:27.690048  F5 4029EDF6 0015 00022 00000000 40840000 .... ...          FCP RETN NORMAL
                           FCP returns to EIP on successful completion of the GET request.
                           Fields A and B contain the VSAM RPL return code information.
08:35:27.690112  F5 402A23D8 1003 00022 1011CB20 00000000 ........          FCP RELEASE
                           EIP copies the record from the VSWA into DFH$PREN's DSA,
                           and calls FCP again to release the VSWA.
08:35:27.690144  F1 7029F900 4004 00022 0011CB20 0111F784 ......7.          SCP FREEMAIN
08:35:27.690176  C9 502ABA6C 0004 00022 0011CB20 8F110068 ........          SCP RELEASED FILE
                                                                             STORAGE
                           FCP requests storage control to release the storage.
08:35:27.690176  F5 4029EDF6 0005 00022 00000000 00000000 ........          FCP RETN NORMAL
                           FCP returns control to EIP.
08:35:27.690208  E1 50205B28 00F4 00022 00000000 00000602 ........          EIP READ RESPONSE
                           EIP traces completion of command 6.
08:35:27.692992  E1 50205B98 0004 00022 0011C200 00000802 ..B.....          EIP WRITEQ-TD ENTRY
                           Command 7 in DFH$PREN is started.
08:35:27.724000  F6 4029844A E103 00022 00000000 00000000 ........ L860      TDP APPL REQ CTYPE LOCATE
                           EIP passes control to the transient data program.
08:35:27.724224  EA 40294E64 0003 00022 01000600 0011B504 ........ L860      TMP DCT LOCATE
08:35:27.724544  EA 402C0E34 0005 00022 01000600 00294B48 ........          TMP RETN NORMAL
                           TDP invokes TMP to check the destination control table (DCT)
                           entry for L860.
08:35:27.724576  F6 402950D8 0005 00022 00000000 00000000 ........          TDP RETN APPL RESP NORMAL
08:35:27.724640  F1 602F39F0 CD04 00022 0000001A 0111F784 ......7.          SCP GETMAIN INITIMG
08:35:27.724704  C8 502ABA30 0004 00022 0011CA20 8D000028 ........          SCP ACQUIRED TRANSDATA
                                                                             STORAGE
                           EIP asks storage control for an area large enough
                           to hold a copy of the record to be written to the transient
                           data queue L860.
```

Figure 58 (Part 6 of 11). Trace table entries for sample transaction

```
   TIME OF DAY    ID  REG 14  REQD TASK   FIELD A  FIELD B  CHARS   RESOURCE  TRACE TYPE

08:35:27.724736  F6 502984BC 4803 00022 00294B48 0011CA28 ........ L860      TDP APPL REQ PUT ADDR=YES
                              EIP invokes TDP to write the data to queue L860.
08:35:27.770976  F0 50297424 C004 00022 20000000 0011CA30 ........           KCP WAIT+CANCADDR DCI=DISP
08:35:27.771040  D0 502F1E6A 0504 TCP   00000000 00000000 ........           KCP DISPATCH
08:35:27.771392  F0 402DE1E8 4004 TCP   40000000 002572E0  .......           KCP WAIT DCI=LIST
08:35:27.771424  D0 502F1E6A 0504 00022 00000000 00000000 ........           KCP DISPATCH
08:35:27.771456  F6 402950D8 0005 00022 00000000 00000000 ........           TDP RETN APPL RESP NORMAL
                              TDP traces completion of the I/O event, and returns control to EIP.
08:35:27.771488  F1 502F39F0 4004 00022 0011CA20 0111F784 ......7.           SCP FREEMAIN
08:35:27.771520  C9 502ABA6C 0004 00022 0011CA20 8D000028 ........           SCP RELEASED TRANSDATA
                                                                                STORAGE
                              EIP releases the storage acquired for the copied transient data record.
08:35:27.771552  E1 50205B98 00F4 00022 00000000 00000802 ........           EIP WRITEQ-TD RESPONSE
                              EIP traces completion of command 7.
08:35:27.771584  E1 50205BCC 0004 00022 0011C200 00001804 ..B.....           EIP SEND-MAP ENTRY
                              Command 8 in program DFH$PREN is started.
08:35:27.771840  FA 50290726 0003 00022 00000460 24000020 ...-....           BMS ERASEAUP SEND-OUT
                                                                                MAP SAVE
08:35:27.771872  FA 7028E0EC 0003 00022 00000460 24000020 ...-....           BMS ERASEAUP SEND-OUT
                                                                                MAP SAVE
                              Once again, MCP detects that the fast-path option is suitable,
                              and calls MCX to execute the request.
08:35:27.771936  F1 40291A4A 8504 00022 00110163 0111F784 ......7.           SCP GETMAIN
08:35:27.771968  C8 502ABA30 0004 00022 0011A240 85110178 ... ....           SCP ACQUIRED TERMINAL
                                                                                STORAGE
                              BMS obtains storage for a TIOA, which is used to write
                              the map to the screen.
08:35:27.772224  FC 502920E8 EE03 00022 00000017 0011F784 ......7.           ZCP EXIT TRACE ZRVX
08:35:27.772224  FC 502920E8 0103 00022 00010000 0011F784 ......7.           ZCP ZARQ APPL REQ WRITE
08:35:27.772256  FC 402DE97C 0105 00022 00010400 0011A240 .......           ZCP RETN ZARQ APPL REQ
                                                                                WRITE DEFER
                              ZCP executes the request on behalf of BMS.
08:35:27.772288  FA 402918E8 0005 00022 00000000 00000000 ........           BMS RETN
08:35:27.772320  FA 4028E10A 0005 00022 00000000 00000000 ........           BMS RETN
                              BMS traces successful completion of command 8 and returns
                              control to EIP.
                              Note the duplicate RETN trace entry indicates that the BMS fast-path
                              option was invoked.
08:35:27.772320  E1 50205BCC 00F4 00022 00000000 00001804 ........           EIP SEND-MAP RESPONSE
                              EIP traces completion of command 8 and returns control
                              to the user code in DFH$PREN.
08:35:27.772352  E1 502059C8 0004 00022 0011C200 00001802 ..B.....           EIP RECEIVE-MAP ENTRY
                              Command 4, at the label RECEIVE in program DFH$PREN, is started.
08:35:27.772384  FA 50290726 0003 00022 00020409 00000020 ........           BMS MAP WAIT IN
08:35:27.772416  FA 5028E0EC 0003 00022 00020409 00000020 ........           BMS MAP WAIT IN
                              EIP requests BMS to read incoming data from the terminal.
08:35:27.772448  FC 50291456 0103 00022 00140000 0011F784 ......7.           ZCP ZARQ APPL REQ
                                                                                READ WAIT
08:35:27.772512  FC 702DB570 1804 00022 00050001 0011F784 ......7.           ZCP ZSDS SEND
                              BMS requests terminal control (ZCP) to perform terminal I/O.
08:35:27.805824  F0 702DEA9C 4004 00022 13000000 0011F784 ......7.           KCP WAIT DCI=TERMINAL
08:35:27.805856  D0 502F0838 0A04 00022 00000000 00000000 ........           KCP SUSPEND
                              Task 22 is suspended, pending I/O completion.
08:35:27.805920  D0 502F1E6A 0504 TCP   00000000 00000000 ........           KCP DISPATCH
08:35:27.806016  EE 702DD140 2214 TCP   00050126 0111F784 ......7. 16331624  VIO SEND OIC DATA RQE1
08:35:27.806048  EE 702DD140 0024 TCP   00C0F1C3 11404012 ..1C.  . 16331624  VIO DATA
08:35:27.806080  FC 702DD3D4 1204 TCP   00200001 1911F784 ......7.           ZCP ZFRE FREEMAIN
08:35:27.806080  F1 602DD646 4104 TCP   0011A240 0111F784 ... ..7.           SCP FREEMAIN
08:35:27.806112  C9 502ABA6C 0004 TCP   0011A240 85110178 ... ....           SCP RELEASED TERMINAL
                                                                                STORAGE
                              The TIOA acquired to send the map is released.
08:35:27.806176  FC 702DD3D4 1604 TCP   00030001 0011F784 ......7.           ZCP ZRVS RECEIVE SPECIFIC
08:35:27.806208  F1 602DA1C6 E304 TCP   000003E8 0011F784 ...Y..7.           SCP GETMAIN CONDITIONAL
                                                                                INITIMG
```

Figure 58 (Part 7 of 11). Trace table entries for sample transaction

```
   TIME OF DAY   ID  REG 14  REQD TASK   FIELD A  FIELD B  CHARS   RESOURCE  TRACE TYPE

08:35:27.806240  C8 502ABA30 0004 TCP    0011A240 850003F8 ... ...8          SCP ACQUIRED TERMINAL
                                                                               STORAGE
                           ZCP sets up a RECEIVE SPECIFIC RPL for VTAM, and acquires
                           storage for a TIOA in which to hold the data expected
                           from the terminal.
08:35:27.811264  F0 402DE1E8 4004 TCP    40000000 002572E0 .......          KCP WAIT DCI=LIST
                           TCP checks for terminal activity.
                           There is none, so KCP relinquishes control by an
                           operating system WAIT macro.
                           After a certain elapsed time (ICV value), an operating system
                           interval timer interrupt causes control to return to KCP.
                           KCP passes control to TCP to scan for terminal activity.
                           This cycle continues until .............
08:35:27.915328  D0 502F1E6A 0904 KCP    40010000 D2251997  ...K...          KCP SYSTEM RESUME
08:35:27.916256  D0 502F1E6A 0504 TCP    00000000 00000000 ........          KCP DISPATCH
08:35:27.917792  F0 402DE1E8 4004 TCP    40000000 002572E0  .......          KCP WAIT DCI=LIST
08:35:28.931840  D0 502F1E6A 0904 KCP    40010000 D22519B1  ...K...          KCP SYSTEM RESUME
08:35:28.932768  D0 502F1E6A 0504 TCP    00000000 00000000 ........          KCP DISPATCH
08:35:28.934336  F0 402DE1E8 4004 TCP    40000000 002572E0  .......          KCP WAIT DCI=LIST
08:35:29.961920  D0 502F1E6A 0904 KCP    40010000 D2251AA9  ...K...          KCP SYSTEM RESUME
08:35:29.962848  D0 502F1E6A 0504 TCP    00000000 00000000 ........          KCP DISPATCH
08:35:29.964512  F0 402DE1E8 4004 TCP    40000000 002572E0  .......          KCP WAIT DCI=LIST
08:35:30.007552  D0 502F1E6A 0904 KCP    00010000 D2251BA5 ....K...          KCP SYSTEM RESUME
08:35:30.008640  D0 502F1E6A 0504 TCP    00000000 00000000 ........          KCP DISPATCH
                           ........ the terminal control program detects incoming data from VTAM
08:35:30.009056  EE 702DD140 2314 TCP    000401A0 0111F784 ......7. 16331624  VIO RECEIVE OIC DATA
08:35:30.009088  EE 702DD140 0024 TCP    00016D00 00000000 ........ 16331624  VIO DATA
                           The resource is the VTAM communications ID, consisting of
                           the origin address field and the destination address field.
                           This can be used to refer to a VTAM trace.
08:35:30.009280  F0 602DD29A 0804 TCP    0011B190 0100017C ........          KCP RESUME
08:35:30.009376  F0 402DE1E8 4004 TCP    40000000 002572E0  .......          KCP WAIT DCI=LIST
08:35:30.009408  D0 502F1E6A 0504 00022  00000000 00000000 ........          KCP DISPATCH
08:35:30.009440  FC 402DE97C 0105 00022  00000000 0011A240 .......           ZCP RETN ZARQ APPL REQ
                           ZCP returns control to BMS.
08:35:30.009664  FA 402918E8 0405 00022  04000000 00000000 ........          BMS RETN CANNOT MAP I/O
                                                                               AREA
08:35:30.009824  FA 4028E10A 0405 00022  00000000 00000000 ........          BMS RETN
                           BMS detects that there is no incoming data
                           (because the terminal operator pressed CLEAR),
                           and returns control to EIP.
08:35:30.010080  F1 502F39F0 4004 00022  0011A240 0111F784 ... ..7.          SCP FREEMAIN
08:35:30.010304  C9 502ABA6C 0004 00022  0011A240 850003F8 ... ...8          SCP RELEASED TERMINAL
                                                                               STORAGE
                           EIP copies the incoming data (in this case there is none)
                           into DFH$PREN's automatic storage and releases the TIOA.
08:35:30.010400  E1 0011BE10 00F4 00022  04000000 00001802 ........          EIP RECEIVE-MAP RESPONSE
                           EIP recognizes the CLEAR response from the terminal operator, and passes
                           control to the HANDLE AID CLEAR label specified in command 1.
08:35:30.010592  E1 50205D9A 0004 00022  0011C200 00001806 ..B.....          EIP SEND-TEXT ENTRY
                           Command 13 in DFH$PREN is started.
08:35:30.010624  F1 602F39F0 CC04 00022  00000018 0111F784 ......7.          SCP GETMAIN INITIMG
08:35:30.010656  C8 502ABA30 0004 00022  0011CA20 8C000028 ........          SCP ACQUIRED USER STORAGE
                           BMS obtains storage for a direct terminal type parameter (TTP),
                           and places its address in the OSPWA.
                           A TTP is obtained for each device type to receive this message.
08:35:30.010720  FA 50290726 0003 00022  00000002 04000020 ........          BMS SEND-OUT ERASE
08:35:30.010816  F1 4028A05A DE04 00022  00000140 0111F784 ... ..7.          SCP GETMAIN INITIMG
08:35:30.010848  C8 502ABA30 0004 00022  0011CA50 9E000148 ...&....          SCP ACQUIRED MAPCOPY
                                                                               STORAGE
                           BMS obtains storage for a MAPCOPY to hold the text to be sent
                           to the terminal.
```

Figure 58 (Part 8 of 11). Trace table entries for sample transaction

```
   TIME OF DAY    ID  REG 14  REQD TASK   FIELD A  FIELD B  CHARS   RESOURCE  TRACE TYPE

08:35:30.010912  F1 702924E6 8504 00022 00110035 0111F784 ......7.           SCP GETMAIN
08:35:30.010944  C8 502ABA30 0004 00022 0011A240 85110048 ... ....           SCP ACQUIRED TERMINAL
                                                                               STORAGE
                            BMS obtains storage for a TIOA, which is used to write
                            the map to the screen.
08:35:30.010976  F1 502925A8 CC04 00022 00000074 0111F784 ......7.           SCP GETMAIN INITIMG
08:35:30.011008  C8 502ABA30 0004 00022 0011CBA0 8C000088 ........           SCP ACQUIRED USER STORAGE
                            BMS obtains a merged data area to continue building the TIOA.
08:35:30.011072  F1 60292EFE 4004 00022 0011CA50 0111F784 ...&..7.           SCP FREEMAIN
08:35:30.011072  C9 502ABA6C 0004 00022 0011CA50 9E000148 ...&....           SCP RELEASED MAPCOPY
                                                                               STORAGE
08:35:30.011104  F1 60292F58 4004 00022 0011CBA0 0111F784 ......7.           SCP FREEMAIN
08:35:30.011136  C9 502ABA6C 0004 00022 0011CBA0 8C000088 ........           SCP RELEASED USER STORAGE
                            The TIOA is complete, so BMS releases its work areas.
08:35:30.011168  FC 4028DD4E EE03 00022 00000017 0011F784 ......7.           ZCP EXIT TRACE ZRVX
                            BMS requests ZCP to write the map from the TIOA.
                            The I/O request to the access method may
                            be deferred until ZCP is called for a later request,
                            to allow SNA indicators to be sent together with
                            message traffic.
08:35:30.011200  FC 4028DD4E 0103 00022 00810000 0011F784 ......7.           ZCP ZARQ APPL REQ ERASE
                                                                               WRITE
08:35:30.011232  FC 402DE97C 0105 00022 00810400 0011A240 .......           ZCP RETN ZARQ APPL REQ
                                                                               ERASE WRITE DEFER
                            ZCP returns control to BMS, indicating that the I/O has
                            been deferred on this occasion.
08:35:30.011264  F1 4028EEA6 4004 00022 0011CA20 0111F784 ......7.           SCP FREEMAIN
08:35:30.011296  C9 502ABA6C 0004 00022 0011CA20 8C000028 ........           SCP RELEASED USER STORAGE
08:35:30.011328  FA 4028EF3E 0005 00022 00000000 00000000 ........           BMS RETN
                            BMS releases the TTP, and returns control to EIP.
08:35:30.011328  E1 50205D9A 00F4 00022 00000000 00001806 ........           EIP SEND-TEXT RESPONSE
                            EIP traces completion of command 13.
08:35:30.011360  E1 50205DC4 0004 00022 0011C200 00001812 ..B.....           EIP SEND-CONTROL ENTRY
                            Command 14 in DFH$PREN is started.
08:35:30.011392  FC 502907C2 0103 00022 00044000 0011F784 .. ...7.           ZCP ZARQ APPL REQ WAIT
08:35:30.011456  FC 702DB570 1804 00022 00060001 0011F784 ......7.           ZCP ZSDS SEND
08:35:30.022208  F0 702DEA9C 4004 00022 13000000 0011F784 ......7.           KCP WAIT DCI=TERMINAL
08:35:30.022240  D0 502F0838 0A04 00022 00000000 00000000 ........           KCP SUSPEND
                            Task control suspends task 22, pending I/O completion.
08:35:30.022336  D0 502F1E6A 0504 TCP    00000000 00000000 ........           KCP DISPATCH
08:35:30.023008  EE 702DD140 2214 TCP    00060126 0111F784 ......7. 16331624  VIO SEND OIC DATA RQE1
08:35:30.023040  EE 702DD140 0024 TCP    001FF5C3 1140C113 ..5C. A. 16331624  VIO DATA
08:35:30.023072  FC 702DD3D4 1204 TCP    00200001 1911F784 ......7.           ZCP ZFRE FREEMAIN
08:35:30.023104  F1 602DD646 4104 TCP    0011A240 0111F784 ... ..7.           SCP FREEMAIN
08:35:30.023136  C9 502ABA6C 0004 TCP    0011A240 85110048 ... ....           SCP RELEASED TERMINAL
                                                                               STORAGE
                            BMS releases the TIOA.
08:35:30.023168  F0 602DD29A 0804 TCP    0011B190 0100017C ........           KCP RESUME
08:35:30.023264  F0 402DE1E8 4004 TCP    40000000 002572E0  ......           KCP WAIT DCI=LIST
08:35:30.023296  D0 502F1E6A 0504 00022 00000000 00000000 ........           KCP DISPATCH
08:35:30.023328  FC 402DE97C 0105 00022 00000000 00000000 ........           ZCP RETN ZARQ APPL REQ
08:35:30.023360  FA 50290726 0003 00022 00000840 04000021 ... ....           BMS SEND-OUT CTRL CONTROL
08:35:30.023552  FA 7028E0EC 0003 00022 00000840 04000021 ... ....           BMS SEND-OUT CTRL CONTROL
                            Once again, MCP detects that the fast-path option is suitable,
                            and calls MCX to execute the request.
08:35:30.023584  F1 40291A4A 8504 00022 00110009 0111F784 ......7.           SCP GETMAIN
08:35:30.023616  C8 502ABA30 0004 00022 0011A240 85110018 ... ....           SCP ACQUIRED TERMINAL
                                                                               STORAGE
                            BMS requests storage for a TIOA.
```

Figure 58 (Part 9 of 11). Trace table entries for sample transaction

```
   TIME OF DAY   ID  REG 14  REQD TASK   FIELD A  FIELD B  CHARS   RESOURCE  TRACE TYPE

08:35:30.023680  FC 502920E8 0103 00022 00010000 0011F784 ......7.           ZCP ZARQ APPL REQ WRITE
08:35:30.023712  FC 402DE97C 0105 00022 00010400 0011A240 .......            ZCP RETN ZARQ APPL REQ
                                                                               WRITE DEFER
08:35:30.023712  FA 402918E8 0005 00022 00000000 00000000 ........           BMS RETN
08:35:30.023744  FA 4028E10A 0005 00022 00000000 00000000 ........           BMS RETN
                           BMS returns to EIP.
08:35:30.023776  E1 50205DC4 00F4 00022 00000000 00001812 ........           EIP SEND-CONTROL RESPONSI
                           EIP traces completion of command 14
08:35:30.023808  E1 50205DEE 0004 00022 0011C200 00000E08 ..B.....           EIP RETURN ENTRY
                           Command 15 in DFH$PREN is started. This is where
                           the user code returns control to CICS.
08:35:30.023936  F1 502F39F0 4004 00022 0011C490 0111F784 ..D...7.           SCP FREEMAIN
08:35:30.023936  C9 502ABA6C 0004 00022 0011C490 8C0000C8 ..D....H           SCP RELEASED USER
                                                                               STORAGE
                           The HANDLE CONDITION label table storage area is released.
08:35:30.023968  F1 502F39F0 4004 00022 0011C400 0111F784 ..D...7.           SCP FREEMAIN
08:35:30.024000  C9 502ABA6C 0004 00022 0011C400 8C000088 ..D.....           SCP RELEASED USER
                                                                               STORAGE
                           The HANDLE AID label table storage area is released.
08:35:30.024032  F2 402F3CC6 1004 00022 00000000 00000000 ........ DFH$PREN  PCP RETURN
08:35:30.024064  F1 602A7040 4004 00022 0011BD80 0111F784 ......7.           SCP FREEMAIN
08:35:30.024096  C9 502ABA6C 0004 00022 0011BD80 8C000678 ........           SCP RELEASED USER
                                                                               STORAGE
                           EIP informs program control of the application program's RETURN.
08:35:30.024128  F0 402A6B56 8004 00022 00000000 00000000 ........           KCP DETACH
                           PCP makes a request to task control to detach task 22.
08:35:30.024192  D8 402EFFF8 0203 00022 02000000 001174C0 ........           SPP SYSTEM
                           The sync point program is invoked with the address of the
                           first deferred work element (DWE).
08:35:30.024288  FC 502E97DA 0B03 00022 03000000 0011F784 ......7.           ZCP ZISP ISC FREE
                           Task control calls ZCP to release the terminal.
08:35:30.024352  FC 402E390A 0304 00022 00200001 0011F784 ......7.           ZCP ZDET DETACH
08:35:30.024384  FC 702DB570 1804 00022 00070001 0011F784 ......7.           ZCP ZSDS SEND
                           ZCP indicates to VTAM that the primary facility
                           attached to task 22 can now be released.
08:35:30.049376  EE 702D7364 2214 00022 00070126 0111F784 ......7. 16331624  VIO SEND OIC DATA RQE1
08:35:30.049440  EE 702D7364 0024 00022 0002F1C2 1140C113 ..1B. A. 16331624  VIO DATA
                           ZCP traces the transmission of the request to VTAM, and
                           returns control to KCP.
08:35:30.049440  F1 502D74C0 6104 00022 00000000 00000000 ........           SCP FREEMAIN ALL
08:35:30.049472  C9 502ABA6C 0004 00022 0011A240 85110018 ... ....           SCP RELEASED TERMINAL
                                                                               STORAGE
                           The TIOA is released.
08:35:30.057504  FC 402D7518 1504 00022 00200001 1911F784 ......7.           ZCP ZRST RESETSR
08:35:30.067232  FC 402E3AC4 0B05 00022 03000000 0011F784 ......7.           ZCP RETN ZISP ISC FREE
08:35:30.067264  D8 402EB3E4 0005 00022 00000000 00000000 ........           SPP RETN
                           The sync point program traces completion of its processing.
08:35:30.067424  F0 502F0002 0304 00022 0011F784 00000000 ..7.....           KCP DEQALL
                           Task control issues a DEQUEUE ALL to release any ENQUEUEs
                           issued by the user code which have not yet been released.
08:35:30.067488  D0 502F0838 0704 00022 08000400 D7D6D9C4 ....PORD           KCP TERMINATE
08:35:30.067520  F1 402F01BA 4A04 KCP   0011B000 00000000 ........           SCP FREEMAIN
                           Storage control is invoked by task control to release
                           all storage areas chained off the TCA for Task 22.
08:35:30.067552  C9 502ABA6C 0004 KCP   0011CA00 8C110018 ........           SCP RELEASED USER STORAGI
                           The storage acquired to hold two copies of the record key (RIDFLD)
                           of the FILEA record retrieved by DFH$PREN is released.
```

Figure 58 (Part 10 of 11). Trace table entries for sample transaction

```
   TIME OF DAY   ID  REG 14  REQD TASK   FIELD A  FIELD B  CHARS   RESOURCE  TRACE TYPE

08:35:30.067584  C9 502ABA6C 0004 KCP    0011C960 8C000098 ..I-....          SCP RELEASED USER STORAGE
                             The file control data-set table is released.
08:35:30.067616  C9 502ABA6C 0004 KCP    0011C880 8C0000D8 ..H....Q          SCP RELEASED USER STORAGE
                             The TTP acquired by BMS is released.
08:35:30.067648  C9 502ABA6C 0004 KCP    0011C560 8C000318 ..E-....          SCP RELEASED USER STORAGE
                             The OSPWA acquired by BMS is released.
08:35:30.074144  C9 502ABA6C 0004 KCP    0011BB20 8C000258 ........          SCP RELEASED USER STORAGE
                             The PL/I automatic storage area is released.
08:35:30.074240  C9 502ABA6C 0004 KCP    0011B000 8A040B18 ........          SCP RELEASED TCA STORAGE
                             Finally, the TCA itself is released.
                             All resources acquired by task 22 have now been returned
                             to CICS, and the task has disappeared from the system.
08:35:30.081792  D0 502F1E6A 0504 TCP    00000000 00000000 ........              KCP DISPATCH
08:35:30.082336  FC 702DD3D4 1204 TCP    00000001 0011F784 ......7.              ZCP ZFRE FREEMAIN
08:35:30.082560  F0 402DE1E8 4004 TCP    40000000 002572E0  .......              KCP WAIT DCI=LIST
                             TCP is given control to check for terminal activity.
                             There is none, so KCP relinquishes control by an
                             operating system WAIT macro.
                             After a certain elapsed time (ICV value), an operating system
                             interval timer interrupt causes control to return to KCP.
                             KCP passes control to TCP to scan for terminal activity.
                             This cycle continues until a new task is created.
08:35:30.148384  D0 502F1E6A 0904 KCP    40010000 D2251BC2  ...K..B          KCP SYSTEM RESUME
08:35:30.149664  D0 502F1E6A 0504 TCP    00000000 00000000 ........              KCP DISPATCH
08:35:30.151168  F0 402DE1E8 4004 TCP    40000000 002572E0  .......              KCP WAIT DCI=LIST
08:35:31.201248  D0 502F1E6A 0904 KCP    40010000 D2251BD2  ...K..K          KCP SYSTEM RESUME
08:35:31.202912  D0 502F1E6A 0504 TCP    00000000 00000000 ........              KCP DISPATCH
```

Figure 58 (Part 11 of 11). Trace table entries for sample transaction

CICS termination

CICS can terminate in a controlled, or an uncontrolled manner. During a controlled shutdown, CICS participates in the termination, and performs some housekeeping before returning to the operating system. In an uncontrolled shutdown, CICS might not be as involved and so has less opportunity to tidy up.

A controlled shutdown results from the use of the master terminal CEMT PERFORM SHUTDOWN command (without the IMMEDIATE option).

An uncontrolled (or immediate) shutdown may result from:

- Use of the master terminal CEMT PERFORM SHUTDOWN IMMEDIATE command.
- A machine failure.
- An operating system failure.
- An abnormal termination in which an ESTAE exit is given control and decides that CICS should not continue execution.
- A program check in a system task or while recovering from a previous program check.
- A program check or system abend occurring while DFHSRP is processing an abnormal termination.

After a controlled shutdown, it is possible to warm start CICS. After an uncontrolled (or immediate) shutdown, an emergency restart or a cold start must be performed.

If the TAKEOVER option is specified, then the active CICS system signs off abnormally from the CICS availability manager (CAVM) before returning control to the operating system. The alternate CICS system, if running, then takes over and completes an emergency restart, whether the shutdown was controlled or uncontrolled.

Abnormal termination

An abnormal termination of CICS may occur if an application program destroys part of the CICS nucleus or key system information maintained in dynamic storage. This is generally the result of an application program error, and may cause an ABEND of the CICS address space. To reduce the impact of such an abend, the CICS system recovery program (DFHSRP) issues an ESTAE macro instruction during initialization to regain control in the event of a CICS ABEND. If an abnormal termination subsequently occurs, DFHSRP attempts either to correct or to circumvent the problem to continue operation (see "The system recovery program (DFHSRP)" on page 275). If recovery is not possible or advisable, DFHSRP shuts CICS down immediately.

If DFHSRP is unable to correct or circumvent the problem causing the abnormal termination, and is further unable to initiate a controlled shutdown (such as in the event of destruction of part of the controlled shutdown routines), an uncontrolled shutdown will occur.

Program check

Any program check will cause an uncontrolled shutdown of CICS if SRT = NO is specified in the system initialization table (SIT). An emergency restart or a cold start must then be performed. When using an SRT, CICS can recover from a program check, possibly abending a task, unless it is a program check while processing an abend, in a system task or during recovery from a previous program check. If it is a program check while processing an abend, an uncontrolled shutdown occurs. Otherwise a controlled shutdown is started.

CEMT shutdown command used

Two types of operator-initiated shutdown are available. An immediate shutdown does not wait for existing tasks to finish. A controlled shutdown waits for existing tasks to finish and allows users to participate in the termination process using the XLT and PLT tables. Descriptions of the table functions are given below.

Restart data set

A controlled shutdown preserves certain vital information about the CICS environment during normal termination of CICS. This information, recorded on the restart data set, may be subsequently used to warm start CICS and reinitialize it to its status at termination. The user may optionally warm start only certain parts of CICS, and allow a cold start (complete reinitialization) of other parts of CICS. The restart data set controls the resolution of **automatic** restarts. The control record holds information concerning whether the last shutdown was controlled or not. The restart data set consists of two logical files:

- The recovery file, which holds relevant recovery information collected from the system log during emergency restart.
- The catalog file, which holds information for each resource manager.

Two CICS tables are used for controlled shutdown and warm start:

- The transaction list table (XLT).
- The program list table (PLT).

Transaction list table (XLT)

During normal CICS termination at the end of a CICS operational period, a transaction list table (XLT) may be loaded. This transaction list table identifies a list of transaction codes accepted by CICS during termination. All other transaction codes will be rejected by CICS except CEMT, CSAC, CSIR, CLS1, CLS2, CSMT, CSNE, CSSF, and CSTE, which do not need to be named in a shutdown XLT, and ATI tasks initiated during phase 1 of termination. The acceptance of transactions can therefore be limited to those transactions that need to be completed during system termination.

Program list table (PLT)

The program list table (PLT) is a list of programs to be used in the following ways:

- Identify various user-written programs that are to be executed during the postinitialization phase of CICS system initialization. These user programs may locate the application-dependent information written by PLT-identified programs during a previous CICS controlled shutdown, and use that information to reestablish the online applications as required by the user.

- Identify various user-written programs that are to be executed during either the first or second stage of CICS operator-initiated controlled shutdown (see Figure 59 on page 257). These user programs may record application-dependent information that will permit user recovery of that information on subsequent system initialization.

- Specify a list of programs that can be enabled or disabled by a master terminal command.

Programs that are defined in the postinitialization PLT execute under the control of the terminal control TCA. This TCA is assembled as part of the module DFHCSA. A PLT program should issue a DFHSC TYPE=GETMAIN to obtain any working storage it requires.

Termination PLT programs execute under the system termination program's TCA, which has a TWA for a CEMT (or CSMT) transaction. This TWA should not be used by the PLT program.

A normal controlled shutdown causes the status of various areas to be written to the restart data set to permit a warm start to take place. It uses the program list table (PLT), as described above, and carries out termination in two stages: the **first quiesce stage** and the **second quiesce stage**. During the first quiesce stage, terminals are still active, but if a transaction list table (XLT) has been specified, only transactions that are defined in that table, or the CICS transactions CEMT, CSAC, CSIR, CLS1, CLS2, CSMT, CSNE, CSSE, or CSTE are permitted. Programs defined in the first section of the PLT are also executed. During the second quiesce stage, terminals are deactivated and programs defined in the second section of the PLT are executed. Figure 59 on page 257 describes the flow of these two stages of termination.

MODULE	ACTION	NOTE/MSG
DFHSTP	Determine if controlled or immediate shutdown. If controlled, perform steps 1 through 22 below; otherwise perform 2, and 15 through 22 only.	DFH1700 - IMMEDIATE SHUTDOWN?
	STAGE 1	
	1. Set **system-termination** flag	X'80' in CSASSI2
	2. Issue **CICS-is-being-terminated**	
	3. Quiesce IRC session	
	4. LOAD XLT, PLT if required	
	5. Prevent TCP starting new transactions if they are not in the XLT and not CEMT, CSAC, CSIR, CLS1, CLS2, CSMT, CSNE, CSSE, CSTE, except ATI tasks, which can be started	X'04' in CSASSI2
	6. Change priority to 0	
	7. Establish PLT abend exit	Use DFHPC TYPE=SETXIT
	8. Link to PLT stage 1 programs The shutdown transaction disconnects from the initiating terminal, allowing the terminal to use CEMT, and so on, to monitor the termination.	
	9. Cancel PLT abend exit	
	10. Stop TCP from accepting inputs	X'08' in CSASSI2
	11. Wait for TCP to serve last output	Wait on TCP ECB
	STAGE 2	
	12. Reestablish PLT abend exit	
	13. Execute PLT stage 2 programs	
	14. Cancel PLT abend exit	
	15. Shut down DL/I	CALL DFHDLI('*QDL')
DFHJCSDJ	16. Shut down journal control	
	17. Call DFHDLI('*TDS') to shut down DBRC and IRLM.	
DFHSTP	18. Flush temporary storage buffers	DFHTS TYPE=FLUSH
DFHSTKC	19. Take statistics	
DFHSTP	20. Cancel SPIE, ESTAE exits	
	21. Take a warm keypoint	
	22. Abnormally close IRC session if still open	
	23. Return to operating system	

Figure 59. CICS termination modules – sequence of execution

Chapter 3.2. Storage management

General storage

Storage management in a CICS system is handled by the storage control program (DFHSCP) and by the storage recovery program (DFHSCR). They control the dynamic storage area of the CICS address space. The size of this area is determined and space is allocated for it by program DFHSIH1 during the initialization of the CICS system. This area is used to satisfy all CICS requests for space, including space for all dynamically acquired control blocks and space for all nonresident application programs, but not including requests of more than four kilobytes that use the FLENGTH parameter under MVS/XA, or main temporary storage of any size. CICS requests for space are made by both CICS system programs and user programs by means of the DFHSC macro. Operating system GETMAIN requests will not receive storage from the CICS dynamic storage area. The acquisition of storage for operating system requests is described under "Storage for use by the operating system" on page 266.

Subpools and the page allocation map (PAM)

The dynamic storage area consists of a number of pages whose size is specified by the PGSIZE operand of the DFHSIT macro and which, for best results, should be equal to the size of the operating system's virtual storage pages. In order to reduce the paging overhead, an attempt is made to satisfy requests for storage for similar types of use from the same pages. To do this, CICS maintains several subpools. At any time, a page either will be not in use (and therefore not allocated to a subpool) or will have been used to satisfy a request for a certain class of storage and will be allocated to the appropriate subpool.

When a page is allocated to a subpool it is eligible to satisfy only those requests specifying a class of storage that is associated with that subpool. Once a page has been allocated to a subpool, it cannot be used by any other subpool until all the storage in it has been released. A page allocated to the program subpool is reused only when the program is no longer in use and a short-on-storage condition occurs. The pages allocated to a subpool need not be contiguous. Each subpool has a 1-byte identification.

The subpool names and identifications are given in the first two columns of Figure 60 on page 260. In that figure, **first-fit** means that CICS searches the queue of free areas, and takes the first area large enough to satisfy the request.

Subpool Type		Storage Allocation Algorithm	SAA		FAQE Chained from
ID	Name		Length (in bytes)	Chained from	
01	Control	First-fit	4	CSA	PAM
02	TP	First-fit	8	TCTTE*	PAM
04	Task	First-fit	8	TCA*	TCA
05	Shared	First-fit	4	—	PAM
06	RPL	First-fit	4	TCTTE	PAM
08	Program	Full pages	4	PPT	—

* Overrun detection (duplicate SAA)

Figure 60. Storage control subpools

When a program makes a request for storage, the request specifies the class of storage required and DFHSCP uses a table lookup to determine from which subpool the request should be satisfied. If one of the pages allocated to that subpool has enough free space to satisfy the request, then no new page will be required, but otherwise another page will be taken from the set of unallocated pages.

The subpools are controlled by means of the page allocation map (PAM), which also serves as a temporary work area for DFHSCP. The PAM contains two maps of the dynamic storage area each containing a 1-byte field for each page. The PAM, including the two byte maps, is illustrated in Figure 61 on page 261. In the first map, each byte contains the subpool identification of the subpool to which the corresponding page is allocated. In the second map, the use depends on the subpool to which the page is allocated.

For the task subpool, the bytes corresponding to the number of pages of the initial storage allocation contain a count of the number of pages in the initial allocation. For the program subpool, the value in the byte corresponding to the first page occupied by any program is the number of pages occupied by that program, unless this is zero, in which case the next two bytes contain the number of pages as a halfword. Other bytes for the program subpool, and all other subpools, are zero.

The address of any page can be calculated by multiplying the offset into the map by the page size (PAMPGSZE) and adding it to the start of the dynamic storage area (PAMDYNAM).

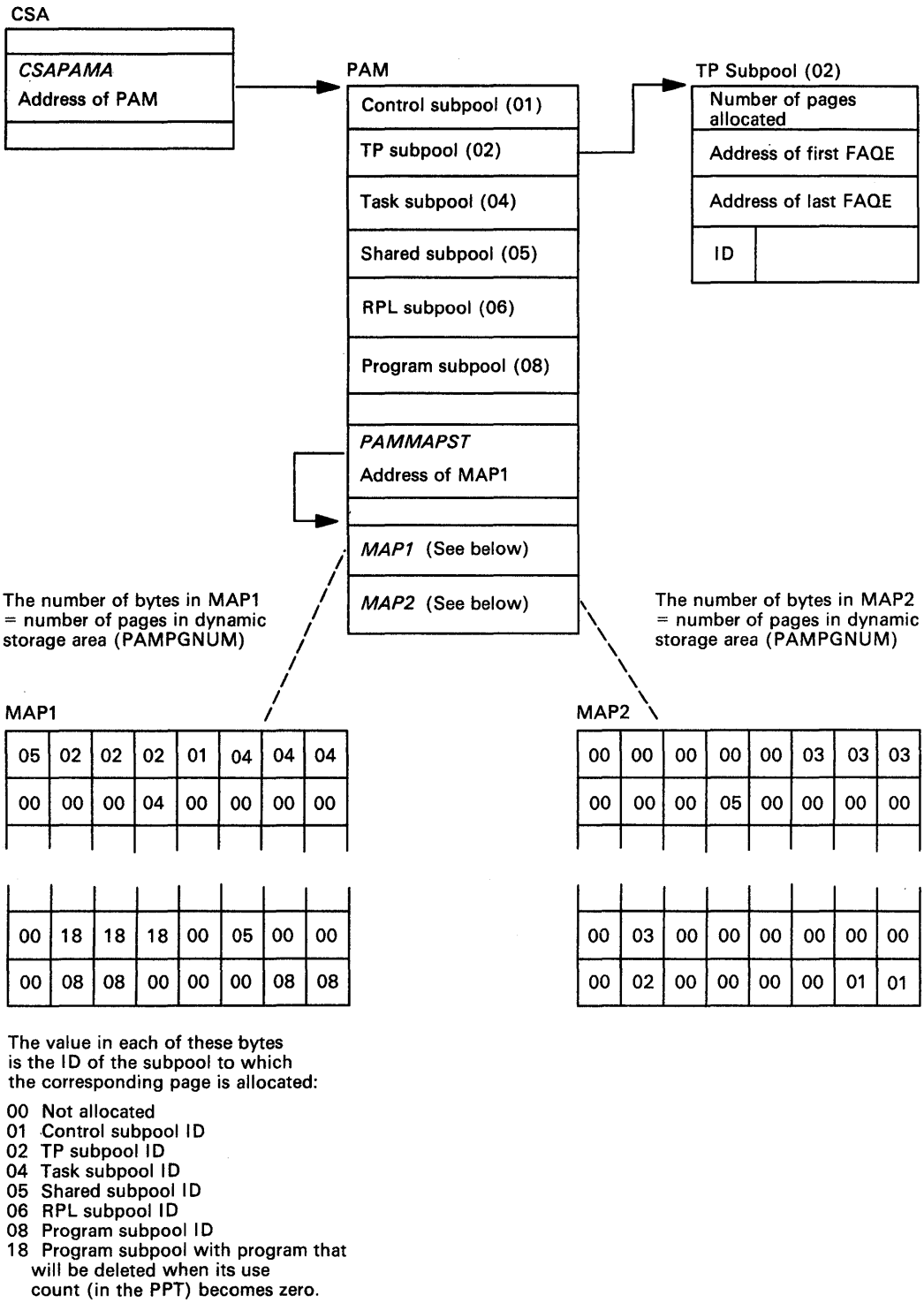


Figure 61. Page allocation map (PAM)

Free area queue elements (FAQEs)

The PAM contains a 16-byte control area for each of the six subpools. This control area or header contains a count of the number of pages currently allocated to the subpool and the subpool identification. For all except the task and program subpools, the subpool header also contains information about the free areas in the subpool. For the control, teleprocessing, shared, and RPL subpools, this information consists of the address of the highest and lowest free areas in the subpool. In these subpools, each free area contains a free area queue element (FAQE) by means of which it is chained to the next and previous free areas.

The FAQE occupies the first 12 bytes of any free area in all except the program subpool. It contains the length of the free area, the address of the next higher FAQE and the address of the next lower FAQE. (If it is the highest or the lowest FAQE, it contains the address of the subpool header.) For the control, teleprocessing, shared and RPL subpools, the subpool header is in the PAM, as mentioned above.

The chaining of free areas is illustrated in Figure 62 on page 263.

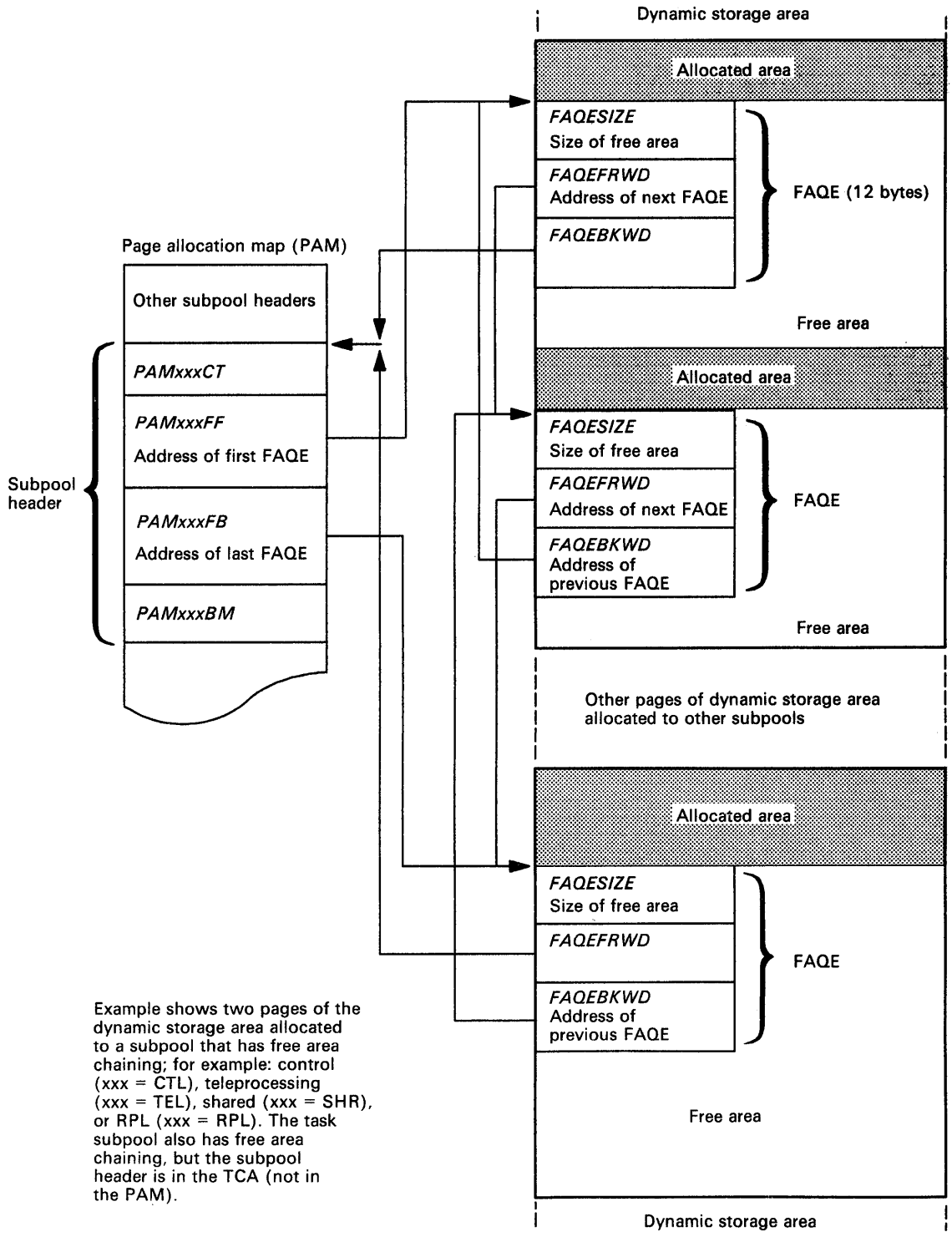


Figure 62. Free area chaining

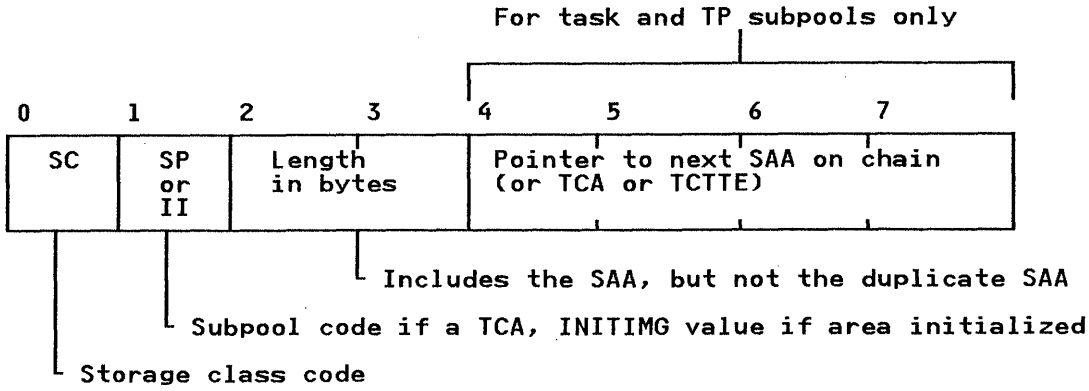


Figure 63. Format of storage accounting area (SAA)

Storage accounting areas (SAAs) for CICS

Each area of allocated storage begins with a storage accounting area (SAA). There are three different formats for an SAA depending on the subpool for which they are used.

SAA for teleprocessing and task subpools

These SAAs are 8 bytes long and are normally chained either from the TCA (for task subpools) or from a TCTTE (for TIOAs in the teleprocessing subpool). An exception to this is line input/output areas. The format of these SAAs is shown in Figure 63.

With the exception of areas assigned for a primed storage task (PRMSIZE specified in the program control table), there is a second SAA at the end of the allocated storage that is an exact copy of the SAA at the beginning of the storage. When the storage is freed these two SAAs are compared. If they are not identical a storage violation has occurred – one of them has been overwritten. The action taken depends on the options selected and is described in "Storage recovery" on page 266.

Storage in these subpools need not be explicitly freed by the task that acquires it. It will be freed automatically either when the task is detached if it is chained from the TCA or when the last output operation has completed if it is chained from the TCTTE.

SAA for control, shared, and RPL subpools

These SAAs are 4 bytes long in the format of the first four bytes of the 8-byte SAA shown in Figure 63. Storage in these subpools is not necessarily associated with a particular task and does not have a chain field in the SAA. Hence it is the responsibility of the acquiring program to ensure that it is eventually freed.

SAA for program subpool

The SAA for the program subpool has a 1-byte storage class code, and a 3-byte length. The length is always a multiple of the page size because storage in the program subpool is always allocated in pages. There are no duplicate SAAs in the program subpool.

Storage classes

When a request for storage is made, in addition to the number of bytes of storage required, the class of storage is specified. From this, DFHSCP uses a table lookup to determine the subpool from which the storage should be allocated.

The storage class is specified by a 5-bit code which is placed in bits 3-7 of the first byte of the storage allocated. Bits 0-2 of the first byte are set to B'100' so the codes are normally seen and specified as a 1-byte code with the first bit set on. The codes are given in Figure 64.

Storage Class		Subpool ID
Code	Names	
81	DCA	01
82	QEA	01
83	ISC TIOA	02
84	LIOA	02
85	TIOA	02
86	ICE	01
87	AID	01
88	Program	08
89	RSA	04
8A	TCA	04
8B	LLA	04
8C	User	04
8D	TRANSDATA	04
8E	TEMPSTRG	04
8F	File	04
90	RPL	06
93	Shared	05
94	Control	01
95	EXTPGM	08
96	TACLE	05
97	TSMAIN	05
98	TSTABLE	05
99	Map	05
9A	Permanent	–
9B	JCA	04
9C	Reserved	–
9D	DWE	04
9E	MAPCOPY	04
9F	DL/I	05

Figure 64. Storage classes and subpool IDs

Storage cushion

When the amount of free storage falls below a certain value (which is specified as the cushion size) CICS sets a short-on-storage indicator (CSASOSON in the CSA) and does not initiate any new tasks until sufficient storage has been released to re-establish the original storage cushion size. (This use of a storage cushion helps to prevent the deadlock that would occur if all tasks tried to acquire storage when none is available.)

The cushion size is specified either by the user on the DFHSIT macro (SCS operand), or by an operator override at initialization time; it can also be changed using the master terminal CEMT transaction for cushion size. The value specified is rounded up to a multiple of the page size. It is stored in the CSA at CSASCNB and the corresponding number of pages is stored in the PAM at PAMCUSHN.

The PAM contains (at PAMAVAIL) a count of the number of unallocated pages that are available before the storage cushion size is reached. Whenever a new page is allocated to a subpool this count is decremented. When it becomes negative, any pages that are in use by the program subpool for programs whose use count is zero are deallocated, and the count is increased accordingly. If this fails to make the count nonnegative then the short-on-storage indicator CSASOSON is turned on in byte CSASOSI of the CSA. No further tasks are initiated until the requisite number of pages becomes available.

Storage for use by the operating system

If a program running under CICS issues an operating system request for storage, the request cannot be met by any of the storage occupied by CICS or by the dynamic storage area. Unless storage is provided from which the operating system may satisfy the request, an abend will occur.

OSCOR

The user can specify a value for operating system storage (OSCOR) operand either on the DFHSIT macro or at startup time. DFHSIP then ensures that after initialization, an area at least equal to the OSCOR specification is available to the host operating system to satisfy GETMAIN requests.

Storage recovery

Storage errors may be detected by DFHSCP itself (for example when an SAA and its duplicate do not match) in which case DFHSCP calls DFHSCR directly, or they may be detected by means of a program interrupt. If a program interrupt occurs while DFHSCP is being executed, the interrupt handler in DFHSRP gives control to DFHSCR.

If storage recovery has not been specified using the SVD parameter of the SIT, DFHSCR issues a trace macro with trace identification X'CA' and abends CICS with the message DFH0501 with "RECOVERY NOT SPECIFIED" included in the message, unless the error was caused by an invalid attempt to free storage, when CICS abends with the message DFH0504. If, on the other hand, storage recovery has been requested, DFHSCR issues a trace macro with trace identification X'CA' and, if requested by the SVD operand, invokes the formatted dump program, DFHFDP. If DUMP = FORMAT has been specified as the dump option on the DFHSIT macro, then DFHFDP dumps

the dynamic storage area as one large block to the CICS dump data set, as well as producing a formatted dump.

DFHSCR then attempts to recover from the storage violation. DFHSCR first checks the subpool header address. If it is not in the PAM, it is assumed to be a task subpool header in a TCA. If it is in the PAM and the subpool is the program subpool, no recovery is attempted and CICS is abended with message DFH0501 and "RECOVERY NOT POSSIBLE" is included in the message.

Recovery of unallocated storage

For all subpools that use FAQEs (that is, all except the program subpool), DFHSCR first checks the FAQE chains. Each FAQE has its length, and forward and backward pointers checked. The scan begins at the header and follows the forward chain until it reaches the end or encounters an error condition. When an error is encountered, the address of the last good FAQE is saved and the chain is then searched backward from the header. If an error is encountered or a FAQE is found whose backward pointer addresses the last good FAQE found on the forward scan or the FAQE addressed by the last good FAQE, then the FAQEs are linked together to restore the chain. This may or may not involve losing some of the dynamic storage. If both forward and backward pointers of a FAQE are destroyed but the length field is not corrupted, then no storage is lost. But if the length field is also destroyed, or if two FAQEs are affected, then something must be lost.

Recovery of allocated storage

DFHSCR then attempts to recover the allocated storage. For a GETMAIN in a subpool with FAQEs, CICS is abended with message DFH0501 if no error was found in the FAQE chain or if a GETMAIN for a TCA in the task subpool fails. Otherwise the GETMAIN is retried with the corrected FAQE chain. For a FREEMAIN in the control, shared, or RPL subpool, the area is made to look like a FAQE if it was not already free. If necessary, the length is set to 16. The FAQE is then added to the FAQE chain.

For a teleprocessing subpool FREEMAIN of an area that is not already free, DFHSCR verifies the storage accounting area. For a TIOA, but not an LIOA, the storage accounting chain is checked, freed areas are removed, addresses are corrected, and the subject area is added to the chain if it is not already on it. If no error has been found, DFHSCR abends CICS, but otherwise it retries the FREEMAIN.

In general, DFHSCR abends CICS with the message DFH0501 only if it fails to find an error or finds one it cannot correct.

Using the storage violation trap

To debug storage violations, you can use the field engineering debug facility known as the storage violation trap or FAQE trap. This trap resides in the trace program (DFHTRP), and is entered before the trace program returns to its caller after making a trace entry.

The storage violation trap provides **two** storage checking functions:

- FAQE chain checking for all or selected subpools
- Verification of user storage associated with the currently active task.

These functions are activated (and deactivated) by the FAQE and TASKSTG operands of the CSFE DEBUG command (see the *CICS-Supplied Transactions* manual).

You can reduce the degradation in performance when checking the FAQE chains of large systems by using the SUBPOOL operand of the CSFE DEBUG,FAQE=ON command to restrict the FAQE chain checking to the subpool with the storage error, where this is known.

Note: TASKSTG checks the FAQE chains associated with running tasks as well as checking the transaction storage chains, so this operand should be specified if the error is known to be in task-related storage.

The storage violation trap can also be activated during CICS system initialization by using the startup override:

```
STGCHK=([FAQE][,TASKSTG])
```

where:

- FAQE specifies that the FAQE chains for **all** storage subpools will be checked (the same as the function provided at execution time using the CSFE DEBUG,FAQE=ON command, with the default of SUBPOOL=ALL).

- TASKSTG specifies that the user storage associated with the currently active task will be checked (the same as the function provided at execution time using the CSFE DEBUG,TASKSTG=ON command).

The ability to activate the storage violation trap during CICS system initialization is a useful function for debugging storage violation problems that occur during the second and third stages of initialization, due to malfunction of either the CICS system or user program list table (PLT) programs. There is no SIT option corresponding to the STGCHK startup override.

When a storage violation is detected by the trap, a formatted dump (storage violation type) is produced, and the storage violation trap is switched off for both the FAQE and the TASKSTG functions. Otherwise, the debugging functions remain in effect until either a CSFE DEBUG,FAQE=OFF command or a CSFE DEBUG,TASKSTG=OFF command is executed, when the corresponding activity will cease.

A message is written to the console when the trap finds an error:

```
DFH0509 - STORAGE VIOLATION DETECTED BY FAQE TRAP,
          ERROR CODE X'02nn' - FAQE TRAP NOW INACTIVE
```

The error code X'02nn', which also appears as the interrupt code in the PSW in the accompanying dump, indicates the type of storage violation detected. Details of the error codes are given in the analysis section of the failure analysis structure table diagram for message DFH0501 in the *Messages and Codes* manual.

In a partition dump, you will find a record of the last storage violation detected by the trap. This record is held in an area of storage in the trace program (DFHTRP). The area is preceded by the eye-catcher '***FAQE TRAP DIAGNOSTIC AREA***', and contains the following information:

- An error description (8 characters) – see Figure 65
- Registers 0-15 at the time of detection of the error by DFHTRP
- The PSW, giving the address in DFHTRP at which the error was detected as the interrupt address, and an error code of the form X'02nn' (as displayed in message DFH0509) as the interrupt code.

```
DCALIMIT   More than 4095 DCAs found in the system
FAQEBKWD   FAQE backward pointer is invalid
FAQEFRWD   FAQE forward pointer is invalid
FAQELGTH   FAQE length is invalid
FAQELIMT   More than 4095 FAQEs found in current subpool
STGLIMIT   More than 32767 areas found on transaction storage chain
STGSAANM   Non-matching duplicate SAA on transaction storage chain
STGSAAZL   Zero length in first SAA for area on transaction storage chain
STGSADER   Invalid area address on transaction storage chain
TCASAAER   TCA has invalid SAA
```

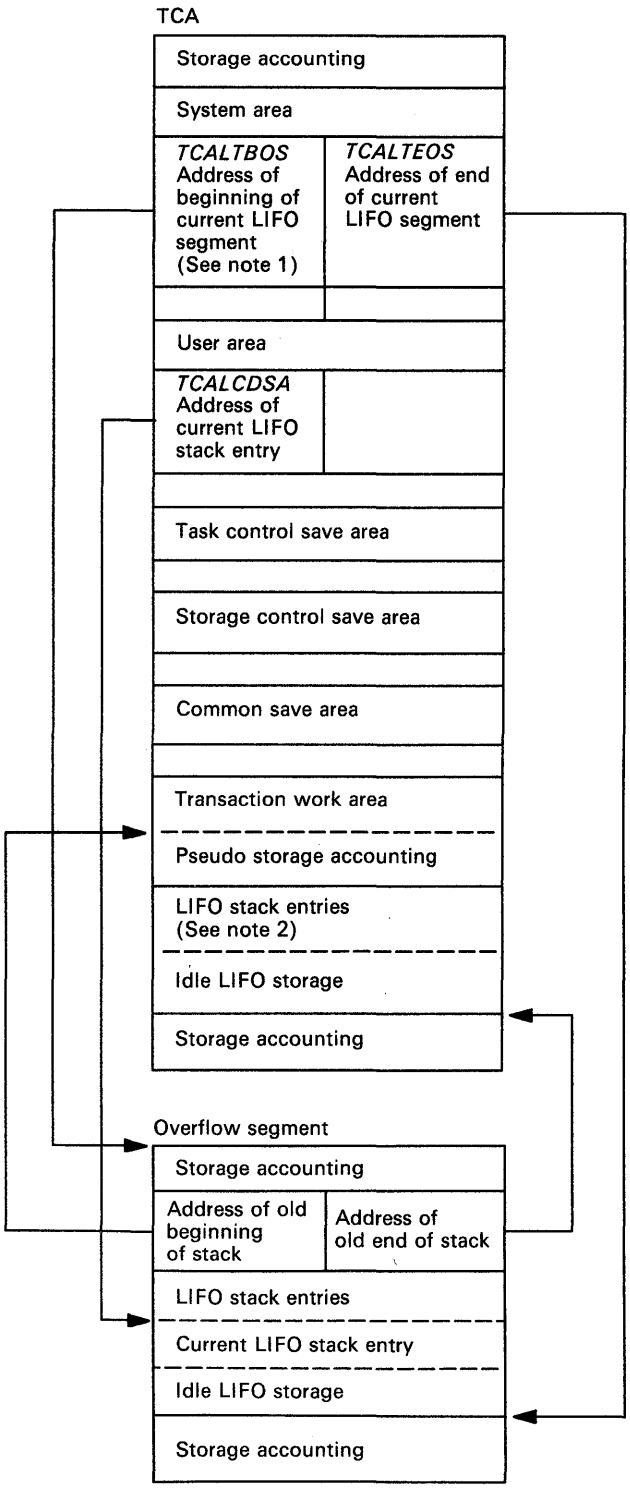
Figure 65. Storage violation errors

Further details of these errors, with a suggested approach for debugging, are given in the *Messages and Codes* manual.

Note: Use of the storage violation trap imposes limits on the numbers of DCAs, FAQEs, and areas on transaction storage chains, to avoid excessive looping in the scanning code, besides making it possible to detect the presence of loops in the chains concerned.

LIFO storage

When a TCA is created, a LIFO stack area (called the initial LIFO stack **segment**) is created. The first LIFO stack entry is created by DFHKCP, which sets control pointers to the start and end of the segment. DFHKCP's LIFO stack entry is flagged with X'42' in the first byte and can be found by using the TCALBGST pointer in the system part of the TCA; see Figure 66 on page 270.



Notes:

1. *The first byte in TCALTBOS and in TCALTEOS contains the number of the current segment (X'FE' for the first, X'FD' for the next, etc.).*
2. *The first LIFO stack entry belongs to DFHKCP and contains X'42' in its first byte.*

Figure 66. TCA and LIFO storage segments

On invocation of a module that uses the LIFO stack macro DFHLFM, the caller's registers are saved in the current LIFO stack entry, and a call is made to a routine DFHLFA, which resides in the DFHCSA module. DFHLFA is passed the module's identity and the required length of LIFO stack entry. Assuming there is sufficient room in the current LIFO segment (determined by comparing the next-available-byte (NAB) pointer plus the required length, against the end-of-stack pointer), then a new stack entry is created immediately. The chain-back and chain-forward slots are set up, the stack entry is initialized with the module identity (a 1- or 2-byte value corresponding to the module's trace ID), and the current LIFO stack pointer (TCALCDSA, at TCA+X'10') is updated to address the new stack entry. Each LIFO stack entry begins and ends on a doubleword boundary.

If, however, there is insufficient room in the current stack segment for the required entry, an **overflow segment** is created by calling DFHSCP to obtain **user** class storage. This overflow segment is then made the current segment and is used to satisfy subsequent LIFO stack requests; it may not be freed until task termination.

An optimization scheme is used to calculate the size of initial LIFO stack segment used during the transaction's life. Thus, the PCT entry for each transaction contains two values:

- The first value (PCTISA01) controls the size of the initial LIFO stack segment to be allocated as part of the TCA. When one transaction terminates, the total size of LIFO storage used is put into PCTISA01, and this value is used as the size of the initial LIFO stack segment for the next invocation of the transaction.

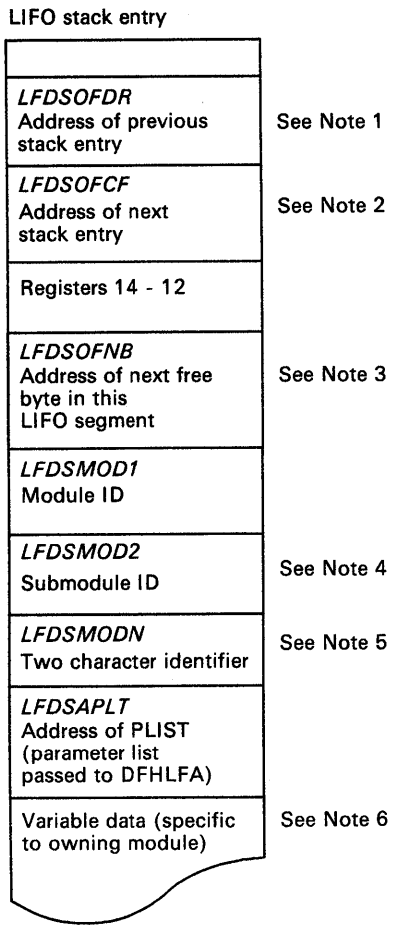
- The second value (PCTISA02) controls the amount of storage to be obtained whenever an overflow segment is required; this value remains constant (at 1K bytes).

Because this optimization scheme is self-tuning, no user control of these values is required.

When a module wishes to return to the module that invoked it, macro DFHLFM TYPE=RETURN is used to reset the current LIFO stack pointer to the previous stack entry, and to restore the invoking module's registers.

The LIFO stack scheme allows for a number of overflow segments to be obtained. Each LIFO stack segment is numbered, starting at X'FE' for the initial segment within the TCA; following segments are numbered X'FD', X'FC', and so on. If the transaction causes the segment number to reach X'CD', the transaction is abended with code ALFA (on the basis that any transaction that tries to exceed this limit is likely to be in a loop).

The main fields of interest within an individual LIFO stack entry are shown in Figure 67.



Notes:

1. *For DFHKCP's LIFO stack entry (the first in the initial segment), the pointer is zero.*
2. *If the first byte of LFDSOFCF is X'FF', then the stack entry addressed by the remaining 3 bytes is no longer active.*
3. *The first byte of LFDSOFNB contains the number of the segment to which this entry belongs (X'FE' for the initial segment, X'FD' for the next segment, and so on).*
4. *For example, DFHZARQ is a submodule of DFHZCP and then LFDSMOD2=X'01'.*
5. *For example, if module is DFHTSP, then LFDSMODN is 'TS'.*
6. *On invocation, some management modules (such as DFHZARQ and DFHTDP) copy the TCA common control area (X'80' onwards) into their LIFO stack entry. These copied areas start after LFDSAPLT in the stack entry and are usually at least 12 bytes long.*

Figure 67. Layout of a LIFO stack entry

Debugging problems caused by overwriting LIFO stack information

In user transactions, the literal string 'LIFOSTOR' is placed at the end of the monitoring control area (if any), or the transaction work area (if any), or the user TCA (if neither is present). Thus, if the literal is corrupted or overwritten, then the problem is probably due to an incorrect TWA size or a transaction writing beyond its TWA allocation. An abend code of ALFE is generated if overwriting occurs.

Each LIFO stack entry has a next-available-byte pointer (NAB), the top byte of which is the segment number. In general, this should be X'FE' unless one or more overflow segments have been allocated. (The number of times overflow segments have been obtained and released is recorded in the system part of the TCA.) An incorrect segment-number value causes an invalid FREEMAIN request to be issued by the overflow module DFHLFO.

DFHKCP's initial LIFO stack entry is always in the initial LIFO stack segment acquired with the TCA. If it is overwritten, then, when the transaction terminates, a program check or invalid FREEMAIN request may be caused while still executing under the terminating task's TCA.

If trace is active, then, at task termination, the updated values (PCTISA01 and PCTISA02) for the transaction (held in the PCT) are placed in the associated X'D0' trace entry.

If a transaction ends with a code of ALFA, ALFB, ALFC or ALFE, it is likely that storage has been overwritten and it is necessary to determine the cause of the overwriting. For abend code ALFA, it is possible that the transaction is in a loop causing LIFO segments to be allocated.

Each LIFO stack entry's length must be a multiple of 8. An abend, ALFB, may be caused if this is not the case.

CICS with MVS/XA

On MVS/XA, when FLENGTH is used instead of LENGTH in the EXEC CICS GETMAIN command, and the requested length is more than 4 kilobytes, and the program is executing in AMODE(31), storage management tries to obtain the storage area starting at an address greater than 16 megabytes, using the MVS/XA GETMAIN macro. If a storage area is obtained by this method, it has no storage accounting area. The length of this storage area is equal to the number of bytes requested, rounded up to a multiple of 8.

The control block DFHSCXDS holds the lengths and starting addresses of a number of these storage areas. Two chains of these control blocks are maintained as needed. One chain, anchored in CSASCXCA and obtained from MVS subpool 1, describes all the storage areas used globally by the CICS system. The second chain, anchored in TCASCXCA and obtained from MVS subpool 0, describes the storage areas used by the associated task.

Task storage areas specified by the second chain are freed implicitly by DFHSCP when DFHKCP issues a FREEMAIN for the TCA.

Chapter 3.3. Abnormal condition handling

In CICS, the following programs are involved in the abnormal termination processing of a transaction:

- DFHSRP – The system recovery program, which handles program checks and operating system abends
- DFHPCP – The program control program, which handles transaction abends
- DFHACP – The abnormal condition program, which reestablishes the correct terminal environment after a transaction abend
- DFHPEP – The user-modified program error program, which may perform any additional processing required
- DFHMGP – The message generating program which writes messages to CSMT.

The system recovery program (DFHSRP)

DFHSRP's save areas

DFHSRP has several save areas contained in its static storage area. To find the static storage area, look at the CSA optional features list (CSAOPFL), where CSASSA points to DFHSSADS. DFHSSADS is a list of addresses of static storage areas, and SSASRP points to that of DFHSRP.

The static storage area for DFHSRP holds the following areas:

- Two program check save areas
- One abend save area
- One system diagnostic work area (SDWA)
- The program check/abend trace table (also addressed by CSASTRTA).

The first two save areas are both 88 bytes long and contain:

- 8 bytes unused (formerly a BC-mode PSW)
- 64 bytes for the contents of registers 0 through 15
- 8 bytes for an EC-mode PSW
- 8 bytes for interrupt information.

Program check handling

Initialization

During CICS initialization, DFHSII1 calls the system recovery program, DFHSRP, which issues a SPIE macro, and initializes abend handling (see "Abend handling" on page 278). As a result of this, if a program check occurs later, the program check handling routine in DFHSRP will be given control by the operating system.

Tracing

DFHSRP makes an entry in the program check and abend trace table (see "Program check and abend trace table" on page 136 for details).

Save areas

In DFHSRP's program check save areas, the first save area contains the PSW and registers of the latest program check and the second one contains the PSW and registers of the previous one. The first actions of the program check routine are to move the contents of the first save area to the second and to fill the first one from the operating system's program interrupt element (PIE) and the registers on entry.

Program check in storage control

The action is to determine if the program check occurred in the storage control program, DFHSCP. If it did, control is passed by the operating system to the recovery routine in DFHSCP.

Abends DFH0601 and DFH0602

Tests are made to see if the task that caused the program check is a system task, that is, the task control task or terminal control task, or a journaling task. If it is, CICS is abended with message DFH0601. This abend will be intercepted by the ESTAE exit, as described under "Abend handling" on page 278, and a dump will result. Another test is made to determine if this task is already in the process of abending because of a program check. This is detected by the presence of the abend code ASRA in TCAPCAC. If the task is already abending with abend code ASRA, CICS is abended with message DFH0602 in order to avoid an abend loop.

Abend DFH0699

This abend code is internal to CICS and should not normally occur. DFHCSA issues the abend when it discovers a runaway task condition. The operating system should pass the abend to DFHSRP by the ESTAE exit so that DFHSRP handles the abend.

Runaway task purge

The next test is to determine if the program check was caused by the interval control program, DFHICP. If a CICS user task does not issue a task control request (other than RESUME), either explicitly or by some other service request, for a specified interval of time (ICVR operand of the DFHSIT macro), DFHKCP is invoked by the timer exit routine. DFHKCP causes the task to program check by overwriting the instruction which it is about to execute. When DFHSRP recognizes this, it passes control to DFHPCP to abend the task with an abend code of AICA.

Formatted dump

At this point in the handling of the program check, the program interrupt exit is terminated. The program check PSW is first saved and the resume address is set to the label SRPSLIH in DFHSRP. After terminating the exit, the program check PSW is restored. The formatted dump program is then invoked if FDUMP = ASRA has been specified on the DFHPCT macro for this transaction.

Note: DFHFDP has its own program check handler because it attempts not to rely on any code or data elsewhere in the CICS address space.

Transaction abend

Finally, the program check PSW and registers are saved in the TCA and a special version of DFHPC TYPE = ABEND is used to abend the task with an abend code of ASRA. This special macro branches into DFHPCP at a secondary entry point which avoids saving the registers. DFHPCP (and hence DFHDCP) then has access to registers 14 through 11 exactly as they were at the time of the program check.

For each transaction abend, a TACB (transaction abend control block) is built containing:

1. The abend code
2. The registers on entry to DFHPCP
3. The PSW at entry for ASRA and ASRB abends only
4. The program active at that time
5. The remote system message (if caused by a remote system abend and a message was sent)
6. If a terminal error was the cause of the abend, the name of that terminal.

Abend handling

Initialization

During CICS initialization and during the same request issued by DFHSII1 (described in "Program check handling" on page 276), DFHSRP, after issuing a SPIE macro, issues an ESTAE macro. As a result of this, if an abend occurs during the execution of CICS, issued either by the operating system or by CICS itself, the abend exit routine in DFHSRP will be given control and may be able to recover from the ABEND.

Note: An ESTAE macro is issued during DFHSIB1 if the environment is suitable for interregion communication. The ESTAE macro nominates module DFHCRC as the exit routine; if the ESTAE exit routine in DFHSRP continues an abend, the operating system passes control to the next exit routine (namely DFHCRC, which performs the interregion cleanup).

Tracing

DFHSRP makes an entry in the program check and abend trace table (see "Program check and abend trace table" on page 136 for details).

Save areas

The first area is filled in by the system recovery program when a program interrupt is intercepted. The second save area immediately follows the first and contains a copy of the previous contents of the first save area. The third area is 104 bytes long and contains a copy of the ESTAE work area.

Recursive abend

After establishing addressability and loading registers 13 and 12 with the addresses of the CSA and the current TCA respectively, DFHSRP tests to determine if it has been reentered while attempting to recover from a previous abend. If it has been reentered from a system abend, message DFH0612 is issued and no recovery is attempted. If reentered from a CICS abend, message DFH0405 is issued and no recovery attempted.

System recovery table search

DFHSRP then searches the system recovery table looking for an entry corresponding to the abend code. If an entry is found, the specified program or routine is invoked after first terminating the abend exit status and reestablishing the SPIE. If the user's recovery program or routine specifies that CICS can continue execution, DFHSRP abends the current task with a transaction dump and, if FDUMP=ASRB is specified in the PCT entry, a formatted dump.

Termination

If no entry is found in the system recovery table search, or if the user does not specify that CICS can continue, the operating system abend is allowed to continue. The operating system is left to close files and terminal data sets.

The program control program (DFHPCP)

When an EXEC CICS ABEND command is issued by an application program or by a CICS module, the abend routine in DFHPCP is given control.

System task abend

If the task that is abending is the task control task or the terminal control task, or is a journaling task abending with other than a journaling abend, the situation is catastrophic and DFHPCP immediately terminates CICS with message DFH0401.

If the task is associated with a VTAM terminal, a bit is set in the TCTTE to inform DFHZCP that the task has abended.

If the task is using the load routine, the load routine is made available to other tasks, the load-in-progress indicator is turned off in the PPT entry of the program being loaded and any storage that has been acquired for the program is released.

Invocation of task abend exit

If the EXEC CICS ABEND command did not have the CANCEL option and there is an active abend exit, it is now invoked. See "Task abend exits" on page 281.

Transaction dump

If the EXEC CICS ABEND command had the CANCEL option or if there is no active exit, the abend process continues. If a dump was requested and has not been suppressed by specifying DUMP = NO on the DFHPCT macro for this transaction, DFHPCP calls DFHDCP to take a transaction dump.

Program elevation

The program elevation routine is then called to terminate the program currently being executed. If this is not the highest-level program in the stack (that is, if it was linked to by a higher level program), DFHPCP loops back to check if the next higher-level had an abend exit and, if so, to invoke it. Program elevation is called again to remove this higher level and so on until the top of the execution stack has been released.

Transaction backout and restart

If the abending task is one for which dynamic transaction backout has been specified (DTB = YES on the DFHPCT macro), DFHPCP calls the dynamic backout program, DFHDBP, to perform transaction backout to restore protected resources to their state before the current logical unit of work began.

In addition, for a transaction designated as **restartable**:

1. Any TIOA, command-level communications area, or terminal user area that existed at task initiation time is reinstated.

2. Interval control AIDs used in the task are re-created from deferred work elements (DWEs).

3. Flags in the TCA are maintained to show (1) what terminal traffic has occurred during the task; (2) whether a sync point has been passed; and (3) whether or not the current activation of the task is the result of a restart.

Assuming DFHDBP's backout was successful, the **proceed** flag (bit TCADBTRP in TCADBRTS) in the system part of the TCA is turned **on** only if:

1. The abend code is ADLD, indicating program isolation deadlock;

2. The **terminal-traffic** flags (bits TCAZRRD and TCAZRWRT in TCAZLUWT) are **off**; and

3. The **sync-point** flag (bit TCAZIOSK in TCAZLUWT) is **off**.

DFHDBP then executes a conditional link to the retry program DFHRTY (optionally user-written). DFHRTY has the task's TCA and all user storage available and may or may not change the setting of the **proceed** flag.

When control is returned from DFHRTY, DFHDBP inspects the **proceed** flag:

- If the **proceed** flag is **on**, DFHDBP executes DFHPC TYPE=RETRY, which:

  1. Releases all user storage

  2. Sets the EXEC interface pointer (TCAEISA) and the BMS pointer (TCAOSPWA) to zero

  3. Clears the TWA

  4. Resets the dynamic log

  5. Updates the retry counter (PCTATRCT)

  6. Executes DFHTC TYPE=WAIT (to let terminal activity finish)

  7. Sets appropriate flags and fields in the TCA

  8. Transfers control to the initial application program.

- If the **proceed** flag is **off**, DFHDBP transfers control to DFHACP with a flag set in the recovery and restart section of the TCA to show that restart was rejected. DFHACP then issues a message to say why restart was rejected.

Task abend exits

A task abend exit is written by the user and may be either a routine or a program. If it is a program, then it must terminate with a DFHPC TYPE=RETURN (as in any other program) and DFHPCP will then issue DFHPC TYPE=RETURN, which terminates the invocation of the program that caused the abend. Using an abend exit program, it may be difficult to pass control back to the program that caused the abend. If an abend exit routine is used however, it may be a subroutine of the program that abended so that return to the point of error is simpler. If recovery from an ASRA abend (program check) is required, then the abend code in the TCA at TCAPCAC must be cleared or a further program check will cause a CICS abend with message DFH0602.

The abnormal condition program (DFHACP)

The abnormal condition program, DFHACP, is invoked by DFHPCP during transaction abend processing. Its purpose is to reset the status of a terminal attached to the transaction and to call DFHMGP passing the message number. DFHMGP then informs the operator that the transaction has abended, calls the user-written program error program, DFHPEP, and writes a message to the transient data destination, CSMT.

DFHMGP is invoked by DFHACP and DFHZEMW, using DFHSPP, to write messages to CSMT, and the operator. DFHMGP uses the set of skeleton messages in DFHMGT to build complete messages and then write them to the appropriate destination. For LU6.2, DFHACP calls DFHZERH.

The program error program (DFHPEP)

The user-written program error program, DFHPEP, is invoked by DFHACP during transaction abend processing.

The version of DFHPEP that is distributed with CICS contains code to establish a base register and addressability to the system portion of the TCA, and to return control to DFHACP by a DFHPC TYPE=RETURN macro instruction. This module may be updated to include any user logic.

For a description of how to write and use a program error program, see the *Customization Guide*.

Chapter 3.4. Storage areas and tables

This chapter briefly describes some of the more important storage areas and tables; the descriptions are presented in alphabetic order. (Readers interested in storage areas should also see "Chapter 2.4. Control block linkages" on page 141.)

AFCB: Authorization facility control block. The AFCB is used as an address vector for the CICS Type 2 SVC, or for the authorization of use of the SVC.

AFCS: Authorized function control shared block. An AFCS is the anchor for data shared by several operating system tasks.

AID: Automatic initiate descriptor. An AID is created to represent a request for the use of a terminal. This may be an automatic task initiation request for a new task to be attached to the terminal, or a request by an existing task for the use of that terminal. The AID chain is sequenced by symbolic terminal identification. A terminal may represent a conversation to a remote system. In this case, AIDs for that conversation are chained from the system entry for that system (TCSESUS). For all other terminals, AIDs requesting that terminal are chained from the TCT at TCSESUSF.

ALT: Application load table. The ALT is used to control the layout of permanently resident application programs in the CICS address space. It is used during initialization.

Bind model: The bind area is a VTAM control block that CICS uses to establish a session with a destination. CICS dynamically allocates storage for the bind area (with the NIB) and uses the bind model to set up the correct values in the area. There is a bind model for each device type and the models are stored together logically as part of the TCT prefix.

CCE: Console control element. The CCE is the interface between an operating system console and CICS.

CDBLK: CONVDATA block. This data area is generated for the CONVDATA option in generalized data streams commands. An application program can include the copy book DFHCDBLK to define the area.

CLT: Command list table. The CLT defines the hierarchy of regions in an XRF environment. There is one for each alternate system, and it controls the use of MVS-authorized services during takeover. It is loaded during takeover, and deleted when it has been processed.

CRB: CICS region block. The CRB is used to control an interregion communication session. It is allocated at the start of a session and is freed at the end of a session. The CRB is addressed by CSACRBA.

CSA: Common system area. The DFHCSA module contains the storage for the CSA control block, the DCAs, the TCAs of terminal control and task control, the CSA optional features list, the timer exit, and the copyright statement. The CSA control block contains the addresses of CICS nucleus programs, the anchor points of various chains, and other control information.

DBLDS: Dynamic buffer area. DBLDS is acquired by the journal control program for use as a dynamic log for transactions for which dynamic transaction backout is applicable. The records (DBRDS) written to this log are equivalent to those written to the system log for use by the transaction backout programs during emergency restart. Should the dynamic buffer area prove to be too small, temporary storage is used.

DBR: Dynamic backout record (within dynamic log buffer). A DBR is put on the dynamic log for each change to a protected file, transient data destination, temporary storage, or DL/I data base by a transaction for which dynamic backout is specified. In the event of the transaction's abnormal termination, the DBRs are used to back out the protected resources.

DCA: Dispatch control area. A DCA is created for each task by CICS. It is placed on a DCA chain and serves as a record of the dispatching priority and current status of the task. Whether a task is active or suspended at a particular time during execution determines whether its DCA is on the active or suspended chain. The active chain is addressed by CSAACTFA or CSAACTBA; the suspended chain is addressed by CSASUSFA or CSASUSBA. The dispatcher uses DCAs rather than TCAs to reduce page faults.

DCT: Destination control table. The DCT contains an entry for each extrapartition, intrapartition, and indirect destination. Extrapartition entries address the DCB. Indirect destination entries address the DCT entry for the destination to which they indirectly refer. Intrapartition destination entries contain the information required to locate the queue in the intrapartition data set.

DDIR: DMB directory (DL/I). A DDIR contains an entry for each data base referenced by the CICS subsystem scheduled by the current task. There is one entry per physical data base, but none for logical data bases.

DLDO: DBRC parameter list. A DLDO represents the parameter list passed to DFHDLI for DBRC online commands.

DLMT1, DLMT2: CEMT parameter list. The DLMT DSECT is used by the master terminal routines when processing DL/I commands.

DLP: DL/I interface parameter list. DLP is used to pass data from the DL/I interface CALL initiation program (DFHDLI) to the DL/I interface application program (DFHDLQ). It holds many ECBs, pointers, and counters used by the CICS-DL/I interface. CSADLI in the CSA optional features list (CSAOPFL) points to this control block.

DMB: Data management block. The DMB is used by the partition specification block (PSB) in a DL/I system.

DRCA: Dependent region control area. DRCA is the control block used by the batch region participating in a shared data base session with a CICS system using IRC.

DRX: DL/I resource block. DRX represents a DL/I resource that resides on another CICS system and is used by the CICS function request shipping transformer programs (DFHXFP and DFHXFQ) for both online and batch systems.

DSN: Data set name. A DSN block holds a name for dynamic allocation when a file is opened, or represents a base cluster (VSAM), or a data set (BDAM), that can be updated and is capable of being backed out.

DWE: Deferred work element. A DWE is created and placed on a DWE chain to save information about an event that must be completed before task termination but is not completed at the present time. For example, a

```
DFHFC TYPE=GET,TYPOPER=UPDATE
```

macro instruction causes a DWE to be created. The DWE remains on the DWE chain until the update operation is performed, until a sync point, or until task termination (if the task terminates without requesting the update operation). DWEs are also used to save information about work to be backed out in case of an abend.

EIB: EXEC interface block. EIB is part of EIS. It holds information about the command being executed, and this information is available to the application program.

EIPDS: Command-level interface DSECTs. EIPDS holds DSECTs for the label table (EIL, EILLAB), register save area (EIR), EIS appendage (EIO), program control link table (EIP), data interchange area (EII), argument list (EIA), and push chain (EIU).

EIS: EXEC interface storage. EIS is used by DFHEIP to control applications that use the EXEC interface. EIS contains the EIB.

EPB: Exit program block. An EPB represents an enabled exit program, and holds the program name, program address, work area address and length, and other data.

EPL: Exit program link. EPL represents the activation of an exit program, and holds pointers to the corresponding EPB, and any further EPLs for other programs at that exit.

FBWA: File browse work area. The FBWA is used during a browse operation against a BDAM data set to maintain position in the data set. The FBWA is acquired when a DFHFC TYPE=SETL macro instruction is issued against such a data set. It is released when the browse operation is terminated by a DFHFC TYPE=ESETL. The address of the FBWA is placed at FCFBWA in the FIOA used in the browse operation.

FCENT: File control operation entry. The FCENT holds information about a function-shipped file control request that may be required for a later file control request. It is chained from EISFCPTR in the EXEC interface structure (EIS), and created by the transformer programs DFHXFP or DFHXFX.

FCT: File control table. The FCT is used to describe the data sets accessed by the file control program. Each entry specifies the types of services to be allowed for a file, and indicates the access method used to get or put a record. Included as an appendage to each BDAM FCT entry, is the data control block (DCB). For a VSAM FCT entry, the appendage contains the access-method control block (ACB) for that file.

FIOA: File input/output area. An FIOA is acquired by the file control program whenever a request is made to a BDAM data set. The data area, beginning at field FIOADBA, is used as the true I/O area from or to which records are read or written.

FMH: Function management header. FMH is a data string containing SNA information. The FMH is optionally added to the data sent or received by SNA devices, according to requirements.

FWA: File work area. An FWA is acquired by the file control program when reading or updating an existing blocked record, when adding a new record to a file, or when retrieving records using the browse feature.

HTA: HPO transaction area. The HTA is associated with the execution of some item of work under an SRB (service request block in MVS). It may be associated with a TCA when the task issues a request to enter SRB mode, or may execute independently when it is the subject of an ATTACH HTA request. The HTA is acquired from a chain anchored at SRANXHTA in the SRA, or, failing that, by a DFHSC TYPE=GETMAIN.

ICE: Interval control element. An ICE is created for each time-dependent request received by the interval control program. These ICEs are chained from CSAICEBA in the CSA in expiration time-of-day sequence.

IRSDS: Interregion communication subsystem blocks. These blocks are allocated by the IRC SVC routine, but may be accessed by the participating subsystems.

ISB: CICS-DL/I interface scheduling block. DFHISB is used by the CICS-DL/I interface CALL initiation program (DFHDLI) during the execution of transactions using DL/I local data bases.

JCA: Journal control area. This control block is the communications vehicle for journal control requests and comprises three major areas: the register save area used by the journal control program, the JCA communication area, and the JCA system prefix build area. It is acquired by a task as the result of issuing a DFHJC TYPE=GET macro. It is addressed by the field TCAJCAAD.

JCR: Journal control record. This contains a standard prefix containing information such as the record's ID, and the record number. If the record has been created by the system, then a system prefix is included, and the data follows. The first record in each block is a journal control label record indicating, among other things, the date and time the block was written.

JCT: Journal control table. JCT consists of a root, followed by JCT entries occupying contiguous storage locations. The root is addressed through the optional features list by field CSAJCTBA. The JCT root addresses the first JCT entry by field JCTTESA; further entries can be found using the length field (JCTTELEN) in each entry.

JCTTE: Journal control table entry. A JCTTE is generated by the DFHJCT macro instruction and holds the DSECTs DFHJCEDD, and DFHJCEXD. Each JCTTE describes one journal, and has the data-set/device description fields: JCTCED and JCTAED. The DFHJCEDD DSECT maps these fields. The journal control data set status control block (DFHJCEXD) holds the status needed for protection against reuse of data sets in connection with the PAUSE mechanism.

KCS: Task control static storage. The KCS is used by task control for event control blocks (ECBs) and working storage. This control block occurs only once in a CICS system and is allocated by DFHSIB1.

KCTWA: Task control transaction work area. A KCTWA is the work area for task control.

LFSE: LIFO stack entry. LFSEs are extended standard save areas that are used by many CICS modules to save registers on entry and to contain work space for dynamic variables. The current LFSE is addressed by TCALCDSA. Each LFSE contains a chain back field, a chain forward field and a next available byte pointer (NAB). The area is of variable length and the end of the area is pointed at by the NAB. For further information, see "Chapter 3.2. Storage management" on page 259.

LFSS: LIFO stack segment. This is an area used to contain one or more LIFO stack entries (LFSEs). The initial segment is allocated as part of the TCA; its length is dynamically controlled to be an optimum size. Overflow segments are obtained whenever there is insufficient storage remaining in the current segment to satisfy a request for a LFSE. For further information, see "Chapter 3.2. Storage management" on page 259.

LLA: Load list area. LLAs are used by DFHPCP to keep track of programs and maps that have been loaded by a task. Each entry in an LLA contains the address of the PPT entry of the program loaded. LLAs are chained from TCAPCLC.

LUC: LU6.2 control block. LUC holds storage definitions for use by the LU6.2 modules. This DSECT is passed to DFHZARL when a DFHLUC macro is executed.

LUM: LU6.2 macro-level control block. LUM holds information for the execution of DFHTC requests by LU6.2. LUM is passed to DFHZARM when a DFHLUCM macro is executed.

MBCA: Transient data buffer common area. The MBCA is allocated by the initialization program, DFHSID1, and addressed from TDSTMBCA in the transient data static storage area, DFHTDST.

MBCB: Transient data buffer control block. There is one MBCB for each I/O buffer, which is allocated by the initialization program, DFHSID1. The MBCBs are chained from MBCAFCN1, MBCAFCN2 and MBCAFCN3 in the transient data buffer common area (MBCA). MBCAFCN1 is the anchor for MBCBs that are unallocated and whose corresponding buffer is empty. MBCAFCN2 is the anchor for MBCBs that are unallocated and whose corresponding buffer is valid. MBCAFCN3 is the anchor for MBCBs that are allocated.

MCB: Message control block. The MCB contains information used by BMS to control the routing and viewing of messages issued in response to paging commands entered by the terminal operator.

MCR: Message control record. The MCR contains a list of the terminals to which a logical message is to be routed. It is created by the DFHBMS macro instruction. The ID of the MCR is in the AID, in the ICE, or in the MCB, depending on the status of the request.

MCT: Monitor control table. MCT contains the definition of user event monitor points (EMPs), and specifies the journal data sets used to record the data.

MGT: Message tables module. The MGT consists of tables of prototype error messages that are accessed by DFHMGP.

MQCB: Transient Data Queue Control Block. There is one MQCB for each I/O buffer, which is allocated by the initialization program, DFHSID1. The MQCBs are chained from TDDCTFCN in the destination control table (DCT) or MBCAFCNQ in the transient data buffer common area (MBCA).

MRCA: Transient data string common area. The MRCA is allocated by the initialization program, DFHSID1, and addressed by TDSTMRCA in the transient data static storage area (TDST).

MRCB: Transient data string control block. There is one MRCB for each VSAM string. A chain of MRCBs is allocated by the initialization program, DFHSID1, and anchored in MRCAFCN1 in the transient data string common area (MRCA).

MWCB: Transient data wait control block. The MWCB is used when more than one task is waiting on a single event or for a buffer wait, post or cancel. MWCB holds the ECB on which the tasks wait. MWCBs are acquired by DFHTDP from shared storage before an intrapartition request, and chained from TDSTFCNW in the transient data static storage area (TDST).

NIB descriptor block: The node initialization block (NIB) and BIND area are VTAM control blocks for each node containing information which CICS uses to communicate with the node. CICS dynamically allocates storage for the NIB and BIND areas and uses the NIB descriptor block to set up the correct values in the NIB as required by VTAM. The descriptor block is logically part of each VTAM TCTTE.

OSPWA: Output services processor work area. The OSPWA is used by all basic mapping support (BMS) routines as a storage area in transmitting data between routines and across BMS calls. It is acquired when the first BMS request is issued and is addressed by the TCAOSPWA field of the TCA.

PAM: Page allocation map. The PAM contains information used by the storage control program to manage dynamic storage. It is addressed by CSAPAMA and includes 16-byte headers for each of the storage subpools defined within CICS. (See "Chapter 3.2. Storage management" on page 259 for a full description of the PAM.)

PCB: Program communication block. A PCB defines an application program's view of a DL/I data base. The types of access allowed (for example, read or update) are included in the definition. The set of PCBs for an application program is held in a program specification block (PSB).

PCS: Program control static storage. A PCS is used by program control for event control blocks (ECBs), and working storage. This control block occurs only once in a CICS system, and is allocated by DFHSIB1 for the lifetime of CICS.

PCT: Program control table. Its purpose is to identify transactions to CICS. The PCT consists of a set of entries, each of which begins with a 4-character transaction identifier. Each entry contains the name of the PPT entry of the first program to be executed for this transaction. Also included are a set of attributes of this transaction, and transaction statistics.

PDIR: PSB directory. The PDIR contains one entry for each PSB to be used during CICS execution. It is loaded during CICS initialization.

PLT: Program list table. The PLT is a list of names of programs that are to be executed during CICS initialization before the terminal control program receives control or during the first or second stages of CICS termination. See "Chapter 3.1. Normal operation" on page 235 of this manual.

PPT: Processing program table. Its purpose is to identify application programs to CICS, including those supplied by CICS itself. Each entry in the table contains control information including the program name and the load address (after the storage for the program has been allocated).

PRA: Primed storage area. If a task runs **primed**, then it is given a large storage area at initial attach by the task control program. This area contains a control header, the TCA and TWA for the task and any task-related storage area that is small enough to fit in the remaining space, allocated as required by calls to the storage control program. The PRA is addressed by TCAPRAA when in use or by PCTNXPRA when free.

PSB: Program specification block (DL/I). A PSB describes a program's intent against one or more data bases. An application program schedules a PSB, and finishes its use by a TERM or SYNCPOINT call. A PSB is a set of program communication blocks (PCBs).

PSDDS: Partition set data area. The PSDDS describes the areas that are constructed by the DFHPSD and DFHPDI macros for partition sets.

PST: Partition specification table. The PST holds information used by DL/I for a particular transaction.

QCA: Queue control area. The QCA is logically part of the CSA. It is assembled in the DFHCSA module and is addressed by CSAQETBA. It contains 64 hash chain anchors for QEAs currently in use (that is, representing a satisfied ENQ request), and the anchor of a chain of free QEAs available for reuse.

QEA: Queue element area. QEAs are used by the task control program to control resources being enqueued upon (by the DFHKC TYPE=ENQ/DEQ macro). A QEA is used to represent a resource that is owned by the task, and also to represent an enqueue request that has not yet been satisfied because the resource is currently owned by another task.

RCS: Recovery control static storage. An RCS is used by recovery control for event control blocks (ECBs), and anchors for thread management. This control block occurs only once in a CICS system, and is allocated by DFHSIB1 for the lifetime of CICS.

RSA: Register save area. An RSA is obtained by DFHPCP during execution of a DFHPC TYPE=LINK request, and is used to save registers and other control information belonging to the program issuing the LINK. The RSA information is copied from the TCA and is restored to the TCA during DFHPC TYPE=RETURN processing.

RSB: Remote scheduling block. RSB is used by CICS-DL/I interface routines when a DL/I request is for a remote data base. The RSB is allocated during a scheduling call for a remote PSB, and is freed during the corresponding DL/I TERM call.

SAA: Storage accounting area. An SAA forms the first part of storage obtained by DFHSC TYPE=GETMAIN macro. It contains control information such as the size of the area. See "Chapter 3.2. Storage management" on page 259 of this manual.

SCD (DL/I): System contents directory. The SCD is the main DL/I control block (similar to the CSA in CICS). The SCD holds pointers to the routines for IWAIT, IPOST, and similar functions, and it is pointed to by DLPSCD.

SCXDS: Extended storage anchor block. The SCXDS holds the addresses and lengths of a set of storage areas whose addresses are greater than 16 megabytes.

SDT: Series definition table. The SDT holds names and attributes of series of volumes.

SECDS: Security table. SECDS holds the addresses of the operator ID, old and new passwords, resource name, and other data for the security program.

SKA: Subtask control area. An SKA describes the storage used by a subtask. It is contained in the static storage for DFHSKP.

SKW: Subtask work queue element. An SKW represents a piece of work to be performed by the subtask management program. The first area is in DFHSKP static storage and subsequent SKWs are obtained from shared storage. The SKWs are chained together and controlled by the subtask management program, DFHSKM.

SNT: Sign-on table. The SNT holds terminal operator data, including the operator name, password, and operator priority.

SNTTE: Sign-on table entry. An SNTTE is created from information in the sign-on table, and attached to the TCTTE at sign-on time.

SQE: Subtask queue element. The SQE holds information that describes an element of work that is to be processed by the VSAM/BSAM subtask program, DFHVSP. It is contained in a journal control table entry (JCTTE) or a VSAM work area (VSWA).

SRA: SRB interface control area. The SRA is special to MVS/HPO systems. It is logically part of the CSA. It is addressed by CSASRAA in the CSA optional features list and is assembled in the DFHCSA module. It contains control information relating to queues of SRB mode work, the number of currently executing SRBs and the queues of work completed and ready to run in TCB mode again.

SRT: System recovery table. The SRT is used by the system recovery program to determine the routine or program to be executed during recovery from an operating system abend. Each entry contains a user or system abend code and a routine address or program name.

TACB: Transaction abend control block. The TACB contains details of, for example, the PSW and registers at the time of an abend. If multiple abends occur, one control block per abend is built, the control blocks being chained together.

TACLE: Terminal abnormal condition line entry. When an abnormal condition associated with a non-VTAM terminal or line occurs, terminal control places the terminal out of service and creates a TACLE, which is chained off the line entry (TCTLE) on which the error occurred. The first 16 bytes of the TACLE contain CICS error information in the format described by the DSECT DFHTACLE. The remainder of the TACLE is in the format described by the DSECT DFHTCTLE. It contains a copy of the data event control block (DECB) at the time the error occurred, plus other valuable information.

TCA: Task control area. A TCA is created for each task when it is first dispatched, and released at task termination. Its contents are organized into several logical sections: (1) CICS system control section (addressed by the first field of the next logical section); (2) application program communication section (always addressed by register 12 during execution of the task); (3) transaction work area (TWA) (optional); (4) monitoring area (optional); (5) load list area; (6) EXEC interface storage (optional); (7) lock block storage; and (8) LIFO storage.

TCT: Terminal control table. The TCT contains line and terminal information required by the terminal control program, and provides space to retain certain desired statistical information. It consists of TCTLEs, TCTTEs, BTAM polling and addressing lists, the terminal wait list, data control blocks (DCB), and for VTAM systems, the LOGON, LOSTERM, RELREQ, LERAD, SYNAD, and TPEND exit bootstrap routines, the node information blocks (NIBs), the NIB descriptors, and the VTAM access method control block (ACB). For systems using CICS function request shipping, the TCT includes the system entry tables (TCTSEs), and mode entry tables (TCTMEs). Figure 68 on page 296 shows the storage layout of the TCT.

TCTLE: Terminal control table line entry. A TCTLE describes a communication path to one or more terminals on the system. (The type of communication line is described to CICS by the DFHTCT TYPE=LINE macro instruction which generates the TCTLE.) It contains the data event control block (DECB) that is used to communicate with the appropriate access method. In addition, other information required by CICS to control the communication link is saved here. The TCTLE does not apply for VTAM terminals.

TCTME: Terminal control table mode entry. The TCTME describes a group of terminals that have the same characteristics for use with LU6.2 systems.

TCTSE: Terminal control table system entry. The TCTSE in the TCT is used to describe a remote system to which CICS applications can send commands to be executed against remote resources. The TCTSE contains (1) a pointer to the TCTTE that describes the session with the remote system, (2) a pointer to the chain of AIDs representing outstanding ALLOCATE requests, and (3) statistics on the number of requests sent to the remote system.

TCTTE: Terminal control table terminal entry. The types of terminals on the system are described to CICS by the DFHTCT TYPE=TERMINAL macro instruction, which generates the terminal control table terminal entry (TCTTE). In addition to describing the terminal, the TCTTE addresses the corresponding TCTLE (RPL for VTAM terminals), and the active TCA, and TIOAs; it also contains control information relating to terminal control requests issued by the CICS application program.

TCTTELUC: TCTTE extension for LU6.2. TCTTELUC holds information for LU6.2 requests.

TCX: TCA extension for LU6.2.

TDST: Transient data static storage area. The TDST is allocated by the initialization program, DFHSID1, and addressed from CSATDSTA in the CSA optional features list (CSAOPFL).

TIE: Transaction interface element. The TIE supports the task-related user exit interface. This control block is built by DFHERM, and includes a DWE.

TIOA: Terminal input/output area. TIOAs are acquired and chained to the TCTTE as needed for terminal input/output operations. The field TCTTESC addresses the first terminal-class storage area obtained for a task (the beginning of the chain) and the field TCTTEDA gives the address of the active TIOA. CICS terminal management passes data received from a terminal to the CICS application program in the TIOA, and writes data from the TIOA to the terminal.

TMDEL: Table management directory element. The TMDEL is used by DFHTMP, and is a set of pointers that address members of chains of directory elements and a pointer to the corresponding directory segment.

TMDSG: Table management directory segment. The TMDSG is used by DFHTMP, and holds the name of an entry in the table.

TMELD: Table management lock block. The TMELD is used by DFHTMP for local and global locks. It holds lock values and counts.

TMRQ: Table management parameter list. TMRQ holds information passed from a calling routine to DFHTMP. It also holds the response code, and working storage for DFHTMP.

TMSKT: Table management scatter table. TMSKT holds pointers to directory elements for use by DFHTMP.

TMSSA: Table management static storage area. TMSSA holds global data for DFHTMP.

TPE: Terminal partition extension. TPE defines the TCTTE partition extension. It is pointed to by TCTTETPA.

TSACA: Temporary storage auxiliary control area. The TSACA controls the use of auxiliary temporary storage. It is allocated by the initialization program, DFHSIH1, and pointed to by TSMACAP in the temporary storage common area (TSMAP).

TSBCA: Temporary storage buffer control area. There is one TSBCA for each buffer. TSBCAs are allocated by the initialization program, DFHSIH1, and chained from TSABCAHD in the TSACA.

TSGID: Temporary storage group identification control block. Each temporary storage queue has at least one TSGID that contains entries addressing each element of the queue. Extra TSGID entries are allocated as required.

TSQE: Temporary storage queue element. A TSQE is allocated for each auxiliary temporary storage queue for which there is an active request. The TSQEs are held in temporary storage request element blocks (TSREBs) together with temporary storage request elements (TSREs). A TSQE is created when the first request is received for a queue and deleted when all requests for the queue have been completed. If the queue is recoverable and it is modified, then the TSQE remains until the end of the logical unit of work (unless there are further requests outstanding for the queue). Allocated TSQEs are chained from TSUTEQEA in the temporary storage unit table entry (TSUTE).

TSRE: Temporary storage request element. A TSRE is allocated for each auxiliary temporary storage request. The TSREs are held in temporary storage request element blocks (TSREBs) and are freed on exit from DFHTSP. Allocated TSREs are chained from TSQECHNP in the temporary storage queue element (TSQE) for this queue.

TSREB: Temporary storage request element block. A TSREB is allocated in shared storage during initialization by DFHSIH1, and more blocks are allocated as required. A TSREB holds temporary storage queue elements (TSQEs) and temporary storage request elements (TSREs). TSREBs are chained from TSAREBHD in the temporary storage auxiliary control area (TSACA) and TSAFFREP in the same area points to the first free request element.

TST: Temporary storage table. The TST is a list of generic mnemonics used to identify temporary storage DATAIDS or to identify DATAIDS on a remote system.

TSUT: Temporary storage unit table. The TSUT contains an entry for each temporary storage identifier. Each entry addresses either the temporary storage record in main or in auxiliary storage, or, in the case of a temporary storage queue, the TSGID.

TS use map: Temporary storage use map. The TS use map is addressed by CSATSATA and it contains temporary storage statistical and control information, and a control interval map. The map contains 1 byte for each control interval available to temporary storage, each byte containing a value that is the number of free segments in the corresponding control interval. A segment is the unit of space allocated by temporary storage management and is 64 bytes when the control interval size is 16K bytes or less. Otherwise it is 128 bytes.

TSVCA: Temporary storage VSWA control area. There is one TSVCA for each VSAM work area (VSWA). TSVCAs are allocated by the initialization program, DFHSIH1, and chained from TSAVCAHD in the temporary storage auxiliary control area (TSACA).

TTP: Terminal type parameter. A TTP is created by DFHRLR and is used to control message building and disposal for each terminal type. If the message is not to be routed a direct TTP is built for the originating terminal and its address is placed in OSPDTTP (OSPWA). If the message is to be routed, a routing TTP for each type of terminal is built (following the user's route list) and the address of the first routing TTP in the chain of TTPs is placed in OSPTTP (OSPWA).

TUL: Tape user label. TUL defines the format of user header and trailer labels for data sets in CICS standard-labeled tape journals. They are usually created and interpreted by DFHVCP, exit routines in DFHJCOCP, or offline utilities.

UD: DL/I updated data base element. UD is the update data base element. This element is created by DFHDLBP for each data base that is backed out for a given PSB/PST pair. If a backout failure occurs, then the chain of elements is used to stop the identified data bases.

UET: User exit table. UET consists of a header, and entries in contiguous storage for each exit. UET is pointed to by CSAUETBA.

UIB: DL/I user interface block. The UIB is used by the interface between CICS and DL/I to pass information back to the application program. It is obtained by the interface when a DL/I scheduling call is executed and the UIB address is returned in one of the arguments to the scheduling call. The UIB contains the address of the PCB address list obtained by the scheduling call and, for every DL/I request, the response from the CICS-DL/I interface.

URD: Unit of recovery descriptor. This control block holds the state of the logical unit of work when CICS uses an external resource manager. The URD is chained off the CSA, and survives any failure of either system.

URL: User route list. A URL is a list of terminals to which a routed message is to be sent by BMS. Each entry in the list contains the terminal identification, any necessary logical device code or operator identification, and a status flag.

VOLDS: Volume manager area. VOLDS maps the volume descriptor, which is the representation of a single volume in a series. The series is anchored in the series definition table (SDT).

VSCA: VSAM subtask control area. VSCA holds data used by the VSAM subtask program DFHVSP.

VSWA: VSAM work area. A VSWA is acquired by the file control program when accessing a VSAM data set. At label VSWARPL is the VSAM request parameter list (RPL) for the request.

XFIOA: Transformer input/output area. The XFIOA is created by the fast-path transformer program, DFHXFX, and contains information describing the function-shipped request or reply. XFRATIOA in the CICS function request shipping request control block (XFR) points to this area.

XFSTG: Transformer storage for CICS function request shipping. An XFSTG area is used to store information for DFHXFP and DFHISP between transformations 1 and 4 (CICS function request shipping originating system) and for DFHXFP and DFHMIR between transformations 2 and 3 (in CICS function request shipping mirror system). Information stored includes the name of the remote system, the address of the session TCTTE, and the address of the DFHEIP parameter list.

XLT: Transaction list table. The XLT defines a group of transaction IDs. An XLT suffix can be passed on a master terminal command. The requested operation is then performed on each of the transaction identifiers in the XLT (for example, they are all disabled). An XLT can be used at CICS shutdown time to restrict the transaction IDs that can be used during the first stage of shutdown.

XTSTG: Transformer storage for transaction routing. XTSTG is used to pass parameters between DFHZTSP and DFHXTP, and between DFHTPS and DFHXTP for transaction routing.

ZCQ: Builder parameter set. A ZCQ is used when creating a terminal control table resource dynamically, for example, using resource definition online (RDO). It is allocated by the RDO front end, by DFHZATD, or by DFHZCQ. This area describes the properties of a terminal.

ZEPD: ZCP Module list table. ZEPD contains the addresses of routines contained in DFHTCP, DFHZCA, DFHZCB, DFHZCX, DFHZCY, and DFHZCZ.

CSECT Name	DSECT Name	TCT Label	Description
DFHTCT	DFHTCTFX IFGACB DFHTCRPA	DFHTCTBA TCTVRSAS TCTVACB TCTVXLST TCTVRAPA	TCT prefix for VTAM: Register save areas VTAM ACB VTAM exit list VTAM receive any pool (ECBs and RPL addresses)
DFHZNIB	TCTENIB	– TCTVDNIB TCTVDRPL TCTNIBLA TCTBNDLA	NIB descriptor blocks (1 per VTAM TCTTE) Dummy NIB Dummy RPL List of NIB models List of BIND models
DFHZEXI	–	Various	VTAM exit routine bootstraps
DFHZMODL	–	–	COMMON module list, VTAM module list
DFHZATTB	–	ZATHTNTB ZATHPGTB ZATHAITB	Transaction code delimiter table Page command table AID trancode table
DFHTDCB	–	–	Data control blocks
DFHDECB	DFHTCTLE	–	Line entries (TCTLEs)
DFHDFTM	–	–	BTAM terminal lists
DFHTERM1	DFHTCTTE	– TCTVDTE	Terminal entries (TCTTEs) non-VTAM and dummy TCTTE
DFHPCCE	DFHTCTCE	–	Pool CCEs
DFHZRPL1	IFGRPL	–	Static RPL
DFHZRPLS	IFGRPL	–	Static RPL
DFHUCTRT	–	TCZUCTAB	Uppercase translate table
DFHZBIND	ISTDBIND	–	VTAM bind area models and model NIBs
DFHZSLDC	DFHSLDC	–	System LDC table
DFHZLLDC	DFHLLDC	–	Local LDC table
DFHSUFX	–	TCTUSAA	User work area (specified by TCTUAL operand of DFHTCTTE macro)
		TCTVWL TCTWLCLE TCTWLA1	ECB list: Timer ECB address VTAM ECB addresses Communication list ECB Console ECB addresses Non-VTAM wait list

Figure 68. Terminal control table storage layout

Chapter 3.5. CICS in a wait state

Mechanism of the CICS dispatcher

The CICS dispatcher (in DFHKCP) is always given control when DFHKC TYPE=WAIT or DFHKC TYPE=SUSPEND is issued, and is sometimes given control when other task control services are requested. It scans the DCAs on the active chain in priority order and will dispatch the first one that can execute. This task then takes control until it issues another task control service request. The effect of each request on the issuing task is outlined below.

- DFHKC TYPE=WAIT,DCI=TERMINAL (TCATCDC=X'13'): The task is transferred to the suspend chain.
- DFHKC TYPE=SUSPEND: The task is transferred to the suspend chain.
- DFHKC TYPE=ENQ: If it is available, the resource is allocated to the task without a dispatcher scan, but otherwise the task is transferred to the suspend chain.
- Other: The task is left on the active chain; there may or may not be a task switch.

Thus, a task is transferred to the suspend chain only when the task itself issues DFHKC TYPE=WAIT,DCI=TERMINAL, SUSPEND or ENQ. Other tasks cannot cause it to be suspended. Once a task is suspended, it cannot execute until some other (active) task issues a DFHKC TYPE=RESUME specifying its TCA. If the suspension was because of DFHKC TYPE=WAIT,DCI=TERMINAL, it will be the terminal control program that resumes it (the address of the TCA comes from TCTTECA in the TCTTE). If the suspension is because of DFHKC TYPE=ENQ, the DFHKC TYPE=RESUME macro is issued by DFHKCP running under the TCA of the task that is relinquishing the resource. DFHKCP determines the DCA of the suspended task by following the wait chain anchored in the QEA. If the suspension is because of a DFHKC TYPE=SUSPEND macro, the RESUME, which can be issued by any other task, is that described in "Suspended tasks" on page 312.

Purging tasks

Tasks that stall (that is, they are held indefinitely in a WAIT or on the suspend chain) can be eliminated from the system in the following ways:

Stall purge
If the CICS dispatcher finds that no tasks can be dispatched, and either the short-on-storage (SOS), or the maximum task (MXT operand in DFHSIT), or maximum active task (AMXT operand in DFHSIT) limit has been reached, it sets the stall indicator **on**. The stall indicator gets set **off** if a task other than the CICS terminal control task is dispatched.

If, after the region exit interval has expired (ICV operand in DFHSIT), the stall indicator is still **on**, the dispatcher determines the type of stall. If it is due to the AMXT limit being reached, the dispatcher temporarily raises the AMXT value to its maximum possible value of 999, thus removing a potential cause of the stall. If it is due to an SOS or MXT condition, then a task will be purged if possible, but only after the stall time interval has expired (ICVS operand in DFHSIT).

This purged task will be the lowest priority suspended task, if there is one, or the lowest priority active task otherwise. A task is only purged in this way if specified by CEDA DEFINE TRANSACTION SPURGE(YES); see the *Resource Definition (Online)* manual.

Time-out purge
When a task issues a terminal input request it is only allowed to stay in the system as long as the time-out value specified by CEDA DEFINE PROFILE RTIMOUT(value); see the *Resource Definition (Online)* manual. If this time expires before the READ completes, DFHKCP abends the task with abend code AKCT, but no dump is produced.

Deadlock time-out
If a value is specified by CEDA DEFINE TRANSACTION DTIMOUT(value) for the deadlock time-out option, and the task is suspended (other than for terminal input) for longer than the time specified, the transaction is abended, provided the task does not hold a critical resource.

CEMT cancel
Any task can be removed from the system by the master terminal CEMT transaction cancel.

In certain BTAM situations, you need to take further action to complete the purging of the task. If a task is waiting for a read from a BTAM nonlocal terminal to complete and that task is canceled, it may require input from another terminal on the same line before the cancel operation completes.

Runaway purge
This is different from the preceding purges, which can only happen when a task is waiting or suspended. Runaway purge happens at the end of the current CICS service if a task is dispatched and does not relinquish control to the CICS dispatcher for the time interval specified on the ICVR operand of the DFHSIT macro, provided the task does not execute a DFHKC TYPE=WAIT macro. (See the *Resource Definition (Macro)* manual for a description of the SIT.)

Waits issued by modules of CICS

User tasks can issue waits directly and it may be necessary to obtain the application program listings in order to understand the reason for these. These will normally specify DCI = SINGLE. However, DCI = SINGLE can lead to the redundant inclusion of the ECB in the wait list if the ECB is actually to be posted by another user task rather than on completion of I/O. If this happens, there is the possibility of an OS/VS 301 abend if two tasks wait on the same ECB.

To prevent this problem, use DCI = CICS, but ensure that the ECB resides in permanent storage. If the task unexpectedly abends and the ECB storage is freed, the post in another task can lead to unpredictable results.

Although user tasks can issue waits directly, it is more common for waits to be issued on their behalf by CICS service modules. Most of these waits and the methods by which they are satisfied are described as follows.

DL/I waits

DCI = CICS,ECADDR = DLPSBECB after labels DLPDWAIT and DLPWAITI in DFHDLI. These waits occur on schedule (PCB call) requests. The first is an optimization that delays the IMS/VS SCHEDULE request if the PSB is in use by another task, and intent scheduling has been requested (PISCHD = NO specified in the SIT). The second follows an unsuccessful IMS/VS SCHEDULE attempt (because of intent conflict or lack of storage). In both cases the ECB is posted on any TERMINATE request that will happen at least on the sync point or end of task of the task that is using the conflicting PSB. See the code after label DLTUNLBP in DFHDLI.

DCI = CICS,ECADDR = DLSCHECB after label DLPDIROK in DFHDLI. During a SCHEDULE call, the ECB is set to X'80'; this causes other tasks requesting a SCHEDULE to wait. The ECB is also set to X'80' during a crucial phase of termination. When the scheduling is complete, the ECB is set to X'40'. It can be confirmed that a task is locked in scheduling code by inspection of field ISBINDS (ISB + X'1B'). Its value will be X'80'. There is a use count, kept at DLSCHCNT in DFHDLI, of the number of tasks in the scheduling logic.

DCI = CICS,ECADDR = DLISBECD after label DLGETISB in DFHDLI. If there is no available ISB for a schedule request, it is necessary to wait for an ISB to be freed by a termination request. The number of ISBs is specified by the DLTHRED operand of the DFHSIT macro. The ECB is posted by the code at label DLFREISB. There is an implied termination request when a task reaches a sync point or task end.

DCI = CICS,ECADDR = DLDBDECB after label MTD355 in DFHDLX. This processing is performed on receipt of a CLSE, DUMP or RCVR request. During this time, the flag DDIRNOSC is set to prevent schedule requests against the data base being processed. Before the CLSE can proceed, it is necessary to wait until all current schedules involving the data base have completed. The wait is posted on each termination after label DLTUNLBP in DFHDLI, and the WAIT is reissued until the active PSB count becomes zero.

DCI = CICS,ECADDR = ISBPSECB after label DLGBISBF in DFHDLR. When IMS/VS requires storage for data base or DL/I control blocks, control is passed to

routine IGETBUF in DFHDLR so that a CICS conditional GETMAIN can be issued. If the GETMAIN fails for lack of storage, the above WAIT is issued on field PSECB in the appropriate ISB. When IMS/VS frees buffer storage, it invokes routine IFREEBUF in DFHDLI, and if any tasks are waiting for buffer storage they will be posted by the code at label DLFBLP. During the time that any tasks are waiting for buffer storage field DLPISECB is set to X'40'. This is not an ECB because it is never waited for; its purpose is merely to allow the IFREEBUF routine to perform a quick test for waiting tasks.

DCI = SINGLE after label IWAIT in DFHDLR. This is the only wait for I/O in the CICS-DL/I interface. The ECB will be posted by the operating system.

DCI = CICS,ECADDR = DLPPSECB after label DLPWAITS in DFHDLI. This wait is on the PSB pool space.

DCI = CICS,ECADDR = DLPSTECB in DFHDLI and DFHDLR. This wait stops all DL/I calls during immediate shutdown, so that the UNLD call, issued by the QUIESDLI routine in DFHDLX, can be issued with the guarantee that no other DL/I activity is taking place. The DFHDLI invocation of the DFHKC TYPE = WAIT macro stops the DL/I calls as they are issued. The DFHDLR invocation stops them immediately after they have been posted by the simulated IWAIT routine.

DCI = CICS,ECADDR = DLPDAECB in DFHDLX. This wait, issued during normal shutdown, ensures that the dynamic allocation task, attached by IMS/VS, has terminated. The subtask termination exit, DYNALEXT in DFHDLR, posts this wait.

DCI = CICS,ECADDR = DLPRCECB in DFHDLX. This wait, issued during normal shutdown, ensures that the DBRC subtask, attached by IMS/VS, has terminated. The subtask termination exit, DBRCEXTR in DFHDLR, posts this wait.

DCI = CICS,ECADDR = DLPSDECB in DFHDLX. This wait ensures that DBRC services are not used until DBRC sign-on has completed. In the case of an alternate system, this does not occur until after takeover.

DFHDLG issues a DFHKC TYPE = WAIT macro when it determines that its work queue is empty. When more work arrives, the appropriate IRLM exits (DFSCMDX0 or DFSNOTI0) post the ECB in the IMS/VS system contents directory (SCD).

DFHDLS issues a DFHKC TYPE = WAIT macro when it determines that its work queue is empty. When more work arrives, the appropriate IRLM exit (DFSSTAX0), running under an SRB, posts the ECB in the common services PST.

See also "Debugging" on page 350 for how to analyze a CICS-DL/I problem.

Task control waits

DFHKCQ issues a **DFHKC TYPE = WAIT** to wait for completion of the initialization program DFHKCRP. The ECB is KCSCPECB in KC static storage, and it is posted by DFHKCRP.

DFHKCQ issues a **DFHKC TYPE = ENQ** to single-thread updates to PCT definitions in the catalog. ENQ is on the address of KC static storage, and the DEQ is issued by DFHKCQ at the end of DWE processing.

Journal control waits

DCI = CICS,ECADDR = JCTBAECB after label JC354 in DFHJCP. There are two main possibilities:

- This is a journal CLOSE request from a user task. The ECB JCTBAECB (buffer available ECB) is posted complete when a journal task starts (label JC269 in DFHJCP), or whenever I/O is initiated and data is shifted up in the buffer (label JC261 in DFHJCP). When the WAIT is satisfied, it is repeated if necessary until all three bits JCTJSIOE, JCTJSC, and JCTJSXC are off.

- There is no space in the buffer to satisfy a journal output request. The task must await the completion of some I/O so that data can once again be accumulated at the start of the buffer area.

DCI = CICS,ECADDR = JCTBAECB after label JTDETACH in DFHJCSDJ. For each journal, the journal shutdown program issues DFHJC TYPE = CLOSE, which causes the flag JCTJSNO to be set in the JCT for that journal, and issues DFHKC TYPE = RESUME for the CICS subtask for the journal. The effect of these two actions is to cause the ECB JCTBAECB to be posted (after label JC275 in DFHJCP) and the journal task to terminate itself. JCTBAECB is also posted by DFHJCC, DFHJCEOV, DFHJCIOE, DFHJCKOJ, DFHJCO, and DFHJCP.

The ECBs, **JCOCRECB and JCOCDECB**, are used to synchronize operation of the open/close subtask, program DFHJCOCP, with journaling routines that use it, but which themselves run under CICS tasks.

While DFHJCOCP is idle, it waits for the request ECB, JCORECB, near label JC205. JCOCRECB is posted by modules expecting some service, for example:

DFHJCO after JC903, to open a data set.
DFHJCC after JC105, to close a data set.
DFHJSDJ after OSTEND, to terminate the operating system task.

Conversely, when DFHJCOCP finishes a requested operation, it posts JCOCDECB, the request-done ECB. Each of the requesting modules waits on this ECB, after posting JCORECB, to initiate a service. These two ECBs, and JCOCAECB, the subtask-termination ECB, are also used during the startup and closedown phases.

DCI = SINGLE,ECADDR = MYDECB after label READ in DFHJCO. Wait for completion of a direct-access READ operation. This I/O is required in the course a search on a disk data set to find the last record written to the journal during the previous run.

DCI = CICS after label JC256 in DFHJCP. The ECB is an LECB which contains a use count. Wait user task after a PUT or WAIT request. Will be posted by DFHJCP when block has been output (after label JC286). The LECB address is kept in JCTCILA − if no LECB is available.

DCI = IOEVENT after label JC280A in DFHJCP. The DECB address is in JCTICA. Wait for I/O on the journal to complete. It is the journal task performing this WAIT, not a user task.

DCI = CICS,ECADDR = JCTBFECB after label JC362 in DFHJCP. A journal OPEN or CLOSE request has been made. If there are records in the buffer still to be output, the bit JCTJSC is turned on, I/O is initiated and the user task is made to wait for the journal task to write the buffer. The journal task will suspend itself while writing the buffer rather than continue I/O. Other tasks must await completion of the status change. After the buffer has been written by the journal task, the ECB, JCTBFECB, is posted after label JC386 in DFHJCP.

Note: Status change may also be requested by the journal task itself, in which case JCTJSEOV is set instead of JCTJSC.

Program control waits

One of the functions of DFHPCP is to load modules, which it does in response to DFHPC TYPE = LINK, LOAD or XCTL requests. Because a LOAD macro would hold up the whole CICS region, and could not use preallocated CICS storage, DFHPCP loads modules using BSAM READ requests. The WAITs in DFHPCP are as follows:

DCI = CICS, after labels PCBLDLR, PCSIZEOK, PCPFTWL, PCSPBYP1 and PCINVBLD. This WAIT is for the loader ECB, which is located at label PCSENQIN in DFHPCP static storage. The ECB is set to X'00' so long as the loader is in use. The address of the TCA currently using the loader can be found at PCENQIN + 1. (Thus the ECB is used as a LOCK word, though DFHKC TYPE = LOCK or UNLOCK is not used.)

The ECB is set as follows:

- After label PCLDFRB for an abnormal task termination
- After label PCBLDLXT after a BLDL
- After label PCCHAPB after a successful LOAD.

DCI = CICS after label WAITOFB in DFHPCP. This wait is for completion of recovery prior to searching the PPT for the entry corresponding to the requested program. This wait uses PCSOBECB in the program control static storage as an ECB. The ECB is posted before returning from DFHPCRP on completion of program control recovery during startup.

DFHPCQ issues a **DFHKC TYPE = WAIT** to wait for completion of the initialization program DFHPCRP. The ECB is PCSCPECB in PC static storage, and it is posted by DFHPCRP.

DFHPCQ issues a **DFHKC TYPE = ENQ** to single-thread updates to PPT definitions in the catalog. ENQ is on the address of PC static storage, and the DEQ is issued by DFHPCQ at the end of DWE processing.

Temporary storage waits

The following are temporary storage waits:

Wait for buffer unlock. If a temporary storage request requires a buffer, but all buffers are locked, a wait is issued on the ECB TSABUECB (at offset X'50' in DFHTSACA). This ECB is posted whenever a buffer is unlocked.

Wait for a write buffer unlock. If a temporary storage request requires a write buffer, but all write buffers are locked, a wait is issued on the ECB TSABUEC (at offset X'68' in DFHTSACA). This ECB is posted whenever a write buffer is unlocked.

Wait for a string. If a temporary storage request requires a VSAM string, but all strings are in use, a wait is issued on the ECB TSAVUECB (at offset X'54' in DFHTSACA). This ECB is posted whenever a string becomes free.

Wait for a data-set extension to complete. If the temporary storage data set becomes full, an attempt is made to extend the data set by writing a new record at the end. This operation is single-threaded, and controlled by the ECB TSAEXECB (at offset X'58' in DFHTSACA). The ECB is cleared when extension is in progress, and posted when it is complete.

Wait for keypointing to complete. While temporary storage activity keypointing is in progress, all temporary storage activity is quiesced using the ECB TSMTQLB (at offset X'34' in DFHTSMAP). The ECB is cleared when keypointing begins, and posted when it is complete. The ECB is checked each time the temporary storage unit table (TSUT) is scanned.

Wait for a queue to be unlocked. Operations on a temporary storage queue are single-threaded. There is an ECB in the request element for the queue (at offset X'10' in DFHTSRE). If a request is already in progress, it is necessary to wait for this ECB. The ECB is posted when the request is complete.

Wait for I/O to complete. When an I/O request is made to VSAM, a wait is issued on an ECB whose address is passed to VSAM (at offset X'08' in DFHTSVCA).

Wait for subtasking I/O to complete. If the VSAM/BSAM subtask is being used, a wait is issued on the ECB VSWAECB (at offset X'70' in DFHVSWA). The VSWA is addressed by TSVSWAP (at offset X'10' in DFHTSVCA).

Terminal control waits

DFHTOR issues a **DFHKC TYPE = ENQ** to single-thread updates to model terminal and TYPETERM definitions in the catalog. ENQ is on a resource whose name is 'DFHTOR:tt:mm', and the DEQ is issued by DFHTOR at the end of DWE processing.

Parts of the terminal control program run under the user TCA, and other parts run under the special TCA for terminal control. When a task has run away, the user's TCA is referenced.

For non-LU6.2 transmissions, when an application program issues a DFHTC TYPE=READ or WRITE request (possibly using BMS), code in DFHZARQ (part of DFHZCP) is executed under the user's TCA. This code turns one or more bits on in the TCTTE to indicate to the terminal control program that I/O is to be initiated. If WAIT was requested in the READ or WRITE macro, or DFHTC TYPE=WAIT is issued, DFHZARQ issues DFHKC TYPE=WAIT, DCI=TERMINAL which places the task on the suspend chain. A RESUME is issued by the terminal control program running under its own TCA when the I/O completes.

For LU6.2 transmissions, the request goes to DFHZARQ, DFHZARQ passes the request, through DFHZARM, to DFHZARL which handles I/O initiation and I/O waits.

DFHZEMW also issues **DFHKC TYPE=WAIT,DCI=TERMINAL** to suspend a task during abend processing. It is invoked to clean up the conversation state of VTAM sessions. It does this by calling DFHZCP functions directly, without involving DFHZARQ, and so uses the direct form of DFHKC TYPE=WAIT, rather than a call to DFHZARQ.

The TCA for the terminal control program task itself is always on the active chain and issues DFHKC TYPE=WAIT,DCI=LIST after label TCZDSP34 in DFHZDSP. The ECB list consists of:

- The timer CSATTECB
- All VTAM RECEIVE ANY ECBs
- System communication ECBs
- All operating system console ECBs
- All BTAM line ECBs.

When I/O completes on a BTAM line the terminal control task will be dispatched by task control like any other task and performs the actions associated with the I/O that has just completed (that is, continue I/O, resume a task, attach a new task, process error, or ignore). In addition, DFHTCP scans all the terminals on that line to see if there are other I/O requests pending. It does not scan terminals on other lines.

When the timer ECB is posted, DFHTCP scans the whole TCT looking for terminal I/O requests. The timer ECB is set by DFHKCP if the region exit time interval (ICV, set in the SIT), has elapsed since the last full scan, or if a DFHTC TYPE=WAIT has been issued and the terminal scan delay time (ICVTSD) has expired since then.

The following applies to local terminals only. DFHTCP will always be dispatched on completion of I/O and therefore the ICV should be made irrelevant by being set to a large value. The ICVTSD time interval is to prevent too many scans of the TCT on high activity systems. On low activity systems the scan delay should be set to zero to improve response times.

Interregion communication waits

CICS Address Space, SLCB ECB. DFHKCP adds this ECB to the external wait list if IRC is active. The ECB is posted by the IRC SVC routine.

DCI = CICS,ECADDR = SCCBDECB (DFHZCX). This ECB represents activity on a given interregion communication connection to CICS. The IORENT routine in DFHZCX waits on this ECB for data from the connected-to address space. The ECB is posted by the interregion SVC routine.

IMS/VS Batch Region, SLCB ECB. This ECB is posted by the interregion SVC routine when the batch region has any input from CICS. It is waited upon by the converse subroutine in DFHDRPE.

Dump control waits

The formatted dump program, which may get control after a task abend, does not perform any CICS requests and therefore no task switches can occur while this is in progress. The ECBs for the other tasks in the system may be posted (particularly if DCI = ECB or DCI = SINGLE), but the tasks cannot run until the task control dispatcher gets control after formatted dump has finished.

The transaction dump program, DFHDCP, does, however, allow task switches to occur while a transaction dump is in progress. This is because DFHKC TYPE = WAIT,DCI = ECB is issued while waiting for I/O on the dump data set.

The LOCK performed at the start of DFHDCP ensures that only one task dump can be in progress at any one time. The ECB used by the LOCK, DCDTFI, is also waited for by DFHOCP when a request is made to close the dump data set.

Transient data waits

Most of the waits issued in DFHTDP are DCI = ECB and are posted by OS/VS when the I/O is complete. The ECB is the first field in an OS/VS DECB which starts at TDIADECB in a TDIA.

Other waits issued by DFHTDP are as follows:

DCI = ECB,ECADDR = RDSEQECB after label TDCALTTR. Wait after writing capacity record after acquiring a new track. The ECB opcode is X'04', meaning WRITE SZ.

DCI = CICS,ECADDR = TDDCTFLB after label TDGLRAC. Normally a **recoverable** queue can only be in use by one task at a time that is, once a task has accessed a queue, no other task can use the queue until the first one terminates (or reaches a sync point). An exception is made, however, to allow GETs from a queue that is being output to by another task, provided the record being obtained was on the queue when the other task started. When a task issues a GET for a record that was placed on the queue by a currently active task, the above wait occurs until the other task terminates. The ECB is posted complete after label TDDWPRL in DFHTDP DWE processing.

TYPE = LOCK,RSOURCE = TDDCTDEL at label TDPLOCK. This LOCK will result in a WAIT if the destination is in use by a transient data request in another task. The

resulting WAIT is DCI = CICS,ECB = TDDCTDEL. TDDCTDEL is in the DCT entry. The WAIT is satisfied when the other task completes its TD request.

The TCA address is not stored in the lock word, but the task holding the lock can be determined either by:

- Using the trace table and searching for use of the destination by a task other than the current one.
- Scanning the TCA chains for a TCA with TCALCKAD = TDDCTDEL.

OPEN/CLOSE waits

If a request is made to close the dump data set, a test is made to see if a dump is currently in progress. If so, DFHOCP awaits completion of the dump before the CLOSE is issued. The ECB is DCDTFI and is located near the start of DFHDCP. It is posted complete by the UNLOCK at DCPEXIT in DFHDCP.

No other WAITs are issued by DFHOCP.

File control waits

If the DCI is X'80' (DCI = ECB or DCI = SINGLE), file control is waiting for BDAM or VSAM I/O. For BDAM, the ECB is is located at offset X'08' in the FIOA. For VSAM, the ECB is located at offset X'10' in a VSWA. The RPL starts at offset X'08' in the VSWA.

VSAM strings

The number of concurrent accesses permitted on a VSAM file is specified by the option STRNO on the FCT. For this purpose, the duration of an access is:

```
GET read only      - until the GET is complete if
                     MODE=MOVE on the DFHFC macro.
                   - until RELEASE if MODE=LOCATE.

GET for update     - until PUT update, PUT DELETE or
                     RELEASE.

SETL GETNEXT       - duration of the browse, that is,
 and so on           until ESETL.

PUT                - until the PUT is complete.

PUT (mass insert) - duration of the mass insert, that is,
                     from GETAREA to RELEASE.
```

If a CICS file control VSAM request is made that would exceed the string count, the task is made to wait until a string is released by another task (for example, by a RELEASE) before the request to VSAM is made. This wait is on the following ECBs:

A read-only request causes the wait
DFHKC TYPE = WAIT,DCI = CICS,ECADDR = FCTDSMSW to be issued if all the strings for the file are in use. The ECB is marked **not-posted** whenever a string is

acquired that brings the number in use to STRNO. This ECB is posted whenever a string is released, for example, by a PUT UPDATE for the file.

A non-read-only request causes the wait
DFHKC TYPE = WAIT,DCI = CICS,ECADDR = FCTDSPSW to be issued if 80% of all the strings for the file are in use. To permit read-only requests at almost any time, requests that are not read-only are not permitted to use more strings than the pseudomaximum string number, which is 80% of STRNO. This is because non-read-only requests take much longer. The ECB, FCTDSPSW, is incomplete while the number of strings for the file currently active is equal to or greater than FCTDSPMS.

If the FCT entry did not specify LSRPOOL = NONE, the VSAM file shares resources (strings and buffers) with files that specify the same share pool number. A VSAM request for a file using shared resources causes the wait
DFHKC TYPE = WAIT,DCI = CICS,ECADDR = FCTSRSWE to be issued if the number of active strings for this pool, FCTSRHAS, is greater than the computed maximum string number, FCTSRSTN, for this pool. The ECB, FCTSRSWE, is posted complete whenever a shared string for the relevant share pool is released.

For the MVS/XA Data Facility Product (MVS/XA DFP), DFHDLI uses LSR pool 0, and CICS uses pools 1 through 8. The number of strings for use by DFHFCP is the number of strings passed to VSAM minus the value of DLITHRED. Thus there are always VSAM strings available for DFHDLI, which does not post the **string-available** ECB on completion of the DFHDLI request.

Debugging VSAM file control waits

Waiting For Strings. If tasks are held up for lack of strings, it is necessary to determine which tasks are holding strings for that file and find out why they cannot run. One way to do this is to follow task storage on both the active and suspend DCA chains, looking for VSWAs. Each VSWA represents one string. If the number of VSWAs referencing the file is not equal to the active string count, FCTDSASC, CICS is in error because strings are not being released when they should be.

Waiting For Exclusive Control. A request to VSAM can fail because an outstanding request may be trying to update the same control interval, or causing the split of a control area containing the required control interval. This is called **exclusive-control-conflict**. CICS makes such tasks wait for the resource by chaining the VSWA to the exclusive-control-conflict tree, and performing a CICS wait on the ECB, VSWAEXW. This chain is anchored in FCTBCVSC in the base cluster control block (BCCB). This ECB is posted complete when the task holding exclusive control releases the string. See "CICS record locking" on page 362 for a discussion of exclusive control.

Waiting For Buffers. A request to VSAM using shared resources can fail because of lack of buffers. In this case, the FCT entry is added to a chain anchored in FCTSRBWC, and CICS waits for the task using the ECB, FCTDSBWE. This ECB is posted when a shared string is released:

- On completion of I/O for a read-only GET
- On a PUT UPDATE, PUT DELETE, or RELEASE following a GET UPDATE
- On completion of a full browse or mass-insert operation.

CICS Enqueue. If a task makes a request to update a record in a protected file (LOG = YES in the FCT), a CICS enqueue is performed using the queue name constructed by concatenating the data set name and the record identifier (which can be a key, a relative block address (RBA), or a relative record number). This enqueue can result in the task being placed in a wait state on the suspend chain.

Waits associated with file state changes

Synchronous Disable. If a file is to be disabled synchronously, using the EXEC CICS SET command with the WAIT option, and the file is in use, it is necessary to wait until all the users have finished. The ECB, FCTDIECB, is posted by the last user when it finishes, and, until then, no new users are allowed.

The users of a file can be found by following the chain of deferred work elements (DWEs) anchored in FCTDSDWE. The chain field in the DWE is DWEFCCHN. Only those DWEs without the **cancel** bit **on** (DWECNLM in DWESTAT) represent TCAs currently using the file. The address of the TCA is in DWEMYTCA, and the number of users is held in FCTDWEUC.

Synchronous Close. If a file is closed synchronously, using the EXEC CICS SET command with the WAIT or FORCE option, it is necessary to wait until all users of the file have finished; the FORCE option that abends those users must wait. When the last user of the file finishes, CICS closes the file and posts the ECB, FCTCLECB. The current users of the file can be found as described in the section "Synchronous Disable" on page 308.

Note: Although close implies disable, the disable ECB, FCTDIECB, is not used for this disable.

Concurrent Requests For Open, Close. If a request to open or close a file is made while the file is being opened or closed, it is necessary to wait until the current operation completes before servicing the new request. The ECB made to wait is FCTOPECB for the open, or FCTCLECB for the close.

Interval control waits

DFHICP never waits for an event, though it does give control to the dispatcher to allow a task switch when DFHIC TYPE = POST is issued. This is achieved by DFHKC TYPE = WAIT,DCI = DISP after label ICPOSTN in DFHICP.

The DFHKC TYPE = WAIT,DCI = DISP issued by DFHICP after label ICNOATT is not really a wait because task control's TCA is in control at this time. The macro acts as a return to DFHKCP (from where DFHICP has just been called).

System termination waits

DCI = CICS,ECADDR = CSASSI2 after label STNOJRN in DFHSTP. Prior to the above wait the flag CSASSI2, X'80' is turned on during CICS termination to prevent terminal control accepting further input. The wait is posted by DFHTCP when it has served the last output operation.

DCI = CICS after label STNOJRNL in DFHSTP. The ECB waited on is VSCATECB. This ECB is posted when the VSAM subtask has finished.

DCI = CICS after label STPNORM in DFHSTP. The ECB is JCOCDECB which is posted when the OS/VS subtask for opening or closing journal files terminates, whether normally or abnormally. The ECB is located as the first byte of the open/close parameter list in DFHJCP.

After label STWAITNT, DFHSTP waits for all the outstanding tasks to terminate (normal shutdown only) by a wait on the ECB, CSAKCNAC. DFHKCP posts this ECB after label KCPDELBP whenever a task terminates. DFHSTP reissues the wait so long as the count of outstanding tasks is greater than zero.

General purpose subtasking waits

The following waits are issued by the subtask management program (DFHSKP):

Wait for subtask initialization to complete: When subtask management decides that an operating system subtask should be started, it attaches a CICS task to execute DFHSKC, which performs this function. DFHSKM issues a DFHKC WAIT macro on an ECB in DFHSKP static storage. This is posted by DFHSKC when subtask initialization is completed.

Wait for a subtask to execute request: When DFHSKM has created a work queue element (SKW) which represents a piece of work to be performed by a subtask, it adds the SKW to the work queue for the chosen subtask. A DFHKC WAIT macro on an ECB in the SKW for the subtask is issued. The ECB is posted by DFHSKE when the SKW has been processed by the subtask.

Synchronous request execution: If the request cannot be executed by an operating system subtask (perhaps due to the subtask's failure), and the request is executed under the CICS task, the routine being executed may issue a DFHSK CTYPE = WAIT macro, which causes a DFHKC WAIT macro to be issued.

Subtask termination: When DFHSTP requests subtask termination (by issuing a DFHSK CTYPE = TERMINATE macro), DFHSKM issues a DFHKC WAIT macro on an ECB in the subtask control area (DFHSKA), which is posted by DFHSKC when the subtask has been detached.

Wait for subtask completion: When DFHSKC attaches an operating system subtask, it issues a wait on the completion ECB for the subtask, which resides in the subtask control area (DFHSKA) for the subtask being processed. This ECB is posted by the operating system when the subtask has terminated, normally or abnormally.

Master terminal waits

DCI = DISP in DFHEMA, DFHEMC-I. These are waits in the master terminal programs to allow task switching.

System spooling interface waits

The SPOOLREAD command completes successfully only if the issuing task has control of the system spooling interface (SSI) single-thread switch. This restriction is because JES can only process one SYSOUT data set for read for each MVS region, or user.

To prevent a low-priority CICS task withholding the single thread from a higher priority waiting task, a DFHKC TYPE=WAIT,DCI=DISP macro is issued at the end of SPOOLCLOSE processing (in the CLOSESUB routine in DFHSPST). This allows any waiting task to seize control of the single-thread switch before the current task has a chance to issue a SPOOLOPEN INPUT request, and lock the single-thread switch once more.

XRF: alternate systems

The takeover process involves several operations; before each operation can be started, one or more events must have completed, for example:

- Terminals with backup sessions can be switched while the active system is running, provided the CICS availability manager (CAVM) has initiated a takeover.
- Passively-shared data sets must not be opened until it is known that the active system has terminated.

  Note: This is the only way an alternate system can be sure that no more data will be written by an active system.
- Resource managers, such as transient data, temporary storage, and data base recovery control (DBRC), rely on the time-of-day clock providing them with a non-decreasing value to ensure the proper management of their resources. The alternate system must not restart a resource manager until the alternate time-of-day clock has been synchronized with the active time-of-day clock.

A system task that issued a takeover request to CAVM waits on the ECB WCSTCECB in the CAVM static control block (DFHWCGPS) until CAVM has decided to accept or reject the request. The DFHKC TYPE=WAIT,DCI=SINGLE requests are issued in DFHWSRTR. The CAVM TCB posts WCSTCECB in either DFHWSTKV (the normal case) or DFHWSSOF (CAVM failure).

The following ECBs each represent an event. The ECBs are located in the static storage for DFHXRP. The ECBs and the events are:

- XRSTIECB – the CAVM has initiated a takeover.
- XRSIAECB – the alternate system is now the incipient active system.
- XRSTCECB – the active system is known to have terminated.
- XRSRAECB – DFHRSD is available for use by DFHCC (catalog control) and DFHRC (recovery control) macros.
- XRSSSECB – the time-of-day clock is synchronized with active system sign off.
- XRSSTECB – the time-of-day clock is synchronized with active system termination.

XRSTIECB

This ECB is posted by DFHXRA, following a successful call to the CAVM to initiate takeover. Once the ECB has been posted, DFHXRA attaches a system transaction to initiate the switch of terminals with backup sessions. DFHXRA is called from either the surveillance task (DFHXRSP), or the console communication task (DFHXRCP). No tasks wait for XRSTIECB to be posted.

XRSIAECB

This ECB is posted by DFHXRA, following notification by the CAVM that an alternate system is now the incipient active system. DFHXRA is called from the surveillance task (DFHXRSP). No tasks wait for XRSIAECB to be posted.

XRSTCECB

This ECB is posted by DFHXRA, following notification by the CAVM that an active system has terminated. There can be a delay in posting the ECB if:

- An SDUMP is being taken as part of the active system termination process.
- The active CEC has failed, and the operator failed to reply to the messages sent to the console. This applies to the 2-CEC environment.

DFHXRA is called from the surveillance task (DFHXRSP). Only one task, the system initialization task (DFHSII1), waits for XRSTCECB to be posted. When the ECB is posted, DFHSII1 opens the restart data set, for DFHRC use as well as for DFHCC use, and then calls DFHXRA to post the XRSRAECB.

XRSRAECB

This ECB is posted by DFHXRA once the restart data set has been opened, for DFHRC use as well as for DFHCC use. DFHXRA is called from the system initialization task (DFHSII1). Two tasks wait for XRSRAECB to be posted:

- The transient data recovery task (DFHTDRP) initializes the entry for the CXRF queue before waiting for XRSRAECB to be posted. When the ECB is posted, DFHTDRP resumes emergency restart processing.
- The terminal control recovery task (DFHTCRP) drains its tracking queue before waiting for XRSRAECB to be posted. When the ECB is posted, DFHTCRP resumes emergency restart processing.

XRSSSECB

This ECB is posted by DFHXRA following notification by the CAVM that the time-of-day clock is synchronized with active sign off. DFHXRA is called from the surveillance task (DFHXRSP). No tasks wait for XRSSSECB to be posted.

XRSSTECB

This ECB is posted by DFHXRA, following notification by the CAVM that the time-of-day clock is synchronized with respect to active system termination. There may be a delay in posting the ECB if the time indicated by the active time-of-day clock is significantly ahead of that indicated by the alternate time-of-day clock. DFHXRA is called from the surveillance task (DFHXRSP).

Only one task, the system initialization task (DFHSII1), waits for XRSSTECB to be posted. When the ECB is posted DFHSII1 links to DFHJRCP to restore journal states before attaching the remaining resource manager tasks.

Suspended tasks

A task starts initially on the active chain. It is transferred to the suspended chain only when it issues one of the following macros:

```
                        Value     Value
                        in        in
                        TCATCTR   TCATCDC
DFHKC TYPE=WAIT,        X'40'     X'13'
 DCI=TERMINAL
DFHKC TYPE=SUSPEND      X'04' or
                        X'84'
DFHKC TYPE=ENQ          X'01' or
                        X'05'
```

The register 14 value saved at TCATCRS or in the trace table gives the address of the macro, and hence the module that issued it. TCASVMID can also be used. See the section "How to find the reason for each wait or suspend" on page 36.

Terminal control suspend

The code in DFHZARQ that analyzes an application request and runs under the user's TCA suspends the task (by method 1 above) when it is waiting for terminal I/O. For local terminals, terminal control will be dispatched as a task when the I/O completes because an ECB is posted. For remote terminals, terminal control may not be dispatched to process the completed I/O until either the ICV time or the terminal scan delay has expired. In both cases, the terminal control task issues a RESUME for the task awaiting I/O.

Intercommunication facility suspends

DFHZISP handles ALLOCATE requests for an intercommunication facility session (TCTTE). If an ALLOCATE request cannot be satisfied immediately, and NOQUEUE has not been specified, the requesting task is suspended, and an AID (representing the request) is chained off the appropriate system entry. The AID is used to remember and, subsequently, to process the ALLOCATE request.

When a session becomes free, the freeing task issues a DFHKC TYPE=AVAIL macro to indicate that the session TCTTE is available. If a session fails, then a DFHKC TYPE=UNAVAIL macro is issued to cancel all the ALLOCATE requests in the queued AIDs.

DFHCRNP, which performs the work scan for interregion communication, issues the DFHKC TYPE=SUSPEND macro when its scan is complete, and it has no more work to do. When more work is available, it will be resumed by DFHKCP, under task CSNC.

Interval control suspend

SUSPEND is issued by the interval control program in response to a DFHIC TYPE=WAIT request. The RESUME is issued by the DFHICP expiry analysis code when the time expires, or when a DFHIC TYPE=CANCEL is issued. The ICE address on whose expiry the task is waiting may be found at TCAICEAD. A SUSPEND is issued by DFHICP in response to an EXEC CICS RETRIEVE WAIT command. A RESUME is issued by DFHALP when more data becomes available, or by DFHKCP if the task is subject to deadlock time-out and the time-out interval expires before more data becomes available, or by DFHSTP during CICS termination.

Journal control suspend

Each journaling task suspends itself when no further I/O on the journal is required for the time being. In response to a user journal request (that is, running under the user's TCA) DFHJCP issues RESUME for the appropriate journal task when it needs to initiate journal output. That is, when one of the following occurs:

- STARTIO=YES is specified on a journal request.
- A TYPE=PUT, (WRITE,WAIT), or WAIT request is issued, subject to the SYSWAIT option.
- The journal buffer (as far as the shift-up value) is full.
- The end of logical unit of work (LUW) is reached.

RESUME of the journal task is also performed:

- When the journal task must be terminated during shutdown processing. (After label JTDETACH in DFHJCSDJ.)
- When a journal is opened. (After label JC907 in DFHJCO.)

Storage control suspend (DCATCDC=X'18')

DFHKC TYPE=SUSPEND is issued by DFHSCP if it is unable to satisfy an unconditional request for storage. When another task issues FREEMAIN, DFHSCP scans the suspended DCA chain looking for tasks suspended for storage. Each request is retried until one is satisfied in which case that task is resumed. Each TCA with X'18' in its DCI will hold in TCASCNB the number of bytes it is waiting to become available.

Mirror task suspend (DCATCDC = X'1A')

The mirror program, DFHMIR, suspends itself on completion of a unit of work, for example a single inquiry READ, a complete browse or a READ UPDATE-REWRITE sequence. As IRC function-shipping requests arrive, the connection manager, DFHCRNP, resumes mirror tasks that were suspended in this way so that they continue processing. To avoid unnecessary tying up of storage, these tasks are automatically purged by DFHKCP if they have been suspended for more than 2 seconds.

Temporary storage suspend (DCATCDC = X'1C')

This occurs when a new track is required for an output request but none are available. RESUME is performed for all tasks suspended for TS whenever a PURGE or RELEASE request is made. All these tasks will then retry their requests.

Suspends as a result of ENQ

System and user resources are represented by queue element areas (QEAs). The QEAs are chained together from QCANXQEA, and the QCA is pointed to by CSAQCAA in the CSA. Each QEA has an owner DCA and possibly a chain of waiting DCAs (see Figure 26 on page 148). Some of the resources that can be held by a CICS task are listed below:

- An address in the FCT (FCTDSBCP). The ENQ is done by DFHFCP when a new record is added to a VSAM ESDS if new records have to be logged. In order to log the record before it is written it is necessary to predict the relative byte address (RBA) that VSAM will return. This predicted value is obtained from the FCT and is updated after each ADD is complete. This method only works if ADDs are prevented from running concurrently, hence the above ENQ. The DEQ is done on return from DFHFCP.

- A resource whose name is the concatenation of the address of the FCT entry and the record identifier (key, RBA, or RRN).

  This ENQ is performed by DFHFCP to prevent concurrent updates to the same record in a recoverable file.

- The address of label DFHFCCQ in module DFHFCN. This ENQ is used to single-thread file OPEN/CLOSE requests. The corresponding DEQ is issued at completion of the module.

- The address of the journal open/close parameter list. The format and length of the open/close parameter list are defined by DFHJCOCL. Requests to the open/close subtask are single-threaded. Hence the ENQs and DEQs as follows:

```
                          ENQ        DEQ

DFHJCO   after label JC902      JC906
DFHJCSDJ after label EOVAWAY    JTDETACH
DFHJCC   after label JC100      JC110
```

- A resource whose name is 'DFHMAINTOCP'. This ENQ is done at the beginning of DFHOCP and the DEQ at the end, because it is necessary to single thread the whole of the code in that module.

- A resource whose name is 'DFHAKPACT'. The ENQ is done at the beginning of DFHAKP and the DEQ at the end, because it is necessary to single thread the whole of the code of that module.

- A resource whose name is 'DFHTDINT.abcd' where abcd is a transient data destination ID. This ENQ is performed by DFHTDP to permit backout to take place for a GET or PUT request on a logically recoverable queue. The DEQ is done by DEQALL at LUW end.

- A resource whose name is the character string 'DFHTDTFR.NNNN'. DFHTDP single threads its code to acquire a new track. (A task switch could occur while the capacity record is being written).

- The resource address of CSASTSLF. The statistics program DFHSTSP is invoked either by the CSTT task entered from a terminal, or, on a regular basis, auto-initiated by ICE. Single threading the code is achieved by means of an ENQ on the above address. DEQ is done on return from STSP, though if control is XCTL to STKC only a DEQALL is done at task end.

- A resource whose name has the form '**allltp' where:

```
** indicates a BMS ENQ and may be replaced by a REQID operand on
     BMS request.
 a = X'FD'
lll = a unique 3-byte message identifier
  t = a printable character, representing a device type
     (refer to the DFHMSD programming note in the
     Resource Definition (Macro) manual).
  p = X'00' (indicating an MCR rather than a page)
```

  The resource name is the temporary storage ID of a BMS MCR. ENQs and DEQs are done by DFHTPQ and DFHTPR while the MCR is being updated, because pages are purged terminal by terminal (by DFHTPR) while messages are purged because of time expiry (by DFHTPQ). ENQ/DEQ is also performed around the GET of the MCR in DFHTPR.

- A resource whose name is 'DFHTS.tsid' where tsid is a temporary storage data-ID. The ENQ is done by DFHTSP, if the tsid is recoverable, that is, in the TST. DFHTS TYPE=EXCL is not part of the application program interface, but is used by DFHTPQ. DEQ is done:

  - On a PURGE or RELEASE of the item if the item was created in the current LUW, that is, flagged as keypointable

  - At the end of a LUW

  - If a write buffer is locked after a DWE is chained.

- The resource address of the table manager's (DFHTMP) static storage. The ENQ is issued by DFHTMP to single thread changes to the tables, such as additions or deletions, performed by the CEDA transaction when an INSTALL command is

executed. The DEQ is issued by the deferred work element (DWE) processor in DFHTMP at the end of a logical unit of work.

- The resource address of a table directory element. The ENQ is issued by DFHTMP to prevent other tasks accessing the table entry using the directory element when the table entry is about to be modified or deleted. This can happen, for example, as a result of the CEDA transaction executing an INSTALL command. The DEQ is issued by the DWE processor in DFHTMP at the end of a logical unit of work and when the directory element is no longer busy.

- The address of the subtask control area (DFHSKA) in DFHSKP's static storage. The ENQ was issued by DFHSKM to prevent more than one task trying to initialize an operating system subtask. The DEQ is issued when DFHSKC indicates to DFHSKM (by posting an ECB) that the subtask has been initialized.

- Statistics program (DFHSTSP). DFHSTSP enqueues on the lock flag CSASTSLF so that only one task at a time can change the statistics flags.

- The address of an LU6.2 systems attach-time security USERID table (TCSEUTA). The ENQ is issued on closure of an LU6.2 session (when the message DFH3462 is issued) by DFHZNAC at NAPES52E to single-thread access to the table for signing off and deletion of USERIDs that have either expired, or been invalidated. This is done because the adds and deletes performed by DFHZSUP can run concurrently.

  The DEQ is performed on completion of the CLEAR request at NAPES52E. In DFHZSUP, the enqueues and dequeues are performed by subroutines ZSUPACNQ and ZSUPACDQ, which are called before and after calls to DFHZUTM, near labels ZSUPAC06 and ZSUPACAD.

- The address of the PPT entry corresponding to a task-related or global user exit. DFHUEM performs the ENQ when it processes one of the following exit-related requests:

  - EXEC CICS ENABLE
  - EXEC CICS DISABLE
  - EXEC CICS EXTRACT

  DFHUEM issues a DEQ when it has completed processing the request.

DFHSPP issues a DFHKC TYPE=ENQ macro to ensure that, at any time, any resource manager (usually task-related user exits) can only issue a single EXEC CICS RESYNC command to resolve in-doubt units of work. The value enqueued on is the resource manager name.

DFHCRNP, running under transaction CSNC, the interregion control program, issues the DFHKC TYPE=ENQ macro to enqueue on the usage of the user-id table built as part of the attach-time support. The ENQ ensures the integrity of access to the table.

The master terminal command processing routine, MTDBRTN in DFHDLX, issues a DFHKC TYPE=ENQ macro against the DDIR entry for the data base to be processed, to ensure that only one master terminal operation can be in progress for any DL/I data base at a time.

Chapter 3.6. CICS with XRF

If you have a problem when using XRF, it may not be easy to determine which system is at fault. In diagnosing these problems, the message content and sequence can be very useful. The CICS availability manager data sets hold these messages, and these are described in "Chapter 2.5. The CAVM data sets" on page 175.

Symptoms of problems in an XRF complex

This section gives examples of the symptoms that can occur in your CICS system when using the extended recovery facility (XRF). For each symptom, a list of questions leads to the possible causes, indicating the nature of the error. If the error is a CICS error, rather than a user error, the module in error is shown.

You should use this chapter in conjunction with the *Messages and Codes* manual.

The symptoms are divided as follows:

- "Active doesn't initialize"
- "Alternate doesn't initialize"
- "No backup sessions established"
- "Takeover starts, but doesn't complete"
- "Not all terminals recovered after a takeover"
- "Unexpected takeover"
- "Takeover didn't occur when expected"
- "Related systems not taken over"
- "Alternate terminated unexpectedly"
- "Takeover took a long time"
- "Overseer didn't restart a job that had failed"
- "Unable to log on to CICS."

Active doesn't initialize

1. Does MVS indicate that the active system is waiting for data sets?

   This could be because you have not specified DISP = SHR for data sets that need to be shared with the alternate system.

2. Are you trying to allocate BTAM terminals to more than one job at a time?

3. Have the CAVM data sets (the control and message data sets) been set up correctly?

4. For a two-CEC configuration, have the JES shared spool requirements been met?

5. Has the active system signed on to the CAVM?

6. If not, is another job signed on to the CAVM as an active system, or is takeover in progress?

7. If the active system signed on to the CAVM, is an alternate system in the process of taking over and awaiting a response to a message?

8. If IRLM is used, is CICS waiting for the IRLM to be started?

   See the IMS/VS message DFS039I

Alternate doesn't initialize

1. Does MVS indicate that the alternate system is waiting for data sets?

   This could be because you have not specified DISP = SHR for data sets that need to be shared with the active system.

2. Are you trying to allocate BTAM terminals to more than one job at a time?

3. Have the CAVM data sets (the control and message data sets) been set up correctly?

4. For a two-CEC configuration, have the JES shared spool requirements been met?

5. Is an alternate system already signed on to CAVM?

6. Have you started VTAM?

   Unlike the active system, the alternate system cannot initialize without VTAM.

No backup sessions established

1. Has the active system initialized successfully?

   See the appropriate section above.

2. Has the alternate system initialized successfully?

   See the appropriate section above.

3. Has the active system seen the alternate system sign on to CAVM? This is indicated by the message DFH6403.

4. Was transaction CXCU attached?

   You can confirm this by message DFH6499: DFHZXCU CATCH-UP STARTING, which is issued by DFHZXCU as it starts.

5. Did the active system finish sending catch-up message DFHxxxx?

   Messages DFH6494, DFH6495, DFH6496, and DFH6497 are issued during normal execution of CXCU. Messages DFH6476 and DFH6492 are issued for an abnormal end of CXCU. Message DFH6498 indicates that CXCU has finished.

6. Did the alternate system fully receive catch-up message DFHxxxx?

   Check for the following messages:

   - Message DFH1041 is issued at the start of tracking.
   - Message DFH1044 is issued when the start of catch up is received.
   - Message DFH1040 is issued as records are received from the active system.
   - Message DFH1045 is issued when the end of catch up is received.
   - Message DFH1024 is issued when an error occurs in tracking.

7. Do the terminals have backup sessions?

   If a terminal **has** a backup session, a VARY NET,ACTIVE command to the active system shows if an XRF active session was created. **If there are no active sessions, there can be no backup sessions.**

   CICS cannot control whether a terminal has, or has not, backup sessions. To qualify for a backup session (which means that it can support both an active session and an backup session, and that CICS can use the SessionC Control=Switch command to switch the session):

   - The APPLID definition must have HAVAIL=YES.
   - The terminal must be a remote SNA device.
   - The terminal must be connected to a 3725.

- The NCP must be at a release that supports XRF.
- The VTAM must be at a release that supports XRF.
- The terminal must not use session level cryptography.
- The terminal must not be ISC (for example, LU6.1 or LU6.2).
- The terminal must not use a negotiable bind (like an LU6.2 single-session terminal, such as Scanmaster).

8. Are the necessary VTAM APPLID definitions and XDOMAIN definitions active?
9. Does the alternate system CSMT log indicate that XRF backup sessions are being established and then unbound?

   (See under "Not all terminals recovered after a takeover" on page 321.)

You can use the VTAM command DISPLAY NET,ID=applid to display the status of active and backup sessions.

Takeover starts, but doesn't complete

1. Has the takeover been accepted by the alternate system?
2. Is the alternate system waiting for confirmation that the active system has ended (particularly in a two-CEC, CEC failure case)?

   If the alternate system has successfully issued the CANCEL, the active system may be unable to terminate abnormally, for example it may be necessary for the system operator to issue the FORCE command.

   If the active system has ended, the alternate system may be unable to determine from JES that it has. If the active system's CEC has failed, JES may still indicate that the active job is running.

   In these cases, a reply will probably be outstanding to message DFH6561 or DFH6562 on the system operator console, or to message DFH6577 or DFH6578 if an error has been detected with the CLT contents. Refer to *Messages and Codes* for the responses to these messages.
3. If IRLM is used, is CICS waiting for the IRLM to be started?
4. Did the alternate system attempt to cancel the active system?

   When the alternate system issues the CANCEL command, this should be displayed on both system consoles in a two-CEC configuration. If MVS on the alternate system CEC did not accept the command, message DFH6560 is displayed by the alternate system.

   If there is a failure of JES communication between the active and alternate CECs, the CANCEL command may not have been routed.

If there are errors in the CLT, the alternate system may not be able to construct the appropriate CANCEL command.

In each of these cases a reply to messages DFH6561, DFH6562, DFH6577 or DFH6578 may be required. Refer to *Messages and Codes* for the responses to these messages.

5. Do you have the correct definitions in the CLT?

   A previous message should be displayed on the alternate system's CEC to report the error with the CLT contents. If CLT errors were detected, the alternate system may not be able to construct the CANCEL command.

   A reply to messages DFH6561, DFH6562, DFH6577 or DFH6578 may be required. Refer to *Messages and Codes* for the responses to these messages.

Not all terminals recovered after a takeover

Terminals with backup sessions

1. Was the terminal definition successfully restored in the alternate system?

   On the alternate system, message DFH1022: TRACKING ERROR FOR xxxxxxxx, or DFH1046I: FLUSHING TERMINAL CONTROL TRACKING may be issued if a tracking record is dropped for some reason.

2. Was a backup bind established for this terminal?

   Message DFH6593: BACKUP SESSION STARTED is issued to the transient data destination CSNE for each terminal that has a backup bind established. Note that subsequently message DFH3462: SESSION TERMINATED might have been issued to the same destination. This means that there was a backup session that was terminated for some reason.

   You could also consult the CSMT log for other messages, for example, network services error.

3. Was the conversation successfully restarted for this terminal?

   Message DFH6590: CONVERSATION RESTARTED is issued for each terminal that has been switched to the alternate system. It is not issued at the time of the switch, but from DFHZNAC when that terminal begins its recovery process.

   Again, you could also consult the CSMT log.

4. Is there any indication in the CSMT log that recovery failed and that this terminal was unbound during takeover?

   This depends upon:

   - The terminal type
   - Its session characteristics as presented in the bind

- How it is defined in its recovery action parameter
- The activity on the session at the moment the switch was issued.

Depending upon the above points, it is possible that the switch command was issued from the active system to transfer the terminal's session to the alternate system, but that then the alternate system issued CLSDST (UNBIND) to release that session. If that happened, the alternate system attempts to reacquire the session if the terminal supports SIMLOGON.

5. Are the necessary VTAM XDOMAIN definitions active/available?
6. Is RECOVACT = NONE specified for this terminal?

   If RECOVACT = NONE is specified, the logon/logoff state of the terminal in the active system is not tracked by the alternate system, and no backup session is established.

7. At takeover, was message DFH1042I: WAITING FOR TERMINAL CONTROL TRACKING TO DRAIN issued?

   This message indicates that there is a backlog of messages in the CAVM message file. It's likely that the information about the terminal that failed to recover was not known to the alternate system.

8. At takeover, did message DFH1046I: FLUSHING TERMINAL CONTROL TRACKING occur?

   Some event concerning the establishment of the backup bind was held up, and this terminal will not be switched. It will be treated as a terminal without a backout session, and the alternate system attempts to reacquire its session.

9. Did the CSMT log indicate that conversation was restarted for one terminal, but not for the rest?

Terminals without backup sessions

1. In a two-CEC configuration, do the control units need switching?
2. Are the necessary VTAM XDOMAIN definitions active/available?
3. Is RECOVACT = NONE specified for this terminal?

   If RECOVACT = NONE is specified for the terminal, the logon/logoff state of the terminal in the active system will not have been tracked by the alternate system. The recovery at takeover is the same as that at normal emergency restart; the alternate system honors the AUTOCONNECT option for the terminal.

4. Should the reconnect transaction (CXRE) have run yet?

   If CXRE could not be scheduled, messages DFH6485, DFH6487, or DFH6488 will have appeared. If CXRE was delayed, to allow physical switching of terminals to be completed, message DFH6481 is issued, telling you how long the delay will be. This delay includes the delay specified in the SIT parameter AUTCONN.

5. Did the reconnect transaction (CXRE) start up on time?

   Message DFH6490 is issued at the start of every reconnection pass. When CXRE has reconnected all the terminals that it believes it is supposed to reconnect, it ends the pass with message DFH6484. Otherwise, if it is unable to reschedule itself, it issues the message DFH6489.

Unexpected takeover

1. Did someone stop the CEC of the active system, with COMMAND=AUTO coded for the alternate system?

2. Does the CICS system have a sufficiently good performance group to allow continuous normal running?

3. Is the ADI value specified in the SIT for the alternate system too low?

4. In a two-CEC system, has there been some presumably unexpected processing that stopped the active system from running?

   Note: If such processing is expected, you can suspend surveillance by using the command CEBT SET SURV OFF, or change the alternate system to TAKEOVER=COMMAND by using CEBT SET TAKE COM for the duration of the activity.

Takeover didn't occur when expected

1. Was the correct TAKEOVR value in effect for the alternate system?

2. If the CEBT PERFORM TAKEOVER command was entered, was it accepted?

3. If you are expecting an automatic takeover, is the ADI value set too high?

4. If you have set up for an automatic takeover, was the missing surveillance signal detected by the alternate system?

5. Did the active system sign off abnormally?

   If so, did the alternate system detect this?

6. Did the alternate system have a large backlog of messages in the CAVM message file?

   If the backlog becomes too large, the alternate system becomes invalid. Do you need a larger message file?

7. Is the alternate system clock significantly behind the active system clock? This can cause the alternate system to wait before completing the takeover. See the messages DFH6682 and DFH6683.

Related systems not taken over

1. Did the master/coordinator alternate system run its CLT successfully?

   If errors were detected in the CLT, message DFH6576 should have been issued by the alternate system. Refer to *Messages and Codes* for this message.

2. Do you have the correct definitions in the CLT, so that the other alternate systems are instructed to take over?

   If other alternate systems of related systems should have initiated a takeover as a result of a command entry in the CLT, but did not, the appropriate command entries may not be correctly specified in the CLT, or the set of commands may be incomplete.

   System operator commands issued as part of CLT processing are normally displayed on the alternate system's CEC system console. If an error was detected in a command, there should be an associated MVS error message displayed on the alternate system's CEC system console. CLT commands are not interpreted by CICS.

   The CLT contents may not be correct for use by this alternate system. For example, the appropriate APPLID may not be correctly specified in the CLT, in which case message DFH6567 should be displayed on the system console reporting the error.

Alternate terminated unexpectedly

1. Did the active system shut down normally, thus causing the alternate system to shut down normally?

2. Was the alternate system shutdown caused by a CAVM event?

   This could be because a new active system has been initialized, thus invalidating the existing alternate system.

3. Did the alternate system's VTAM fail, thus causing the alternate system to terminate?

   If the alternate system has been defined with VTAM = YES in its SIT, it does not complete initialization until VTAM is available and the VTAM ACB has been opened.

   Similarly, if the VTAM in the alternate system fails while the alternate system is tracking the active system, the alternate system cannot continue, and it is abended. You should bring up VTAM again, and restart the alternate system.

4. Did the alternate system become invalid because of an unacceptable message backlog in the CAVM message file?

Takeover took a long time

1. Did the active system take a long time to close down after it was canceled?
2. Did message DFH6480: WAITING FOR BACKUP SIMLOGON PROCESSING TO DRAIN, or message DFH6475: xxxx BACKUP SIMLOGONS ABANDONED appear?
3. Did message DFH1042: WAITING FOR TERMINAL CONTROL TRACKING TO DRAIN, or message DFH1046: FLUSHING TERMINAL CONTROL TRACKING appear?

Overseer didn't restart a job that had failed

1. Is restart in place disabled?
2. Is restart in place not active for this system?
3. Is the job to be restarted not defined to the overseer?

   If the overseer determines that a takeover is taking place, or is likely to take place, it does not perform a restart in place.
4. Was the overseer unable to open/read the CAVM data sets associated with this job?

Unable to log on to CICS

1. Is the VTAM USERVAR set to the correct value?

   The VTAM commands D NET,USERVAR, and F procname,USERVAR, can be used to inspect, and change, the setting of the USERVAR whose ID is the APPLID used in the failing logon attempts. The VTAM USERVAR is the CICS generic APPLID.
2. Has CICS issued the necessary MODIFY procname,USERVAR command?

   The MODIFY command is logged by both MVS and VTAM. CICS issues message DFH6479: MODIFY USERVAR ISSUED UNSUCCESSFULLY. RETURN CODE xxxx, if it is unable to issue the MODIFY command.
3. Has the new USERVAR value been propagated around the network after initialization of the active system, or after a takeover?

   The VTAM commands D NET,USERVAR, and F procname,USERVAR, can be used to inspect and change the setting of the USERVAR.
4. Are the necessary VTAM XDOMAIN definitions not active, or not available?
5. Did you set up the specific and generic APPLIDs correctly (are they the right way round?) in your SIT?

Part 4. Interfaces

Chapter 4.1. CICS-BTAM interface

The CICS terminal control program, DFHTCP, works as a BTAM application program, so all the rules regarding BTAM macros are followed as described in the **BTAM** manuals. The BTAM requests issued are dependent on the type of terminal and type of connection.

Function of DFHTCP

- Polls terminals
- Initiates transactions
- Handles communication between user written application programs and terminals through BTAM
- Provides additional services, for example:
  - Code translation
  - Terminal storage management
  - Error handling interface.

DFHTCP overview

1. Scans TCT for things to do
2. Calls on BTAM (to poll lines)
3. Continues TCT line scan
4. Waits on TCT-ECB list (CICS wait)
5. Operating system interrupt posts ECB
6. DFHTCP is dispatched through DFHKCP
7. Line is posted ECB (line event initiated bit is turned on in TCTLE)
8. Checks for other activities pending online

9. Handles terminal request (attaches task, transfers data to application programs)
10. Other device interrupts post ECBs
11. DFHTCP continues TCT scan loop to end of TCT
12. Issues CICS wait on TCT-ECB list
13. ECB posted or EXIT TIME expires causes TCP dispatch.

The main block of communication between DFHTCP and BTAM is the TCTLE which is the DECB in BTAM. Most of the fields in the block are modified and referenced by DFHTCP as well as by BTAM. Figure 69 illustrates the interfaces between the user transaction and DFHTCP, and between DFHTCP and BTAM.

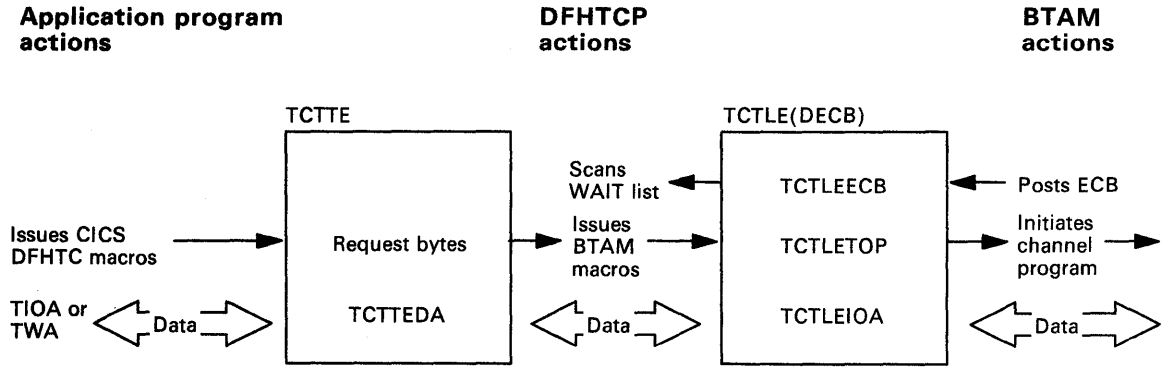


Figure 69. Relationship between the application program, DFHTCP, and BTAM

Debugging

When examining the trace table for CICS-BTAM problems, you should look for the X'E6' entries. If FE trace is active, trace entries are made for each line during a line scan, on I/O event initiation and termination, and on entry to the autopoll change entry routine. See "Chapter 2.1. Trace" on page 45 for a description of the DFHTCP (X'E6', FE type) trace entries.

Examples of DFHTCP operation

This section shows input and output operations under DFHTCP. Terminal storage is chained from the TCTTE and will, in general, remain after a transaction has terminated. DFHTCP is responsible for freeing any remaining storage when all terminal I/O operations have completed. See also Figure 27 on page 150.

Input operation for 3270

CICS performs the following sequence of operations:

- Obtains line I/O area (size of INAREAL) – becomes TIOA for local terminals.
- Issues initial read. Application program may also issue READs.
- Combines 256-byte blocks and obtains larger line I/O area, if needed, for remote terminals.
- Identifies terminal answering poll.
- Obtains a TIOA or uses a user-specified TIOA for the terminal.
- Removes line control characters (STX ETX for remote 3270s).
- Stores AID (attention identifier) at TCTTEAID.
- Stores cursor address at TCTTECAD (converted to binary).
- Stores the data stream in TIOA starting at TIOADBA.
- Translates data if required (lower to upper, ASCII to EBCDIC).
- Calculates and stores the length of data stream at TIOATDL.
- Initiates a transaction or resumes the current transaction associated with the TCTTE if it has issued a WAIT.

The application program performs the following operations:

- Establishes addressability to the TCTTE and TIOA.
- Tests the AID for type of data stream in TIOA.
- Interprets the data stream.

See also Figure 70.

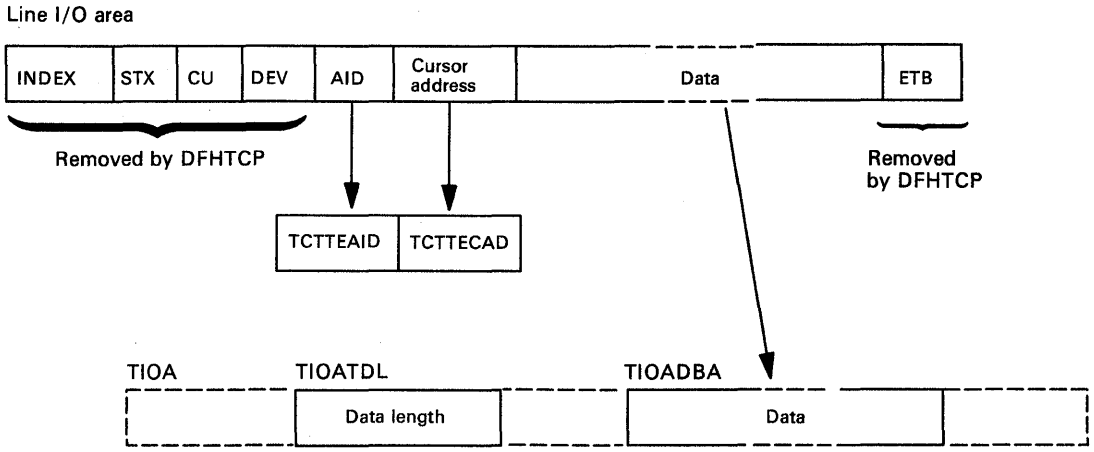


Figure 70. DFHTCP input operation (for remote 3270)

Output operation for 3270

The transaction performs the following operations:

- Gets TIOA and establishes addressability.
- Builds data stream in TIOA starting at TIOADBA.
- Supplies length of data stream to TIOATDL.
- Issues WRITE supplying control information with macro.
- Optionally waits for the WRITE to complete.

CICS then performs the following sequence of operations:

- Obtains line I/O area; CICS adds STX, ESC and ETX for remote 3270s
- Interprets DFHTC request to provide command code, write control character (WCC), and copy control character (CCC)
- Moves data from TIOA into the LIOA; end of data is calculated from length specified
- Provides translation if needed (EBCDIC to ASCII)
- Performs WRITE operation addressing terminal.

When the I/O completes, CICS:

- Frees TIOA after WRITE (unless SAVE specified)
- Resumes the current transaction if a WAIT has been issued.

If a task completes without waiting for the WRITE to complete, CICS will free all remaining storage when it next inspects the TCTTE for work to be done on the line. This will not occur until the output operation has completed and may be later if activity is high.

See also Figure 71.

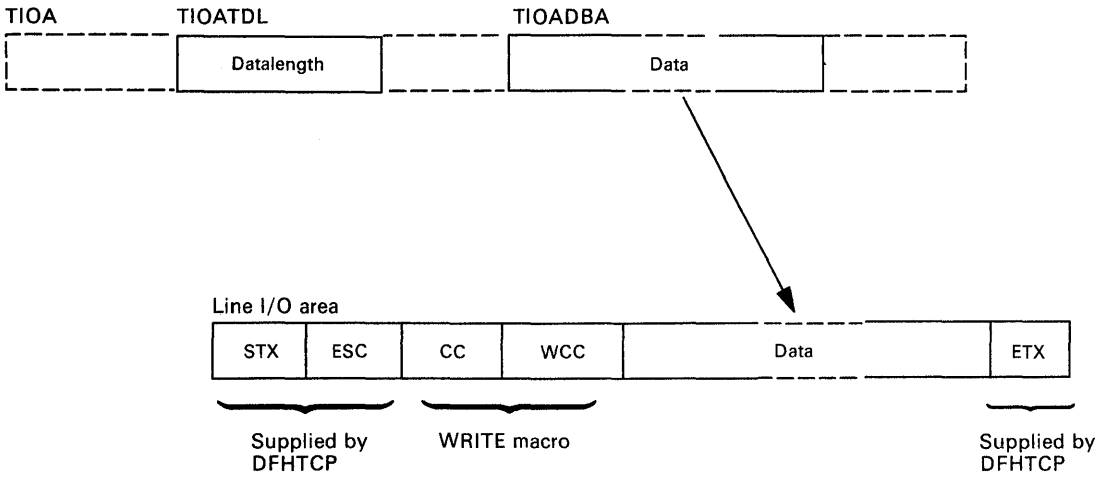


Figure 71. DFHTCP output operation (for remote 3270)

Chapter 4.2. CICS-VTAM interface

The CICS-VTAM interface consists solely of the VTAM macros that are used by CICS.

This chapter lists the VTAM macros used by the modules in DFHZCP, by DFHZNAC, and by the VTAM exit modules in the TCT. Where certain modules are the primary users of any particular VTAM interface, this is indicated by an asterisk (*). Where a module issues an asynchronous request, the location of the RPL exit (if used) is also given.

CHECK

DFHZCLX

DFHZEMW

DFHZOPX

DFHZRAC

DFHZRLX

DFHZRVX

DFHZSAX

DFHZSDX

DFHZSLX

DFHZSSX

DFHZSEX

DFHZSIX

DFHZTAX

CLSDST

| ***DFHZCLS,** the CLSDST RPL exit used is DFHZCLX.

DFHZLGX issued if a logical unit not defined to CICS attempts to logon.

INQUIRE

***DFHZOPN**	issued to read logon data.
DFHZSIM	determines the name of the application program currently associated with a USERVAR.

OPNDST

***DFHZOPN**	the RPL exit used is DFHZOPX.

OPNSEC

DFHZOPN	

RECEIVE

DFHZERH	handles LU6.2 error conditions.
DFHZRAC	issues a synchronous RECEIVE SPECIFIC to retrieve the remainder of a request unit which is larger than the receive any input area (RAIA) used for the RECEIVE ANY. DFHZRAC reissues RECEIVE ANY after processing completion of a RECEIVE ANY.
DFHZRLX	is the VTAM exit routine for LU6.2 RECEIVEs.
DFHZRVL	issues RECEIVE SPECIFIC for LU6.2 to obtain data to satisfy a terminal control read request − the RPL exit used is DFHZRLX. DFHZRVL issues synchronous RECEIVE to purge data in error conditions.
DFHZRVS	issues RECEIVE SPECIFIC to obtain data to satisfy a terminal control read request − the RPL exit used is DFHZRVX. DFHZRVS issues synchronous RECEIVE to purge data in error conditions.
DFHZSDS	issues a synchronous RECEIVE to check for the device end condition on a VTAM-supported native 3270 if there is an intervention-required condition. DFHZSDS also issues a synchronous RECEIVE to purge unsolicited data from a VTAM-supported native 3270.
DFHZSLS	issues first set of RECEIVE ANYs during initialization.

REQSESS

DFHZSIM	used to request a session if a secondary TCTTE.

RESETSR

DFHZRST	used to change VTAM processing mode of a logical unit from CONTINUE ANY to CONTINUE SPECIFIC, or vice versa.

SEND

DFHZERH	handles LU6.2 error conditions.
DFHZRAC	sends exception responses for some of the error conditions that may be detected in DFHZRAC.
DFHZRVS	sends a null RV with change direction indicator set to prompt half-duplex flip-flop devices to send data to CICS – the RPL exit used is DFHZTAX.
DFHZRVX	sends a positive response when a normal data flow command is received.
***DFHZSDA**	sends expedited flow commands – the RPL exit used is DFHZSAX.
DFHZSDL	sends normal flow data for LU6.2, mainly as a result of terminal control write requests and normal data flow commands – the RPL exit is DFHZSLX.
***DFHZSDR**	sends responses to normal data flow requests.
***DFHZSDS**	sends normal flow data, mainly as a result of a terminal control write request, and normal data flow commands – the RPL exit normal flow data is DFHZSDX, and for commands is DFHZSSX.
DFHZSKR	sends responses to VTAM commands including BIND, SDT, and STSN.
DFHZSLX	is the VTAM exit routine for LU6.2 SENDs.
DFHZSYX	send an exception response if an exception request drives the SYNAD exit.

SESSIONC

***DFHZSES**	passes session control commands such as CLEAR and SDT to VTAM – the RPL exit used is DFHZSEX.
DFHZSCX	used to give BIND responses if no logical unit is defined to CICS.

SETLOGON

***DFHZSLS**	issues SETLOGON START during initialization of DFHZCP to allow CICS to accept logons.
DFHZSHU	issue SETLOGON QUIESCE to prevent further logons. This macro is issued when the logon exit is entered during the termination of CICS.
DFHZSCX	issues SETLOGON QUIESCE to prevent further logons. This macro is issued when SCIP exit is entered during termination of CICS.

SIMLOGON

***DFHZSIM**	generates a logon on behalf of a logical unit.

TERMSESS

DFHZCLS	issues TERMSESS when an abnormal error causes the secondary logical unit to terminate the session immediately.

Chapter 4.3. CICS-IMS/DB interface

CICS and the IMS/VS Data Base system can be used together to provide the user with a complete data base/data communication facility. A full description of the system generation, system programmer responsibilities and application programming considerations of this joint use of CICS and IMS/VS DB is provided in the appropriate CICS and IMS/VS publications. The purpose of this chapter is to describe in more detail the various interactions between the two licensed programs. An understanding of these various interactions should facilitate the determination of which area of which of the two licensed programs is the cause of any error that may be experienced.

Before describing each of the various interactions in turn, it is necessary to present some of the basic principles underlying the interface between these two licensed programs.

Note: The principles stated here apply only to the online operation of IMS/VS DB with CICS. They do not necessarily apply either to the batch operation of IMS/VS DB or to the online operation of IMS/VS DB under IMS/VS DC (the data communication feature of IMS/VS). Where appropriate, comparisons between these three different environments are described.

The batch IMS/VS DB system operates as a single batch OS/VS job. It allows a single batch program to access DL/I data bases under the control of that job. It is possible to run more then one such job concurrently but, if this is done, the two jobs will be entirely independent of each other. That is, they will access separate data bases, use separate logs, each undergo full DL/I initialization and termination, and so on. The batch IMS/VS DB environment can be represented schematically as in Figure 72 on page 340.

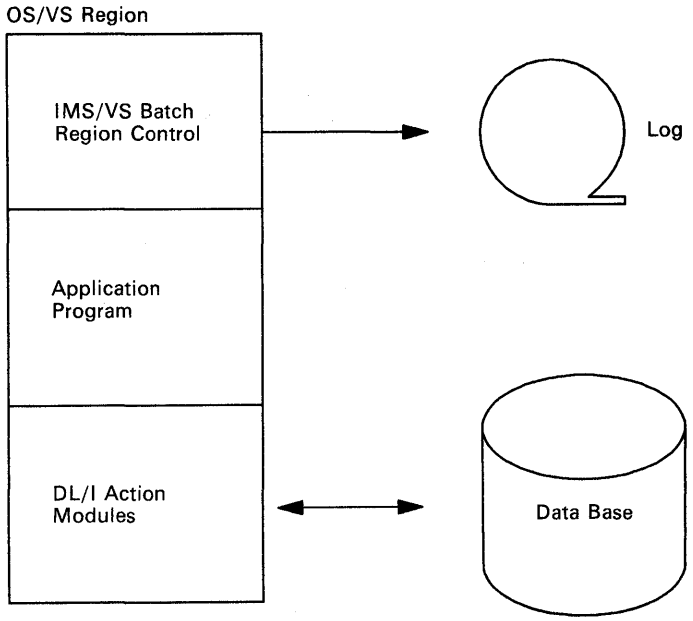


Figure 72. Batch IMS/VS DB system structure

The online IMS/VS DB/DC system has a completely different structure to that of the batch system. It is a multiregion structure. The primary region is termed the IMS/VS Control Region. The region contains IMS/VS function such as terminal support, DL/I, and logging. The application programs themselves reside in one or more dependent regions, one program per region. Each region contains IMS/VS region control functions and the application program. All requests for DL/I or for terminal requests are passed from the dependent region to the control region for processing. Programs in dependent regions are either initiated from the IMS/VS control region as a result of a transaction request from a terminal (message processing programs (MPPs)) or are initiated as batch jobs through batch JCL (batch message processing programs (BMPs)). Through this mechanism, batch and online jobs can concurrently access the same data bases under the control of the IMS/VS control region. The online IMS/VS DB/DC environment can be represented schematically as in Figure 73 on page 341.

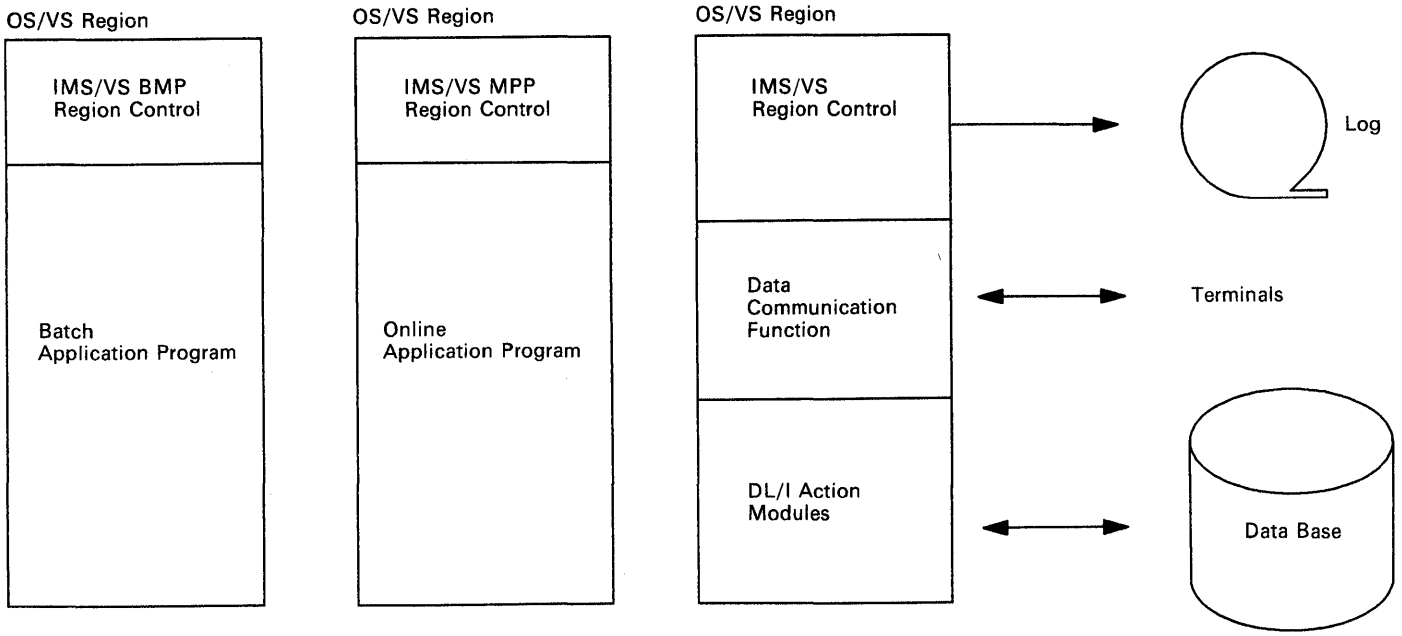


Figure 73. Online IMS/VS DB/DC system structure

The online CICS-DL/I system has a third different structure, in some ways similar to the batch IMS/VS DB system. Instead of the DL/I action modules servicing requests from a single batch application program under control of the IMS/VS batch region controller, they service requests from multiple transactions under the control of CICS, which provides the region control services (such as initialization, error handling, multithreading, storage management, and logging) that are provided in the batch IMS/VS DB system by IMS/VS batch region control and in the online IMS/VS DB/DC system by the IMS/VS control region. The online CICS-DL/I environment can be represented schematically as in Figure 74 on page 342. This figure also illustrates a CICS batch application program that accesses a DL/I data base.

When operating under CICS, the DL/I action modules operate within the normal CICS environment. That is to say, they reside in the same partition, region or address space and run under the same OS/VS task as does CICS. In addition, while servicing requests from a particular CICS task, the DL/I action modules operate under the CICS TCA for that task. This mode of operation is made possible by the fact that the DL/I action modules themselves are written so that they will operate within the different environments of the IMS/VS DB and the IMS/VS DB/DC system. To achieve this, they invoke service routines for any environment-sensitive function they require. These routines differ between the two different IMS/VS environments; the CICS-DL/I interface provides its own service routines for the following functions:

- Task switching
- Buffer pool management
- Master terminal messages
- Logging
- Abend handling.

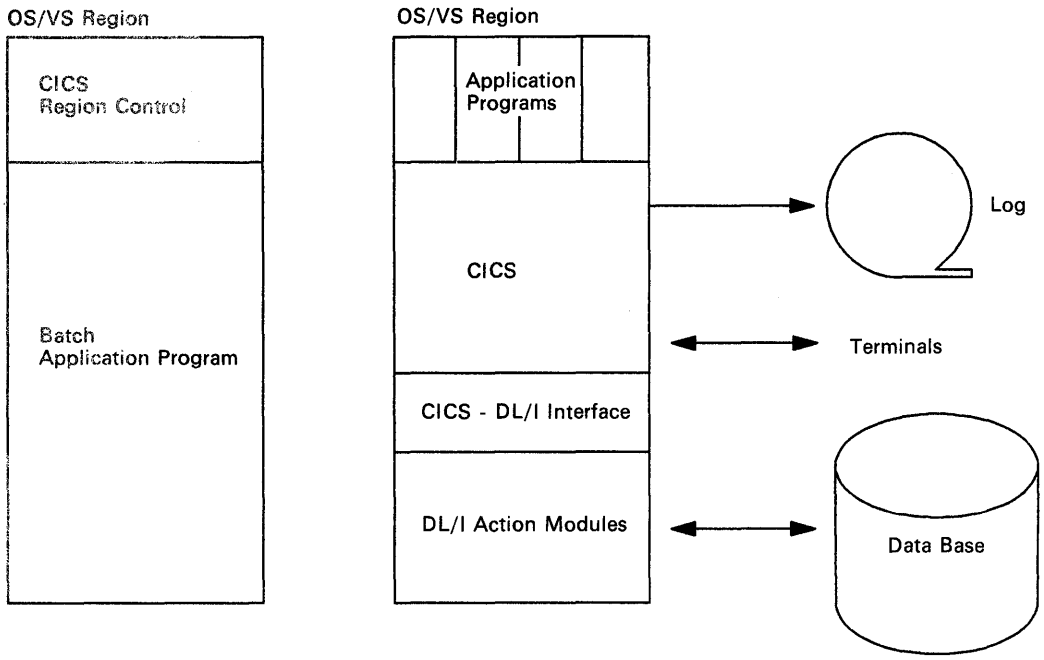
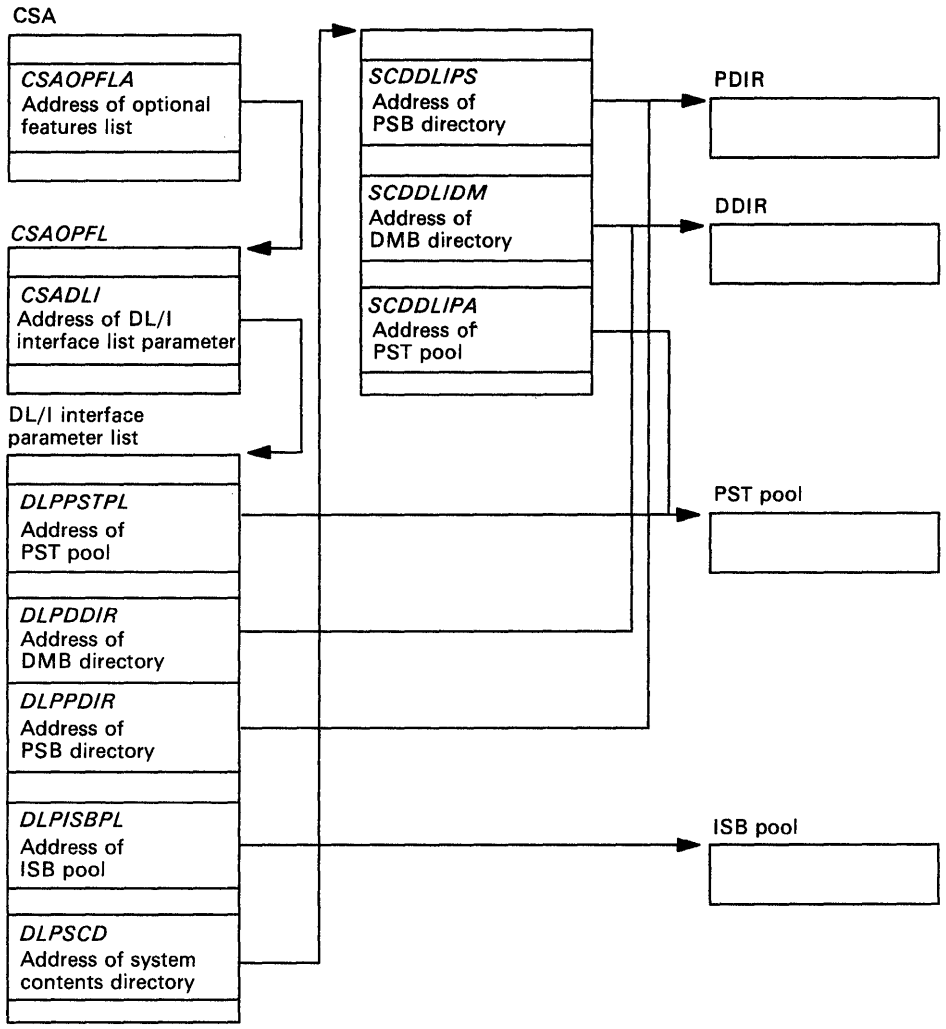


Figure 74. Online CICS – DL/I system structure with batch shared data base

The CICS-DL/I interface comprises six principal modules, DFHDLG, DFHDLI, DFHDLQ, DFHDLR, DFHDLS, and DFHDLX together with localized DL/I-related function in other modules concerned with startup, shutdown, and recovery. DFHDLQ contains all the code necessary to initialize the CICS-DL/I environment. DFHDLR contains the service routines described above. DFHDLI contains the interface code required to service individual DL/I calls, and DFHDLX contains routines invoked internally by DFHDLI. DFHDLG is used by the CICS-DL/I interface as an exit for the IMS/VS resource lock manager (IRLM) to drive global commands, and DFHDLS is the status condition task invoked when a local IRLM abends, or when communication with a remote IRLM fails.

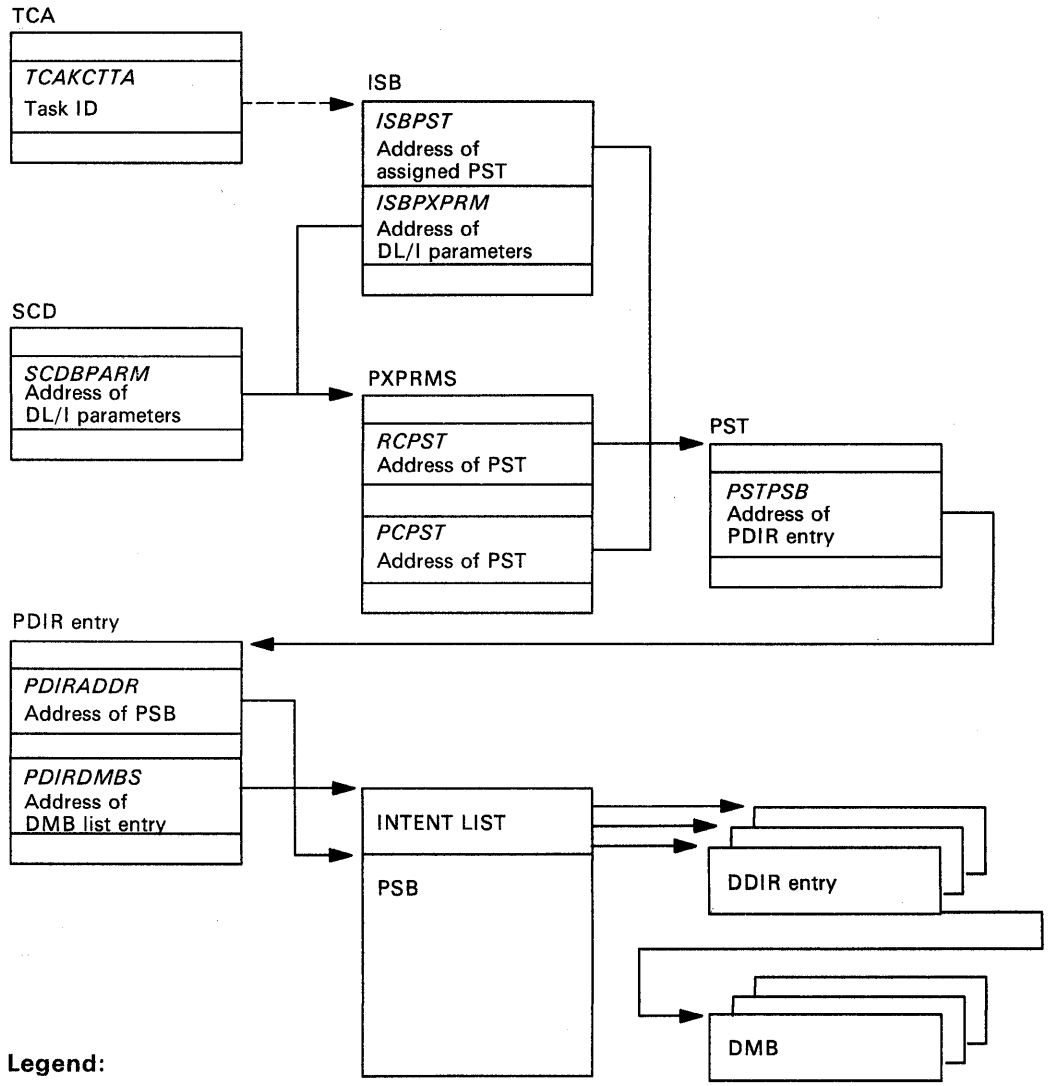
An understanding of the CICS-DL/I interface also requires an understanding of the various control blocks associated with the interface. Figure 75 on page 343 gives a representation of the various control blocks and how they statically relate to each other. Figure 76 on page 344 gives a representation of how these control blocks dynamically relate to each other during the execution of a specific CICS-DL/I task.



Legend:

DDIR	*The DL/I DMB directory as represented by the load module DFHDMB.*	*PDIR*	*The DL/I PSB directory as represented by the load module DFHPSB.*
DFHDLPDS	*CICS - DL/I interface parameter list, as described by the DSECT DFHDLPDS.*	*PST POOL*	*A pool of DL/I PSTs (one for each value of the parameter DLTHRED) allocated and initialized by DFHDLQ.*
ISB POOL	*A pool of CICS - DL/I interface scheduling blocks (one for each value of the parameter DLTHRED) allocated and initialized by DFHDLQ.*	*SCD*	*The DL/I system contents directory.*

Figure 75. CICS-DL/I control blocks – static representation



Legend:

DDIR entry — *The specific entries in the DMB directory corresponding to the data bases referenced by the PSB scheduled by the current task.*

DMB — *The data management block(s) corresponding to the data base(s) referenced by the PSB scheduled by the current task.*

INTENT LIST — *The PSB intent list which represents the scheduling intent against specific segment types in specific data bases defined for this PSB. The intent list will often, but not necessarily always, physically precede the PSB itself.*

ISB — *The CICS - DL/I interface scheduling block. This block connects a CICS task (TCA) to a DL/I thread (PST) dynamically (at DL/I 'PCB' call time). The TCA does not actually address the ISB; the connection is determined by the presence in the ISB of the task ID (TCAKCTTA).*

PDIR entry — *The specific entry in the PSB directory corresponding to the PSB scheduled by the current task.*

PSB — *The program specification block scheduled by the task.*

PST — *The DL/I partition specification table. This is the control block representing a DL/I thread, analagous to the TCA.*

PXPRMS — *The DL/I parameter block. There is one parameter block for each PST, and hence for each DL/I thread generated for CICS. The DL/I parameter block comprises four parts:*

PXPARMS - parameter anchor block
RCPARMS - region control parameter block
LIPARMS - language interface parameter block
PCPARMS - program control parameter block

Figure 76. CICS-DL/I control blocks – dynamic representation

Having given this general overview of the structure of the CICS-DL/I interface, the remainder of this chapter is devoted to a more detailed description of the various aspects of the interface. These aspects are:

- System generation
- System initialization
- DL/I scheduling
- Data base calls
- DL/I schedule termination
- System termination
- Master terminal function
- Service routines
- Recovery.

System generation

The generation of the DL/I description blocks, the data base description (DBD) and program specification block (PSB) is the same for the CICS environment as it is for the IMS/VS environment.

Note: CICS (like IMS/VS DB/DC but unlike IMS/VS DB) requires the object form of the PSBs and DBDs (DMBs) to be generated using the IMS/VS ACB generation and the maintenance utility program.

In addition to generating the PSBs and DBDs, the PSB directory (PDIR) and DMB directory (DDIR) must be generated. These directories have an entry for each PSB and DMB referenced by the CICS-DL/I system. The online IMS/VS DB/DC system also requires generation of these directories; the batch IMS/VS DB system does not because they are built dynamically at batch initialization time from the PSB specified on the batch JCL. The CICS PDIR and DDIR generation macro, DFHDLPSB and DFHDLDBD, build the PDIR and DDIR by invoking the corresponding IMS/VS macro DFSPSBD and DFSDMD.

In addition to generating the required DL/I control blocks, DL/I-related code must be generated in CICS modules (see the *Customization Guide*).

System initialization

CICS-DL/I initialization is performed jointly by CICS and IMS/VS modules. Broadly speaking, the initialization functions performed can be divided into three categories:

- Initialization of the interface itself
- Simulation by CICS of initialization functions performed, in the IMS/VS environment, by IMS/VS itself
- Invocation by CICS of initialization functions that exist in the IMS/VS DB system for batch initialization.

These functions are performed or controlled by modules DFHSIB1, DFHSIH1 and DFHDLQ.

DFHSIB1 is responsible for loading, as CICS nucleus modules, DFHDLI, DFHDLR, DFHDLX, DFHPSB and DFHDMB. DFHSIB1 also copies the control block DFHDLPDS from DFHSIB1 to DFHDLI.

DFHSIH1 loads DFHDLQ, using an OS/VS LOAD, and invokes it, passing the address of DFHDLPDS. DFHDLQ performs the major initialization functions. For START=STANDBY, phase 1A of DFHDLQ is executed. For other values of START, phase 1 of DFHDLQ is executed.

The steps performed during phase 1 initialization are:

1. DFHDLQ ensures that the JCT meets the requirements of DL/I.
2. DFHDLQ initializes the DFHDLPDS save area set.
3. DFHDLQ then OS/VS links to DFSRRC00, passing the parameter list. IMS/VS performs the interface initialization, including signing on to DBRC, and identifying with IRLM. CLIIOPID is set to zero.
4. On return from DFSRRC00, DFHDLQ obtains storage for the ISB pool, formats the ISBs, and allocates one ISB to each PST, up to the number of threads specified by the value of DLITHRED.
5. Statistics blocks are set up.
6. The DDIRs are refreshed.
7. The DBRC sign-on-complete ECB (DLPSOECB) is posted.

For phase 1A initialization, steps 1 to 4 are performed, and DFHSRRC00 sets CLFLG1 (CLCWBU) for an alternate CICS system.

The DL/I restart program (DFHDLRP):

- Calls DFHDLQ, if required, passing the address of DFHDLPDS.
- Handles the recovery of EEQEs for the DL/I input/output error condition.

- Calls DFHDLBP to perform DL/I backout.

When DFHDLRP calls DFHDLQ, DFHDLQ executes phase 1B initialization. Phase 1B is the same as steps 3, 5, 6, and 7 on page 346 for phase 1 initialization.

DL/I scheduling

This section describes the processing that is performed when a CICS task requests DL/I scheduling, by means of a DL/I 'PCB' call. The steps performed during this processing are as follows:

1. Determine whether the PSB to be scheduled is local or remote. If remote, allocate an RSB (remote scheduling block), and issue a DFHIS TYPE=CONVERSE to ship the scheduling request to the owning system, and continue at step 7.

2. If the PSB is local, obtain a free ISB for this new DL/I user. If there is no free ISB, the task must wait until a currently scheduled DL/I user releases an ISB using a TERM call (implicitly or explicitly issued).

3. Locate the PDIR entry for the requested PSB and check for validity. If the PSB has update intent against some data base segment, and the PSB is already in use by another task, then if PISCHD=NO has been specified in DFHSIT (or in the override in the JCL), the current task must wait until the PSB is released by that other task. If PISCHD=YES has been specified, then scheduling can proceed.

4. Invoke the DL/I block loader DFSDBLM0. This composite module is responsible for loading and checking the DL/I control blocks required for the current schedule operation (the PSB intent list, the PSB and the DMBs) and for checking for intent conflicts between this task's PSB and any PSBs already scheduled. A conflict will result in the current task waiting for the conflicting PSB to be released.

   If PISCHD=NO has been specified, a conflict arises if this task's PSB has update intent on any segments for which PSBs currently scheduled also have update intent. If PISCHD=YES has been specified, a conflict arises if this task's PSB and currently scheduled PSBs share intent on a segment for which one of the PSBs has exclusive intent.

   A conflict also arises if the PSB contains a data base that cannot be authorized for usage by this CICS subsystem. This preserves data-base integrity in a data sharing environment.

5. Test the result of the schedule by DFSDBLM0; if successful, the PCB address list is constructed for the application program.

6. If the PSB has update intent, construct and write out a scheduling record.

7. If a UIB pointer argument has been passed on the scheduling call, return the address of UIB to the application. The UIB address is also stored in TCADLUIB. UIB allows the application programmer to obtain the address of the PCB address list and the CICS-DL/I interface response without reference to the TCA.

Data base calls

Normal data base calls require very little action of the CICS-DL/I interface itself. This action is limited to:

1. Optionally, constructing a standard DL/I argument list if the call was invoked by the DFHFC macro.

2. Selecting the appropriate ISB and related control blocks for the current task.

3. Invoking DL/I and checking that the call was processed satisfactorily before returning to the application programmer.

Alternatively, if a **remote** PSB is scheduled (see step 1 on page 347 in "DL/I scheduling" on page 347), the action consists of issuing a DFHIS TYPE=CONVERSE to ship the data base call to the owning system, and returning any return codes to the user (using the UIB if one exists).

DL/I schedule termination

The actions performed at DL/I schedule termination time are largely the converse of these performed at DL/I schedule time. A DL/I termination can result from an explicit DL/I TERM or T call or can be issued implicitly by CICS as a result of a task termination or user synchronization point.

If an application program issues TERM, DFHDLI is invoked. DFHDLI calls DFHSPP, which performs a sync point and calls DFHDLI with the TERM parameter. When DFHDLI determines that it is called by DFHSPP, it terminates DL/I.

The actions performed by DFHDLI when called by DFHSPP are:

1. Locate the appropriate ISB and related control blocks for the current task.

2. Call IMS/VS (DFSFXC50) to flush the data base buffers and DL/I log record buffers for this task.

3. If the PSB is an update intent PSB, write out a DL/I termination log record.

4. Build a deferred work element (DWE) for DFHSPP and return to that program.

DFHSPP makes a phase II sync point call to DFSFXC50 to release all locks, and a RELPSB call to the same routine to free the data base and the PSB.

The ISB is released for use by another task.

Alternatively, if a remote PSB is scheduled, the actions performed are:

1. Free the RSB and local PCB storage.

2. Unless the DL/I termination was caused by a CICS sync point, request one now.

System termination

During normal system termination (DFHSTP), CICS terminates DL/I by issuing a CALLDLI('*QDL') call. This has the effect of closing all DL/I data bases.

Master terminal function

The CEMT command supports data base START, STOP, dump (DUMPDB) and recover (RECOVERDB) commands. Only one command can be processed at a time.

The START command allows access to the data base. This command has options for four different modes of ACCESS. If the data base is currently open for input only (that is, ACCESS = RO or ACCESS = RD, or a DUMPDB command has been issued), and you require output mode (that is, ACCESS = UP or ACCESS = EX), then the data base is quiesced, closed, and opened for output. If the data base is currently open for output and you require input only, then the data base is quiesced, closed, and opened for input only.

The STOP command performs the opposite function to the START command. If the specified data base is currently in use, it waits until it is no longer in use. The IMS/VS open/close routine, DFSDLOC0 is then invoked to explicitly close the data base.

The purpose of the DUMPDB command is to temporarily forbid any schedule request for a PSB with update intent on the data base, while a copy of the current state of the data base is taken. This is achieved by indicating in the DDIR entry that no PSB with update intent can be scheduled against the data base. After all currently scheduled PSBs against this data base have completed then the data base is explicitly closed, as above, and the command is complete.

The purpose of the RECOVERDB command is to temporarily forbid any access (read or update) to the data base while some recovery operation is performed against the data base. You can issue RECOVERDB commands for several data sets at the same time. This is performed in an identical manner to the DUMPDB command above except that:

- The DDIR entry is set to indicate that no PSB with any intent against this data base may be scheduled.
- When all the data bases have been quiesced and closed, the current volume of the system log is closed and a new one opened. This is to facilitate the use of the current log in any recovery operation.

Service routines

Module DFHDLR contains various CICS service routines which replace equivalent service routines in IMS/VS. Briefly, these service routines perform the following actions:

- Task switching. When DL/I must wait for some action before continuing operation on a particular thread (for example, wait for I/O or schedule conflict), the DL/I action modules issue an IWAIT macro. The CICS simulation of IWAIT function invokes a CICS task control wait on the same ECB.

- Buffer pool management. IMS/VS has buffer pool management mechanisms for DMB and PSB control blocks. CICS replaces these mechanisms by straightforward storage control operations.

- Master terminal messages. Any messages which would normally be output to the IMS/VS master terminal are intercepted and directed to the CICS master terminal destination.

- Logging. Any records which would normally be written to the IMS/VS system log are intercepted and directed to the CICS system log.

  Log records, representing data base I/O errors, are written to the CICS catalog, so that the data base control blocks that had errors can be identified at restart, and integrity ensured.

- DL/I internal abends which are produced when DL/I detects an error on a specific thread are intercepted by CICS and are converted into CICS task abends.

Recovery

CICS takes advantage of the DL/I records that are written on the CICS system log to extend the CICS dynamic transaction backout and emergency restart mechanisms to include modifications made to DL/I data bases. To achieve this, CICS reads any relevant records from the dynamic or system log respectively and passes them directly to the IMS/VS data base backout module DFSRDBC0. DFSRDBC0 is loaded during initialization.

Execution of DL/I commands

For online applications, an EXEC DLI command is translated to a call to DFHEIP. DFHEIP passes the parameters of the command to DFHEDP. DFHEDP transforms the parameters to use in calling DFHDLI, which interfaces with the DL/I system. For batch applications using CICS shared data base, the application calls DFHDRPG, which calls DFHDRPE.

Debugging

When IMS/VS detects a data base error, a snap dump is written to the CICS system log. This should be taken into account when specifying the size of the log. The snap dumps can be printed using the journal print utility (DFHJUP).

If the system is in a wait state and you suspect that the wait is caused by a DL/I problem, find the CSA and locate the CSA optional features list (CSAOPFL) from the address in CSAOPFLA. In CSAOPFL, CSADLI points to the DL/I interface parameter list (DLP) . The DLP holds the number of DL/I threads (DLPTHRED) and a pointer to the ISB pool (DLPISBPL).

Look at each ISB to determine whether it is in use or not. If an ISB is in use, the task save area (ISBSA) has a pointer to the corresponding TCA. If the transaction was in an IWAIT, register 5, saved in the TCA before the DFHKC wait was reached, gives the IMS/VS save area address. From this, you should be able to determine which IMS/VS module issued the WAIT.

You might find other useful information in the DL/I system contents directory (SCD) which is addressed from DLPSCD.

The journal print utility (DFHJUP) displays each CICS journal record in both hexadecimal and character form. DL/I records are formatted, but not CICS records.

Data sharing

Data sharing is the facility for a CICS system to share the same DL/I data base with another system such as CICS, IMS/VS DB, or IMS/VS DC. Data sharing can be done at two levels – at the data base level where one system updating a data base locks that data base out from the other systems, or at block level. For data sharing, data base level sharing is controlled by the IMS/VS Data Base Recovery Control (DBRC) component, and block level sharing is controlled by the IMS/VS Resource Lock Manager (IRLM).

DBRC is a program which supervises access to, integrity of, and recovery of data bases.

In a CICS environment, DBRC runs as an operating system subtask. Since DBRC handles only one request at a time, a call from a CICS program is routed through the IMS/VS program, DFSRCQR0, which queues the request.

If IRLM is unable to execute a request immediately, a special exit (the SUSPEND exit) is made to the caller and a task control wait macro is issued by CICS. When the requested resource is available, the CICS task is posted.

Data sharing is implemented by the DL/I interface module, DFHDLX. The initialization program, DFHSII1, calls DFHDLX to complete the initialization for DL/I. If the previous CICS run abended, DFHDLX frees any retained authorizations by issuing a DBRC sign-on-recovery-end call. It also issues a purge call to IRLM to release locks still held, and attaches the transactions CSGX and CSSX.

All requests by CICS to DBRC are executed by a call to DFHDLI with register 1 holding the address of a parameter list, the first parameter being the string '*Dxx' where 'xx' specifies the request. These requests are processed by subroutines in the module DFHDLX which are called by DFHDLI.

DFHDLX handles requests to IRLM by using the DFSLR macro for lock requests and DFSLM for other requests. Requests to IRLM are issued by DFSLMGR0 and IRLM runs in a separate address space, so that the requests can run in Cross-Memory mode.

When CICS is shut down, the system termination program, DFHSTP, calls DFHDLI to shut down DBRC by issuing DFSPCCC SIGNOFF NORMAL for normal shutdown or DFSPCCC SIGNOFF ABNORMAL for immediate shutdown. For IRLM, DFHDLI issues a DFSPCCC QUIT NORMAL for normal shutdown or DFSPCCC QUIT ABNORMAL for immediate shutdown. The calling argument in DFHSTP is '*QDL'. When an abend occurs, DFHCRC issues the corresponding DFSPCCC macros.

The global command "ITASK, CSGX", which invokes DFHDLG, processes master-terminal data-base commands issued by other subsystems. It is IPOSTed by the command exit DFSCMDX0, which is running under a service request block (SRB) and, if not resumed for a detach, checks the work queue on its ECB and processes any elements in the queue.

The status "ITASK, CSSX", which invokes DFHDLS, handles IRLM abends. It is invoked when the IRLM status exit, DFSSTAX0, running under an SRB, IPOSTs DFHDLS. If the local IRLM failed, then DFHDLS calls DFSRDSH0 to abend tasks using SHARELVL=2 or 3 data bases. If the remote IRLM failed, or the link between a local and a remote IRLM failed, then DFHDLS calls DFSDBAU0 to close data bases in which the other system needed to reference and to abend tasks using these data bases.

When DFHJCO opens a journal volume or data set, it calls DFHDLI to inform DBRC of the change in the system log. When the system log is closed, DFHJCC calls DFHDLI to inform DBRC of the closure. If an I/O error occurs in the system log, DFHJCIOE calls DFHDLI to tell it of the defective tape.

The master terminal program, DFHEMI, handles the keywords ACCESS and GLOBAL, and calls the corresponding routine in DFHDLX. DFHDLX issues the DFSLM NOTIFY macro to call IRLM to transmit GLOBAL commands to other systems.

Obtaining a trace of the IRLM activity

For the IMS/VS resource lock manager (IRLM), traces are recorded using the MVS/XA Generalized Trace Facility (GTF) when the USR option is specified. The IRLM also provides internal traces that can be activated without starting the GTF trace.

Tracing of the IRLM is invoked normally by the MVS console operator. The command to start or stop the trace is:

```
F        irlmjobn,(START|STOP)
                 ,ITRACE
                 ,GTRACE
                 ,PTBTRACE
                 ,TRACE
```

where **irlmjobn** is the MVS job-name of the IRLM (the default job-name is the same as the procedure name).

The GTF trace data set contains data for the IRLM in that system. The trace records go to a spooled output data set for offline printing and formatting. The arrangements for running the trace need to be coordinated between the sharing subsystems. The IRLM records in the GTF trace data set can be selected by specifying USR=(FC8,FC9) in the control statements for the AMDPRDMP service aid.

Chapter 4.4. CICS-TCAM interface

The interface between CICS and TCAM is through sequential message queues. The interface routine uses READ and CHECK macros to obtain messages from the queue, and WRITE and CHECK macros to place messages on to the queue. Messages are in U-format, as shown in Figure 77 on page 354.

The TCAM Message Control Program (MCP) must provide messages of the expected format; in particular for SNA, a CCB of 2 bytes must be inserted in all messages. On output, the CCB must be removed by the MCP. The CICS-TCAM interface comprises an input routine and an output routine.

The input routine performs the following basic functions:

1. Obtains line I/O area and issues READ macro.
2. Checks completion of the I/O request and exits if incomplete.
3. Compares DESTINATION NAME with TCTTE NETNAME field to determine which terminal (TCTTE) is to receive the message.
4. Checks status of the terminal to ensure that the terminal is in service and is not receive only.
5. Obtains TIOA and copies FMH and message from LIOA to this TIOA.
6. If a task is associated with a terminal, ensures READ has been issued; otherwise tries to attach a task.
7. Any failure of these tests results in the terminal abnormal condition program, DFHTACP, being called immediately and delayed calling of DFHTACP if the situation has not changed after a specified time.

The output routine scans all TCTTEs, performing the following functions:

1. If a write is requested, the routine obtains an LIOA, builds in it the CCB and device-dependent data, if necessary, copies the FMH and message from the TIOA, and issues the WRITE macro.
2. The routine checks for completion of the WRITE macro and exits if incomplete.

3. The routine calls DFHTACP if any errors occurred on the I/O.
4. If no errors occurred, frees the TIOA (unless the SAVE option is specified) and RESUMEs if WAIT was issued.
5. If the TCTTE is inactive (no task and no write requests), the routine tests if automatic task initiation (ATI) is required, and performs the appropriate function.

Problem pointers

- A mismatch of message formats between MCP and CICS-TCAM interface.
- A mismatch of message lengths expected to/from MCP and CICS-TCAM interface.
- Insufficient negative poll delay specified on TCTTE causing UNSOLICITED INPUT message to be given.
- MCP terminal generation does not match CICS DFHTCT generation; for example, terminal names do not correspond.
- The CICS-TCAM interface provides only the equivalent of CICS sequential terminal support for TCAM terminals. All real terminal handling is provided by the MCP and TCAM routines. Therefore, the handling of hardware errors is left entirely to the MCP and TCAM routines. The CICS-TCAM interface works solely on the message queue, never with the terminals directly; for example, TCAM control macros are not issued by the interface.

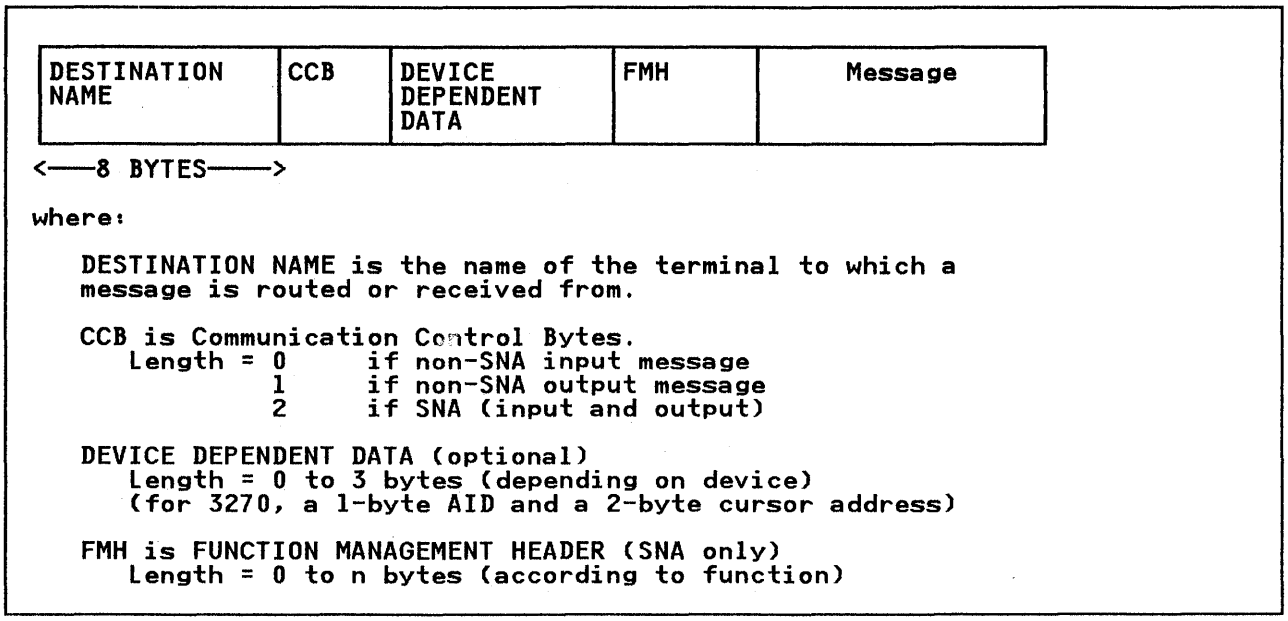


Figure 77. Format of CICS-TCAM interface messages

Chapter 4.5. CICS file control-VSAM interface

VSAM files

Subtasking

VSAM and BSAM read/writes are performed by the VSAM/BSAM subtask program, DFHVSP. The CICS file control, transient data, temporary storage and journal programs use the DFHSQEM macro to add a subtask queue element (SQE) to DFHVSP's work chain and call that program.

During opening, or closing, of a VSAM file (and also a BDAM file), file control (DFHFCN) uses the CICS subtask management program (DFHSKP) for various operations, which might cause operating system waits.

The VSAM ACB

The ACB for each VSAM file accessed by CICS resides in the FCT and is generated at open time. Figure 78 on page 356 gives the correlation between the ACB options (see the *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide* or the *MVS/XA VSAM Administration: Macro Instruction Reference* manual), and the FCT entry (see the *Resource Definition (Macro)* manual).

ACB Option	FCT Entry
AM=VSAM	ACCMETH=VSAM
DDNAME=name	DATASET=name
PASSWD=password	PASSWD=password
BUFND=number)	BUFND, BUFNI, and STRNO (for nonshared resource
BUFNI=number)	(NSR) files only), unless the file is REUSE,
STRNO=number)	empty, or to be emptied, in which case all values are set to 1
BSTRNO=number	The accumulation of STRNOs from all FCT entries with the same base name, and the FCT entry with DATASET name equal to the base name (if it is in the FCT). BSTRNO is specified for paths only. For base clusters, this number replaces the STRNO number.
MACRF=	
ADR	KSDSs and ESDSs
KEY	KSDSs, RRDSs, and AIX paths
DSN	All files apart from load-mode files and files opened for input only with DSN=UPDATE specified in the FCT entry
DFR	All files using local shared resources (LSR) unless specified to VSAM as share option 4
DIR	All files except load-mode files
SKP	KSDSs, RRDSs, and AIX paths
OUT	Files with a SERVREQ of ADD, DELETE, or UPDATE
RST	Files with a SERVREQ of REUSE, with an empty request outstanding, or actually empty
LSR	Files that do not have LSRPOOL=NONE specified, unless to be opened in load mode
SHRPOOL=number	The LSRPOOL specification in the FCT if multiple resource pools are supported

Figure 78. Correlation between ACB options and the FCT entry

The options MACRF=ADR, AMODE31, DIR, NUB, SEQ, and SKP are always used in the VSAM ACB; the ACB options CATALOG, CRA, EXLST, and MACRF=AIX, CFX, CNV, GSR, ICI, SIS, and UBF are never used by CICS VSAM files.

Opening the ACB

VSAM ACBs are opened during CICS initialization for cold start, warm start, or emergency restart if FILSTAT=OPENED is specified in the corresponding FCT entry. Otherwise, they are opened on the first reference. During the opening of a VSAM file, the information put in the FCT includes:

The data set name of the file (for BDAM files also)

The base cluster data set name

The type of object – path, base, or alternate index (AIX) (see note 1 on page 357)

The type of file – ESDS, KSDS, or RRDS

The key length

The relative key position

The logical record length

BUFND, BUFNI, and STRNO (see note 2)

The high RBA for a logged ESDS (see note 3)

Notes:

1. *CICS does not support the processing of an AIX as a data set. If one is opened, CICS treats it as a KSDS, but does not ensure its integrity. It is the user's responsibility to make sure that the file is processed correctly.*

2. *If BUFND, BUFNI, or STRNO are specified in the JCL, VSAM takes these values in preference to those specified in the ACB for NSR files. You are advised not to specify them in the JCL. If you do, CICS updates the values in the FCT entry for NSR files in order to avoid some of the problems that can arise.*

3. *If the file is protected, changes must be logged before they take place. In order to do this, it is necessary to provide the relative byte address (RBA) before a new record is added. The current RBA is therefore maintained by CICS in the base cluster block. It is set up at the opening of the first file updating the base, and updated each time a new record is added.*

VSAM file operations

VSAM file operations requested by CICS applications are controlled by DFHFCP. When DFHFCP is invoked, it communicates the request to VSAM by means of a request parameter list (RPL), described in the *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide* or the *MVS/XA VSAM Administration: Macro Instruction Reference* manual.

For each file operation there are usually three references to the RPL. These are:

- VSAM request, for instance, GET, PUT
- CHECK macro issued after I/O is complete
- ENDREQ issued at termination of request, or when RELEASE specified.

The RPL is built dynamically by DFHFCP as part of a VSWA. Its duration is the same as that of the VSWA, and the same as the length of time for which the transaction holds one of the strings on the VSAM file. This is shown in Figure 79 on page 358.

Setting of fields in VSAM RPL

The VSAM RPL is located at offset 8 in a VSWA and set up by DFHFCP according to options on the DFHFC macro, or the FCT entry for the file being referenced. The fields in the VSWA used by CICS are shown in the *Data Areas* manual. The values of the option fields VSWAOPT1 and VSWAOPT2 are shown in Figure 80 on page 359 and Figure 81 on page 360.

Request	Duration of RPL (in a VSWA)
GET request (move mode)	From the GET until I/O is complete and DFHFCP returns to the user.
GET request (locate mode)	From the GET until user issues RELEASE.
GET for update TYPOPER=UPDATE	From the GET until the PUT TYPOPER=UPDATE, PUT TYPOPER=DELETE, or RELEASE.
PUT TYPOPER=NEWREC	From the PUT until the end of the PUT.
PUT TYPOPER=NEWREC part of mass insert sequence	From the GETAREA TYPOPER=MASSINSERT until RELEASE.
Browse operation, such as SETL, GETNEXT	From SETL to ESETL. RESETL does not release the string or the VSWA.
DELETE	Until completion of the DELETE. It may be a generic delete, which involves several VSAM requests.

Note:
In all cases there is an implicit RELEASE at end of task or sync point, which is implemented by the DWE mechanism.

Figure 79. Duration of a VSAM RPL in a VSWA

Bit	Name	When Set
X'80'	RPLLOC	Locate mode request. Set on for: TYPE=GET and not TYPOPER=UPDATE TYPE=SETL, MODE=LOCATE.
X'40'	RPLDIR	Direct request. This bit is set on by default. Set on for: A GETNEXT request, if the user has lowered the key since the previous GETNEXT A GETNEXT request, if the previous request was for a backward browse A GETPREV request, if the previous request was forward. Set off for: A mass-insert A generic delete after the first erase.
X'20'	RPLSEQ	Sequential request. Set on for: A mass-insert request A generic delete after the first erase A GETNEXT, or GETPREV if the KEY field has not been changed by the user.
X'10'	RPLSKP	Skip sequential request. Set on for: A mass-insert request if the file is an RRDS A GETNEXT request if the user has increased the key since the previous GETNEXT.
X'08'	RPLASY	Asynchronous request. Set on for: Files using nonshared resources (NSR).
X'04'	RPLKGE	Set on for: A request for a greater-or-equal key, as opposed to an equal key.
X'02'	RPLGEN	Set on for: A generic key search. The first byte of the key is the length.
X'01'	RPLECB	Always off. The ECB address is in VSWAECB.

Figure 80. Option code settings in VSWAOPT1

Bit	Name	When Set
X'80'	RPLKEY	Locate record by key. Set on for: A file that is RRDS A file that is ESDS, or KSDS, and the access is by key.
X'40'	RPLRBA	Locate record by RBA. Set on for: A file that is ESDS, or KSDS, and the file is by RBA.
X'20'		Not used.
X'10'	RPLBWD	Read backward request. Set on for: A GETPREV request The initial browse key being X'FF's, that is, the request is specified to start at the end of the data set.
X'08'	RPLLRD	Last record processing. Set on for: The initial browse key being X'FF's.
X'04'	RPLWTX	User UPAD exit. Always set on if shared resources are being used.
X'02'	RPLUPD	Update request. Set on for: A GET UPDATE request A DELETE request that is interpreted as a VSAM GET UPDATE followed by an ERASE request.
X'01'	RPLNSP	Note string position. Set on for: A load-mode file A generic delete A mass-insert request A browse request.

Figure 81. Option code settings in VSWAOPT2

VSAM ENDREQ request

A RELEASE operation may be explicitly coded by the user. There is also an implicit RELEASE at:

- End of task
- Sync point
- Termination of PUT (not mass insert), DELETE and move-mode GET requests.

In each case DFHFCP performs a VSAM ENDREQ operation if the bit VSWAEREQ is on (see Figure 82). If the ENDREQ operation is not performed, subsequent VSAM requests may cause the string count specified for the file to be exceeded, resulting in the dynamic acquisition of resources and consequent performance degradation.

CICS maintains its own count of the current number of operations outstanding on a file, and puts a transaction in a CICS wait before attempting a VSAM request if the request would cause VSAM to use more strings than originally allocated. This is described in the section on DFHFCP waits in "Chapter 3.5. CICS in a wait state" on page 297 of this manual.

	FWA acquired	VSWA acquired	VSWAEREQ bit set - ENDREQ required	Shared string acquired
GETAREA	Yes			
GETAREA mass insert	Yes	Yes	Yes	Yes
GET move mode	Yes	*	A	*
GET locate mode		Yes	Yes	Yes
GET UPDATE	Yes	Yes	Yes	Yes
PUT UPDATE	B	C	B	C
PUT DELETE	B	C	B	C
PUT NEWREC	B	*	A	*
PUT subsequent mass insert	B	B	B	Yes
DELETE	D	*	Yes	*
SETL	E	Yes	Yes	Yes
RESETL	B	B	B	B
GETNEXT/GETPREV	B	B	B	B

Legend
* Acquired and released for this operation.
A Normally not set, but may be set in an error situation (DUPREC, ILLOGIC, IOERROR, NOTFND).
B Already done by previous operation.
C Already acquired and released by this operation.
D No, but VSWA includes data area.
E If move mode.

Notes:

1. *Acquiring a VSWA will put the transaction in a CICS wait if all the strings specified in the FCT for the file are already in use. This happens whether or not resources are shared.*

2. *Release of FWA, VSWA and shared strings, and the issuing of VSAM ENDREQ if required, are all performed on DFHFC TYPE=RELEASE or TYPE=ESETL.*

Figure 82. Summary of actions performed by DFHFCP VSAM requests

CICS record locking

If the FCT declares a file as protected, then a task can change a record on that file (by update, delete or add), but no other task may change the record until the first task has reached a sync point or has terminated. To achieve this, DFHFCP performs an ENQ on the record. There is no explicit DEQ; this is performed by the DEQALL issued in DFHSPP.

VSAM exclusive control

VSAM does not allow concurrent updates to the same control interval (CI), or to a CI in a control area (CA) that is being split by an update request to another CI. For example, if a request is made to GET UPDATE (including DELETE requests) a record that is being updated by another string, VSAM produces the return code X'14', which means **exclusive-control-conflict**.

When this happens, DFHFCP immediately reissues the request with an RPL message area. The address returned by VSAM is the address of the RPL which already has exclusive control of the CI. This address is used to find the VSWA in the base cluster control block chain. The failing VSWA is added to the chain (VSWAOWND) of VSWAs waiting for the executing task to relinquish exclusive control. The failing task waits on the ECB, VSWAEXW. When the RPL completes its operation, each task on this chain is posted, and each VSWA is removed from the chain of waits.

Note on variable length records

At the EXEC level, the record length is specified separately both for input and output operations. The user should not append an LLBB field, the FROM or INTO operand should identify the point immediately following the LLBB, and the value in the LENGTH operand should be the length of the data **excluding the length bytes.**

At the DFHFC level, the data comes and goes in FWAs for move-mode operations. For variable-length records, the data commences at FCUWA (offset X'10' in the FWA) with the LLBB field. The value of LLBB **includes the four length bytes.**

At the VSAM level, the length and the data are specified separately as at the EXEC level. The data address (excluding the length) is at VSWAREA, and the length (excluding LLBB) is at VSWALEN. These values match what the EXEC user originally specified.

Appendixes

Appendix A. Documentation required when submitting an APAR

The first section of this appendix lists documentation that should be submitted for **all** types of CICS problems. The second section deals with additional documentation for problems in specific areas of CICS.

By following these guidelines, you have a better chance of a quick response.

General documentation (for all types of problem)

A good rule of thumb when gathering documentation to submit with an authorized program analysis report (APAR) is to include anything which you would need yourself to do problem determination and diagnosis. As a further guide, however, consider the following recommendations.

APAR error description

The basic document that a program support representative (PSR) submits with an APAR is the "APAR Error Description." To make this document effective, it should contain all the descriptive details that might not appear in any listings.

Subject to the type of problem, the error description should include (for example):

- The frequency of the problem, for example once a day, once an hour, or every few minutes.
- The circumstances in which the problem occurs, for example, when a particular program (or version of program) is used, or under certain stress conditions.
- What traps (if any) were applied to produce the submitted listings.

All the above suggestions should be judged individually to ensure their validity for a particular problem being reported. Remember, too much information is always better than too little.

Listings

For all types of CICS problems the following items are normally required:

- The problem symptoms.
- A CICS formatted or address-space dump together with any relevant transaction dump(s).
- A CICS trace (preferably an auxiliary trace), but if an internal trace is supplied, it should contain about 2000 entries.
- Listings of CICS management modules (such as DFHTCP and DFHKCP).

  Note: These should be unnecessary if preassembled modules are used.
- Relevant CICS tables. Again, these should be unnecessary if preassembled versions are used.
- Any necessary listings of application programs.
- Console logs.
- CICS logs (for example, the CSMT log) wherever possible; these contain much information that is often overlooked, and are particularly useful when VTAM is in use.
- JCL listings. These may appear on dumps, and need not be sent twice.
- A list of PTFs and/or APARs applied. The System Modification Program (SMP) CICS control data set (CDS) provides this information and should be included.
- Details of any user modifications.
- Any ideas you have of the cause of the problem.

Additional documentation for problems in specific areas of CICS

For categorizing the documentation required for the diagnosis of CICS problems, we divide CICS into four major areas:

- System/general area
- Application area
- Data base area
- Data communication area.

The documentation required is described in the following sections.

System/General area

Storage control

- DFHSCP and DFHSCR listings
- Output using the storage-violation trap (transaction CSFE)
- A CICS dump taken using the SIT parameter DUMP = FULL, and SVD = nn, or SVD = NO (where applicable)
- For TIOA problems, the output using storage freeze (transaction CSFE)
- Auxiliary trace.

Task control, wait

- DFHKCP listing
- Application program(s) listing(s) associated with a wait problem
- Dump taken using the SIT parameter DUMP = FULL
- Auxiliary trace.

SYSGEN

- Input to stage 1
- Output from stage 1
- Run-time JCL listings.

System initialization

- DFHSIP or DFHSIA1-DFHSIJ1 listing causing problem
- DFHSIT listing
- Listing of DFHSIT override parameters
- Complete JCL listing for startup
- Dump, with trace active, using the SIT parameter DUMP = FULL.

Program control

- DFHPCP listing
- DFHPPT listing (if defined using macros)
- DFHPCT listing (if defined using macros)
- Application program listing associated with the problem
- Dump, with trace active, using the SIT parameter DUMP=FULL.

Resource definition online (RDO)

- CEDA input parameters
- Listing of the CICS system definition (CSD) file
- IDCAMS print of the CSD (where applicable)
- List of groups installed in the system
- Dump, with trace active, using the SIT parameter DUMP=FULL.

DFHCSDUP offline utility

- Input JCL, and associated output
- IDCAMS print of the CSD **before** and **after** failure (where applicable)
- MVS address-space dump.

Monitoring

- DFHCMP listing
- DFHMCT listing
- DFH$MOLS print of data
- Dump, with trace active, using the SIT parameter DUMP=FULL
- Auxiliary trace
- Hexadecimal print of monitor data set (where applicable).

Statistics

- Statistics output
- Dump, with trace active, using the SIT parameter DUMP=FULL.

XRF

- For terminal control problems, the CXRF transient data destination contains diagnostics.
- For the CICS availability manager (CAVM), send an SDUMP.

Application area

Mapping

- Application program listing associated with the problem
- Listings of maps and DSECTs
- Storage freeze, using the CSFE transaction
- Dump, with trace active, using the SIT parameter DUMP=FULL (**not** using CEDF)
- Special trap for fast-path problems, see your IBM Support personnel
- PROGCK information from terminal, for partition support problems
- Hard copy screen print(s), if available.

EXEC interface

- Application program(s) listing(s)
- User source on tape
- Punch output from translator on tape (option FE specified)
- Dump, with trace active, using the SIT parameter DUMP=FULL.

Translator

- Application program(s) listing(s)
- User source on tape
- Punch output from translator on tape (option FE specified)
- Output listing from translator (option FE specified)
- Output from compiler, if problem caused by bad input to compiler.

CEDF

- TRACE = YES in PCT entry for transaction CEDF
- Information using single terminal mode, unless problem happens only when using two terminals
- Dump, with trace active, using the SIT parameter DUMP = FULL
- Auxiliary trace, started before using CEDF.

Data base area

Emergency restart

- DFHRUP listing
- Print of system log (affected area only)
- Print of restart data set, **after** emergency restart
- Dump, with trace active, using the SIT parameter DUMP = PARTN.

DL/I interface

- DFHDBP listing
- DFHDLBP listing
- DFHDLI listing
- DFHDLR listing
- DFHDLQ listing
- DFHDLX listing
- Dump, with trace active, using the SIT parameter DUMP = PARTN.

File control

- DFHFCT listing
- Listing of the relevant file control module, if you know where the problem is.
- Dump, with trace active, using the SIT parameter DUMP=PARTN.

Journal control

- DFHJCP listing
- DFHJCT listing
- DFHJCO listing, if journal open problem
- DFHJCOCP listing, if journal open problem
- Print of journal data set
- Dump, with trace active, using the SIT parameter DUMP=PARTN.

Sync point

- DFHSPP listing
- DFHSPZ listing
- Listing of program issuing the sync point request
- Dump, with trace active, using the SIT parameter DUMP=PARTN.

Temporary storage

- DFHTSP listing
- Print of temporary storage data set (affected areas only)
- Dump, with trace active, using the SIT parameter DUMP=PARTN.

Transient data

- DFHTDP listing
- DFHDCT listing
- Print of intrapartition/extrapartition data sets (affected areas only)
- Dump, with trace active, using the SIT parameter DUMP=PARTN.

Data communication area

TCT TYPE = GEN

- Assembler output listing
- DFHTCT macro listing
- Tape and listing of user source statement input.

ZCP/VTAM terminals (CONNECT/DISCONNECT)

- Listing of user NEP (if applicable)
- VTAM internal trace with options API, PIU, SSCP, MSG
- Auxiliary trace, for whole problem
- Dump, with trace active, using the SIT parameter DUMP = FULL
- Active CSFE trace, with option ZCPTRACE (only if difficult to discover where the offending request was issued)
- Name and address of failing terminal
- Specify whether HPO = YES or NO.

ZCP/VTAM terminals (not CONNECT/DISCONNECT)

- VTAM buffer trace
- Auxiliary trace, for whole problem
- Active CSFE trace, with option ZCPTRACE
- Name and address of failing terminal
- Specify whether HPO = YES or NO.

IRC

- Dumps for **both** sides, with trace active and using the SIT parameter DUMP = FULL, which together contain all the IRC control blocks.
- CDS listing for **both** sides
- Auxiliary trace for **both** sides, taken at the same time
- CSMT/CSTL logs for **both** sides.

ISC

- Auxiliary trace for **both** sides, if CICS to CICS, taken at the same time
- VTAM buffer or VIT trace, depending on the type of problem
- Dump, with trace active, using the SIT parameter DUMP = FULL, for **both** sides if CICS to CICS.

TCP/BTAM terminals

- Active CSFE trace, with option ZCPTRACE, and, also, with option FETRACE if it is difficult to discover where the offending request was issued
- Auxiliary trace, if error does not show up in main-storage trace
- SIO/IO data trace for the error sequence
- Dump, with trace active, using the SIT parameter DUMP = FULL, taken as soon possible after the error
- DFHTCP listing
- DFHTACP listing
- DFHTEP listing
- DFHTCT listing
- Print of CSMT log.

Appendix B. Common user errors

The following list shows some of the user errors that are known to have caused problems:

- Incorrect system programming
- Restrictions on COBOL programs not followed
- CICS areas addressed incorrectly
- Rules for quasi-reentrancy not followed
- Transient data managed incorrectly
- File resources not released
- Storage corrupted by programs
- Return codes (from CICS requests) ignored
- Advice on performance considerations (in *Performance Guide*) not followed
- Restrictions on programs executing under the terminal control task not followed
- Errors in a CICS function request shipping environment
- Errors when using DL/I.

The following sections give examples of how these errors can occur.

Incorrect system programming

Some system programming errors are listed below:

- Incorrect I/O generation
- Wrong system initialization parameters leading to assumption that something is not working when in fact:
  - Program never got into library
  - Table filed under wrong suffix

- SIT not updated
- SIT updated but overridden in startup JCL
- CICS programs not altered for new function required.

- Resource definitions inadvertently replaced using the CEDA INSTALL transaction
- TWASIZE inadequate
- OSCOR value was too small, causing a loop in OPNDST SIMLOGON.

Attention to the following points may prevent errors occurring:

- For very large tables, an assembler error can occur because the assembler uses a default value for the conditional assembly loop counter, ACTR. To avoid this, set ACTR explicitly.
- The warning message IHB061, "DCB ddnames operand missing or invalid", that is issued when DFHJCT is assembled can be ignored. The ddname will be inserted at execution time.
- An end-of-file does not terminate input to a sequential file. Use the CSSF GOODNIGHT transaction.
- To cause the terminal to be disconnected when CSSN LOGOFF or GOODNIGHT is issued on a VTAM terminal, specify RELREQ=(NO,YES) in the DFHTCT macro.
- OPERSEC must not be specified in the TCT (except by default of 1) for any terminal that can have CSSN entered. If the security has a preset value in the TCTTE, and an attempt is made to sign on, the message DFH3526 is issued.
- OPERRSL must not be specified in the TCT for any terminal that can have CSSN entered.
- INBFMH=ALL must be specified for CSMI, CSM1, CSM2, CSM3 or CSM5 in the PCT.
- The name "CICS" must be specified in the subsystem name table (see the *Installation Guide*).
- CICSSVC must be link-edited into the SVCLIB as a type 2 SVC, and the SVC number entered in the SVC name table. The local-lock option must not be specified.
- If using resource definition online, invoke the CEDA CHECK transaction.

Restrictions on COBOL programs not followed

The *Application Programmer's Reference* and the *CICS/VS Application Programmer's Reference Manual (Macro Level)* tell you the restrictions you must observe when writing COBOL programs to use in a CICS environment. If you do not, you may have the following errors:

- The CICS partition runs out of GETMAIN space. This happens because some COBOL modules perform a FREEMAIN only when a job is finished.
- The CICS partition can cancel with a COBOL error message.
- CICS goes to the end of the job without an error message, or a dump. This happens if a COBOL application does not terminate with an EXEC CICS RETURN command or GOBACK statement, and the compiler added a STOP RUN statement.

CICS areas addressed incorrectly

Ways in which CICS areas can be addressed incorrectly:

- Not loading control block fields (such as TIOABAR and TDIABAR) in the correct manner immediately before or after a CICS service request as appropriate.
- Not recognizing that some address fields (such as TCAFCAA and TCATDAA) are in fact the same, and therefore that a DFHFC TYPE=GET will corrupt the address returned in a DFHTD TYPE=GET if the field has not been saved.
- Not recognizing that a BMS (OUT,SAVE) request alters TCTTEDA, so that the user must set a separate address (pointer) for the saved area.
- Not realizing that the DFHTDIA, DFHTSIOA, and DFHTDOA DSECTs define 12 bytes before the user data, but that the user must **ask** for the last **four** of these (the variable-length prefix) as part of a DFHSC TYPE=GETMAIN.
- Not realizing that a COBOL program with a LINKAGE section structure (TCA and TWA, for example) that exceeds 4K bytes requires definition and the setting of more than one contiguous BLL cell.
- Not realizing that a TIOA has been freed. Suppose an application program saves the TIOA address, writes to a terminal without specifying the SAVE option, and then restores the TIOA address to TCTTEDA. A subsequent operation on that terminal could use the TIOA that has already been freed.

Rules for quasi-reentrancy not followed

When using the high-level language macro-level interface, PL/I STATIC storage can be used as work areas only between CICS requests; it cannot be assumed that any modifiable fields (as opposed to read-only constants) retain their values across a CICS service request. It is recommended that these areas be reserved for read-only information.

Transient data managed incorrectly

The conditions for automatic task initiation (ATI) are:

- Queue record count is greater than or equal to trigger level; and
- No task is already initiated (TDTIBM in DCT entry is off).

These conditions are only tested:

- After a new record has been added to the queue; or
- During emergency restart.

When ATI occurs, TDTIBM is set on. It is only set off:

- On reaching QUEZERO; or
- By a PURGE request; or
- During emergency restart; or
- When a task completes.

Modifying the trigger level using CEMT does not cause a task to be initiated immediately.

A task initiated as a result of the trigger level being exceeded on a transient data queue should read that queue to exhaustion (that is, until QUEZERO is signaled) or should purge it. If DESTFAC = FILE and the queue is empty, another task cannot be initiated, even though the trigger level is reached.

The input area used for DFHTD TYPE = GET macros must not be used for anything else, or be freed by the user. Processing of the input area must be complete before the next DFHTD request is made. If the contents of the input area are to be written out, they must first be copied to a separate transient data output area (TDOA). Any independent DFHTD TYPE = PUT operation may also cause destruction of the transient data input area (TDIA) contents, because the TDIA is also used as a work area in chain record handling for intrapartition queues.

File resources not released

A DFHFC TYPE=RELEASE **must** be issued if a DFHFC TYPE=GET or TYPE=PUT, TYPOPER=UPDATE fails. This is imperative when accessing VSAM files or when files are recoverable, because otherwise the use of VSAM strings or access to specific records is restricted by ENQ for unduly long periods.

Storage corrupted by programs

Ways in which storage can be corrupted:

- Insufficient TWA. For further information, see "Debugging problems caused by overwriting LIFO stack information" on page 272. (The LIFO storage mechanism is described in "Chapter 3.2. Storage management" on page 259.)
- Runaway loops through arrays.
- Inconsistency between GETMAIN area sizes and overlays (COBOL LINKAGE SECTION areas, PL/I BASED structures, assembler DSECTs).
- Failure to allow for padding between aligned fields.
- Implied lengths for MVCs in assembler language may cause overlong data to be moved; this is particularly dangerous with multiply-defined areas in assembler programming, and, to a lesser extent, with COBOL or PL/I programming.
- Corruption or destruction of RDIDADR area indirectly referenced by a DFHFC TYPE=GETNEXT for VSAM files. For safety, never alter this area (1) between a GET/UPDATE and the corresponding PUT/RELEASE, (2) between a SETL and the corresponding ESETL, or (3) between GETAREA/MASSINSERT and RELEASE. If two files have to be handled, use two RDIDADR areas.
- Insufficient terminal control table user area (TCTUA). If the program uses a TCTUA, and it is too small or not specified, other TCTUAs, or the TCT wait list, can be destroyed.

Return codes (from CICS requests) ignored

As good programming practice (especially in a telecommunications environment), the result of **every** invocation of a CICS service should be checked. The minimum action should be to use the NORESP (normal response) or ERROR operands to branch to an error routine that abends the task. If special action is taken for a particular return code, remember to test for the other return codes.

If a DFHPC TYPE=ABEND macro is invoked in response to an error return code, the macro expansion saves 6 bytes of TCA information that would otherwise be corrupted by the abend request. The 2 bytes overwritten by the abend request code are saved at TCACCSV1 and the four bytes overwritten by the abend code are saved at TCACCSV2.

For the IBM 6670 Information Distributor (Logical Unit Type 4), when CICS receives a SIGNAL command, which forces a change of direction, CICS will ensure that the application program does not continue to send. If the application issues a SEND after the SIGNAL condition has been raised, control is returned without performing the output operation and the IGREQCD condition is raised. If another SEND is issued, then the transaction is abnormally terminated with ABEND code ATCS.

Advice on performance considerations not followed

The *Performance Guide* gives advice on good application programming practice for performance. Some examples of poor programming practice in this respect are listed below.

- Programs modularized the wrong way:
  - Browse and direct-access functions are in the same program when they are logically separate and could be switched by XCTL.
  - LINKs from within a frequent loop.
- Too many records browsed in one task.
- Use of DFHTC (or DFHIC) TYPE=WAIT where it can be avoided by redesign.
- Invoking a complex subroutine to do what is usually a simple test – eliminate the simple case first.
- Repeated conversion of data; use work fields to hold data that has been converted for use, if the converted form is needed several times.
- Repeated subscripted references as a source of data; it is better to transfer the data once into a work field and avoid wasting processing power in unnecessary array address calculations.
- Data not defined in its most readily usable form; for example, do not make array indexes zoned decimal.
- OPENLST used for BSC lines in the TCT (though they may be mandatory in VM/MVS configurations).
- AUTOPOLL not used where available for start-stop.
- VERIFY used unnecessarily on advanced technology DASD files.
- Unreasonable transaction, terminal and operator priority values.
- Tables poorly ordered.
- Maximum task too high.
- Maximum active task too low.

- Virtual storage address space too large or too small, leading, respectively, to too much paging, or too much program compression and too many stalls.

Some suggestions:

- Be aware that DFHFC GET/UPDATE and any activity on recoverable resources implies a CICS ENQ. Ensure that tasks cannot be deadlocked because of crossed ENQs – insist on a standard order in which resources are acquired.

- Always free resources as soon as possible, and acquire them as late as possible, but do not go extremes; for example, do not free 200 bytes and GETMAIN them after 20 in-line instructions. If all you have to do is RETURN to CICS, let CICS free all that is outstanding. Otherwise free what you can before you do an I/O operation.

  Where CICS recovery forces retention of resources to end-of-task (or, in rare cases, to user sync point), reduce resource retention by acquiring as late as possible. That is, issue the GET for UPDATE only after the completion of other work such as editing and table lookups. Then issue the PUT/UPDATE macro, and return.

- Never cause operating system WAITs unless the transaction has very low priority and does a DFHKC TYPE=WAIT,DCI=DISP first.

  Note: DFHTD PUTs to extrapartition files (QSAM).

Restrictions on programs executing under the terminal control task not followed

There are severe restrictions on code included in:

- DFHKCP user exits
- DFHZCP user exits
- DFHTCP user exits
- Initialization PLT programs.

In particular:

- No WAITs/ENQs/SUSPENDs, explicit or implicit, may be issued.
- All possible requests must be conditional, for example, GETMAIN or LOAD.

If these restrictions are ignored, CICS abends will occur with messages such as DFH0501, DFH0401, or DFH0601.

Errors in a CICS function request shipping environment

Intersystem communication only

Errors:

- The TCT generation and the VTAM generation do not contain the same logical unit names.
- The TCTs have been generated incorrectly; one system must have SESTYPE = SEND and the other system must have SESTYPE = RECEIVE.
- The TCTTEs generated for the local and remote systems are both primary or both secondary.
- ACF/VTAM is not being used.

Hint:

- If a START command with the NOCHECK option is issued, then the application program may not receive an error response from a remote system. The EIBRCODE (response code) may be set to zero (normal response) although errors did occur on the remote system.

Intersystem communication and multiregion operation

Errors:

- The mirror transactions have not been generated or CSMI, CSM1, CSM2, CSM3 and CSM5 have been generated without INBFMH = ALL,DTB = YES specified in both systems. (Use DFHPCT TYPE = GROUP, FN = ISC and DFHPPT TYPE = GROUP, FN = ISC.)
- The resources defined as remote in the local system were not defined properly in the remote system.
- The code for intersystem communication has not been generated. (At system generation time, use DFHSG PROGRAM = TCP, VTAMDEV = LUTYPE6, and/or ACCMETH = IRC.)

Hints:

- DFHIRP must be in the link pack area.
- If transaction security is used on the remote system, DFHTCT TYPE = SYSTEM must have OPERSEC specified to match the security code in the PCT entry.

Errors when using DL/I

Attention to the following points may prevent errors:

- When using the shared data base, DFHXFP must be generated with DL/I support.
- Modules with DL/I-level dependencies must be assembled with the correct level of DL/I.
- When using batch shared data base, the value specified in SESNUMB in the TCT must be one more than the number of sessions required. This number must be allowed for in calculating the value of DLTHRED.
- The default value of OSCOR (8192) is too small for any system using IMS/VS. Choose a large value, and reduce it as far as possible, rather than trying to specify the exact value.
- If you are not using MVS/XA Data Product Facility (multiple pool support), VSAM files are defined using the VSAM shared resource pool, and VSAM data bases are used, the value of STRNO as specified in the DFHFC TYPE=SHRCTL macro must be large enough to allow for all DL/I requirements. DL/I needs DLTHRED+1 strings as a minimum. IMS/VS obtains its own buffers and DL/I requirements do not need to be reflected in the BUFFERS parameter.
- If your IMS/VS system has been changed by maintenance, you are advised to reassemble the corresponding CICS modules.

Appendix C. List of abbreviations

This list is primarily intended to give definitions of the most frequently used abbreviations. "Chapter 3.4. Storage areas and tables" on page 283 (which gives descriptions, in alphabetic order, of CICS storage areas) may also be useful to the reader seeking explanations of abbreviated names.

No attempt is made here to define the abbreviations used in CICS commands and macros.

ACB access method control block (VTAM and VSAM)

ACF advanced communications function

ACP abnormal condition program

AFCB authorization facility control block

AFCS authorized function common control block

AICB application interface control block

AID automatic initiate descriptor (CICS)

AID attention identifier (VTAM)

AIX alternate index (VSAM)

ALT application load table

APAR authorized program analysis report

ASCB address space control block

ASXB address space extension block

ATI automatic task initiation

BCA batch control area

BCCB base cluster control block

BDAM basic direct access method

BLL base locator linkage (COBOL)

BMP batch message processing program (IMS/VS)

BMS basic mapping support

BSC binary synchronous communication

BTAM basic telecommunications access method

CA control area

CAVM CICS availability manager

CCE console control element

CDBLK CONVDATA block

CDS control data set

CEC central electronic complex

CI control interval

CLT command list table

CRB CICS cross-region block

CSA common system area

CSD CICS system definition

DASD direct-access storage device

DB data base

DBD data base definition (DL/I)

DBLDS dynamic buffer area

DBR dynamic backout record

DBRC data base recovery control

DCA dispatch control area

DCB data control block (operating system)

DCI dispatch control indicator

DCT destination control table

DDIR DMB directory (DL/I)

DECB data event control block (operating system)

DLP DL/I interface parameter list

DL/I data language − one

DMB data management block (DL/I)

DRCA dependent region control area

DRX DL/I resource block

DSA dynamic storage area

DWE deferred work element

ECB event control block (operating system)

EDF execution diagnostic facility (command-level)

EIB exec interface block

EIP exec interface program

EIS exec interface storage

EOT end of transmission

EPB exit program block

EPL exit program link

ESDS entry sequenced data set (VSAM)

ESTAE extended specify task asynchronous exit

FAQE free area queue element

FBWA file browse work area

FCP file control program

FCT file control table

FCTE file control table entry

FE field engineer(ing) (IBM)

FIOA file input/output area

FMH function management header (SNA)

FWA file work area

GDS	generalized data stream
GTF	generalized trace facility
HLL	high-level language
HPO	high-performance option
HTA	HPO transaction area
I/O	input/output (from and to terminals, primarily)
ICE	interval control element
ICV	region exit time interval
ICVTSD	terminal scan delay
ID	identifier
IMS	information management system
IRC	interregion communication
IRLM	IMS resource lock manager
IRS	interrecord separator
IRSDS	interregion communication subsystem blocks
ISA	initial storage area (PL/I)
ISB	interface scheduling block (DL/I)
ISC	intersystem communication
JC	journal control
JCA	journal control area
JCL	job control language
JCR	journal control record
JCT	journal control table
JCTTE	journal control table entry
K	called "kilo" (1000), but means 1024
KCP	task control program
KP	key point

KSDS key-sequenced data set

LDC logical device code

LECB line event control block

LERAD logical error address

LFSE LIFO stack entry

LFSS LIFO stack segment

LIFO last-in/first-out (storage)

LIOA line input/output area

LLA load list area

LSR local shared resources

LUC LU6.2 control block

LUM LU6.2 macro-level control block

LUW logical unit of work

M called "mega" (1000000), but means 1048576

MCB message control block

MCR message control record

MCT monitoring control table

MRO multiregion operation

MSG message

NAB next available byte pointer

NACP node abnormal condition program

NCCF Network Communications Control Facility

NCP network control program

NIB node initialization block (VTAM)

NL new line

NLT nucleus load table

NSR nonshared resources

OLTEP online test executive program

OSPWA output services processor work area

PAM page allocation map

PCB program communication block (DL/I)

PCP program control program

PCT program control table

PDIR PSB directory (DL/I)

PEP program error program (usually user-written)

PGT program global table

PIDS component name in short symptom string

PIE program interrupt element (operating system)

PLT program list table

PPT processing program table

PRA primed storage area

PSB program specification block (DL/I)

PSDDS partition set data area

PSR program support representative

PST partition specification table (DL/I)

PSW program status word

PTF program temporary fix

PXT program translation table

QCA queue control area

QEA queue element area

RACF resource access control facility

RBA relative byte address

RIDS module name in short symptom string

RLA route list area

RLN relative line number

RMSR recovery management support recording (BTAM)

RPL request parameter list

RSA register save area

RSB remote scheduling block (DL/I)

SAA storage accounting area

SCD system contents directory (DL/I)

SCP storage control program

SCR storage recovery program

SCXDS extended storage anchor block

SDT series definition table

SDWA system diagnostic work area

SECDS external security table

SIT system initialization table

SMI standard message indicator

SMP System Modification Program

SNA systems network architecture

SNT sign-on table

SOS short on storage

SPIE specify program interrupt element

SQE subtask queue element

SRA SRB interface control area

SRB service request block

SRP system recovery program

SRT system recovery table

SSF software support facility

SVC supervisor call

TACB transaction abend control block

TACLE terminal abnormal condition line entry

TC terminal control

TCA task control area

TCAM telecommunications access method

TCB task control block

TCP non-VTAM terminal control program

TCT terminal control table

TCTLE terminal control table line entry

TCTME terminal control table mode entry

TCTSE terminal control table system entry

TCTTE terminal control table terminal entry

TCX TCA extension for LU6.2

TDIA transient data input area

TDOA transient data output area

TDP transient data program

TGT task global table

TIE transaction interface element

TIOA terminal input/output area

TMDEL table management directory element

TMELD table management lock block

TMP table management program

TMRQ table management parameter list

TMSKT table management scatter table

TMSSA table management static storage area

TOLTEP terminal online testing error program

TP teleprocessing (subpool)

TPE terminal partition extension

TR transaction restart

TS temporary storage

TSGID temporary storage group identifier

TSIOA temporary storage input/output area

TST temporary storage table

TSUT temporary storage unit table

TTP terminal type parameter

TTR track/record (disk address)

TUL tape user label

TWA transaction work area

UET user exit table

UIB user interface block (DL/I)

URD unit of recovery descriptor

URL user route list

VCP volume control program

VOLDS volume descriptor

VSAM virtual storage access method

VSCA VSAM subtask control area

VSWA VSAM work area

VTAM virtual telecommunications access method

WRE write request element

XFSTG transformer storage for CICS function request shipping

XLT transaction list table

XRF extended recovery facility

XTSTG transformer storage for transaction routing

ZCP VTAM terminal control program and common VTAM and non-VTAM routines

ZEPD ZCP module list

Index

C

E

F

G

H

I

J

K

L

Q

R

S

T

U

V

W

X

Z

Numerics

Customer Information Control System
CICS/MVS Version 2 Release 1
Problem Determination Guide

READER'S COMMENT FORM

Order No. SC33-0516-0

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative, or the IBM branch office serving your locality.

Number of latest Technical Newsletter for this publication...

Note: Staples can cause problems with automated mail-sorting equipment. Please use pressure-sensitive or other gummed tape to seal this form.

If you want an acknowledgement, give your name and address below.

Name .....

Job Title ..... Company .....

Address .....

..... Zip .....

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

SC33-0516-0

Reader's Comment Form

Fold and tape | Please do not staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

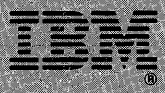
BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 6R1H
180 Kost Road
Mechanicsburg, PA 17055, USA

Fold and tape | Please do not staple | Fold and tape

IBM®



Problem Determination Guide
SC33-0516-0

Version 2.1

CICS/MVS

Program Number
5665-403

Printed in U.S.A.

SC33-0516-00