SC27-0433-5
File No. S370/4300-30

Network Communications Control Facility

Program Product

Customization

Program Number 5735-XX6

Release 2

IBM

Notice to Network Communications Control Facility (NCCF) Users

NCCF Release 1 Users:

- If you are not migrating to Release 2, continue to use your current Release 1 book.
- If you are migrating to Release 2, use this book for planning for Release 2, but continue to use your current book for NCCF Release 1 information.

When ordering additional Release 1 copies, use order number, ST27-0433-0

NCCF Release 2 Users:

Use this book.

Sixth Edition (July 1982)

This is major revision of, and obsoletes, SC27-0433-4. This edition applies to the Release 2 of the Network Communications Control Facility (NCCF), program number 5735-XX6, an IBM program product. The program product described in this manual, and all licensed material available for it, are provided by IBM under terms of the Agreement for IBM Licensed Programs. Your branch office can advise you on ordering procedures.

Before using this publication in connection with the operation of IBM systems consult your IBM representative to find out which editions are applicable and current.

Copies of this and other IBM publications can be obtained through IBM branch offices.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

A form for reader's comments has been provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Department E03, P.O. Box 12195, Research Triangle Park, North Carolina, U.S.A. 27709. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Preface

This manual describes the programming procedures for tailoring and modifying the Network Communications Control Facility (NCCF). It is intended for the system programmer who defines the IBM-supplied NCCF program product and decides in what ways the program can be customized to suit the individual requirements of an installation.

How This Book Is Organized

Chapter 1,"Introduction," is a brief overview of NCCF customization and describes the syntax rules that are observed throughout the manual.

Chapter 2, "Command Lists," describes how control statements, access method commands, NCCF commands, user-written commands, and other command lists can be coded into command lists that are invoked by name when required.

Chapter 3, "Service Facilities and Macro Instructions," describes the NCCF service facilities and macro instructions that can be used in coding command processors, exit routines, and subtasks. This chapter is a prerequisite to Chapters 4, 5, and 6.

Chapter 4, "Command Processors," describes the service facilities and macro instructions that can be used to code command processors. The operating environment and control block considerations are discussed in detail.

Chapter 5, "Exit Routines," describes the location and uses of exit routines throughout NCCF, as well as guidelines for user-coded exit routines. A sample exit routine is included.

Chapter 6, "Subtasks," describes the procedures for coding a user-written subtask. A sample subtask is shown.

Appendix A, "Command Summary," summarizes all commands available to NCCF operators and the places from which the commands can be entered.

Appendix B, "NCCF Log and Hard-Copy Log," describes the JCL, formats, and output of the online disk log, as well as the hard-copy log that records operator transactions.

Appendix C, "NCCF Control Blocks," shows the formats of the control blocks that are of concern to the NCCF user in customizing NCCF.

Appendix D is an example of a user-written command processor.

Appendix E contains two examples of user-written data services command processors.

Appendix F is an example of a user-written full-screen command processor.

Appendix G is a glossary that defines terms and abbreviations applicable to NCCF.

Prerequisite Publications

Readers should be familiar with the basic functions and capabilities of NCCF as described in:

Network Communications Control Facility General Information, GC27-0429

In addition, the following publication explains the program structure of NCCF and how the program is installed:

Network Communications Control Facility Installation, SC27-0430

The following publication describes NCCF operator commands:

Network Communications Control Facility Terminal Use, SC27-0432

Publications for Additional Information

Advanced Communications Function for VTAM (ACF/VTAM) programmers may also find useful the information in:

ACF/VTAM Programming, ST27-0449 (formerly SC27-0449): ACF/VTAM Version 1 Release 2

ACF/VTAM Programming, SC27-0449: ACF/VTAM Version 1 Release 3

ACF/VTAM Version 2 Programming, SC27-0611

ACF/VTAME Programming, SC27-0442

Advanced Communications Function for TCAM (ACF/TCAM) programmers may find useful information in:

ACF/TCAM Application Programming, SC30-3135: ACF/TCAM Version 2

The following publications contain information on 3270 systems:

3271, 3272, 3275 Control Unit Description and Programmer's Guide, GA23-0060

3274 Control Unit Description and Programmer's Guide, GA23-0061

3276 Control Unit Description and Programmer's Guide, GA18-2081

Notes on Terms Used in This Manual

Throughout this publication, unless otherwise noted, references to ACF/VTAM include ACF/VTAM Version 1 Release 2, ACF/VTAM Version 1 Release 3, ACF/VTAM Version 2, and ACF/VTAME. References to ACF/TCAM include ACF/TCAM Version 2, Releases 2, 3, and 4.

References to ACF/VTAM and ACF/TCAM cross-domain environments presume the presence of the Multisystem Networking Facility or ACF/VTAM Version 2. ACF systems can, of course, also operate with a single domain.

References to VSE in this manual refer to the DOS/VSE system control programming with the VSE/Advanced Functions program product.

References to VSE/OCCF in this manual refer to the Virtual Storage Extended Operator Communications Control Facility program product.

The term network has at least two meanings. A *public network* is a network established and operated by common carriers of telecommunication Administrations for the specific purpose of providing circuit-switched, packet-switched, and leased- circuit services to the public. A *user application network* is a configuration of data processing products, such as processors, controllers, and terminals, established and operated by users for the purpose of data processing or information exchange, which may use transport services offered by common carriers or telecommunication Administrations.

Network, as used in this publication, refers to a user application network.

Other terms used in this manual are defined in the glossary.

Contents

Chapter 1. Introduction 1-1
Syntax Used to Describe Macro Instructions 1-1

Chapter 2. Command Lists 2-1
Introduction to Command Lists 2-1
- What Is a Command List? 2-1
- What Can Command Lists Help You Do? 2-1
- Who Can Use Command Lists? 2-1

Naming and Filing Command Lists 2-2
Defining Command Lists to NCCF 2-2
- Defining Command Lists in OS/VS 2-2
- Defining Command Lists in VSE 2-2

Invoking Command Lists 2-3
- Invoking a Command List from an Operator Terminal 2-3
- Invoking a Command List from a User-Written Command Processor 2-3
- Invoking a Command List from Another Command List 2-3
- Invoking a Command List from an Access Method Message 2-4

Command Lists Running under the Primary POI Task 2-5
The Command List Language 2-6
Coding Guidelines 2-6
Variables 2-6
- Parameters 2-7
- Control Variables 2-8
- User Variables 2-9

Command List Labels 2-9
Command List Statements 2-9
- Null Statements 2-10
- Comments 2-10
- Commands 2-10
- Assignment Statements 2-11
- Control Statements 2-11
  - &BEGWRITE 2-12
  - &CONTROL 2-12
  - &EXIT 2-13
  - &GOTO 2-13
  - &IF 2-13
  - &PAUSE 2-15
  - &WRITE 2-16

Command List Built-In Functions 2-16
  - &CONCAT 2-16
  - &LENGTH 2-17
  - &SUBSTR 2-17
  - &NCCFID 2-18
  - &NCCFSTAT 2-18

How NCCF Analyzes a Command List 2-19
Sample Command Lists 2-21
Sample Conditional Command Lists 2-22
- Sample 1 2-22
- Sample 2 2-24
- Sample 3 2-25

Chapter 3. Service Facilities and Macro Instructions 3-1
Service Facilities Guide 3-1
- Main Vector Table Addressability 3-1
- Control Block Considerations 3-4
  - DSICBS Macro Instruction 3-6
  - Service Work Block (SWB) 3-6
  - Task Vector Block (TVB) 3-7
  - Buffer Header (BUFHDR) 3-7
  - Example of BUFHDR Usage 3-9
  - Internal Function Request (IFR) 3-9
  - Control Block Header (CBH) 3-9
  - Parse Descriptor Block (PDB) 3-10
- Getting and Freeing Storage 3-11
- Getting, Freeing, and Locating a Control Block 3-12
- Disk Services 3-12

Presentation Services 3-13
DSIPSS 3-14
Message Queuing 3-15
Resource Location (ACF/VTAM Only) 3-16
Macro Reference 3-18
DSICBS Macro Instruction 3-18
DSICES Macro Instruction 3-19
DSIDATIM Macro Instruction 3-20
DSIDEL Macro Instruction 3-20
DSIDKS Macro Instruction 3-21
DSIFRE Macro Instruction 3-23
DSIGET Macro Instruction 3-24
DSIKVS Macro Instruction 3-26
DSILCS Macro Instruction 3-27
DSILOD Macro Instruction 3-29
DSIMBS Macro Instruction 3-30
DSIMDS Macro Instruction 3-33
Format 1: Start Message 3-33
Format 2: Message Text 3-33
Format 3: End Message 3-34
DSIMQS Macro Instruction 3-34
DSIOIS Macro Instruction 3-36
DSIPAS Macro Instruction 3-36
DSIPOS Macro Instruction 3-37
DSIPRS Macro Instruction 3-38
DSIPSS Macro Instruction 3-39
DSIRDS Macro Instruction (ACF/VTAM Only) 3-46
DSISSS Macro Instruction (ACF/VTAM Only) 3-47
DSIWAT Macro Instruction 3-49
DSIWCS Macro Instruction 3-50
DSIWLS Macro Instruction 3-50
Data Services Macro Instructions 3-51
DSIZCSMS Macro Instruction 3-51
DSIZVSMS Macro Instruction 3-53

Chapter 4. Command Processors 4-1
Operating Environment 4-2
Regular Command Processors 4-2
Immediate Command Processors 4-3
Both Regular and Immediate Command Processors 4-4
Command Processors Executed Under the Primary POI Task (PPT) 4-4
Command Processors Executed Under a Data Services Task (DST) 4-5
Control Block Considerations 4-6
Command Work Block (CWB) 4-6
System Command Entry (SCE) 4-9
Data Services Request Block (DSRB) 4-9
Invoking a Command Processor 4-11
Obtaining a Command Work Block (CWB) 4-11
Obtaining a Service Work Block (SWB) 4-11
Building a Command Buffer 4-12
Obtaining a Parse Descriptor Block (PDB) and Parsing the Command 4-12
Looking Up the Command Processor Address 4-12
Calling the Command Processor 4-12
Initial DSCP Invocation 4-13
Passing a Command to Another Subtask in the Same Domain 4-13
Forwarding a Command to Another Domain for Execution 4-13
Returning a Command to Another Domain for Output 4-15
Passing Commands to the Access Method 4-15
Output 4-15
Regular Commands 4-15
Immediate Commands 4-16
Completion of a VSAM I/O Request 4-16
Completion of a CNM I/O Request 4-17
Completion of Receipt of Unsolicited CNM Data 4-18
Full-Line Command Processor Considerations 4-19
NCCF Title-Line Processing 4-19
Coding Requirements 4-20
Full-Screen Command Processor Considerations 4-20
Types of Full-Screen Command Processors 4-20
Operations of a Full-Screen Command Processor 4-21

Asynchronous Full-Screen Command Processors 4-21
Asynchronous Full-Screen Command Processor Parameter List 4-21
Processing Asynchronous Full-Screen Input 4-22
Testing if NCCF Events have Occurred 4-22
Noninterruptible Command Processors 4-22
Ending an Asynchronous Full-Screen Command Processor 4-22
Canceling an Asynchronous Full-Screen Command Processor 4-22
For More Information 4-22
Synchronous Full-Screen Command Processors 4-23
Synchronous Full-Screen Command Processor Parameter List 4-23
Processing Synchronous Input 4-24
Establishing a Full-Screen Subroutine 4-24
Ending a Synchronous Full-Screen Command Processor 4-24
General Guidelines 4-24
Screen Formatting for the 3270 Data Stream 4-24
The Escape Key 4-25
The Reshow Option 4-25
The Reshow Key 4-25
Logging Full-Screen Input/Output 4-25
DSIPSS Return Code from a Full-Screen Command Processor 4-25

Chapter 5. Exit Routines 5-1
What Can NCCF Exit Routines Do? 5-1
Overview of NCCF Exit Routines 5-1
DSIEX01: Input from the Operator 5-4
DSIEX02: Output to the Operator 5-5
DSIEX03: Input Before Command Processing 5-5
DSIEX04: Log Output 5-6
DSIEX05: Before Output to the Access Method 5-6
DSIEX06: Solicited Message Input from the Access Method 5-6
DSIEX07: Before Cross-Domain Output 5-7
DSIEX08: Before Cross-Domain Input Processing 5-7
DSIEX09: Output to the System Console 5-7
DSIEX10: Input from the System Console 5-8
DSIEX11: Unsolicited Access Method Messages 5-8
DSIEX12: Logon Validation 5-8
DSIEX13: OST/NNT Message Receiver 5-9
DSIEX14: Before Logoff 5-9
DSIEX15: Before Logoff with MVX/OCCF or VSE/OCCF 5-9
XITDI: Data Services Task (DST) Initialization 5-10
XITCO: CNM Interface Output 5-10
XITCI: CNM Interface Input 5-10
XITVN: VSAM Empty Data Set 5-11
XITVI: VSAM Input 5-11
XITVO: VSAM Output 5-11
DSITRE: ACF/TCAM Read 5-11
Installation 5-13
Coding Guidelines 5-13
Input Parameters 5-14
Registers 5-14
Control Block Considerations 5-14
User Exit Parameter List (DSIUSE) 5-15
Service Work Block (SWB) 5-17
Output Parameters 5-17
Exit Routine Prototype 5-18
Sample User-Written Exit Routine 5-20

Chapter 6. Subtasks 6-1
Why Write Your Own Subtask? 6-1
Defining the Subtask to NCCF 6-1
Subtask Organization 6-1
Requirements 6-3
Coding Guidelines 6-3
Entry and Exit Linkage 6-3
Subtask Attachment 6-3
Indicating that the Subtask is Ready 6-4
Subtask Termination 6-5
Optional Facilities 6-5
LIST Command 6-5
Queued Storage Management 6-6

Reading the Subtask Initialization Deck 6-6
Logging Messages 6-7
Issuing Messages 6-7
Receiving Messages 6-7
Freeing DSIMQS Buffers 6-8
Command Processing 6-9
Control Block Considerations 6-9
Main Vector Table (MVT) 6-9
Task Vector Block (DSITVB) 6-11
Task Information Block (DSITIB) 6-13
Sample User-Written Subtask 6-14

Appendix A. Command Summary A-1

Appendix B. NCCF Log and Hard-Copy Log B-1
NCCF Log B-1
NCCF Hard-Copy Log B-5

Appendix C. NCCF Controls Blocks C-1
How to Read Data Maps C-1

Appendix D. Sample User-Written Command Processor D-1

Appendix E. Sample Data Services Command Processors E-1
DSITDSRD Command Processor E-1

Appendix F. Sample Full-Screen Command Processor F-1

Appendix G. Glossary of Terms and Abbreviations G-1

Index X-1

Figures

2-1. Control Variables for Command Lists 2-8
2-2. Summary of Control Statements for Command Lists 2-11
2-3. Built-in Functions for Command Lists 2-16
2-4. How NCCF Analyzes a Command List 2-20
3-1. Summary of NCCF Macro Instructions 3-2
3-2. Overview of the Control Blocks Used by NCCF Service Routines 3-5
3-3. Buffer Header (BUFHDR) 3-8
3-4. Use of NCCF Macro Instructions for Communication from an 3-13
3-5. Examples of Using the DSIPSS Macro Instruction 3-15
3-6. Table Field Relationships 3-17
3-7. Search of the Span Name Table (DSISNT) 3-48
4-1. Example of Program Design for Data Services Requests 4-6
4-2. Command Processor Input Parameter Control Blocks 4-7
4-3. Example of DSCP Processing Logic 4-14
4-4. Effect of Command Processor Return Codes for Terminal-Originated Commands 4-16
4-5. Example of Full-Line Title-Line Output 4-20
4-6. Sample 3270 Data Stream 4-23
4-7. Interpreting the TVBRESET and TVBPNMOD Bits 4-26
5-1. NCCF Exit Routine Interfaces 5-2
5-2. Environment of NCCF Exit Routines 5-3
5-3. Message Formats for DSITRE: ACF/TCAM Read 5-12
5-4. Return Codes Set by Exit Routines 5-18
6-1. Subtask Organization 6-2
6-2. Subtask Input Parameter Control Blocks 6-4

Summary of Amendments (July 30, 1982) to SC27-0433-4 by Revision SC27-0433-5

NCCF changes have been made to the control blocks to support Network Logical Data Manager (NLDM).

Changes have been made to the DSIPSS macro instruction and to the sections on full-screen command processors to clarify synchronous and asynchronous full-screen information.

A new exit routine, DSITRE: ACF/TCAM Read, has been added.

Various technical and editorial changes have been made.

Summary of Amendments (March 18, 1982) to SC27-0433-4 by TNL SN31-0802

Information on using Terminal Access Facility commands in a command list has been added to Chapter 2.

Appendix A has been revised.

Various technical corrections and clarifications have been made.

Summary of Amendments (April 30, 1981) to SC27-0433-3 by Revision SC27-0433-4

The order and contents of several chapters have been changed as follows:

Chapter 4, "Command Lists," has been made Chapter 2.

The section of Chapter 2 titled "Service Facilities and Macro Instructions" has been expanded and made Chapter 3.

The section of Chapter 2 describing command processors has been made Chapter 4.

Chapter 3, "Exit Routines," has been extensively revised and made Chapter 5.

A new chapter, "Subtasks" has been added as Chapter 6.

The data areas in Appendix C have been updated.

This edition also incorporates various technical corrections and clarifications.

Chapter 1. Introduction

This manual discusses those Network Communications Control Facility (NCCF) features - command lists, command processors, exit routines, and subtasks - that allow an installation to customize NCCF to fit its own requirements.

- **Command lists.** Control statements, NCCF commands, access method commands, user-written commands, and other command lists can be coded into command lists, which are stored in a file during NCCF definition and invoked by name for execution when required.
- **Command processors.** IBM supplies a number of command processors as part of NCCF. NCCF service facilities can be used in user-written command processors. An installation can code its own command processors in assembler language and define the command verbs as described in *NCCF Installation.*
- **Exit routines.** The user can write programs to screen or edit messages and data to and from NCCF at various points in the program.
- **Subtasks.** An installation can code its own subtask to provide central control of a processor or of a resource.

In MVS, NCCF executes in a single address space. In OS/VS1, NCCF executes in its own partition. In VSE, NCCF executes in its own partition or as a subtask in a partition belonging to ACF/VTAM, ACF/VTAME, or the Virtual Storage Extended/Operator Communication Control Facility (VSE/OCCF). NCCF executes as a user program, in problem program state and user key; command processors and exit routines execute as subroutines of NCCF.

It is possible for code executing on behalf of one NCCF operator to affect the code supporting another operator. To prevent such undesirable interaction and possible impact on the security controls imposed on NCCF operators, the installation should control and assure that command processors or exit routines refer only to those data areas described in this manual; coding should refer only to the control blocks belonging to the operator issuing a request.

User command processors, exit routines, and command lists written for NCCF Release 1 are source compatible with NCCF Release 2. Reassembly using the current level of NCCF macro libraries is required.

The following chapters describe how to code command processors, exit routines, and command lists, and each chapter contains examples of each type. The systems programmer for an installation should decide the functions desired beyond those supplied as part of NCCF, and plan accordingly.

Syntax Used to Describe Macro Instructions

Throughout this manual, the following rules or syntax apply:

> Capital letters represent values that must be coded without change. Brackets ([ ]), braces ({}), "or" bars (|), and elipses (...) should not be coded.

Lowercase letters represent operands for which a value, address, or name must be supplied.

Brackets ([ ]) enclose operands or symbols that are optional. If brackets are not present, an item or group of items *must* be coded. Optional operands are those that may be coded or omitted independently of other operands. In some cases, the omission of an operand may cause the corresponding feature or function to be omitted; in other cases, specific values are assigned by default when an operand is omitted.

An "or" bar (|) between operands or braces ({}) enclosing operands indicates that one operand from among those listed must be coded.

An underlined value represents the default of a particular operand. If such an operand is omitted, NCCF uses the underlined value.

Chapter 2. Command Lists

This chapter describes NCCF command lists. It explains what command lists are, why they are useful, and how to code them. Sample NCCF command lists are shown at the end of the chapter.

Introduction to Command Lists

What Is a Command List?

A command list is a group of commands and special instructions with a name that applies to the whole group. To run that group of commands and instructions, the operator calls that command list by entering its name at a terminal, and all of the commands and instructions are run automatically.

What Can Command Lists Help You Do?

Command lists can help simplify routine or repetitive tasks. With a command list, an operator can get many different functions by typing just one name. A command list also allows an operator to supply values for a complex command without typing the command or understanding the command and the values in it. Command lists can be written to process an access method message automatically, with no operator action.

Here is an example of a command list called STATUS.

```
LIST STATUS=TASKS
LIST STATUS=SPAN
LIST STATUS=OPS
MSG SYSOP,ALL CHECKS ARE COMPLETED
```

By typing "STATUS," an operator displays the status of tasks, spans, and operators, and sends a message to the system console operator.

The following command list, "CHECK" allows the operator to vary the LIST commands issued:

```
LIST SCOPE=&1
LIST TIMER=&2
LIST STATUS=&3
```

If the operator types "CHECK ALL,OPER2,SPANS," the status of all spans, all scope commands, and timer commands for operator OPER2 are displayed.

The simplest form of command list is a list containing NCCF commands to be run in order. Control statements, variables, parameter, and assignment statements may also be used in a command list. These provide even more functions and are described later in the chapter. It is best to start with a simple command list first, and then add additional functions as necessary.

Who Can Use Command Lists?

Once a command list has been created, any NCCF operator can use that command list by entering the command list name. A command list may be restricted to a group of operators by using the scope of commands facility described in *NCCF Installation.*

Naming and Filing Command Lists

The system programmer writes the command lists. They can be built prior to starting NCCF or while NCCF is running, and are members or B books stored in the file defined as DSICLD. The name of the command list is used as the member or book name.

Note: If you plan to update or create command lists while NCCF is running, you should define DSICLD without secondary extents. This prevents a member from being filed in a new extent that NCCF cannot reference until it is closed and restarted.

Defining Command Lists to NCCF

Defining Command Lists in OS/VS

Defining command lists to NCCF in OS/VS is required in certain circumstances only.

If scope checking is desired, define the command list as follows:

```
clistname   CMDMDL   MOD=DSICCP
```

To call a command list from an access method message, the message identifier must be defined to NCCF as the name of a command list using a CMDMDL statement, as follows:

```
messageid   CMDMDL   MOD=DSICCP
```

For more information, see "Invoking a Command List from an Access Method Message."

For more information on defining command lists to NCCF, refer to the section on the CMDMDL statement in *NCCF Installation.*

Defining Command Lists in VSE

In VSE, a command list must be defined to NCCF in one of two ways. The command list can be defined using a CMDMDL statement in the form:

```
clistname   CMDMDL   MOD=DSICCP
```

The command list can also be defined to NCCF by including as the first statement in the command list the following command list definition statement:

```
[label]   CLIST
```

The label is not examined by NCCF. If present, it must begin in column one. "CLIST" begins in column 2 or later, and is preceded by at least one blank. (For OS/VS, this VSE definition statement is not required, and will be ignored if present.)

If scope checking is desired, define the command list as follows:

```
clistname   CMDMDL   MOD=DSICCP
```

To call a command list from an access method message, the message identifier must be defined to NCCF as the name of a command list using a CMDMDL statement, as follows:

```
messageid    CMDMDL    MOD=DSICCP
```

For more information, see "Invoking a Command List from an Access Method Message."

For more information on defining command lists to NCCF, refer to the section on the CMDMDL statement in *NCCF Installation.*

Invoking Command Lists

Invoking a Command List from an Operator Terminal

The operator can enter a command list name from the terminal in the same way as a command and operands are entered. When the command list name is entered, the command list begins processing. Message responses and other information are sent to the operator, depending on the contents of the command list.

Invoking a Command List from a User-Written Command Processor

User-written command processors may call command lists. Command lists initiated in this manner are queued until execution of the command in progress and other stacked commands or command lists has been completed. See Chapter 4, "Command Processors," for information on how to write a command processor.

Invoking a Command List from Another Command List

A command list referred to within a command list is executed completely before execution of the calling command list continues. This process is called nesting. NCCF allows up to 16 nested command lists. If you concatenate the DSICLD command list data sets, file the calling command list and all its nested command lists in the same data set.

Variables can be passed to a command list if the variable is defined in the invoking command list, and if the value of the variable is not longer than 255 characters. Nested variables are allowed. If after one substitution, the value generated is still a variable (preceded by &) substitution is performed again. For example, if the ABC command list is defined as:

```
XYZ LINES,CLSTRS,TERMS,CDRMS,ACT,INACT,EVERY,&1,&2
```

and if the XYZ command is defined as:

```
D NET,&&8,&&9
```

the following ACF/VTAM commands are generated for the following operator input:

Input	ACF/VTAM Command Output
ABC 1,5	D NET,LINES,ACT
ABC 4,7	D NET,CDRMS,EVERY
ABC 2,5	D NET,CLSTRS,ACT
ABC 3,6	D NET,TERMS,INACT

Invoking a Command List from an Access Method Message

A command list can be invoked by a message received by NCCF from ACF/VTAM or ACF/TCAM.

Note: Messages from the Terminal Access Facility, or from other IBM program products that work with NCCF cannot invoke a command list.

The command list can be used to send a response to the message or to reword or delete the message. If a command list is used to reword or to delete an access method message, the original message is sent to the NCCF disk log, but not to the operator's console or to the hard-copy log. The operator receives only those messages issued by the command list.

When an access method message is used to call a command list, the message identifier must be identified to NCCF as a command list, using the CMDMDL statement.

Each word of a message (as separated by blanks and commas) is considered a separate parameter when the command list is called. The first word of the message after the message identifier is the first parameter. By using the parameters, each word of the message may be indexed separately. This helps the command list to reword or to respond to the message.

An access method message that requires a reply is preceded by by *Lnn; nn* represents the reply number. In this case, the message identifer is still considered the name of the command list. *Lnn* is the first parameter, and the first word of the message is the second parameter.

Note: There are special considerations for unsolicited access method message. These messages invoke command lists that run under the primary POI task (PPT) rather than under the operator station task (OST). Certain commands cannot be used in PPT command lists. See "Command Lists Running under the PPT" for more information.

The following is an example of using a command list to reply to an ACF/VTAM message:

```
Lnn IST284A OPTION TO RELOAD ncpnm AVAILABLE - REPLY 'YES'
            OR 'NO' OR 'YES, LOADSTA=LINKSTANAME'
Lnn IST284A OPTION TO RELOAD ncpnm AVAILABLE - REPLY 'YES'
            OR 'NO' OR 'YES, LOADSTA=LINKSTANAME'
```

The message identifier is defined as a command list, using the CMDMDL statement (see *NCCF Installation*):

```
IST284A CMDMDL MOD=DSICCP (OS/VS)

5C84A   CMDMDL MOD=DSICCP (VSE)
```

When the message is received, it is prefixed by *Lnn; nn* represents the reply number. The command list treats the *Lnn* as &1 and the reply YES is sent to ACF/VTAM by a command list.

For OS/VS, member IST284A contains the following:

```
REPLY &1,YES
```

For VSE, book DSI5C84A contains the following:

```
REPLY &1,YES
```

The VSE message identifier 5C84A must have the DSI prefix added to the book name before the book is filed in the VSE source statement library.

In VSE, the DSI prefix must also be used to list a command list based on a message identifier; for example:

```
LIST CLIST=DSI5C84A
```

Command Lists Running under the Primary POI Task

Most command lists run under the operator station task (OST). However, some command lists are run under the primary POI task (PPT). Command lists run under the PPT if they are:

- Called by an *unsolicited* access method message.
- Specified with an NCCFIC statement to execute as soon as NCCF is initialized.
- Called with an AT or EVERY command that specifies PPT as an operand. (PPT allows the command to be run even when the operator is not logged on.)

The PPT command list is user-written and defined to NCCF in the same way as any other command list. All output messages produced as a result of the command list are sent to the authorized message receiver and logged under that task. Messages originating under the PPT are flagged with a "P" in the seventh character of the domain name field.

The following commands cannot be used in a command list executing under the primary POI task:

AUTOWRAP	INPUT	RESET	SWITCH
CANCEL	LOGOFF	ROUTE	
CLOSE	MOVE	START	
GO	PAUSE	STOP	

In addition, the &PAUSE control statement, Terminal Access Facility commands, and immediate commands cannot run under the PPT.

The Command List Language

Command lists are written in a special command list language. This command list language is described in the following sections.

Coding Guidelines

The following are some guidelines to keep in mind when coding command lists:

- The command list statement must be within the first 71 characters of an 80-character record. Column 72 should be left blank. Columns 73-80 are reserved for optional sequence numbers.
- Continuation of a command list statement to the next line is not allowed.
- The end of a command list is not indicated in any special manner. There is no END statement. The command list ends when the last command list statement is processed.
- A command list statement may contain any number of leading or trailing blanks. Anywhere one blank may be used within a statement, any number of blanks may be used.
- The suppression character (defined with the SUPPCHAR operand of the NCCFID definition statement) may be coded to prevent a command or any statement of a command list from appearing on the operator's screen, hard-copy log, and NCCF log. If used, the suppression character must be coded in column one of the command list statement. In the example below, ? has been defined as a suppression character:

```
?* COMMAND LIST UPDATED 2/5/80 BY OPERATOR IRENE
START DOMAIN=&1
PAUSE ENTER GO WHEN MESSAGE DSI809A ARRIVES FROM &1
?ROUTE &1,OPER1,123456
```

  The first and last lines of the command will be suppressed.

Variables

A command list statement may contain parameters or variables to be replaced by actual values at execution time. A command list variable is a symbol with an ampersand (&) as the first character, followed by 1 to 11 alphanumeric characters. There are three kinds of variables:

- Parameters
- Control variables
- User variables

Parameters

Parameters are passed to a command list when the command list is invoked. Up to 31 positional parameters may be passed. The parameters appear after the command list name and are delimited by either a blank or a comma. A parameter may be up to 255 characters; parameters longer than 255 characters are truncated. The following special characters may be used within a parameter if the parameter is enclosed in single apostrophes: blank, period, equal sign, apostrophe, comma. Text within single apostrophes is treated as a single operand. Two commas in a row indicates a null parameter.

For example, assume that the command list FLAG is invoked as shown:

```
FLAG  RED,BLUE WHITE '=',,'THE U.S. FLAG'
```

This command list has six parameters. The fourth parameter is an equal sign, the fifth is null, and the sixth is a phrase: THE U.S. FLAG.

When the command list is invoked, each of the parameters is substituted in the command list wherever there is an ampersand followed by a number from 1 through 31 (&*n*). The ampersand indicates substitution, while *n* indicates which of the 31 positional parameters is substituted. The &*n* is deleted and replaced by the parameter.

Consider the FLAG command list again. If one line in the command list read:

```
&6 IS &1, &3, AND &2
```

after substitution, this line would read:

```
THE U.S. FLAG IS RED, WHITE, AND BLUE
```

When the parameter is substituted, the text to the right of the &*n* is moved enough to make room for the parameter. No spaces are added or deleted, and no other characters are affected.

Parameters may be referenced as many times as necessary in any given line. There are no restrictions about parameter sequence. In other words, &5 may be referenced before &2. Substitution is from right to left for each command list line and the right-most &*n* is treated first.

The maximum number of parameters allowed is 31. If an ampersand is followed by a number other than 1 through 31, an error message results.

If the ALTER command list is defined as:

```
VARY NET,ACT,ID=&1
VARY NET,INACT,ID=&2
```

and the command list is invoked by this entry:

```
ALTER ABLE,BAKER
```

the commands executed are:

```
VARY NET,ACT,ID=ABLE
VARY NET,INACT,ID=BAKER
```

Control Variables

Control variables are variables that are predefined by NCCF; their substitution values are initialized and maintained by NCCF. The control variables are shown in Figure 2-1.

Variable	Value
&APPLID	Application program identifier for the task under which the command list is running (NCCF domain ID appended with a 3-character alphanumeric value assigned by NCCF).
&DATE	The current date in the form *mm/dd/yy*.
&HCOPY	Resource name for the hardcopy device started by this operator. (If there is no hard-copy device for this operator, &HCOPY is null.)
&LU	Resource name for this physical operator station.
&MSGMOD	For command lists invoked from an ACF/VTAM message, the 5-character ACF/VTAM module identifier. This module identifier is removed from the message by NCCF before the command list is invoked. &MSGMOD is valid only if the ACF/VTAM MSGMOD facility is in effect; if this facility is not in effect, &MSGMOD is null.
&NCCFCNT	Total number of domains with which this operator can establish a session.
&OPID	This NCCF operator's identifier.
&PARMCNT	Number of parameters specified by the invoker of the command list.
&PARMSTR	Character string following the command that invoked this command list. (If there are no parameters, &PARMSTR is null).
&RETCODE	Return code from a command processor or another command list. The user may set &RETCODE with the &EXIT control statement to any positive value or to -1. &RETCODE may be tested to determine command list processing. All negative return codes are reserved for definition by NCCF. Return codes -1, -2, and -3 may be useful for command lists. -1 may be set by the user with the &EXIT control statement. -2 and -3 are set by NCCF, but the user may test for them in a command list using &RETCODE. -1 NCCF forces the termination of the executing command list and all nested command lists. -2 Invalid command; no command is executed, but the command list is not terminated. -3 Command is not in the operator's scope of commands; no command is executed, but the command list is not terminated.
&TASK	Character string "PPT", "OST", or "NNT", depending on the task under which the command list is running. &TASK allows the same command list to run under any task (using conditional processing for PPT restrictions).
&TIME	The current time in the form *hh : mm*.

Figure 2-1. Control Variables for Command Lists

User Variables

User variables are any variables that are not parameters or control variables. A user variable name is specified as an ampersand (&) followed by 1 to 11 alphanumeric characters. A-Z, 0-9, #, @, $ are valid alphanumeric characters. The first character following the ampersand must be nonnumeric. For example, examine the following sample variables:

Valid	Invalid	
&A	&2ABC	(&2 will be substituted as a parameter)
&USERVARNAME	&INVALIDVARNAME	(too long)
&@23456	&A%	(invalid character)

User variables are initialized to null with a length of 0 if the first use does not provide a value. The user can initialize a variable by using it on the left side of an assignment statement (&USERPARM = 8) or by providing it as a variable on a &PAUSE statement.

Command List Labels

A label, if present, is the first nonblank in a command list record and consists of a dash (—) followed by 1 to 11 alphanumeric characters (A-Z, 0-9, #, @, $). Any command list statement except a comment line may have a label. The command list statement follows the label and is separated from the label by at least one blank. If a label is the only word on a command list statement, the statement is assumed to be a null statement and may be used as the target of a &GOTO or &BEGWRITE statement. Labels must be unique; if a duplicate label is encountered, the command list is terminated. A label is not scanned for variable substitution unless it is an operand on a &GOTO or &BEGWRITE control statement.

Examine the following samples:

```
VALID:
   -LABEL1  MSG READER,THIS IS CORRECT
   -$IRENE  MSG READER,THIS IS ALSO CORRECT
   -NULL
INVALID:
  -INVALIDLABEL  MSG READER,THIS LABEL IS TOO LONG
  -GLENN    *LABELS ARE NOT ALLOWED ON COMMENT LINES
  -&PRIS    MSG READER,LABEL CANNOT BE SUBSTITUTED HERE
  MSG READER,LABEL MUST BE FIRST NON-BLANK  -LABEL1
```

Command List Statements

There are five types of command list statements:

- Null statement
- Comment
- Command
- Assignment statement
- Control statement

Null Statements

A command list statement containing all blanks or only a label is a null statement. If a label is present, the null statement may be the target of a &GOTO or &BEGWRITE statement. Otherwise, the null statement is ignored.

Comments

A command list statement that contains an asterisk (*) as the first nonblank character is treated as a comment. variable substitution is performed on comments, so if the comments are written to the screen they can reference the specific values of variables. (If you wish to write comments to the screen without variable substitution, use &BEGWRITE NOSUB, discussed later in this chapter.)

Comments in command lists can be helpful for headings or simply as a way to display information. For example, a command list coded entirely with comments could be used to show the current network configuration to an operator. The following command list uses comments as a heading prior to the actual display of information requested by the command list:

```
****************************
*    STATUS OF ALL LINES     *
****************************
D NET,LINES,&1
```

Commands

NCCF commands and user-written commands defined as "regular" or "both" may be issued in a command list. ACF/VTAM and ACF/TCAM commands may be issued in a command list. In addition, a command list may invoke other command lists. Immediate commands, the AGAIN command, and data services commands are not allowed in a command list.

If the command list is invoked by an operator at a terminal or by a solicited access method message, the commands that can be issued within the command list are limited by the operator's span of control and scope of commands.

If the command list is executed under the PPT, no span or scope checking is done, and certain commands cannot be issued. For more information, see "Command Lists Executing under the PPT."

Asynchronous full-screen commands (DSIPSS TYPE=ASYPANEL) should be coded only as the last commands in a command list. For more information on asynchronous commands, see the section in Chapter 4 titled "Full-Screen Command Processor Considerations."

Terminal Access Facility commands may be used in a command list. BGNSESS FLSCN and RTRNSESS commands will cause the command list to stop running until the full-screen session is disconnected or ended. When the operator returns to NCCF mode, the command list will continue.

For a quick reference of which commands can be used in command lists, see Appendix A.

Assignment Statements

An assignment statement is a statement of the form

variable = expression

The equal sign (=) must be delimited by blanks.

expression may be one of the following:

A constant or variable

A built-in function (see "Command List Built-In Functions")

An arithmetic expression consisting of numbers and/or variables separated by a plus (+) or minus (-) arithmetic operator. For example:

```
3 + 4     &PARMCNT + 3              8 + -4
5 - 2     &3 - &4 - &USERVARNAME    2 - -&P2
```

Note that the arithmetic operators + and - must be delimited by blanks unless they indicate a negative or positive number (-4, +2). Thus, the expressions 4 - 2 and 4 - -2 are valid, but 4 -2 is invalid and will cause the command list to terminate.

Control Statements

Control statements are used to control the processing sequence of a command list. Control statements also allow the command list to send messages to the NCCF operator and to receive input from the operator. Control statements are processed by the command list processor. Figure 2-2 shows a summary of the control statements in NCCF.

Control Statement	Operands	Description
&BEGWRITE	[SUB | NOSUB] [label]	Causes subsequent lines to be written to the terminal until the specified label is reached.
&CONTROL	[ALL | CMD | ERR]	Controls the writing of command list statements to the operator station.
&EXIT	[number]	Terminates command list processing.
&GOTO	label	Transfers control to the command list line beginning with the specified label.
&IF	logical expression &THEN command list statement	If the logical expression is true, the command list statement will be executed.
&PAUSE	[{ NOINPUT | VARS variable [. . .] | String variable }]	Suspends the execution of a command list.
&WRITE	[text]	Writes a message to the operator station.

Figure 2-2. Summary of Control Statements for Command Lists

Each command list control statement begins with a control symbol in the form *&word*. All operands and operations must be delimited by blanks. A control statement must be coded on one line and cannot be continued on the following line. Only one control statement can be coded on a line. If an error is detected in a control statement, the control statement and an error message are written to the operator's terminal; if the command list cannot recover from the error, command list processing is terminated.

&BEGWRITE

&BEGWRITE causes subsequent lines to be written to the terminal, until the specified label is reached. Labels can be coded on lines being written out by &BEGWRITE, and used later as targets of a &GOTO statement or another &BEGWRITE statement.

Statement	Operands
&BEGWRITE	[SUB | NOSUB][label]

SUB
causes substitution of variables in lines written to the terminal. If there are blanks before the first message character, the line is shifted left until the first nonblank character is in column 1. If you want the blanks sent to the screen, code a nonblank character in column 1.

NOSUB
suppresses substitution of variables in lines written to the terminal. NOSUB is the default.

label
is a standard command list label that is used to indicate the point at which no more lines are to be written to the terminal. The line on which *label* is coded is not written to the terminal and is treated as the next command list statement to be processed. *label* may be a variable that has been assigned a value earlier in the command list. If *label* is not specified, one line will be written to the terminal. If *label* cannot be found, the remainder of the command list is written to the terminal, and the command list is terminated.

&CONTROL

&CONTROL controls the writing of command list statement to the operator station.

Statement	Operands
&CONTROL	[ALL | CMD | ERR]

ALL
specifies that all command list statements are to be written to the operator station, after variable substitution and before execution. This includes:

- Comments
- Null statements

- Control statements
- Assignment statements
- Commands

ALL is the default if &CONTROL is not specified.

CMD
specifies that only commands are to be written to the operator station, after variable substitution and before execution. Other command list statements are not displayed.

ERR
specifies that only command list statements in error and commands that return a nonzero return code are to be written to the operator station, after execution.

&EXIT

&EXIT terminates command list processing.

Statement	Operands
&EXIT	[number]

number
provides a return code to the caller of the command list (see &RETCODE, in the section titled "Control List Variables"). If a number is not specified, a zero return code is generated. Return code -1 causes this command list and all nested command lists to terminate. Other negative return codes are reserved for use by NCCF.

Note: Reaching the end of the file also terminates command list processing, and generates a zero return code.

&GOTO

&GOTO transfers control to the command list statement beginning with a specified label.

Statement	Operands
&GOTO	label

label
is a standard command list label. *Label* may be a variable that has been assigned a value earlier in the command list.

&IF

&IF defines a logical expression and tests the truth of that expression. If the expression is true, then the command list statement is executed. Otherwise, the statement is ignored and the next sequential statement in the command list is executed.

Statement	Operands
&IF	logical expression &THEN command list statement

logical expression
is an expression of the form

expression logical-operator expression

where *expression* is anything that can appear on the right side of an assignment statement, and *logical-operator* is one of the following logical operators:

Logical Operator	Meaning
= (or EQ)	Equal
¬ = (or NE)	Not equal
< (or LT)	Less than
> (or GT)	Greater than
<= (or LE)	Less than or equal
>= (or GE)	Greater than or equal
¬ > (or NG)	Not greater than
¬ < (or NL)	Not less than

The logical operator must be separated from the two expressions by blanks.

&THEN
is a required keyword. It must be separated from the logical expression and the command list statement by blanks.

command list statement
refers to any unlabeled command list statement. If the logical expression specified after &IF is true, this statement is executed.

Note: In a statement of the form

```
&IF &variable1 = &variable2 &THEN . . .
```

the variables are substituted prior to syntax checking. If either *&variable1* or *&variable2* has a null value, a syntax error results. To avoid this problem, prefix both variables with the same character. For example:

```
&IF A&variable1 = A&variable2 &THEN . . .
```

&PAUSE

&PAUSE suspends the execution of the command list. A "P" is displayed in the upper-right corner of the operator screen while the operator is in pause state. The &WRITE or &BEGWRITE statement should be used prior to &PAUSE to tell the operator the reason for the pause, and to describe the actions that should be taken by the operator. Execution of the command list is resumed when a GO command is received from the terminal or ended when a CANCEL command is received. The GO command may also be used to provide input to the command list. For more information on GO and CANCEL, see *NCCF Terminal Use.*

&PAUSE should not be coded in a command list executing under the primary POI task. If &PAUSE is coded under the PPT, an error message is issued, the pause is ignored, and processing continues with the next command list statement.

Note: If the CANCEL command is issued in a nested command list, all command lists in the invoking chain are terminated.

Statement	Operands
&PAUSE	{ NOINPUT VARS variable1[. . .] STRING variable }

NOINPUT
specifies that no operands are permitted on the GO command. An error message will be issued if any operands are encountered. (&PAUSE NOINPUT is equivalent to the PAUSE command with no text.) This is the default.

VARS
specifies that the operands on the GO command are to be assigned to *variable1, variable2,* and so on. All operands are treated as positional. If the number of operands on the GO command exceeds the number of variables on &PAUSE, the extra operands are discarded, an error message is issued, and command list processing continues. If the number of variables on &PAUSE exceeds the number of operands on the GO command, the remaining variables are set to null. Two commas in a row on the GO command results in one null variable. For example, if the GO command is:

```
GO 1,,5
```

and the &PAUSE control statement is:

```
&PAUSE VARS &A,&B,&C
```

&A will be set to 1 and &C will be set to 5. &B will be null.

STRING
specifies that the entire operand string on the GO command is to be treated as one operand and assigned to *variable* No quotes are needed; if quotes are entered they will become part of the variable. If no operand is specified on the GO command, *variable* is set to null.

&WRITE

&WRITE writes a message to the operator station.

Statement	Operands
&WRITE	[text]

text
is a character string to be written to the terminal operator's screen after variable substitution is performed. If no text is specified, a blank line is written to the screen.

If there are blanks before the first message character, the line is shifted left until the first nonblank character is in column 1. If you want the blanks sent to the screen, code a nonblank character in column 1. To send a single quote or an apostrophe in a message, code two apostrophes. Also, if a single quote has several blanks before it, these blanks are changed to one blank.

Command List Built-In Functions

Command list built-in functions are used to perform evaluations of expressions and character strings. Built-in functions are features of the command list language that provide capabilities otherwise unavailable to the user. Built-in functions can be used only as an expression on the right side of an assignment statement (see "Assignment Statements"), or as an expression in an &IF statement. They cannot be part of an arithmetic expression. Figure 2-3 provides a summary of the built-in command list functions in NCCF.

&CONCAT

Function	Operands
&CONCAT	{variable | constant} {variable | constant}

&CONCAT concatenates the values of two parameters to form a new value. The result must be a valid value for a variable. If the value of both parameters is null, the result of &CONCAT is null. If the resulting value is greater than 255 characters, it is truncated to 255 characters.

Function	Operands	Description
&CONCAT	{variable | constant} {variable | constant}	Concatenates the value of two operands to form a new value.
&LENGTH	{variable | constant}	Provides the length of the operands in characters.
&SUBSTR	variable i [j]	Substitutes a part of a string of characters for the total character string.
&NCCFID	{variable | constant}	Provides the identifier of the domain specified by the numeric operand.
&NCCFSTAT	{variable | constant}	Indicates the status of the specified domain name.

Figure 2-3. Built-in Functions for Command Lists

For example, assume you coded the following command list statements:

```
&A = NCCFA
&B = 001
&C = &CONCAT &A &B
```

The variable &C would be set to NCCFA001.

&LENGTH

Function	Operands
&LENGTH	{variable | constant}

&LENGTH provides the length of the parameter in characters. If the parameter is null, the result of &LENGTH is zero.

For example, assume you coded the following command list statements:

```
&A = KEVIN
&C = &LENGTH &A
```

The variable &C would be set to 5.

&SUBSTR

Function	Operands
&SUBSTR	variable i [j]

&SUBSTR substitutes a part of an indicated string of characters as a real value during command list statement execution. The string of characters to be used is the value of *variable* starting at position *i* with length *j*. *i* and *j* may be either constants or variables. If *j* is not specified or exceeds the number of characters remaining, the remaining length is used. The value of *i* must be greater than zero; the value of *j* must be zero or greater. If *i* exceeds the length of the variable or *j* is zero, the value of the function is considered null. If either the variable or *i* are null, the results will not be as expected.

For example, assume you coded the following command list statements:

```
&A = NCCFA003
&B = &SUBSTR &A 6
&C = &SUBSTR &A 1 5
&D = &SUBSTR &A 6 10
```

The variables &B and &D would be set to 003, and the variable &C would be set to NCCFA.

&NCCFID

Function	Operands
&NCCFID	{variable | constant}

&NCCFID provides the identifier of the domain specified by the parameter. The value of the parameter must be a number in the range 1 to &NCCFCNT (see "Control Variables"). &NCCFID serves as an index to the list of domains with which the operator is authorized to establish a session. For an operator with specific authority (as defined on the AUTH statement in the operator's profile) this list is derived from the DOMAINS statement in the profile. For an operator with global authority, this list is derived from the RRD NCCF definition statements. Refer to *NCCF Installation* for more information on AUTH, DOMAINS, and RRD definition statements.

To obtain the domain identifier of the domain in which the command list is running, use &AMP.SUBSTR &APPLID 1 *j*, where *j* is the length of the application program ID (&APPLID) minus three. For example, user variable &DOMID could be set to the name of the local domain by the following statements:

```
&DOMID = &LENGTH &APPLID
&DOMID = &DOMID - 3
&DOMID = &SUBSTR &APPLID 1 &DOMID
```

In the example above, assume &APPLID = NCCFA001. In the first line, &DOMID is set to 8. In the second line, the length of the three-digit NCCF identifier 001 is subtracted from the length of &APPLID, setting &DOMID to 5. in the third line, &DOMID is set to the part of &APPLID starting at position 1 for a length of 5. This last statement results in setting &DOMID to the name of the local domain, NCCFA.

&NCCFSTAT

Function	Operand
&NCCFSTAT	{variable | constant}

&NCCFSTAT indicates the status of the specified domain. The operand must be a valid NCCF domain identifier of 1 to 5 characters. If the operator has a session with the domain whose identifier is specified by the operand, &NCCFSTAT is set to the characters "ACT". If the operator does not have a session with the specified domain, &NCCFSTAT is set to the characters "INACT".

For example, assume you coded the following command list statement to determine the status of domain NCCFA:

```
&A = &NCCFSTAT NCCFA
```

If NCCFA is active, &A is set to ACT. If NCCFA is not active, &A is set to INACT.

How NCCF Analyzes a Command List

Command list statements are analyzed as shown in Figure 2-4. Each command list statement is parsed into separate syntactic elements, using blanks and commas as delimiters. Multiple blanks are considered as one delimiter; multiple commas are treated as multiple delimiters. Labels are removed from the command list statements. Each statement is scanned from right to left and substitution is performed on one element at a time, according to the following rules:

- Substitution is not performed on a &PAUSE statement or the &THEN clause of an &IF statement (the &THEN clause is substituted only when it is to be executed).
- Each element is scanned from right to left for an &. If an & is found, then it, along with the rest of the element to the right, is taken as the name of a variable and is replaced by the value of the variable. This substitution may increase or decrease the length of the element.
- Variables for which a value cannot be found, are considered to be null.
- Command list control symbols and built-in functions are not substituted.
- If the first character to the right of the ampersand is numeric, the variable is assumed to be a parameter.
- If a special character (nonalphanumeric) is encountered, it delimits the variable name. (For example, if an element contains &A=&XYZ first &XYZ is substituted, then &A is substituted.)
- The scan resumes at the next character to the left, and the search for an ampersand continues. If another ampersand is found, it and the entire syntactic element to the right, including the previous substitution are taken as the name of a variable and replaced by its value. Note that the value substituted is not scanned for an ampersand.

  If the element is the target of an assignment statement, the scan stops on the second character to preserve the variable name to be assigned a value. For example:

  ```
  &B = 1
  &A&B = 2
  ```

  will set user variable &A1 to 2.

- This process is repeated until all syntactic elements have been analyzed.

  The statement is then analyzed to determine whether it is null, a comment, a control statement, an assignment statement, or a command. No further processing is done on null and comment statements.

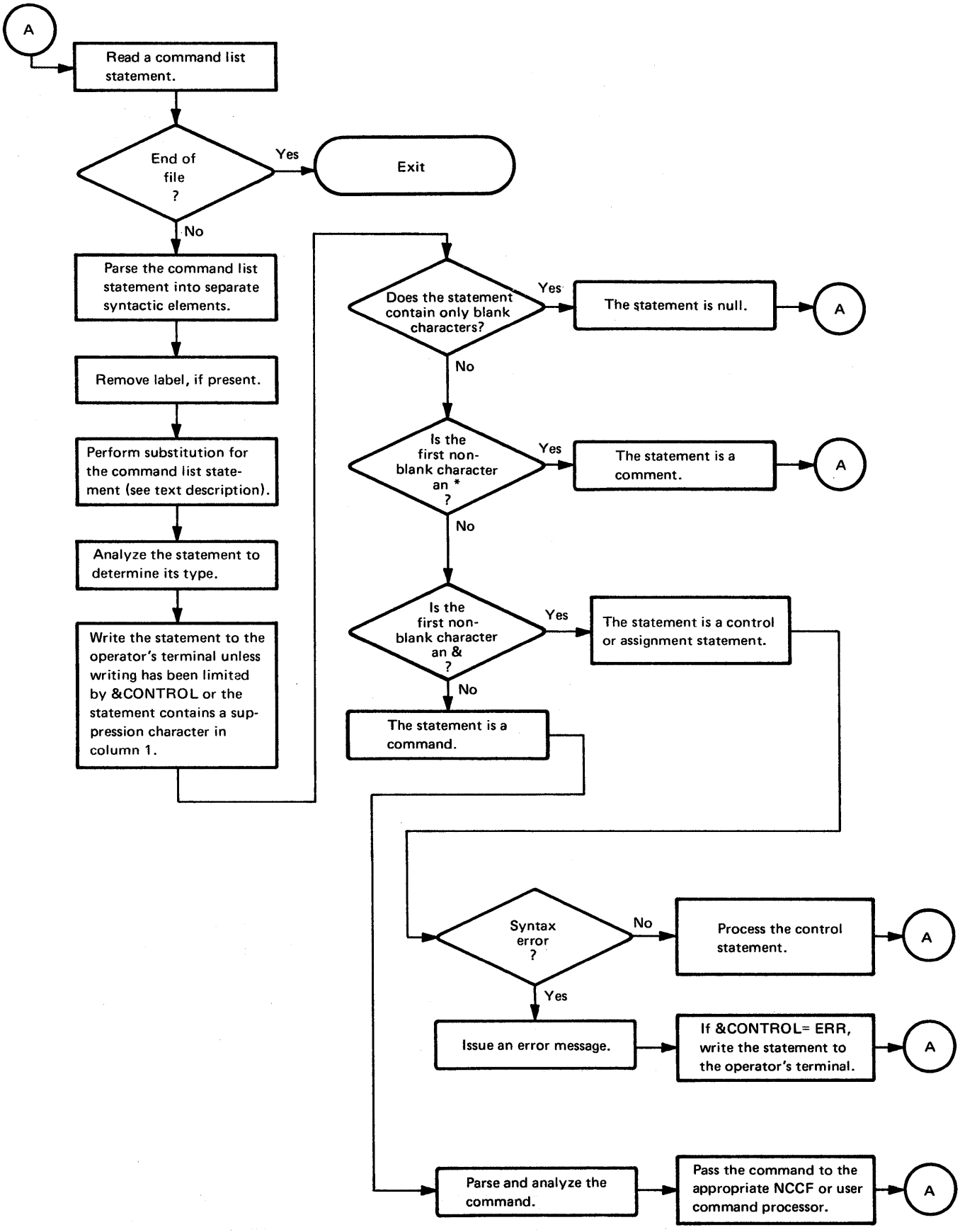


Figure 2-4. How NCCF Analyzes a Command List

Control and assignment statements are checked for syntax errors and then processed. If a syntax error is found, an error message is sent and the statement in error is written to the operator's terminal. Severe errors cause the command list to terminate. Some errors cause warning messages only; a default value will be used or the command list statement will be ignored. Processing will continue. If there are no major errors, the commands are parsed, analyzed, and then passed to the appropriate NCCF or user command processor for processing. The command processor return code is available as &RETCODE. &RETCODE is checked to determine if the command statement should be written to the terminal.

After substitution, the command list statement is written to the terminal unless either writing has been limited by &CONTROL or the statement contains a suppression character in column 1.

Sample Command Lists

The IOBUF command list allows the invoker to start and stop buffer and I/O traces for a physical unit as follows:

```
F NET,&2TRACE,TYPE=IO,ID=&1
F NET,&2TRACE,TYPE=BUF,ID=&1
```

The following entry starts I/O and buffer traces for PU7:

```
IOBUF PU7
```

and generates the following commands:

```
F NET,TRACE,TYPE=IO,ID=PU7
F NET,TRACE,TYPE=BUF,ID=PU7
```

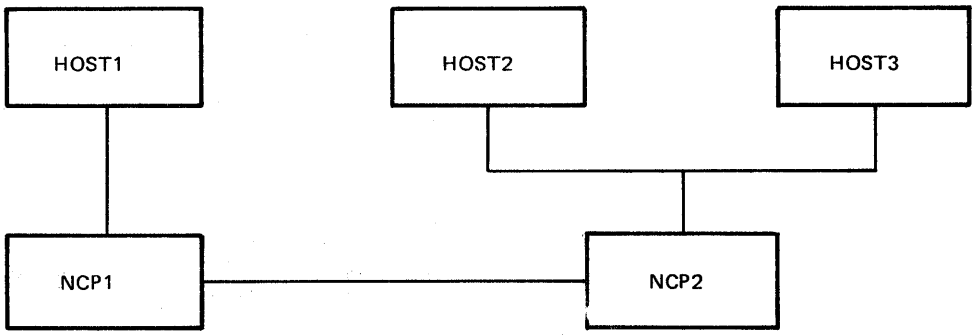
The following entry stops the I/O and buffer traces for PU7:

```
IOBUF PU7,NO
```

and generates the following commands:

```
F NET,NOTRACE,TYPE=IO,ID=PU7
F NET,NOTRACE,TYPE=BUF,ID=PU7
```

A command list can be used in a multiple-domain environment to aid the orderly reinstatement of HOST2 after HOST3 has backed up HOST2's resources.



The command list is entered as follows:

```
REINSTAT &1,&2,&3
```

&1 is the NCP to be reinstated (NCP2); &2 is the host that was performing back-up (HOST3); &3 is the host that was backed up (HOST2). Thus, the command entered from HOST2 is:

```
REINSTAT NCP2,HOST3,HOST2
```

The REINSTAT command list is as follows:

```
ROUTE &2,V NET,REL,ID=&1,CDLINK=ACT
*ENTER GO TO CONTINUE WHEN READY
*OR ENTER CANCEL TO TERMINATE PROCESSING
PAUSE
ROUTE &2,V NET,ACT,ID=HST3CDRS
ROUTE HOST1,F NET,ID=HST3CDRS,CDRM=(&3,&2)
V NET,ACT,ID=&1
MSG ALL,REINSTATEMENT COMPLETE FOR NCP &1 and HOST &3
ROUTE HOST1,MSG ALL,REINSTATEMENT COMPLETE FOR NCP &1 AND
 HOST &3
ROUTE HOST3,MSG ALL,REINSTATEMENT COMPLETE FOR NCP &1 AND
 HOST &3
```

Sample Conditional Command Lists

Sample 1

Member or B book STOPCD contains the following command list statements:

```
        &CONTROL CMD                                                    00000001
* THIS COMMAND LIST STOPS ALL CROSS-DOMAIN SESSIONS FOR                  00000002
* ONE OPERATOR                                                          00000003
*                                                                       00000004
*                                 INITIALIZE COUNT OF POSSIBLE          00000005
*                                 CROSS-DOMAIN SESSIONS                 00000006
   &I = &NCCFCNT                                                        00000007
*                                 IF ALL POSSIBLE SESSIONS              00000008
*                                 CHECKED, EXIT                         00000009
 -LOOP  &IF &I = 0 &THEN &EXIT                                          00000010
*                                 NAME OF DOMAIN                        00000011
   &ID = &NCCFID &I                                                     00000012
*                                 STATUS OF DOMAIN                      00000013
   &STAT = &NCCFSTAT &ID                                                00000014
*                                 IF THE DOMAIN IS ACTIVE,              00000015
*                                 STOP THE DOMAIN                       00000016
*                                                                       00000017
   &IF &STAT = ACT &THEN STOP DOMAIN=&ID                                00000018
*                                                                       00000019
*                                 SUBTRACT 1 FROM COUNT                 00000020
  &I = &I - 1                                                           00000021
*                                 CONTINUE LOOP                         00000022
  &GOTO -LOOP                                                           00000023
```

The command list statements in Sample 1 (STOPCD) are explained below:

Statement Number	Explanation
1	Sets the &CONTROL value to CMD; only commands are sent to the operator's terminal screen.
2-6,8-9,11,13 15-17,19,22	Comment statements to explain what the command list is doing.
7	Assignment statement to set user variable &I to the value of the NCCF control variable, &NCCFCNT. &I now contains the number of domains with which the operator can establish a session. &I will be used to control the number of times the command list should loop.
10	Defines the label -LOOP. Tests the value of &I. If &I is equal to 0, the command list will exit (&EXIT). If &I is not equal to 0, the command list will continue processing with the next statement.
12	Sets the user variable &ID to the value of the built-in function &NCCFID. &I is used as the operand for the &NCCFID function. &ID is set to domain name &I as specified on the DOMAINS definition statement.
14	Sets the user variable &STAT to the value of the built-in function &NCCFSTAT. &ID is used as the operand for the &NCCFSTAT function. The value of &STAT will be the characters "ACT" if domain &ID is active, and "INACT" if domain &ID is not active.
18	Tests the value of &STAT to determine if a STOP command should be issued for that domain. If the value of &STAT is "ACT", the &THEN clause of the &IF statement will be executed. If the value of &STAT is not "ACT", no further processing will be done on this statement.
21	Subtracts 1 from the value of &I. Statement 10 checks &I to decide if the command list should continue.
23	Transfers control to statement 10, where -LOOP is defined. Processing continues with statement 10.

When the STOPCD command list is executed, the operator will see the following on the terminal screen:

```
STOPCD
STOP DOMAIN=domain1
STOP DOMAIN=domain2
           .
           .
           .
STOP DOMAIN=domainn
DSI013I  COMMAND LIST STOPCD COMPLETE
```

Sample 2

Member or B book NOWEVERY contains the following command list statements:

```
* TO ISSUE A COMMAND NOW AND EVERY HH:MM MINUTES,                    00000001
* ISSUE:   NOWEVERY HH:MM,COMMAND                                    00000002
*                                                                    00000003
  &CONTROL ERR                                                      00000004
  &LENP1 = &LENGTH &1                                               00000005
  &STARTP2 = &LENP1 + 2                                             00000006
  &LENPARMSTR = &LENGTH &PARMSTR                                    00000007
  &LENCMD = &LENPARMSTR - &LENP1 - 1                                00000008
  &CMD = &SUBSTR &PARMSTR &STARTP2 &LENCMD                          00000009
* ISSUE COMMAND NOW                                                 00000010
  &CMD                                                              00000011
* ISSUE COMMAND EVERY HH:MM, AS SPECIFIED                           00000012
  EVERY &PARMSTR                                                    00000013
  &EXIT                                                             00000014
```

The command list statements in Sample 2 (NOWEVERY) are described below:

Statement Number	Explanation
1-2,10,12	Comment statements to explain what the command list is doing.
3	Blank statement; this statement will be sent to the terminal as a blank line.
4	Sets the &CONTROL value to ERR. From this point on in the command list, only error statements are sent to the operator's terminal screen.
5	Assignment statement that uses the &LENGTH built-in function to set the user variable &LENP1 to the length of the first parameter entered on the command list invocation (hh:mm).
6	Assignment statement that sets the user variable &STARTP2 to the value of the user variable &LENP1 plus 2. &STARTP2 will be used to obtain the entire character string of the command parameter on the command list invocation.

Statement Number	Explanation
7	Uses the built-in function &LENGTH to set the user variable &LENPARMSTR to the length of the parameters entered on the command list invocation (&PARMSTR).
8	Sets the user variable &LENCMD to the value of &LENPARMSTR (see statement 7) minus &LENP1 (length of first parameter) minus 1.
9	Uses the built-in function &SUBSTR to isolate a section of the &PARMSTR NCCF variable. &CMD is set to the part of &PARMSTR starting at position &STARTP2 (see statement 6) for a length of &LENCMD (see statement 8). &CMD now represents the command entered on the command list invocation.
11	Executes the command specified by &CMD (see statement 9).
13	Issues the EVERY command using the NCCF variable &PARMSTR as the operand for the command.
14	Causes the command list to terminate. This statement is optional for this command list. If the &EXIT is omitted, the command list terminates after statement 13 because an end-of-file is reached.

When the NOWEVERY command list is executed, the operator will see the following on the terminal screen:

```
NOWEVERY HH:MM,COMMAND
TO ISSUE A COMMAND NOW AND EVERY HH:MM MINUTES,
ISSUE:    NOWEVERY  HH:MM,Command
```

Since &CONTROL ERR is specified, no more statements will appear at the operator's terminal unless errors are detected.

Sample 3

Member or B book VLOGON contains the following command list statements:

```
       CLIST                                                          00000001
       &CONTROL CMD                                                   00000002
       &IF A&1 = A? &THEN &BEGWRITE -TEXT                             00000003
* THIS CLIST GENERATES A VARY ACTIVATE COMMAND ACCORDING TO           00000004
*    3 PARAMETERS PASSED.  EACH PARAMETER HAS A DEFAULT AND          00000005
*    DOES NOT HAVE TO BE SPECIFIED.                                  00000006
* PARAMETERS:                                                        00000007
*    &1 = NAME OF ID TO BE ACTIVATED; DEFAULT IS NCPNAME             00000008
*    &2 = NAME OF CONTROLLER APPLICATION; DEFAULT IS SNAPPL          00000009
*    &3 = TYPE OF ACTIVATION--COLD OR WARM; DEFAULT IS WARM          00000010
* FOR EXAMPLE:                                                       00000011
*    VLOGON                                                          00000012
*    VLOGON IDNAME,CONTROLAPPL,COLD                                  00000013
*    VLOGON IDNAME,,WARM                                             00000014
*                                                                    00000015
```

```
-TEXT   &IF A&1 = A? &THEN &EXIT                                    00000016
        &IF &LENGTH &1 NE 0 &THEN &GOTO -LAB1                       00000017
        &1 = NCPNAME                                                00000018
-LAB1   &IF &LENGTH &2 NE 0 &THEN &GOTO -LAB2                       00000019
        &2 = SNAAPPL                                                00000020
-LAB2   &IF &LENGTH &3 NE 0 &THEN &GOTO -CMD                        00000021
        &3 = WARM                                                   00000022
-CMD    V NET,ACT,ID=&1,LOGON=&2,&3                                 00000023
```

The command list statements in Sample 3 (VLOGON) are described below:

Statement Number	Explanation
1	VSE CLIST statement to define this command list to NCCF. In VSE, the CLIST statement is required if a CMDMDL statement does not exist for this command list. In OS/VS, the CLIST statement is not required and will be ignored.
2	Sets the &CONTROL value to CMD; only commands will be sent to the operator's terminal screen.
3	Tests the first parameter (&1) on the command list invocation. "A" is used as part of the variable name tested to ensure that a syntax error will not result if no &1 parameter is entered. If &1 is a question mark (?) the &THEN clause is executed. This is a "help" function coded into the command list. &BEGWRITE writes out statements 4-15 at the operator's terminal and statement 16 causes the command list to terminate. If &1 is not a question mark, the command list continues processing with statement 4.
4-15	Comment statements to explain what the command list will do. These comments will be written to the screen only if statement 3 is true.
16	Defines the label -TEXT which is used as an operand on the &BEGWRITE control statement (see statement 2). This statement tests to see if the first parameter is a question mark (?). If it is, the command list terminates. If not, the command list continues processing with statement 17.
17	Tests whether the first parameter (&1) was supplied on the command list invocation. If it was supplied, processing continues with statement 19; if not, processing continues with statement 18.
18	Sets the first parameter (&1) to the default value, NCPNAME, if &1 was not supplied on the command list invocation.

Statement Number	Explanation
19	Defines the label -LAB1 and tests whether the second parameter (&2) was supplied on the command list. If it was supplied, processing continues with statement 21; if not, processing continues with statement 20.
20	Sets the second parameter (&2) to the default value, SNAPPL, if &2 was not supplied on the command list invocation.
21	Defines the label -LAB2 and tests whether the third parameter (&3) was supplied on the command list. If it was supplied, processing continues with statement 23; if not, processing continues with statement 22.
22	Sets the third parameter (&3) to the default value, WARM, if &3 was not supplied on the command list invocation.
23	Defines the label -CMD. The symbols &1, &2, and &3 are replaced with their values in the command "V NET ACT,ID=&1,LOGON=&2,&3". After value substitution, the command is displayed on the operator's terminal and executed. The command list terminates after this statement because an end-of-file is reached.

If the VLOGON command list is executed with an ID of IRENE and a controller application of NCCF2 specified, the operator will see the following on the terminal screen:

```
VLOGON IRENE,NCCF2
-CMD V NET,ACT,ID=IRENE,LOGON=NCCF2,WARM
DSI013I  COMMAND LIST VLOGON COMPLETE
Output from the ACF/VTAM VARY command would appear here.
```

When the VLOGON command list is executed with a question mark (?) as the first parameter, the operator will see the following on the terminal screen:

```
VLOGON ?

Statements 4-15 from the command list will appear here

DSI013I  COMMAND LIST VLOGON COMPLETE
```

The VLOGON statement will appear exactly as it is specified on the command list invocation.

Chapter 3. Service Facilities and Macro Instructions

This chapter describes the NCCF service facilities and macro instructions that may be used when coding your own command processors, exit routines, and subtasks. The chapter is divided into two parts, a guide explaining some of the service facilities available and how to use them effectively, and a reference section listing the macro instructions that invoke the NCCF service facilities. You should become familiar with this chapter before reading Chapters 4, 5, and 6.

Service Facilities Guide

NCCF provides service facilities for user-written command processors, exit routines, and subtasks. User-written programs that use these service facilities may need to have addressability to the main vector table for NCCF (MVT) and must use the DSICBS macro instruction (see below) to get a copy of the DSECT for the service routine vector list (SVL), and other control blocks that are needed. Figure 3-1 is an overview of the NCCF macro instructions and describes which control block fields must be initialized by the user and which macro instructions require addressibility to the MVT.

Main Vector Table Addressability

To establish addressability to the main vector table (MVT), code the following:

For a command processor:

```
USING DSICWB,1
L     register,CWBTIB
USING DSITIB,register
L     register,TIBTVB
USING DSITVB,register
L     register,TVBMVT
USING DSIMVT,register
```

For an exit routine:

```
USING DSIUSE,1
L     register,USERTVB
USING DSITVB,register
L     register,TVBMVT
USING DSIMVT,register
```

For a subtask:

```
LR    register,1
USING DSITVB,register
L     register,TVBMVT
USING DSIMVT,register
```

Macro Name	Function	User Input	NCCF Output	Control Block Input Fields User Must Set	Control Block Fields Set by NCCF	MVT Addressability Required
DSICBS	Includes control blocks during compilation.	Control block name	Control block is included (optionally listed).	None	None	No
DSICES	Analyzes a command.	Address of parse descriptor block (PDB) or command buffer	Address of entry in system command table (SCT)	None	None	Yes
DSIDATIM	Obtains and formats date and time.	Output area and format desired	Time and date.	None	None	Yes
DSIDEL	Deletes a user-specified module.	Module name	Module is deleted.	None	None	No (OS/VS) Yes (VSE)
DSIDKS CONN	Obtains a buffer; connects subtask to a file.	File name	Buffer is obtained; subtask is connected to file.	None	HDRBLENG, HDRMLENG, HDRTDISP	Yes
FIND	Finds a book or member and reads first record.	Book or member name	Book or member is found; record is read.	None	HDRMLENG, HDRTDISP	Yes
READ	Reads a record into buffer obtained by CONN.	Book or member name	Record is read, or end-of-data is indicated.	None	HDRMLENG, HDRTDISP	Yes
DISC	Frees a buffer; disconnects subtask from a file.	File name	Buffer is freed.	None	None	Yes
DSIFRE	Releases storage obtained by DSIGET.	Address and amount of storage	Storage is freed.	None	None	No if Q=NO Yes if Q=YES
DSIGET	Obtains storage.	Address and amount of storage	Storage is obtained.	None	None	No if Q=NO Yes if Q=YES
DSIKVS	Determines whether an operator is authorized to use a given keyword or value.	Command and keyword or value to be checked	Authorization return code	None	None	Yes
DSILCS CWB	Obtains a command work block (CWB).	Address of area to return CWB address	Address of CWB	CWBTIB (after CWB is obtained)	CWB header, address of next CWB.	Yes
	Frees a command work block (CWB).	Address of CWB	CWB is freed.	None	None	Yes
SWB	Obtains service work block (SWB).	Address of area to return SWB address	Address of SWB	SWBTIB (after SWB is obtained)	SWB header, address, of next SWB	Yes
	Frees service work block (SWB).	Address of SWB	SWB is freed.	None	None	Yes
TVB	Finds task vector block (TVB) for a given subtask.	Address of TVB where search is to begin; LU name, or operator ID, or next active operator station task, hardcopy task or NCCF-to-NCCF task	Address of TVB that matches specified input.	None	Address of TVB	Yes
DSILOD	Loads a user-specified module.	Address of BLDL list.	Module is loaded.	None	None	No (OS/VS): Yes (VSE)

Figure 3-1 (Part 1 of 3). Summary of NCCF Macro Instructions

Macro Name	Function	User Input	NCCF Output	Control Block Input Fields User Must Set	Control Block Fields Set by NCCF	MVT Addressability Required
DSIMBS						
SIZE	Calculates message length.	Message number, message inserts	Length of message	None	HDRMLENG (length of message)	Yes
BFR	Builds a message.	Address of area where message is to be returned	Message is built.	All BUFHDR fields except HDRMLENG	HDRMLENG	Yes
DSIMDS	Generates NCCF message definition module.	Number and text of message	Message is added in message definition module.	None	None	No
DSIMQS	Queues a message to a task.	Address of buffer with message in it, task ID of destination	Message is queued and then sent to display screen or hard-copy log.	All BUFHDR fields	HDRSENDR	Yes
DSIOIS	Searches operator identification table (OIT).	Operator identification	Bit position of operator identification in OIT (also used as input to DSISSS Macro instruction)	None	None	Yes
DSIPAS	Searches for aliases for command parameters.	Address of parse descriptor block (PDB) and number of the entry in it	Alias value or entered value or blanks	None	None	Yes
DSIPOS	Posts completion of an event.	Address of event control block (ECB) and completion code to be put in ECB	Event control block is posted.	None	None	No
DSIPRS	Builds a parse descriptor block (PDB).	Address of storage in which PDB is to be built	PDB is built.	PDB header, indicating length (to avoid overlay)	All PDB fields except header, PDBCMDA, PDB flags	Yes
DSIPSS	Writes a message to the display screen or sends input to NCCF-to-NCCF task (NNT).	Address of data to be sent, name of destination	Message is written.	All BUFHDR fields including HDRMLENG	HDRMLENG modified	Yes
DSIRDS*	Searches authorization and resource table (ART); optionally marks entry as active or inactive.	LU name to be located in ART	Position of entry in ART	None	None	Yes
DSISSS*	Searches span name table (SNT).	Bit position to be checked in SNT (value obtained from DSIOIS) and address of entry in SNT where search is to begin	Address of first entry in SNT with operator bit set to 1	None	None	Yes
DSIWAT	Waits for completion of an event.	Name of event control block (ECB) or address of ECB list	None	None	None	No (OS/VS) Yes (VSE)
DSIWCS	Writes a message to the system operator's console.	Address of buffer with message in it	Message is written.	HDRMLENG, HDRTDISP	None	Yes
DSIWLS	Writes a message on the NCCF log and hard-copy log.	Address of buffer containing record to be logged	Message is written.	All BUFHDR fields	None	Yes

Figure 3-1 (Part 2 of 3). Summary of NCCF Macro Instructions

Macro Name	Function	User Input	NCCF Output	Control Block Input Fields User Must Set	Control Block Fields Set by NCCF	MVT Addressability Required
DSIZCSMS	Requests CN' ata across the CNM interface.	Address of SWB and DSRB; input buffer address and length; RU address and length; destination name; target name	Requested CNM data is returned.	None	None	No
DSIZVSMS	Requests VSAM services for a DSCP.	Address of SWB and DSRB; type of VSAM request; type of access; VSAM key address and length; address of user work buffer.	Appropriate VSAM function is performed.	All BUFHDR fields	None	No

*ACF/VTAM only.

Figure 3-1 (Part 3 of 3). Summary of NCCF Macro Instructions

Control Block Considerations

Figure 3-2 is an overview of the control blocks used by the NCCF service routines. You should become familiar with the NCCF control blocks described in Appendix C before beginning design of your command processor, exit routine, or subtask. In addition, the following chapters each contain detailed descriptions of the control block fields needed for the particular task. The service work block (SWB) and task vector block (TVB) are particularly important. The complete NCCF control block structure is described in *NCCF Logic.*

Some of the more important control blocks are described in detail in this chapter and in the following chapters in this book.

Chapter 3:

DSISWB Is the parmameter list for most of the NCCF service routines.

BUFHDR Is the standard NCCF buffer header.

DSIIFR Maps an internal fuction request which is a formatted buffer that is transmitted to a subtask's message queue using the DSIMQS macro instruction.

DSICBH Is the control block header for most of the NCCF control blocks.

DSIPDB Is used to analyzes input to NCCF.

Chapter 4:

DSICWB Is the parameter list for a command processor.

DSISCE Contains information about the command.

DSIDSRB Is used for communication between the data services task (DST) and a data services command processor (DSCP).

Chapter 5:

DSIUSE Is the parameter list for an exit routine.

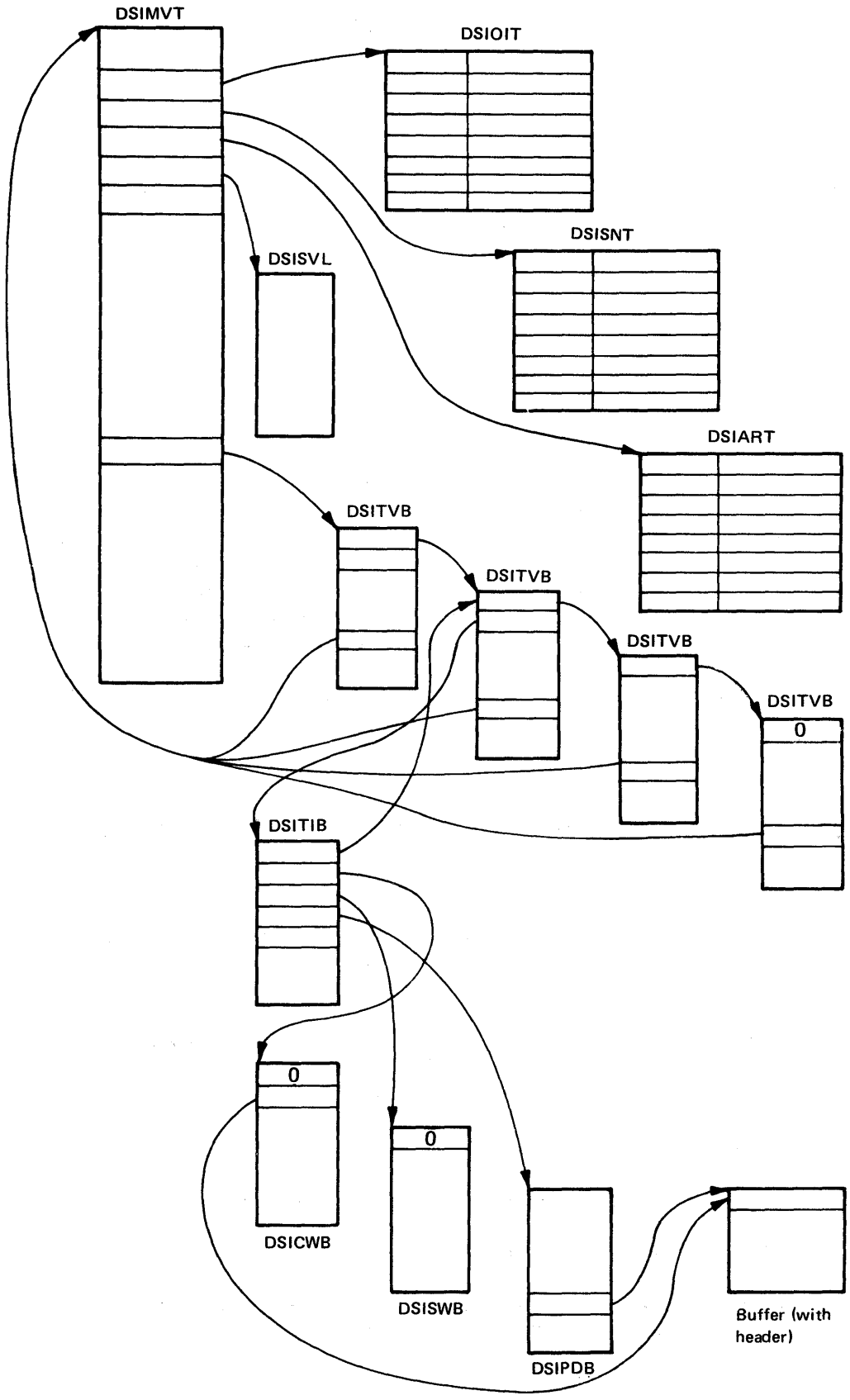


Figure 3-2. Overview of the Control Blocks Used by NCCF Service Routines

DSIMVT Is the main control block for information throughout NCCF.

DSITVB Represents potential and active NCCF subtasks and subtask parameters.

DSITIB Stores information about an active subtask.

DSICBS Macro Instruction

The DSICBS macro instruction is used to gain access to the control blocks needed by your command processor, exit routine, or subtask. In order to use the NCCF service facilities, you need access to the DSISVL and DSISWB. You probably also want to include DSITIB to obtain a buffer header. The other control blocks are optional depending on the type of program you are writing and the service facilities you need. See Figure 3-1 under "User Input" and "Control Block Fields User Must Set" for a guideline. In addition, for a command processor, include DSICWB; for an exit routine, include DSIUSE; for a subtask, include DSITVB.

The DSICBS macro might be coded as follows in a command processor:

```
DSICBS DSICWB,DSISVL,DSISWB,DSIMVT,DSIPDB,DSITVB,DEFER=ALL
```

DEFER allows you to specify exactly where the control blocks should be expanded in your program. DEFER=ALL specifies that all subsequent DSICBS macro instructions are not to be expanded until DSICBS DEFER=INCLUDE is encountered.

Service Work Block (SWB)

The service work block (SWB) contains equates for most of the service routine return codes returned in register 15 (DSILCS return code equates are in DSIMVT). DSISWB is also used as a parameter list for most of the NCCF service routines. The parameter list passed to the command processor (CWB) or to the exit routine (USE) contains the address of an SWB that can be used by the invoked routine. If this SWB is being used for some other purpose, such as a work area, the control block location services macro DSILCS should be used to request another SWB. The DSILCS macro might be coded as follows:

```
DSILCS   CBADDR=(R2),SWB=GET
```

If you obtain another SWB with the DSILCS macro, be sure to initialize the SWBTIB field of the DSISWB with the address of the caller's DSITIB before you request NCCF services.

When the program no longer requires the SWB obtained with the DSILCS macro, you must free this DSISWB. To free the DSISWB in the example above, you would code:

```
DSILCS   CBADDR=(R2),SWB=FREE
```

Note: If you use an SWB as a work area, be careful not to overlay the SWBTIB or SWBCBH fields because these fields are not reinitialized by NCCF. If you must change either of these fields, reinitialize them before returning control to NCCF.

Task Vector Block (TVB)

There is one task vector block (TVB) for each subtask in NCCF. The TVB contains information about the status of the subtask. Certain service routines, such as DSIPSS, use the TVB to store control information that is important for processing their code. The task information block (TIB) is an extension of the TVB and represents an active task.

The TVB contains pointers to the MVT and the TIB. From these control blocks, you can obtain the addresses of other important control blocks.

Buffer Header (BUFHDR)

The buffer header (BUFHDR) portion of the task information block (DSITIB) is shown in Figure 3-3. The BUFHDR DSECT is included in the DSITIB DSECT; it must be included in every message or command buffer and must precede all text in the buffer. The fields are described below and must be initialized as shown in Figure 3-1 under "Control Block Input Fields User Must Set."

Field	Description
HDRMLENG	Indicates the length in bytes of the text data in the buffer as a number between 0 and 32,767.
HDRBLENG	Contains the actual length of the entire buffer: header, plus text, plus unused space. If the buffer is to be released with DSIFRE, this length is used. The length may be up to 32,767 bytes.
HDRIND	Is used by NCCF in certain situations, in general it should be set to zero.
HDRMTYPE	Contains a character that indicates the current usage of the buffer. It may also indicate the origin of the command. If the buffer is written out using the DSIPSS macro, this field is displayed and logged. The values for this field are defined in the BUFHDR expansion and are described in Appendix C of this manual.
HDRTDISP	Is the offset from the start of the buffer header to the first byte of text.
HDRTSTMP	Contains the time that the command was received, in the packed decimal form X'*hhmmss*0C' where *hh* is the hour of the day from 00 to 23, *mm* is the minutes of the hour from 00 to 59, *ss* is the seconds of the minute from 00 to 59, and 0C is a packed decimal sign. See the DSIDATIM macro instruction.
HDRDOMID	Shows the identifier of the domain that originated the message. This field is displayed and logged. The domain identifier for the NCCF under which a particular program is running is shown in the MVTCURAN field of DSIMVT.

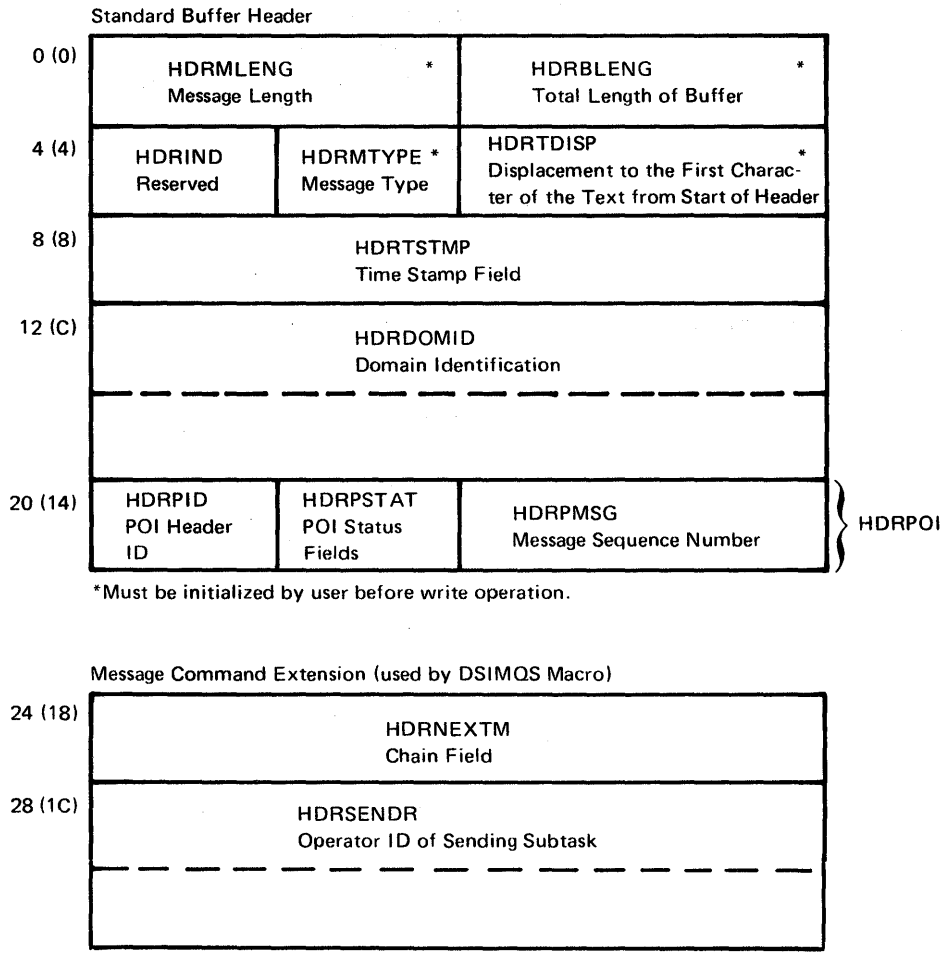


Figure 3-3. Buffer Header (BUFHDR)

Field	Description
HDRPOI	Is a reserved field.
Message command extension	Is an extension to the BUFHDR that is used when a buffer is transferred from one subtask to another. It is built by the DSIMQS macro when creating a buffer for the destination task. Other buffers do not need these fields.
HDRNEXTM	Is an internal NCCF field that is used to chain buffers together.
HDRSENDR	Contains the originator's operator ID, which is the contents of the sender's TVBOPID field.
Text	Can start anywhere after HDRPOI in a standard buffer or after HDRSENDR in a buffer with a message command extension. Use HDRTDISP to locate the start of text.

Example of BUFHDR Usage

The DSIDKS macro uses the buffer header to read a disk data set, which is blocked according to a user-specified blocking factor. The disk services module DSIDRS prefixes the physical read buffer with a BUFHDR. When the first *record* is requested, a disk read is done for the first *block*. Then HDRTDISP is adjusted to index the first logical *record,* and HDRMLENG is set to reflect the logical record length. When the DSIDKS macro is issued for a subsequent logical record, HDRTDISP is adjusted to index to the next logical record, until the block is exhausted. Then another disk read is done, and the process starts again from the first logical record in the block.

Internal Function Request (IFR)

The internal function request (IFR) is a formatted buffer that is transmitted to a subtask's message queue using the DSIMQS macro instruction. An IFR has HDRMTYPE specified as I HDRMTYPE=X'C9'; symbol HDRTYPEI). Because an IFR is transferred by DSIMQS, it always contains a message command extension when it is received. (When building an IFR, the extension is optional). If a command processor receives control with a command buffer and HDRMTYPE=HDRTYPEI, it is assumed that there is a command extension and an IFR.

The IFRCODE is 2 bytes, specified as X'0003', X'0008', or X'000B'. All other values are reserved for NCCF use. Code 3 indicates that the remainder of the buffer is a command to be executed. Code 8 only applies to an OST or NNT; it is user-defined and the IFR is passed to DSIEX13, the message receiver exit routine. Code B indicates that the command is input from a full-screen panel. The HDRTDISP field in an IFR should contain the displacement to the IFRCODE. For IFR codes 3 and B, NCCF modifies HDRTDISP and HDRMLENG so that all commands appear the same to the command processor; the command verb is followed by the operands. The IFR section is logically removed. HDRTDISP contains the offset to the command verb.

Control Block Header (CBH)

The NCCF control block header (CBH) is a 4-byte header that identifies all NCCF control blocks (except BUFHDR and IFR).

Field	Description
CBHID	Is a 1-byte field that identifies the control block type. The DSICBH DSECT defines the permissible values.
CBHTYPE	Is a 1-byte field. The task information block (TIB) and task vector block (TVB) each contain an identifier for the type of subtask that the block represents. Values allowed are PPT, OST, NNT, HCT, TCT, and optional subtask. The DSILCS macro instruction also uses this byte in management of CWBs and SWBs. In all other cases, this byte is reserved and should be set to zero.

Field	Description
CBHLENG	Is a halfword that contains the length of the control block. It represents the length that is preallocated or the length that is obtained by the DSIGET macro instruction. For example, a PDB has a fixed size portion and a variable number of entries. CBHLENG for a PDB contains the length of both parts.

Parse Descriptor Block (PDB)

The fields of the PDB are described below.

Field	Description
PDBCBH	Identifies the storage as a PDB and gives its size. Since PDBs are of no fixed length, this length is important. Most PDBs in NCCF are 160 bytes, which allows for the fixed portion of the PDB plus 37 entries. If the 160-byte PDB is overrun, DSIPRS issues an 8 return code.
PDBCMDA	Points to the entry in the system command entry (DSISCE) for the verb in the buffer (the verb that caused this command processor to be called). This entry is used as a parameter by the DSIPAS (parameter alias services) and DSIKVS (KEYCLASS and VALCLASS lookup services) macros.
PDBBUFA	Contains the address of the command buffer, as does CWBBUF (described earlier).
PDBIMMED	Is a flag that indicates whether the command processor is regular or immediate. When the PDBIMMED bit is on, the command processor is executing as an immediate command processor, as a subroutine of the receive (terminal input) exit (in OS/VS, under an IRB). A command processor can only be immediate if it is running under an operator station task (OST) or a cross-domain task (NNT). The user defines whether a command is regular or immediate on the CMDMDL definition card with the TYPE operand (see *NCCF Installation*). When the first bit in PDMIMMED is off, the command processor is executing as a regular command processor or as a data services command processor under the control of the subtask mainline (PRB in OS/VS). The DISPSS TYPE=OUTPUT macro should be used to write to the terminal. However, command processors running under a data services task (DST) may not use the DSIPSS macro. The DSIMQS macro should be used to send text to a terminal for the appropriate subtask (such as the request originator or the receiver of authorized messages). A queued DSIGET request must code the EXIT=NO operand when operating as a regular command processor.

Field	Description
PDBNOENT	Is the number of syntactical element entries in the PDB, including the verb and all operands. The delimiters used for this command's parse are blank, comma, period, and equal sign.
PDB Syntactical Element Entries	Each syntactical element creates one entry in this portion of the PDB. The verb is always the first entry. The number of syntactical element entries is in PDBNOENT. Each entry contains the length, the delimiter, and the offset from the beginning of the buffer.
PDBLENG	Contains the length of the particular syntactical element. It does not include the length of the delimeter. When two delimiters (except blanks) occur sequentially, the value of the length is zero; two delimiters separated by a blank or blanks also create a zero length entry. The offset is set to point to the second delimiter. The standard NCCF parsing delimiters are the blank, comma, period, and equal sign.
PDBTYPE	Contains the delimiter character that separates this element from the succeeding one. When a command processor is given control, the delimiters used in parsing the command are blank, comma, period, and equal sign. The end of the record is treated as if it is delimited by a blank. Multiple blanks are treated as one blank and blanks preceding a syntactical element are ignored. For example, 'bbbverbbboperand1,bbboperand2', creates an entry for the verb first, ignoring the preceding blanks, an entry for operand1, delimited by a comma, and one for operand2 delimited by a blank.
PDBDISP	Contains the offset from the start of the buffer to the first character of the nth syntactical element; for example, element addr(n) = PDBBUFA + PDBDISP(n).

Getting and Freeing Storage

DSIGET is used to obtain storage and DSIFRE is used to free that storage after use.

```
DSIGET  LV=4096,A=(REG2),BNDRY=PAGE
```

This example specifies that 4096 bytes of storage are to be obtained and the address placed in the the fullword pointed to by register 2.

DSIGET may also be used to queue the obtained storage to the user's task vector block (TVB). This allows NCCF to free the storage at logoff in the case of abnormal termination. An example to obtain this queued storage is:

```
DSIGET  LV=2032,A=(REG2),BNDRY=PAGE,REENT=YES,LISTA=(REG3),
        Q=YES TASKA=MYTVB,EXIT=YES
```

This macro specifies that 2032 bytes of storage are to be obtained and the address placed in the fullword pointed to by register 2. The storage is to be aligned on a page boundry. Since the first 16 bytes of the page are used by NCCF in Q=YES, only 2032 bytes were requested, 16 less than the page size. (Page size is 2048 bytes for OS/VS1 and VSE, 4096 bytes for MVS.) REENT=YES (OS/VS only), specifies that the reentrant form of DSIGET is to be used. In this example, the storage is to be queued to the TVB (Q=YES) specified by the symbolic name MYTVB (TASKA=MYTVB). LISTA=(REG3) specifies that register 3 contains the address of a 14-byte area in dynamic storage which DSIGET uses to obtain the queued storage. EXIT=YES must be coded if the storage request is from DSIEX01, DSIEX02 (if TVBINXIT flag is on), or an immediate command.

Getting, Freeing, and Locating a Control Block

DSILCS is used to get and free a DSISWB or a DSICWB. This macro instruction may also be used to locate a DSITVB.

The following example obtains a DSISWB and places the address of the DSISWB in the SWBAREA:

```
DSILCS   CBADDR=SWBAREA,SWB=GET
```

A register that points to SWBAREA may also be specified. To free the storage, use SWB=FREE instead of SWB=GET.

To obtain a DSICWB, use the same process used for a DSISWB, substituting CWB=GET for SWB=GET.

DSILCS may also be used to locate a DSITVB by operator identification or LU name, to locate the next active DSITVB for a specific task type, and to locate the DSITVB for the authorized message receiver.

Disk Services

Disk services retrieves data from NCCF partition data sets (OS/VS) or the source statement library (VSE). DSIDKS then locates a specified book or member and reads the records in that book or member. DSIDKS is used to obtain storage for the disk service area and initialize the data services block (DSIDSB) and the buffer header of the input buffer.

Here is a series of examples using DSIDKS:

```
DSIDKS   SWB=(REG2),DSBWORD=DISKADDR,TYPE=CONN,NAME=DSIPRF
DSIDKS   SWB=(REG2),DSBWORD=DISKADDR,TYPE=FIND,NAME=MEMNAME
DSIDKS   SWB=(REG2),DSBWORD=DISKADDR,TYPE=READ
DSIDKS   SWB=(REG2),DSBWORD=DISKADDR,TYPE=READ
DSIDKS   SWB=(REG2),DSBWORD=DISKADDR,TYPE=DISC,NAME=DSIPRF
```

In the examples above, DSIDKS initializes the disk service control blocks and input buffer, and returns the address of the DSIDSB in DISKADDR. DSIPRF is the NCCF definition name to be used. DSIDKS then finds the member or book name MEMNAME, and reads the first record. The next two sequential records are also read. When the three records have been read, DSIDKS frees the control blocks and the input buffer.

Presentation Services

Messages can be sent by NCCF using several different macro instructions: DSIPSS, DSIMQS, DSIWCS, and DSIWLS. DSIPSS is used to control screen formats, organize the data for a specific device, and send the data. Another form of DSIPSS is used to send a command to an NCCF in another domain. DSIMQS is used to send messages to the operator. DSIWCS sends a message to the system operator console, and DSIWLS sends a message to the NCCF log and the operator's hard-copy log. Figure 3-4 shows how these macro instructions are used for communication from an NCCF operator's OST. DSIPSS and DSIMQS are described in greater detail below.

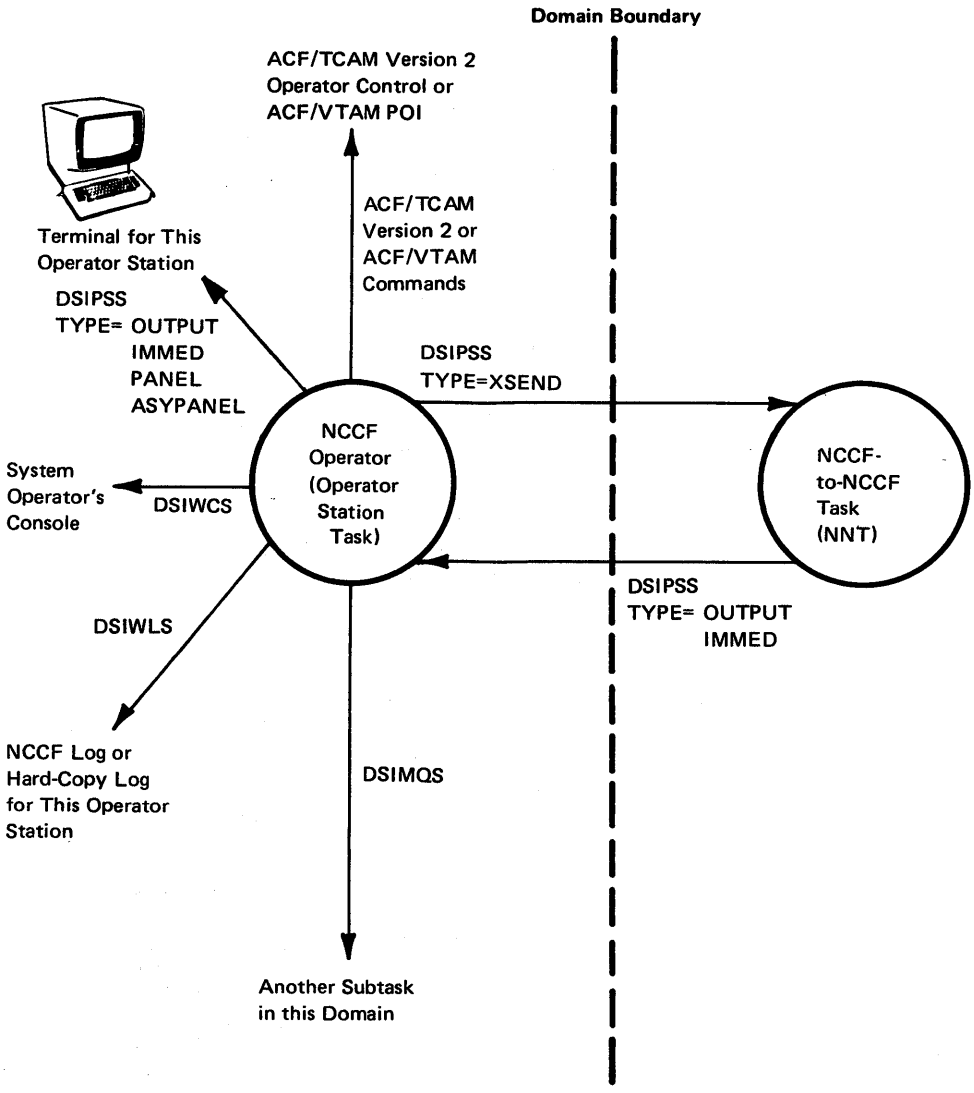


Figure 3-4. Use of NCCF Macro Instructions for Communication from an Operator Station Task

Several different types of presentation services are available through the DSIPSS macro instruction. The NCCF screen modes are described in greater detail in *NCCF Terminal Use*. They include the following:

Standard NCCF Mode

Messages sent to the NCCF screen consist of a 12-byte prefix, followed by 68 bytes of data. The prefix includes a 1-character code for the entry type and a domain name field indicating the domain that generated the message. If the message exceeds 68 bytes, it is broken between words and the message is continued on the next screen line, indented 12 characters.

Full-Line Mode

Messages sent to the NCCF screen appear as 80 bytes of data with no prefix; messages longer than 80 bytes are truncated. Except for this difference, the full-line screen appears the same as the standard NCCF screen. The messages appear in a line-by-line protocol. Full-line mode supports application programs such as NPDA, a separately orderable IBM program product, that use full-line, 80-byte output.

From a subtask other than an OST, full-screen title-line output is supported, allowing full-line messages to be displayed with title headings. (See "Full-Line Command Processor Considerations" in Chapter 4.)

Full-Screen Mode

Application-built 3270 data streams containing commands, orders, and data are sent to the NCCF screen. In this way, information can be presented with a full screen of data. (See "Full-Screen Command Processor Considerations" in Chapter 4).

For synchronous full-screen mode, 3270 data streams built by a command processor are sent to the terminal. Any input causes a command to be scheduled. The CLEAR key is used to escape from full-screen mode.

For asynchronous full-screen mode, full-screen input and output can be processed asynchronously, allowing a command processor to obtain input without scheduling a command and to issue a series of requests and responses without interrupting processing. The asynchronous full-screen command processor may process an event control block (ECB) list of multiple events while waiting for operator input. The escape from asynchronous full-screen mode is user-coded.

Use the chart shown in Figure 3-5 as a guide to help you code the DSIPSS macro instruction to obtain the type of output you desire. Other coding combinations are also possible.

To send a command to an NNT, the DSIPSS TYPE=XSEND macro is used. To return data to the OST from the NNT, DSIPSS TYPE=OUTPUT is used. This data may be messages that the OST places on the operator's screen or commands to be executed by the OST.

DSIPSS Function	Format of DSIPSS Macro Instruction
Standard NCCF mode or NNT-to-OST communication	DSIPSS SWB=(R2),TYPE=OUTPUT,OPTIONS=MSG,BFR=(R3) or DSIPSS SWB=(R2),TYPE=OUTPUT,OPTIONS=SEG,BFR=(R3)
Full-line mode: First line Middle line Last line Only line	 DSIPSS SWB=(R2),TYPE=OUTPUT,OPTIONS=FIRST,BFR=(R3) DSIPSS SWB=(R2),TYPE=OUTPUT,OPTIONS=MIDDLE,BFR=(R3) DSIPSS SWB=(R2),TYPE=OUTPUT,OPTIONS=LAST,BFR=(R3) DSIPSS SWB=(R2),TYPE=OUTPUT,OPTIONS=ONLY,BFR=(R3)
Determine display screen size.	DSIPSS SWB=(R2),TYPE=SCRSIZE,SIZE=SIZEAREA
Determine output area size.	DSIPSS SWB=(R2),TYPE=WINDOW,SIZE=SIZEAREA
Send a single line to the immediate message area.	DSIPSS SWB=(R2),TYPE=IMMED,BFR=(R3)
Send from an OST to an NNT in another domain.	DSIPSS SWB=(R2),TYPE=XSEND,APPLID=APPLNAME,BFR=(R3)
Send a formatted 3270 data stream synchronously. Optionally, receive input.	DSIPSS SWB=(R2),TYPE=PANEL,PANEL=PARMLIST
Send a formatted 3270 data stream and receive input asynchronously.	DSIPSS SWB=(R2),TYPE=ASYPANEL,PANEL=PARMLIST
Test if the OST has work pending.	DSIPSS SWB=(R2),TYPE=TESTWAIT
Wait in a command processor for NCCF and command processor events.	DSIPSS SWB=(R2),TYPE=PSSWAIT,ECBLIST=ECBPARM

Figure 3-5. Examples of Using the DSIPSS Macro Instruction

Message Queuing

DSIMQS is used to queue a user-supplied message to the message queue of a subtask's DSITVB in the same domain. The message may be sent to either the operator's screen or to the hard-copy log. Here is an example of DSIMQS:

```
DSIMQS   SWB=(REG2),BFR=BUFADDR,TASKID=MYTASK
```

In the example above, the message buffer pointed to by BUFADDR is to be queued to the subtask with the subtask identification of MYTASK.

The subtask identifiers can be found by checking the TVBOPID field of DSITVB. TVBOPID is initialized with the following:

- For an OST or an NNT, the DOMAINID operand of the NCCFID definition statement appended with the identifier of the operator.

- For a PPT, the PPT APPL name, which is the NCCFID appended with the characters PPT.
- For a DST, the TSKID operand of the TASK definition statement for the data services task.
- For an HCT, the LU name of the hard-copy device.
- For an optional subtask, the user initializes this field.

If AUTHRCV=YES is coded instead of TASKID, the message is sent to the authorized message receiver specified by the AUTH definition statement. If there is no authorized message receiver, the message is sent to the system console.

The message buffer must have a properly initialized buffer header (BUFHDR), including the message command extension. Buffers that are formatted as internal function requests (IFRs) are not displayed. Instead, they cause the receiving subtask to take the action requested by the IFR. Refer to the sections "Buffer Header (BUFHDR)"
and "Internal Function Request (IFR)" for more information.

Resource Location (ACF/VTAM Only)

The DSIRDS macro instruction is used in ACF/VTAM systems to locate an entry address for the resource in the authorizaton and resource table, DSIART. DSIRDS might be specified as follows:

```
DSIRDS   SWB=(REG2),LUNAME=LUADDR,ARTPOS=ENTRYADR
```

For this example, the DSIART entry address for the resource pointed to by LUADDR will be returned in ENTRYADR. The resource will be marked as active.

Figure 3-6 shows the relationships between the operator identification table (DSIOIT), the span name table (DSISNT), and the authorization and resource table (DSIART). The relative position of an entry in the operator identification table is represented by the bit position of each entry in the span name table (n bits). The relative position of an entry in the span name table is represented by the bit position of each entry in the authorization and resource table (m bits).

For example, if a user wishes to find whether a particular operator is authorized to issue commands for a particular resource, follow this procedure:

- Use the DSIOIS macro instruction to find the position of the operator's identification in the DSIOIT table. The identification is put in the fullword area pointed to by the OPID operand of the macro instruction. The relative position is returned to the fullword area pointed to by the OITPOS operand.
- Use the DSISSS macro instruction to search DSISNT for the bit position that corresponds to the location of the operator identification entry in DSIOIT. The bit position is specified by the OITPOS operand of the macro instruction. It is best to begin the search at the beginning of the span name table. (The DSISNT address is found in the NCCF main vector table, DSIMVT; see *NCCF Logic.*) The address of the first span entry that corresponds with a bit set to 1 is returned to the fullword area specified by the SNTADDR operand of the macro instruction. Because it is the address of the entry and not its

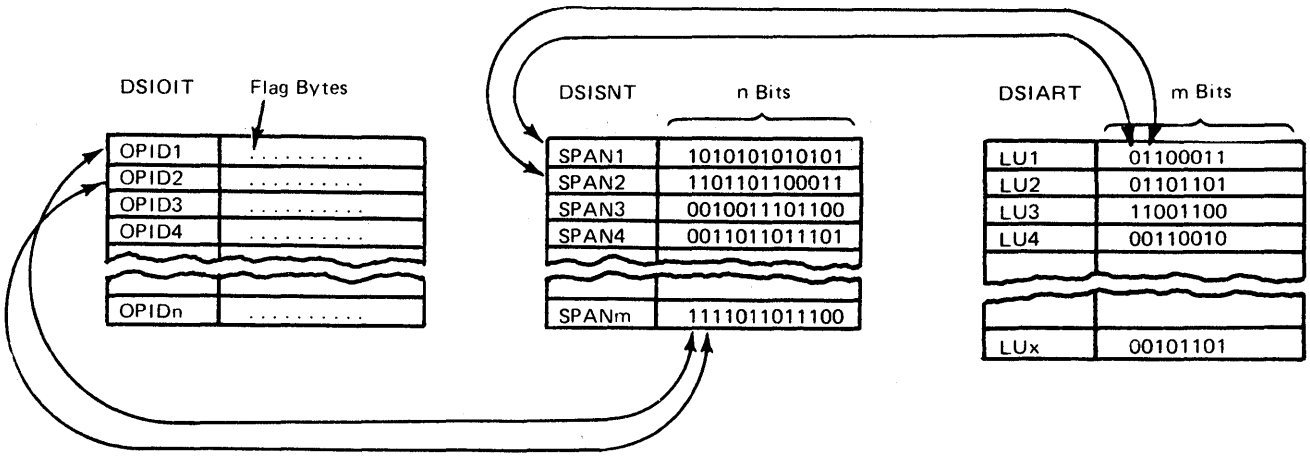


Figure 3-6. Table Field Relationships

relative position that is returned, the starting address should be stored in another area to be used in any calculations that may be required to establish the entry's position.

- Create a mask byte to check the bit position of the authorization and resource table DSIART that corresponds to the span name table entry position.

- Use the DSIRDS macro instruction to find the address of the specific entry for the resource. The address of the entry is returned to a fullword area specified by the ARTPOS operand of the macro instruction. The resource name is specified on the LUNAME operand.

- Use the mask byte to check whether the corresponding bit is set to 1.

In the example shown in Figure 3-6, the DSIOIS macro can be used to determine the position of the identification OPID2 in DSIOIT. Position 2 is returned to the area specified by OITPOS. DSISSS can then be used to check bit position 2 in DSISNT. The first span name with a 1 in that position is SPAN2. The address of that entry is returned to the area specified by SNTADDR. Using the starting address and the address returned, and dividing the difference by the length of the DSISNT entries (found in DSIMVT), the relative position of SPAN2 can be calcualted. A mask byte can then be prepared to test the bit position corresponding to SPAN2 in DSIART. The DSIRDS macro instruction can then be used to find the address of the resource name in DSIART. If LU2 is specified in the area pointed to by the LUNAME operand, it is the second entry in DSIART. The mask byte can then be used at that location, showing that the operator whose identification is OPID2 can issue commands for LU2. If a match is not found, DSISSS can be invoked again to find another span. The starting address specified for the SNTADDR operand should be the address of the entry immediately following SPAN2. This process can be repeated until a span is found or the end of the table is reached.

Macro Reference

This section explains how to code NCCF macros to request various service facilities. NCCF uses registers 0, 1, 14, and 15 for macro instruction expansion; the user should avoid these registers when using NCCF macro instructions. NCCF expects register 13 to point to a standard 72-byte save area.

DSICBS Macro Instruction

The DSICBS macro instruction is used to include the NCCF control blocks that are necessary for particular functions, such as a user-written command processor. The macro instruction ensures that a control block is included only once, that any necessary inner control blocks are included, and that all definitions for inner control blocks precede the definition of the outer control block. DSICBS also controls the format, and printing or suppression, of DSECTs for the control blocks.

Name	Operation	Operands
[name]	DSICBS	[cbname,...] [,EJECT= {YES | **NO**}] [,DEFER= {ALL | THESE | INCLUDE | **NO**}] [,PRINT= {**YES** | NO}]

cbname
specifies the name of an NCCF control block (starting with DSI) to be included. Names must be separated by commas. *cbname* may not be used with DEFER=INCLUDE. Valid control block names are those in Appendix C.

EJECT
specifies that EJECT statements are performed between each control block expansion and after the last expansion.

DEFER=ALL
specifies that all subsequent DSICBS macro instructions are not expanded until a DSICBS DEFER=INCLUDE is encountered.

DEFER=THESE
specifies that these control block expansions are delayed until a DSICBS DEFER=INCLUDE is encountered.

DEFER=INCLUDE
specifies that any deferred control block expansions are to be expanded at this point in the program.

DEFER=NO
specifies that the control block or blocks are to be expanded immediately.

PRINT=YES
specifies that the control block expansion is to be printed.

PRINT=NO
specifies that the control block expansion is not to be printed.

DSICES Macro Instruction

The DSICES macro instruction uses the specified buffer or a parse descriptor block (PDB) name to locate a system command entry (DSISCE) that corresponds to the verb. The routine can also use a module name to locate a particular module in the system command table. The routine returns the address of the DSISCE entry to the area specified by the SCTADDR operand.

Name	Operation	Operands
[name]	DSICES	SWB={(register) | symbolic name} { BFR={(register) | symbolic name} } { , PDB={(register) | symbolic name} } { MODNAME=modulename } ,SCTADDR={(register) | symbolic name}

SWB
is a register containing the address of a service work block (SWB) or the symbolic name of a fullword area that contains the address of an SWB.

BFR
is a register containing the address, or the symbolic name of a fullword area that contains the address of the buffer that contains the verb to be analyzed. **Note:** *This buffer must have a properly initialized BUFHDR.*

PDB
is a register containing the address of a completed parse descriptor block to be used as input or the symbolic name of a fullword area containing the address of that PDB.

MODNAME
specifies the module name to be located in the system command table. The *modulename* may be specified as the field containing the module name or as the module name itself. If you specify the module name itself, this name must be enclosed in single apostrophes.

SCTADDR
is a register containing the address of a user-provided fullword area, or the symbolic name of that area, where the address of the name or verb in the system command entry that corresponds to the verb or module name will be returned. This area is mapped by the DSISCE DSECT shown in Appendix C.

The return codes for the command analysis routine are found in register 15. They are as follows:

0 A regular command was found in the system command table; the address was returned.

4 The command found can be processed as either a regular or immediate command; the address was returned.

8 An immediate command was found in the system command table; the address was returned.

12 No match was found for the input verb; no address was returned.

20 The command found is incompatible with the task type invoking the routine; the address was returned.

DSIDATIM Macro Instruction

The DSIDATIM macro instruction obtains and formats the time and date and places them in an output area. This macro instruction can be used, for example, to show the date and time on a message.

Name	Operation	Operands
[name]	DSIDATIM	AREA={(register) | symbolic name}
		,FORMAT={EBCDIC | BINARY}

AREA
is a register containing the address of, or the symbolic name of, an area into which the date and time are returned. The area does not have a buffer header.

FORMAT
specifies the format of the output. EBCDIC returns the date and time in 17 bytes, formatted as follows:

```
'mm/dd/yy hh:mm:ss'
```

BINARY returns the date and time in 8 bytes as follows:

```
x'00yydddChhmmss0C'
```

yy is the year and *ddd* is the Julian date. *hh* is hours, *mm* is minutes, and *ss* is seconds. If no value is specified, EBCDIC is assumed. **Note:** *A VSE installation may change the date format during system definition.*

DSIDEL Macro Instruction

The DSIDEL macro instruction is used to delete user-defined modules. The user specifies the name or address of the module to be deleted.

Name	Operation	Operands
[name]	DSIDEL	{EP=modulename | EPLOC=address}

EP=modulename
specifies the name of the module to be deleted.

EPLOC=address
specifies the address of an 8-byte field containing the module name to be deleted. The module name should be left-justified and padded with blanks.

The return codes for DSIDEL are in register 15, as follows:

Zero — Module has been deleted.

Nonzero — Attempt to delete module was unsuccessful.

DSIDKS Macro Instruction

The DSIDKS macro instruction obtains storage for the disk service area, initialize the disk service block (DSB) and the buffer header of the input buffer, locate a book or member specified by the file name, and read the records in that book or member upon request. Users must have a copy of the DSECT for DSIDSB, the NCCF disk services block (see Appendix C).

For more information on DSIDKS, see "Disk Services" earlier in this chapter.

Name	Operation	Operands
[name]	DSIDKS	SWB={(register) | symbolic name} ,DSBWORD={(register) | symbolic name} { ,TYPE={CONN | FIND | DISC} ,NAME={(register) | symbolic name} ,TYPE=READ }

SWB
is a register containing the address of a service work block (SWB) or the symbolic name of a fullword area that contains the address of an SWB.

DSBWORD
is a register containing the address of a user-provided fullword area or the symbolic name of that area. When the routine completes processing for TYPE=CONN, this area contains the address of the disk service area that has been obtained and initialized. For other disk service requests, this disk service area address must be returned to the service routines.

TYPE=CONN
specifies that the service routine is to obtain storage for and initialize the disk service control blocks and input buffer. The address of the disk service block (DSB) is returned in the area specified by the DSBWORD operand. The DSIDKS macro must be issued with this option before any other options may be chosen.

TYPE=FIND
specifies that the service routine is to determine whether the book or member specified by the NAME operand is in the appropriate NCCF library and read the first record if it is found. When the routine has completed processing, the address of the input area containing a buffer header and the record is in the DSB field DSBBUFF. This option can only be issued after the CONN option has been issued.

TYPE=DISC
specifies that the service routine is to free the control blocks and input buffer used by the disk service routines.

NAME
for TYPE=CONN and TYPE=DISC, is a register containing the address of an 8-character user area or the symbolic name of that area. The area should contain the caller's NCCF definition name (left-justified and padded with blanks): for example, DSIPRF or DSIPARM. (See *NCCF Installation.*) For TYPE=FIND, NAME is a register containing the address of an 8-character user area that contains the name of the book or member to be read, or is the symbolic name of that user area.

TYPE=READ
specifies that the service routine is to read the next sequential record in the book or member or return an end-of-data indication if there are no more records in the book or member. This option can only be issued after the FIND option has been issued.

The return codes from DSIDKS (types CONN and DISC) are found in register 15. They are as follows:

CONN	0	Disk service area obtained and initialized successfully.
	4	NCCF definition name specified was not found in the data control table.
	12	No storage was available for a disk service area.
DISC	0	The control blocks and buffer were freed successfully.
	20	The control block identifier specified was invalid and no storage was freed.

The return codes from DSIDKS (types FIND and READ) are found in register 15. They are as follows:

FIND	0	The book or member was found, and the first record was read.
	4	The book or member was not found in the source statement library or the specified library.
	8	The book or member was found but an I/O error occurred on the first read.
	12	The specified NCCF definition name has not been opened.
READ	0	The record was read successfully.
	4	An end of data was reached.
	8	An I/O error occurred during reading.
	12	The reading of this record is prohibited; an I/O error may have occurred, end of data may have been reached, or the caller did not issue TYPE=FIND first.

DSIFRE Macro Instruction

The DSIFRE macro instruction frees storage that was obtained using the DSIGET macro instruction. Optionally, DSIFRE dequeues the storage buffer from the user's task vector block (TVB). Registers 2 through 12 may be used for register notation. DSIFRE is intended to allow the user to free the buffer from the TVB chain. DSIFRE macro always generates a reentrant code.

For more information on DSIFRE, see "Getting and Freeing Storage" earlier in this chapter.

Name	Operation	Operands
[name]	DSIFRE	[{E | R}]
		,LV={n | (register)}
		,A={(register) | symbolic name}
		[,SP={(register) | number}]
		[,LISTA={(register) | symbolic name}]
		[,Q={YES | NO}]
		[,TASKA={(register) | symbolic name}]
		[,EXIT={YES | NO}]

E
specifies the element form of FREEMAIN (OS/VS only). This value must not be coded if Q=YES is specified.

R
specifies the register form of FREEMAIN (OS/VS only). This value must not be coded if Q=YES is specified.

LV
is the number of bytes, or is a register containing the number of bytes, of storage to be freed.

A
is a register containing the address of the storage to be freed, or is the symbolic name of the fullword area containing the address of the storage to be freed.

SP
specifies the subpool number (MVS only) from which the storage is to be freed, or specifies a register loaded with the subpool number. 0 through 127 are acceptable values, and 0 is the default value.

LISTA
is a register containing the address of a 14-byte area in the user's dynamic storage or is the symbolic name of that area. The value must be on a

fullword boundary. DSIFRE builds a parameter list in that area to pass to DSIQFM, which frees the storage and dequeues it from the user's task vector block (TVB). This parameter is required if Q=YES is specified.

Q
specifies whether the storage is to be dequeued from the user's TVB. If Q=YES is specified, LISTA, TASKA, and EXIT must also be specified, and register 13 must point to a 72-byte save area. NO is the assumed value.

TASKA
is a register containing the address of the task vector block (TVB), or is the symbolic name of that TVB, from which the storage is dequeued. This operand is required if Q=YES is specified.

EXIT
YES specifies that the storage release request is from DSIEX01, DSIEX02, or an immediate command processor. NCCF has an exit queue and a NCCF mainline queue, and the EXIT specification prevents queuing problems. NO is the assumed value. EXIT is required if Q=YES is specified.

The return codes for DSIFRE are in register 15:

0 Storage was successfully freed (and dequeued, if specified).

4 Storage was found on the queue and was dequeued but was not freed (FREEVIS or FREEMAIN failure).

20 Storage was not found on the queue.

DSIGET Macro Instruction

The DSIGET macro instruction obtains storage. Optionally, DSIGET can be used to queue the obtained storage to the user's task vector block (TVB). Registers 2 through 12 may be used for register notation. DSIGET is intended to allow the user to queue storage on the TVB chain so that, if abnormal termination occurs, NCCF can free the storage at logoff.

For more information on DSIGET, see "Getting and Freeing Storage" earlier in this chapter.

Name	Operation	Operands
[name]	DSIGET	LV={n | (register)} ,A={(register) | symbolic name} [,SP={(register) | symbolic name}] [,REENT={YES | **NO**}] [,BNDRY={PAGE | **DBLWD**}] [,LISTA={(register) | symbolic name}] [,Q={YES | **NO**}] [,TASKA={(register) | symbolic name}] [,EXIT={YES | **NO**}]

LV

is the number of bytes, or is a register containing the number of bytes, of storage to be obtained.

A

is a register containing the address of the fullword, or is the symbolic name of the fullword, into which the address of the obtained storage is returned.

SP

specifies the subpool number (MVS only) from which the storage is to be obtained, or specifies a register loaded with the subpool number. 0 through 127 are acceptable values and 0 is the default value.

REENT

For OS/VS, REENT=YES must be coded. For VSE, this operand is not required, and is ignored if present. REENT specifies whether the reentrant form of DSIGET is used.

BNDRY=PAGE

specifies that the obtained storage is to be aligned on a page boundary.

BNDRY=DBLWD

specifies that the obtained storage is to be aligned on a doubleword boundary.

LISTA

is a register containing the address of a 14-byte area in the user's dynamic storage, or is the symbolic name of that area. The value must be on a fullword boundary. DSIGET builds a parameter list in that area to pass to DSIQGM, which obtains the storage and queues it to the user's task vector block (TVB). This parameter is required if Q=YES and REENT=YES are specified.

Q

specifies whether the obtained storage is to be queued to the user's TVB. If Q=YES is specified, REENT=YES, LISTA, TASKA, and EXIT must also be specified, and register 13 must point to a 72-byte save area. For OS/VS, REENT=YES must be specified if Q=YES is coded. NO is the assumed value.

TASKA

is a register containing the address of the task vector block (TVB), or is the symbolic name of the TVB to which the obtained storage is queued. This operand is required if Q=YES is specified.

EXIT

YES specifies that the storage request is from DSIEX01, DSIEX02, or an immediate command processor and Q=YES was specified. NCCF has an exit queue and an NCCF mainline queue, and the EXIT specification prevents queuing problems. NO is the assumed value.

The return codes for DSIGET are in register 15:

0 Storage was successfully obtained.

4 No storage was obtained.

8 GETVIS or GETMAIN was issued by a program running in real mode.

12 No storage was available, or the length specified was less than zero, or no continuous area of storage of the size requested was available.

DSIKVS Macro Instruction

The DSIKVS macro instruction is used in a command processor to determine if a particular keyword or a particular keyword and its scope value are in the operator's command scope. A return code is shown in register 15 indicating whether the operator who issued the command has been authorized to issue it with the particular keyword or value or both.

Name	Operation	Operands
[name]	DSIKVS	SWB={(register) | symbolic name} {CMD={(register) | symbolic name} {SCTADDR={(register) | symbolic name} ,KEYWORD={(register) | symbolic name} [,VALUE={(register) | symbolic name}]

SWB
is a register containing the address of a service work block (SWB) or the symbolic name of a fullword area that contains the address of an SWB.

CMD
is a register containing the address of an 8-byte field or the symbolic name of an 8-byte field that contains the command name left-justified and padded with blanks. Either CMD or SCTADDR must be specified.

SCTADDR
is a register containing the address of a fullword field or the symbolic name of a fullword field that contains the SCT entry address for the command that is to be checked. Either CMD or SCTADDR must be specified.

KEYWORD
is a register containing the address of an 8-byte field or the symbolic name of an 8-byte field that contains the keyword left-justified and padded with blanks. This operand is required.

VALUE
is a register containing the address of an 8-byte field or the symbolic name of an 8-byte field that contains the value left-justified and padded with blanks. VALUE is an optional keyword that is specified when VALCLASS checking is desired.

Note: If both KEYWORD and VALUE are specified, KEYWORD is scope checked before VALUE. If KEYWORD results in a nonzero return code, VALUE will not be checked.

The return codes for the KEYCLASS and VALCLASS lookup service routine are found in register 15. They are as follows:

0 The specified keyword and value are in the operator's scope of commands.

4 The specified keyword was not in this operator's scope of commands.

8 The specified value was not in this operator's scope of commands.

12 A required parameter was missing or an invalid parameter was specified in the macro instruction.

16 No storage is available.

DSILCS Macro Instruction

The DSILCS macro instruction:

- Gets a service work block (SWB) for the caller and places the address of that SWB in a fullword area specified by the CBADDR operand.
- Frees an SWB after use.
- Gets a command work block (CWB) for the caller and places the address of that CWB in a fullword area specified by the CBADDR operand.
- Frees a CWB after use.
- Locates a task vector block (TVB) by operator identification.
- Locates a task vector block (TVB) by LU name.
- Locates (from a specified starting position) the next active TVB for an NCCF-NCCF task, a hard-copy task, an operator station task, or an optional task.
- Locates a task vector block for an operator designated as a receiver of authorization messages by the AUTH statement of a profile definition.

For more information on DSILCS, see "Getting, Freeing, and Locating a Control Block" earlier in this chapter.

Name	Operation	Operands
[name]	DSILCS	CBADDR={(register) | symbolic name} { CWB={GET | FREE} SWB={GET | FREE} TVB={(register) | symbolic name} { LU={(register) | symbolic name} / OPID={(register) | symbolic name} } , AUTHRCV=YES NEXT={ OST / HCT / NNT / OPT } }

CBADDR
is a register containing the address of user-provided fullword area or the symbolic name of that area. The specified SWB, CWB, or TVB address is returned to this area (for the GET option).

CWB=GET
specifies that the caller needs a command work block. The address of the CWB is returned to the area specified by the CBADDR operand. The user must initialize the CWBTIB field with the address of his TIB.

CWB=FREE
specifies that the caller wishes to free the command work block whose address is found in the area specified by the CBADDR operand.

SWB=GET
specifies that the caller needs a service work block. The address of an SWB is returned to the area specified by the CBADDR operand. The user must initialize the SWBTIB field with the address of his TIB.

SWB=FREE
specifies that the caller wishes to free the service work block whose address is found in the area specified by the CBADDR operand.

TVB
is a register containing the address of the task vector block where the routine begins the search for the TVB specified by LU, OPID, NEXT, or AUTHRCV. The symbolic name of an area containing the address of this TVB may also be supplied. The address of the beginning of this TVB chain is found in the main vector table, DSIMVT. The TVB address found is placed in the area specified by CBADDR after the routine has completed processing. **Note:** *The routine searches from the address specified to the end of the TVB chain; it does not loop to the beginning of the TVB chain.*

LU
is a register containing the address of an 8-byte LU name field or the symbolic name of the 8-byte LU name field. This name is used to find a TVB with a matching LU name.

OPID
is a register containing the address of an 8-byte operator identification field or the symbolic name of the 8-byte operator identification field. This name is used to find a TVB with a matching operator identification.

AUTHRCV
specifies that the routine is to search for the first TVB for an operator authorized to receive messages related to successful and unsuccessful logons and lost station messages. (See the discussions of the AUTH statement and of unsolicited message routing in *NCCF Installation.*)

NEXT=OST
specifies that the TVB associated with the next active operator station task is to be located.

NEXT=HCT
specifies that the TVB associated with the next active hard-copy log task is to be located.

NEXT=NNT
specifies that the TVB associated with the next active NCCF-to-NCCF task is to be located.

NEXT=OPT
specifies that the TVB associated with the next optional task is to be located.

Return codes for the control block location routine are found in register 15. They are as follows:

0 Successful; the address was returned, or the control block was freed.

4 No TVBs of the type specified were found.

8 End of the TVB chain, if TVB was specified.

8 No storage was available, if SWB=GET or CWB=GET was specified.

8 Defective control block, if SWB=FREE or CWB=FREE was specified.

DSILOD Macro Instruction

The DSILOD macro instruction is used to load user-defined modules. The user specifies the name or address of the module to be loaded and the address of a BLDL list.

Name	Operation	Operands
[name]	DSILOD	{ EP=modulename } ,LISTA= { (register) } { EPLOC=address } { symbolic name } [,DCB=address]

EP=modulename
specifies the name of the module to be loaded.

EPLOC=address
specifies the address of an 8-byte field containing the *modulename* to be loaded. The *modulename* should be left-justified and padded with blanks.

LISTA
is a register containing the address of a 62-byte area in the user's dynamic storage or is the symbolic name of that area. The value must be on a fullword boundary. DSILOD uses this area as a BLDL list.

DCB
optionally specifies the address of the DCB for a partitioned data set to be searched for the module.

The return codes for DSILOD are in register 15, as follows:

Zero Module has been loaded.

Nonzero BLDL could not locate the module.

If the module is successfully loaded, register 0 contains the load point address of the module. Register 1 contains the authority code in the high-order byte and the module length in the low-order three bytes.

DSIMBS Macro Instruction

The DSIMBS macro instruction can be used to determine the size of the buffer required to accomodate the message to be edited. The routine can then be used to edit an NCCF or user-supplied message into a buffer provided by the caller. Callers may supply variable fields to be inserted into NCCF messages or supply unique messages of their own with varying positional fields. A total of nine positional parameters may be replaced with user substitutions. DSIMBS may also be used to pass to NCCF the address of a message table generated with the DSIMDS macro instruction.

Name	Operation	Operands
[name]	DSIMBS	SWB= {(register) | symbolic name}
		{ ,MID= { nnn | (register) | symbolic name | *equatename } [,P1= (text,lngth[,padlng,side,fill]) . . . ,P9= (text,lngth[,padlng,side,fill])] ,MSGA= (pdb1addr,pdb2addr) }
		{ ,BFR= {(register) | symbolic name} | ,MSGSIZE={(register) | symbolic name} }
		[,MSGTBL={(register) | symbolic name}]

SWB
is a register containing the address of a service work block (SWB) or the symbolic name of a user area that contains the address an SWB.

MID

specifies the identification of the NCCF message that is to be edited for the caller. The message may be specified by the message number (*nnn*), in a register, in a user area specified by symbolic name, or by the equate name preceded by an asterisk. (For example, if MSG999 EQU 999, you could specify MID=*MSG999.)

MSGA=(pdb1addr,pdb2addr)

pdb1addr is a register containing the address of the parse descriptor block (PDB) or the symbolic name of a fullword area that contains the address of the PDB that contains the addresses and lengths of the variable fields to be substituted into the message text. *pdb2addr* is a register containing the address of the parse descriptor block (PDB) or the symbolic name of a fullword area that contains the address of the PDB that contains the message skeleton to be edited. This is not an NCCF message; it is supplied by the user.

Because the variable field information is contained in *pdb1addr*, the P1...P9 operands cannot be used if MSGA is specified.

BFR

is a register containing the address of the user area where the edited message is to be returned or the symbolic name of a fullword area that contains the address of the user area. **Note:** *This buffer must have a properly initialized BUFHDR, except for the HDRMLENG field, which is initialized by DSIMBS.*

MSGSIZE

is a register containing the address of a user-provided fullword area or the symbolic name of that area. This operand should only be used to request the service routine to determine the size of the buffer needed for the message to be edited. When the routine has completed processing, the required size is returned in this area.

P1...P9

are used only in combination with the MID operand. They specify the positional fields in an NCCF message that are to be replaced by user-supplied text. The first two values, *text* and *lngth* must be specified; the others are optional.

text

is a register containing the address of the variable text, or the symbolic name of the area that contains the text that is to be substituted into the edited NCCF message.

lngth

is the length of the variable text that is to be substituted into the edited NCCF message. The maximum length is 255 characters specified in character format or a binary value in a register or in a user area specified by symbolic name.

padlng
is the total length of the variable field to be padded with fill characters. This length must be equal to or less than the length specified by the *lngth* operand. The maximum size is 255 characters specified in character format or a binary value in a register or in a user area specified by symbolic name.

side
may be specified as L for left-fill or R for right-fill. The default is right-fill.

fill
is the character to be used as the fill character for the area to be padded. The default fill character is a blank (hex 40).

MSGTBL
is a register containing the address of a user-defined message table or the symbolic name of a fullword that contains the address of the message table. The table must be generated using the DSIMDS macro instruction.

Return codes for the DSIMBM macro instruction are found in register 15. They are as follows:

0 Successful: (1) The edited message is in the provided buffer and the length of the message is stored in the message length field of the buffer header, or (2) the size of the message buffer required has been calculated and stored in the area specified by MSGSIZE.

4 The edited message is in the provided buffer, but the message skeleton contained a parameter for which the caller did not supply text. The message contains the characters &*n* where *n* may be from 1 to 9.

8 Unsuccessful: The buffer overflowed, and the message has been truncated. The size of the truncated message has been stored in the message length field of the buffer header.

12 The message number specified could not be found in the NCCF message module, DSIMDM. NCCF message DSI000A is edited into the caller's buffer. If the buffer size only was requested, the size of message DSI000A is returned.

16 The caller did not supply a buffer address.

20 Combined conditions 4 and 8 occurred.

24 Combined conditions 8 and 12 occurred.

28 A validity check failed on the user message definition module. The address passed in the MSGTBL operand does not point to a message definition module that was created with the DSIMDS macro instruction.

DSIMDS Macro Instruction

The DSIMDS macro instruction generates an NCCF message definition module (DSIMDM) that contains all messages issued by NCCF. It can also be used to generate a user-defined message module to contain messages issued by the user in exit routines, command processors, and command lists. After a message definition module has been defined, it must be link-edited into the NCCF load library.

Three forms of the DSIMDS macro instruction are required to generate a message definition module. These forms are described below and must be coded in the sequence shown.

Format 1: Start Message

Name	Operation	Operands
[name]	DSIMDS	prefix,TYPE=START

name
is required to start the message definition module. *name* becomes the CSECT name for the module.

Note: DSIMDM must be specified for name if the NCCF message definition module is being defined or modified.

prefix
is a required positional operand that becomes the prefix for the messages in the module. *prefix* must be DSI for the NCCF message module.

TYPE=START
specifies the beginning of generation for the message definition module.

Format 2: Message Text

Name	Operation	Operands
[name]	DSIMDS	xxx,'message text &&n'c,TYPE={A | I}

xxx
is the message number to be given to the message. It may be any number from 001 through 899; for user-originated messages, numbers 900 through 999 should be used.

Note: When coding your own message CSECT, you must code a message 000 to be issued when an invalid message number is specified. The user-coded message 000 should have one insert containing the invalid message number. You may want to use wording similar to NCCF's message DSI000I:

```
DSI000I  NCCF MESSAGE xxx ISSUED BUT DOES
NOT EXIST - CALL IGNORED.
```

message text
is the text of the message to be added or changed.

&&n
signifies that variable information is to be substituted at this position in the message. &&1 through &&9 may be specified.

TYPE=A
specifies that the message is to be an action message (one for which an appropriate action must be taken).

TYPE=I
specifies that the message is for information only. No specific action need be taken.

Format 3: End Message

Name	Operation	Operands
	DSIMDS	TYPE=END

TYPE=END
specifies the end of the message defintion module. This is the last statement specified.

DSIMQS Macro Instruction

The DSIMQS macro instruction queues a user-supplied message to the message queue of a task's task vector block (TVB). This message appears on the operator's screen or hard-copy log, depending upon which identification is specified. Buffers that are formatted as internal function requests (IFRs) are not displayed; they cause the receiving subtask to take the action requested by the IFR.

For more information on DSIMQS, see "Message Queuing" earlier in this chapter.

Name	Operation	Operands
[name]	DSIMQS	SWB={(register) | symbolic name} ,BFR={(register) | symbolic name} {TASKID={(register) | symbolic name} {AUTHRCV={YES | NO}

SWB
is a register containing the address of a service work block (SWB) or the symbolic name of a fullword area that contains the address of an SWB.

BFR
is a register containing the address of a buffer or the symbolic name of a fullword area containing the address of a buffer. **Note:** *This buffer must have a properly initialized BUFHDR.*

TASKID
is a register containing the address of a user-provided 8-byte area or the symbolic name of that area. The area should contain the 8-character operator identification of the task for which the message is to be queued. If TASKID is specified, AUTHRCV should not be specified.

AUTHRCV
specifies that the first operator designated as the receiver of authorization messages (by the AUTH statement of profile definition) is to receive the message. If no operator is authorized, the message is queued for the system console. (See the discussions of the AUTH statement and of unsolicited message routing in *NCCF Installation.*)

The return codes for the message queuing service routine are found in register 15. They are as follows:

0 Successful completion.

4 The format of the buffer that was passed was invalid.

8 The task identification that was passed could not be found.

12 A buffer could not be obtained.

DSIOIS Macro Instruction

The DSIOIS macro instruction locates the specified operator identification in the DSIOIT table and returns the relative position of the entry to a user-provided fullword area.

Name	Operation	Operands
[name]	DSIOIS	SWB= {(register) | symbolic name}
		,OPID= {(register) | symbolic name}
		,OITPOS= {(register) | symbolic name}

SWB
is a register containing the address of a service work block (SWB) or the symbolic name of a fullword area that contains the address of an SWB.

OPID
is a register containing the address of an 8-byte (left-justified) operator identification field or the symbolic name of that field.

OITPOS
is a register containing the address of a fullword area or the symbolic name of that area. When the routine has located the specified operator identification in DSIOIT, that entry's relative position is returned to this fullword area. For example, the third entry results in a fullword 3 being returned.

The return codes for the DSIOIM routine are found in register 15. They are as follows:

0 Successful; the position of the entry has been returned.

4 Unsuccessful; the entry could not be found in DSIOIT.

DSIPAS Macro Instruction

The DSIPAS macro instruction receives a command operand as input and searches the system command table (DSISCT) to determine whether the operand is an alias for an NCCF operand. If it is an alias, the regular NCCF value is returned to a user-provided area. If it is not an alias, the input value itself is returned to the user area. If the value is invalid, blanks are returned to the input area.

Name	Operation	Operands
[name]	DSIPAS	SWB= {(register) | symbolic name}
		,PDB= {(pdbname,entrynumber)}
		,OUT= {(register) | symbolic name}

SWB
is a register containing the address of a service work block (SWB) or the symbolic name of a fullword area that contains the address of an SWB.

PDB=pdbname
is a register containing the address of a fullword area or is the symbolic name of a fullword area that contains the address of that area. The area should contain the name of the parse descriptor block to be used as input.

PDB=entrynumber
(1) is the number of a register containing the entry number; or (2) is the symbolic name of an area containing the entry number of the field in the parse descriptor block that is to be examined; or (3) is an entry number in character form specified in single quotation marks. For example, the third entry is specified as '3'.

OUT
is a register containing the address of a user-provided 8-byte area to which the NCCF equivalent of the input operand is returned if found, or the symbolic name of that user area.

The return codes for the parameter alias service routine are found in register 15. They are as follows:

0 A regular NCCF opreand value was returned.

4 No equivalent was found; the same operand is returned.

8 Invalid operand; blanks are returned.

DSIPOS Macro Instruction

The DSIPOS macro instruction indicates completion of an event by posting an event control block (ECB).

Name	Operation	Operands
[name]	DSIPOS	ecbaddress[,compcode]

ecbaddress
is a register from 1 through 12 containing the address of the ECB (an aligned fullword), or the symbolic name of the ECB. If a register is specified, it must be coded in parentheses.

compcode
is the value of the completion code to be placed in the ECB (0 through 16,777,215), or a register (0, 2 through 12) containing the value. If a register is specified, it must be coded in parentheses. If no value is specified, 0 is assumed.

Note: These operands are positional and must be specified in the indicated order.

DSIPRS Macro Instruction

The DSIPRS macro instruction uses an input buffer with an initialized buffer header to determine the size of the parse table that must be built to accommodate the data contained in the buffer. The routine may then be invoked again to build the parse table in a user-provided area. DSIPRS then delimits the segments of the data contained in the buffer and puts the number of segments found and the indicators to these segments in the table. See the description of the parse descriptor block (PDB) later in this chapter.

The following delimiters are recognized:

b
,
.
=

Name	Operation	Operands
[name]	DSIPRS	SWB= {(register) | symbolic name} ,BFR= {(register) | symbolic name} ,{PDBSIZE= {(register) | symbolic name} ,{PDB= {(register) | symbolic name} [,FIRST= {YES | NO }] [,DELIM= ('D1', 'D2',...'Dn')]

SWB
is a register containing the address of a service work block (SWB) or the symbolic name of a fullword area that contains the address of an SWB.

BFR
is a register containing the address of the buffer that is to be used for input or the symbolic name of a fullword area that contains the address of that buffer. **Note:** *This buffer must have a properly initialized BUFHDR.*

PDBSIZE
is a register containing the address of a fullword area, or the symbolic name of that area, to which the size of the parse table is to be returned.

PDB
is a register containing the address of a fullword area or the symbolic name of that area, where the parse table is to be built. The parse table must include a user-initialized DSICBH header (see Appendix C) containing control block identification and length before the data can be parsed.

FIRST
is an optional parameter that indicates whether the first word of the buffer can be delimited only by a blank (YES) or by any delimiter (NO).

DELIM
is an optional parameter that allows the user to specify delimiters to be used instead of the NCCF defaults. NCCF default delimiters are blank, comma, period, equal sign. Blank is always considered a delimiter, ever if the user specifies his own delimiters.

The return codes for the parsing service routine are found in register 15. They are as follows:

0 The correct size of the table was found or the command was parsed successfully and the table was built.

4 The command was parsed, but there was no data in the buffer received and only the buffer address and number of entries (0) in the table could be returned.

8 The parse table size was too small for the command; a partial table was built, and the number of entries was set to the number that the table could hold. The size should be increased.

12 Unpaired apostrophes were found.

100 No PDB was passed.

DSIPSS Macro Instruction

The DSIPSS macro instruction invokes the appropriate presentation service routine, DSIPS1-DSIPS15. These modules are an interface between NCCF and various devices. They control the screen formats, organize the data into a specific form for each device, and send the data.

For more information on the DSIPSS macro instruction, see "Presentation Services" earlier in this chapter.

Name	Operation	Operands
[name]	DSIPSS	SWB= {(register) | symbolic name} [,APPLID= {(register) | symbolic name}] ,TYPE= { OUTPUT | IMMED | XSEND | SCRSIZE | WINDOW | PANEL | ASYPANEL | CANCEL | PSSWAIT | TESTWAIT } [,ECBLIST= {(register) | symbolic name}] [,BFR= {(register) | symbolic name}] ,OPTIONS= { MSG | SEG | FIRST | MIDDLE | LAST | ONLY } [,SIZE= {(register) | symbolic name}] [,PANEL= {(register) | symbolic name}]

SWB

is a register containing the address of a service work block (SWB) or the symbolic name of a user area that contains the address of an SWB. This operand is required.

APPLID

is a register containing the address of an 8-byte area, or the symbolic name of an 8-byte area, that contains the name (left-justified and padded with blanks) of the application program to which the data is to be sent. This name should be the same as the name specified on the START command when a session is started. APPLID is specified only when TYPE=XSEND.

TYPE=OUTPUT

specifies that the routine is to send a message to the operator's terminal. This option should not be used by immediate command processors or user exit routines DSIEX01, DSIEX02, or DSIEX03. The maximum message length allowed (before truncation occurs) is 37,767 characters. Upon completion of the macro, the length of the text in the HDRMLENG field of the BUFHDR is set to the length of the data after any trailing blanks have been truncated.

TYPE=IMMED

specifies that the routine is to send a message to the operator station's immediate message area. The maximum message length allowed before truncation occurs is 71 characters. This option can be used only with immediate command processors or in the DSIEX01 exit routine. When this operand is specified, no message header information is sent to the display screen. TYPE=IMMED terminates full-screen mode and causes subsequent terminal input to be treated as commands.

TYPE=XSEND

specifies that the routine is to send data to another NCCF with which a session exists. The maximum length allowed before truncation occurs is 256 characters.

TYPE=SCRSIZE

specifies that the routine is to return the screen size in row-column format.

TYPE=WINDOW

requests information on the output area size of the standard NCCF screen. This option is valid only from an operator station task (OST). (Under any other task, the request is considered null; register 15 contains a return code of zero, but no function is performed.) Three output area sizes are returned:

- Minimum
- Current
- Maximum

The minimum window size may be used to produce screens that are independent of the current window size. The current window shows what screen size is currently in effect. The maximum window size is useful for calculating the maximum storage needed to produce full-line panels.

Note: TYPE=WINDOW applies only to command processors that use DSIPSS in standard NCCF mode. In full-line mode, the screen is automatically changed to single line input. When standard NCCF mode is reentered, the input area is set to the size indicated by the INPUT command.

TYPE=PANEL

specifies that the issuing routine is to format the screen using synchronous full-screen mode, as opposed to the standard NCCF mode. This option is allowed only from an operator station task (OST). When requesting output, the issuer will build a 3270 data stream; when requesting input, the issuer will receive the untranslated 3270 data stream.

The first operator input received for TYPE=PANEL is used to satisfy a first panel request for input. NCCF treats subsequent terminal input as commands. The operator should enter one request and then wait for a new screen panel before entering data.

For more information on synchronous full-screen mode, see the section titled "Full-Screen Command Processor Considerations" in Chapter 4. For more information on the 3270 data stream, refer to the appropriate 3270 manual.

TYPE=ASYPANEL
specifies that the issuing routine is to format the screen using asynchronous full-screen processing through the posting of event control blocks (ECBs).

While the asynchronous full-screen command processor is running, input to the NCCF terminal is treated as input to the command processor and not as a direct input to NCCF. To allow the next input to be treated as a command, issue TYPE=OUTPUT or TYPE=IMMED.

For more information on asynchronous full-screen mode, see the section titled "Full-Screen Command Processor Considerations" in Chapter 4. TYPE=ASYPANEL is allowed only from an operator station task (OST).

TYPE=CANCEL
cancels pending asynchronous full-screen input after the receive from the terminal has been issued. This option is used when changing the characteristics of the asynchronous full-screen processor, such as the ECB address or the panel address. TYPE=CANCEL is allowed only from an operator station task. This option can be invoked whether or not a DSIPSS TYPE=ASYPANEL is active or the input from TYPE=ASYPANEL has been posted as complete.

After TYPE=CANCEL is issued, no further input is received from the terminal until TYPE=OUTPUT, TYPE=IMMED, TYPE=PANEL or TYPE=ASYPANEL is issued.

TYPE=PSSWAIT
specifies that a command is to wait for both a list of its own events and a list of NCCF events that should be allowed to interrupt the command events. TYPE=PSSWAIT is allowed only from an operator station task. This option should be used with TYPE=ASYPANEL.

Note: Use the DSIWAT macro instruction if you do not want the command to wait for the completion of NCCF events.

TYPE=TESTWAIT
allows a command processor to test whether an NCCF event has occurred that should interrupt the asynchronous full-screen command processor. TYPE=TESTWAIT is allowed only from an operator station task. This option can be used before a DSIPSS TYPE=ASYPANEL is issued to determine if the asynchronous full-screen panel input/output should be attempted. If DSIPSS TYPE=PSSWAIT is used to wait for NCCF events, this option can prevent unnecessary screen input/output by allowing testing before panel input/output is requested.

ECBLIST
for TYPE=PSSWAIT is a register containing the address of an ECB list or the symbolic name of an ECB list. An ECB list is a list of addresses of user-defined event control blocks that will be copied and combined with an NCCF ECB list. NCCF waits on this combined list; when one of the

events associated with this list is posted, control is returned to the next sequential instruction. The input ECB list is made up of fullword ECB addresses. The last address in the list must have the first bit set on to specify that this is the last entry.

BFR
is a register containing the address of a user-provided buffer or the symbolic name of a fullword area that contains the address of the buffer. This buffer should contain the data that is to be processed. **Note:** *This buffer must have a properly initialized BUFHDR.*

OPTIONS=MSG
specifies that the data to be sent is a complete message.

OPTIONS=SEG
specifies that the data to be sent is only a segment of a message. When this operand is specified, no message header information is sent to the screen.

OPTIONS=FIRST
specifies that the data to be sent is to start at the top of the screen, with full 80-byte line width.

OPTIONS=MIDDLE
specifies that the data to be sent is a continuation line.

OPTIONS=LAST
specifies the end of a screen of data. The screen is locked until the operator signals for the screen to be refreshed.

OPTIONS=ONLY
specifies that one full line is to be written at the top of the screen with the rest of the screen left blank.

Note: For OPTIONS FIRST, MIDDLE, LAST, or ONLY, a command processor that is invoked by an operator station task is always out of full line mode.

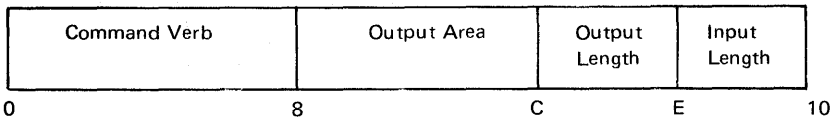
SIZE
for TYPE=SCRSIZE is a register containing the address of a user-provided 4-byte area or the symbolic name of a fullword area that contains the address of a 4-byte area to contain the size of the display screen, in row-column format. For example, a 1920-character screen is defined as X'00180050', since the screen is 24 rows (X'0018') by 80 characters (X'0050').

For TYPE=WINDOW, SIZE is a register containing the address of a 12-byte area or the symbolic name of a 12-byte area in which the window sizes are returned in binary. The format of the area is shown below:

Minimum Window Size		Current Window Size		Maximum Window Size	
Rows	Columns	Rows	Columns	Rows	Columns
0	2	4	6	8	A C

PANEL

for TYPE=PANEL is a register containing the address of a 16-byte parameter list or the symbolic name of a fullword area that contains the address of a 16-byte parameter list. This parameter list contains a command verb, an output area address, an output length, and an input length. The parameter list is formatted as follows:

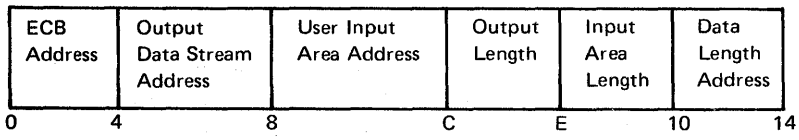


If full-screen output is requested, the output area contains the address of a 3270 data stream containing a 3270 command, WCC, and data. The output length field indicates the length in bytes of the 3270 data stream. The command verb and the input length fields are not used.

To read full-screen input from a terminal, the command verb area contains a valid command to be executed when input is available. The input length field indicates the maximum input data to be expected by NCCF. The command is executed with the 3270 data. Because NCCF does not translate or modify the 3270 data, the parse descriptor block (DSIPDB) may not contain meaningful data. No input buffer is passed to NCCF; NCCF supplies the input buffer.

Note: A sample full-screen command processor is shown in Appendix F.

For TYPE=ASYPANEL, PANEL is a register containing the address of a 20-byte parameter list or the symbolic name of a 20-byte parameter list. The parameter list is formatted as follows:



If asynchronous full-screen output is requested, the output data stream address field contains the address of a 3270 data stream including a 3270 command, WCC, and orders to be written to the terminal. The command must be coded using remote EBCDIC values. The output length field indicates the length, in bytes, of the 3270 data stream (32,767 bytes maximum). The ECB address, input area length, user input area address, and data length address fields are not used if only output is requested.

To read asynchronous full-screen input from a terminal, the ECB address area contains the address of an event control block to be posted when the asynchronous input is received. The user input area address contains the address of a user area into which the address contains the address of a user area into which the full-screen panel data is read. (If the length of the data being read is greater than the user input area, the data will be truncated.) The input area length field indicates the length of the input data area in bytes (32,767 bytes maximum). The data length address field contains the address of a halfword field that is set to the amount of data actually read when the ECB is posted.

Note: For more information on full-screen processing, see "Full-Screen Command Processor Considerations" in Chapter 4. For more information on the 3270 data stream, refer to the appropriate 3270 publications.

The return codes for the presentation service routine are found in register 15. They are as follows:

0 Successful completion. If running under a full-screen processor, see the section titled "DSIPSS Output from a Full-Screen Command Processor" for more information. For TYPE=PSSWAIT, a user ECB has been posted. Check the ECB list to determine which event has completed.

4 For TYPE=XSEND, no RPL was found and no data was sent.

8 Parameter error. There is an error in the formatting of the message buffer header. For TYPE=XSEND, the session is not active and no data is sent. For TYPE=PANEL, the input or output length is invalid, that is, greater than 32,767 bytes (X'7FFF'). For TYPE=ASYPANEL, the parameter list is inconsistent. If the output buffer is specified, its length must also be specified. If the input ECB is specified, the input area address, input area length, and the data length address of the returned length must be specified.

12 There is not enough storage available in NCCF to complete the request. No output will be sent, and the input command processor will not be scheduled.

16 DSIPSS TYPE=OUTPUT was issued for an immediate command or in an IRB system exit routine. Use DSIPSS TYPE=IMMED or DSIMQS instead. Too many OPTIONS=MIDDLE were specified, and the screen is full. This OPTIONS=MIDDLE is treated as an OPTIONS=LAST. If another MIDDLE is issued, it will be treated as an OPTIONS=FIRST. The screen will wrap around, and return code 24 will be issued.

20 No terminal session exists. For TYPE=PANEL, the panel request came from a task other than an operator station task (OST). No output will be sent, and the input command processor will not be scheduled. For TYPE=ASYPANEL, the panel request came from a task other than an OST. No input will be received. For TYPE=CANCEL, the panel request came from a task other than an OST.

24 OPTIONS=FIRST, MIDDLE, LAST, or ONLY was specified in the incorrect order.

28 For OPTIONS=FIRST, MIDDLE, LAST, or ONLY, user exit DSIEX02 specified that full-line output was to be deleted. The output was not written to the screen. Note that this return code does not indicate a severe error.

32 For TYPE=PANEL, no input command processor is scheduled. The operator requested escape to NCCF mode by selecting option 1 when prompted by message DSI817A. See the section titled "DSIPSS Return Codes from a Full-Screen Command Processor" for more information.

36 For TYPE=PANEL or TYPE=ASYPANEL, a temporary error occurred. The contents of the screen have been modified. Reformat the screen using an Erase/Write or Erase/Write Alternate 3270 command. Then retry the request.

40 For TYPE=PANEL, ASYPANEL, or CANCEL a permanent input/output error occcured. Do not retry the request. No output will be sent, and no input processor will be scheduled. For TYPE=ASYPANEL, no input will be received. For TYPE=CANCEL, NCCF is unable to restart normal terminal activity.

44 For TYPE=PANEL, no input is scheduled, because the operator requested reset by selecting option 3 when prompted by message DSI817A. See the section titled "DSIPSS Output from a Full-Screen Command Processor" for more information.

48 For TYPE=ASYPANEL, no input/output is scheduled because the command processor issued a second DSIPSS TYPE=ASYPANEL before the previous request had completed.

56 For TYPE=PSSWAIT or TYPE=TESTWAIT, at least one NCCF ECB was posted.

The ECB post codes for PSS TYPE=ASYPANEL are found in the event control block if one was specified. They are as follows:

0 Successful completion. The requested data is available.

12 There is not enough storage available in NCCF to complete the request. The output data was sent, but the input data is not available.

36 A temporary error occurred during a full-screen read. Retry the request. The output data was sent, but the input data is not available.

40 A permanent error occurred during a full-screen read. Do not retry the request. The output data was sent, but the input data is not available.

52 The requested input was canceled using DSIPSS TYPE=CANCEL. You should not retry the request immediately. The output data was sent, but the input data is not available.

DSIRDS Macro Instruction (ACF/VTAM Only)

The DSIRDS macro instruction locates the specified resource in the authorization and resource table, DSIART, and returns the address of the DSIART entry to a user-provided fullword area.

For more information on DSIRDS, see "Resource Location" earlier in this chapter.

Name	Operation	Operands
[name]	DSIRDS	SWB= {(register) | symbolic name}
		,LUNAME= {(register) | symbolic name}
		,ARTPOS= {(register) | symbolic name}
		[,STATUS= {ACT | INACT}]

SWB
is a register containing the address of a service work block (SWB) or the symbolic name of a fullword area that contains the address of an SWB.

LUNAME
is a register containing the address of a user-provided area or the symbolic name of that area. The area should contain the 8-byte (left-justified) LUNAME that is to be located in the authorization and resource table, DSIART.

ARTPOS
is a register containing the address of a fullword area or the symbolic name of that area. When the routine has located the specified entry in DSIART, that entry's address in the table is returned to this area.

STATUS
specifies whether the LUNAME entry in DSIART is to be marked as active (ACT) or inactive (INACT).

The return codes for the DSIRDM routine are found in register 15. They are as follows:

0 Successful; the entry was found and its address returned.

20 Unsuccessful; the specified entry was not found in DSIART, or the entry is inactive.

DSISSS Macro Instruction (ACF/VTAM Only)

The DSISSS macro instruction checks a specified bit position (as shown in Figure 3-7) in the span name table (DSISNT) and returns the address to a user-provided fullword area of the first entry whose corresponding bit is set to 1.

Name	Operation	Operands
[name]	DSISSS	SWB= {(register) | symbolic name}
		,OITPOS= {(register) | symbolic name}
		,SNTADDR= {(register) | symbolic name}

SWB
is a register containing the address of a service work block (SWB) or the symbolic name of a fullword area containing the address of an SWB.

OITPOS
is a register containing the address of a user-provided fullword area or the symbolic name of that area. This area should contain the bit position to be checked (for the first bit set to 1) in DSISNT. **Note:** *The bit positions in the span name table correspond to entry positions in the operator identification table DSIOIT; for example, the first bit corresponds to the first entry in DSIOIT.*

SNTADDR
is a register containing the address of a user-provided fullword area, or the symbolic name of that area. On input to the DSISSM routine, this area should contain the address of the entry in the span name table where the search is to begin. When the routine has completed processing, this area contains the address of the first entry that the search encountered whose corresponding operator bit was set to 1. **Note:** *The starting address specified in SNTADDR may also be stored elsewhere in case relative location calculations are necessary for searching the authorization and resource table (DSIART).*

The return codes for the DSISSM routine are found in register 15. They are as follows:

0 Successful; an entry was found and its address returned.

12 Unsuccessful; no entry was found; the address originally submitted is still in the area specified by SNTADDR.

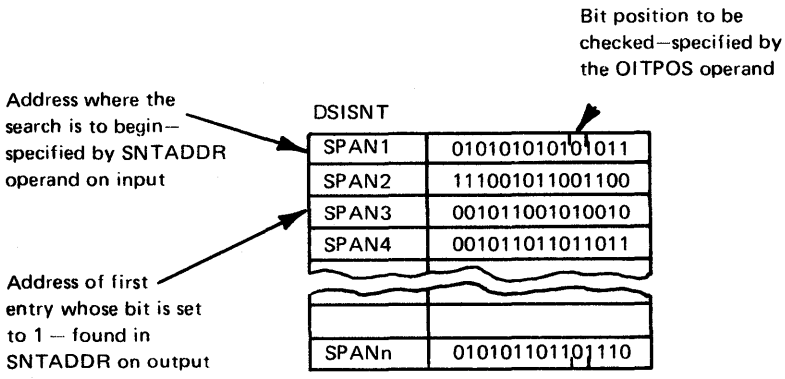


Figure 3-7. Search of the Span Name Table (DSISNT)

DSIWAT Macro Instruction

The DSIWAT macro instruction causes an NCCF subtask to wait for completion of an event.

Name	Operation	Operands
[name]	DSIWAT	{ ECB= {(register) | symbolic name} } { ECBLIST= {(register) | symbolic name} } [,DPR= {(register) | address}]

ECB

is the symbolic name of an aligned fullword to be used as an event control block (ECB), or the address in a register (1 through 12) of an aligned fullword.

ECBLIST

is the symbolic name of a contiguous list of fullword addresses of ECBs, or a register containing the address of the list. The last entry in the list of ECB addresses has the high-order bit (bit 0) set to 1 to indicate the end of the list.

DPR

is the address of the NCCF dispatcher (DSIDPR; VSE only). If this operand is not specified, the address in main vector table field MVTPRAD is used. Addressability to the main vector table (MVT) is required.

The following example shows how DSIWAT can be coded.

```
             DSIWAT ECBLIST=LISTAREA
               .
               .
               .
ECB1         DC F'0'
ECB2         DC F'0'
LISTAREA     DC A(ECB1)
             DC A(ECB2)
             DC A(ECB3)
             DC A(ECB4+X'80000000')
               .
               .
               .
ECB3         DC F'0'
ECB4         DC F'0'
```

Execution resumes when any one ECB is posted. The DSIPOS macro instruction is used to set bit 1 of the ECB to 1. A completion code can also be set in the low-order 3 bytes of the ECB. The VSE supervisor POST macro instruction can also be used to post an ECB. POST sets bit 0 in the second byte of the ECB to 1.

DSIWCS Macro Instruction

The DSIWCS macro instruction writes a message to the system operator console. If the message exceeds 121 characters, it is truncated. The message must have the NCCF buffer header prefixed to it.

Name	Operation	Operands
[name]	DSIWCS	SWB= {(register) | symbolic name}
		,BFR= {(register) | symbolic name}

SWB
is the name of a fullword that contains the address of the service work block (SWB) or is a register containing the address of the SWB.

BFR
is the symbolic name of a fullword that contains the address of a buffer with the message in it or is a register containing the buffer address. **Note:** *This buffer must have a properly initialized BUFHDR.*

DSIWLS Macro Instruction

The DSIWLS macro instruction sends messages to the NCCF log and the operator's hard-copy device.

Name	Operation	Operands
[name]	DSIWLS	SWB= {(register) | symbolic name}
		,BFR= {(register) | symbolic name}

SWB
is a register containing the address of a service work block (SWB) or the symbolic name of a user area that contains the address of an SWB.

BFR
is a register containing the address of a user-provided input buffer or the symbolic name of a fullword area that contains the address of an input buffer. This buffer should contain the record that is to be logged. **Note:** *This buffer must have a properly initialized BUFHDR.*

The DSIWLM service routine's return codes are found in register 15. They are as follows:

0 Successful.

4 No storage is available for logging.

12 The operation was successful but there is no active HCT for this task.

Data Services Macro Instructions

NCCF data services macro instructions allow the recording and retrieval of data from the VSAM data base. NCCF data services are used only as part of a data services command processor. Data services require the data services subtask, as defined in *NCCF Installation.* **Note:** *DSIZCSMS requires a background in SNA request/response units (RUs), as described in* Systems Network Architecture Reference Summary, *GA27-3136. DSIZVSMS requires background in the virtual sequential access method (VSAM), as described in* OS/VS VSAM Programmer's Guide, *GC26-3838, or* Using VSE/VSAM Commands and Macros, *SC24-5144.*

DSIZCSMS Macro Instruction

The DSIZCSMS macro instruction embeds the caller's network services RU (REQMS) in a Forward RU that is passed to the SSCP over the access method's CNM interface. The SSCP then sends the embedded RU to the specified destination.

Name	Operation	Operands
[name]	DSIZCSMS	SWB= {(register) | symbolic name}
		,DSRB= {(register) | symbolic name}
		,INPUT= {(register) | symbolic name}
		,LENGTH= {(register) | symbolic name}
		,RU= {(register) | symbolic name}
		,RULENG= {(register) | symbolic name}
		,DEST= {(register) | symbolic name}
		[,TARGET= {(register) | symbolic name}]
		[,TYPE=CHAIN]

SWB
is a register containing the address of a service work block (SWB) or the symbolic name of a fullword user area that contains the address of an SWB.

DSRB
is a register containing the address of a data services request block (DSRB) or the symbolic name of a fullword user area that contains the address of a DSRB.

INPUT
is a register or the name of a fullword storage location containing the address of a user input buffer. This buffer is used to construct a 28-byte Forward RU to be sent to the specified DEST. This buffer must contain a buffer header followed by text; it also holds the Deliver RU that is returned by the access method.

LENGTH
is a register or the name of a fullword user area that contains the length in binary of the input buffer.

RU
is a register or the name of a fullword storage location containing the address of a user area. That area is an RU that is to be embedded within the Forward RU.

RULENG
is a register or the name of a fullword user area that contains the length in binary of the embedded RU buffer.

DEST
is a register containing the address, or the symbolic name of a fullword user area that contains the address, of the network destination to which the embedded RU is sent. DEST must be 8 characters long, left-justified and padded with blanks if necessary.

TARGET
is a register containing the address, or the symbolic name of a fullword user area that contains the address, of the network component that is the object of the embedded RU. TARGET must be 8 characters long, left-justified and padded with blanks if necessary.

TYPE=CHAIN
indicates that the data services request block (DSRB) has received data and should remain in use to accept further RUs associated with the specific request. If TYPE=CHAIN is specified, the SWB and DSRB operands are required; all other operands are invalid. This operand is invalid with an unsolicited DSRB.

The major return codes for the DSIZCSMS macro instruction in register 15 are:

0	The requested function was performed.
4	The requested function could not be performed.
8	The input buffer was too small.
12	An error was found in parameter specification.
16	The program was not executing under a data services task.

The minor return codes for DSIZCSMS are found in register 0:

0	The function was successful.
4	SWB was invalid.
8	DSRB was invalid.
12	The DSRB that was passed was in use.
16	An unsolicited DSRB was passed.

20 An invalid operator ID was specified in the DSRB.

24 Reserved.

28 There was insufficient NCCF storage to process the request.

32 The CNM interface is inactive.

36 The request was rejected by the access method.

40 A user exit routine rejected the request.

44 Data truncation occurred during the user exit routine processing.

Further information may be found under "Completion of a CNM I/O Request" later in this chapter.

DSIZVSMS Macro Instruction

The DSIZVSMS macro instruction provides access to VSAM services which perform I/O to the specified problem determination file or data set. The operands allow access for data recording, data retrieval, and data deletion.

Name	Operation	Operands
[name]	DSIZVSMS	SWB= {(register) | symbolic name}
		,DSRB= {(register) | symbolic name}
		,FUNC= { GET | PUT | POINT | ENDREQ | ERASE }
		,KEY= {(register) | symbolic name}
		[,KEYLEN= {(register) | symbolic name}]
		[,OPTION= { SEQ | DIR | SKP / ARD | LRD / FWD | BWD / NUP | NSP | UPD / KEQ | KGE / FKS | GEN }]
		[,DATAREA= {(register) | symbolic name}]

SWB
is a register containing the address of a service work block (SWB) or the symbolic name of a fullword that contains the address of an SWB. An SWB contains a save area, work area, and TIB address data. The caller must initialize the SWBTIB field in the SWB with a valid TIB address.

DSRB

is a register containing the address of a data services request block (DSRB) or the symbolic name of a fullword that contains the address of a DSRB. The DSRB contains request information such as RPL, ACB, ECB, and fields used by the DST VSAM service routine for VSAM I/O.

FUNC

describes the VSAM request macro to be issued. See the appropriate VSAM programming manual for a description of the VSAM request macros.

KEY

is a register containing the address or the name of a fullword that contains the VSAM key to be used for access to the requested data.

KEYLEN

is a register or the name of a fullword containing the length in bytes of the key pointed to by KEY. This parameter is required only if OPTION=GEN is specified.

OPTION

specifies the type of access to the file through requests defined by the RPL. Options are arranged in groups; only one option may be specified within each group. This operand has no defaults; therefore, on initial use one option from each group must be specified to set up the RPL. This operand is not valid when FUNC=ERASE or FUNC=ENDREQ is specified.

Note: See the appropriate VSAM programming guide for descriptions and details on how to specify FUNC and OPTION fields.

DATAREA

is a register containing the address, or the name of a user work buffer. The buffer must be large enough to contain the maximum size record in the file or data set and is used by VSAM in the processing of records. This buffer must contain an initialized buffer header (BUFHDR, described later in this chapter under "Control Block Considerations") followed by text.

The return codes for VSAM I/O services are shown below. The minor return codes provide additional information about the condition indicated in the major return codes:

Register 15 = major return code	**Register 0 = minor return code**
0 Successful completion.	0 Successful completion.
	16 A user exit routine rejected the request.
	24 Data truncation occured during substitution of data in a user exit routine.
	28 A user exit returned an invalid return code.

4	Manipulative macro instruction instruction error occurred during processing.	See the explanation of RPL feedback codes in the appropriate VSAM programming guide.
8	An error occurred in the EXEC form of a manipulative macro instruction; a parameter was not in the list.	
12	Unsuccessful completion.	4 The specified DSRB was invalid or in use.
		8 An ACB was unavailable or was not open.
		20 The VSAM I/O request was invalid or there was an I/O scheduling error.
		24 Control block storage could not be obtained.
16	The macro instruction was issued while not executing under a DST.	

For major return codes 4 and 8, the high-order byte of register 15 contains a character that indicates which VSAM manipulative macro instruction failed (for example, M=MODCB). Further information about VSAM I/O requests may be found under "Completion of a VSAM I/O Request" in Chapter 4.

Chapter 4. Command Processors

NCCF allows the user to tailor, modify, or extend the NCCF program. Command processors can be used to process input commands received from operator stations, other command processors, or access method messages. These command processors are invoked by user-defined verbs that are filed during NCCF definition (see *NCCF Installation*). If an ACF/VTAM or ACF/TCAM message number is included in the definition of user-defined verbs, and a command processor is written having that phase or load module name, the command processor is invoked when NCCF receives the message. User-written command processors must be reentrant, written in assembler language, assembled, and link-edited into phases or load modules under the name specified by the MOD operand of the CMDMDL statement.

NCCF provides service facilities that may be used when writing command processors. These facilities and the macro instructions that call them are discussed in Chapter 3. Users who intend to write command processors should also become familiar with the control blocks described in Appendix C before beginning design. NCCF service facilities require an understanding of the service work block (SWB) and the task vector block (TVB), in particular. "Control Block Considerations" later in this chapter also discusses control blocks and fields of interest to the coder of command processors.

Appendix D is an example of a user-written command processor; Appendix E contains two examples of data services command processors; Appendix F is an example of a full-screen command processor.

The following guidelines must be followed in coding command processors:

- Make all command processors reentrant.
- Save registers at entry and restore them before returning control.
- Do not rely upon the contents of registers 0 and 2 through 12 for constant values. Register assignments may vary from task to task or from one program release to another.
- Do not use registers 0, 1, 14, or 15, as they are used by NCCF for macro instruction expansion.
- Register 13 should always point to a standard 72-byte save area.
- Avoid wait states. The DSIWAT macro instruction must not be issued in any immediate command processor.
- No error handling macro instructions (STAE, STXIT) should be used that could override similar specifications being issued by or on behalf of the task.
- Do not return control to any location in the NCCF program other than that specified by register 14.
- If a command processor is designed to handle more than one verb, carefully determine which command is meant.

- Do not use names that begin with the fourth, fifth, and sixth letters of NCCF control blocks. For example, do not use the names SWBAREA and CWBLIST as these names may already be defined in DSISWB and DSICWB, respectively. Do not use names that begin with DSI.

Standard CALL and RETURN sequences transfer control to and from user-written command processors; registers should be restored using standard linkage conventions. Upon entry to the command processor, the registers contain the following information:

Register	Contents
1	The address of the NCCF command processor parameter list (the command work block DSICWB)
13	A save area address
14	The return address of the NCCF program
15	The entry address of the command processor
0, 2-12	Unspecified information

DSICWB contains a user save area, the address of the command buffer, the address of a service work block (SWB) to be used when invoking NCCF service routines, and the address of a parse descriptor block (PRS). The command buffer is described in Chapter 3, under "Buffer Header (BUFHDR)"

When NCCF regains control, it expects to find registers 0 through 14 unchanged and a return code in register 15.

Return codes are documented in Figure 4-4 later in this chapter.

Operating Environment

Regular Command Processors

Regular commands run under the subtask mainline routine (under the PRB in OS/VS). Output to the operator's terminal is sent using the DSIPSS macro with the TYPE=OUTPUT operand. A regular command may execute under an OST, NNT, or PPT.

A regular command processor receives control when one of the mainline event control blocks (ECBs) is posted indicating work to be done. While this command processor is operating, no other event completion is recognized. Regular command processors may be interrupted by system or access method exit routines (not user exit routines). The RESET command causes a regular command processor to stop executing by setting the TVBRESET flag. A regular command processor should periodically examine this flag and, if the flag is on, terminate itself prematurely. It is recommended that this be done within program loops.

Regular command processors are invoked because a command is received from any of the following sources:

- Terminal input
- A command in a command list
- A message from another subtask
- A message from another domain

A regular command processor may also be dispatched by an access method message. The command processor can determine the origin of the command that invokes it by checking the HDRMTYPE field, described below under "Buffer Header." The actual values are shown in the description of the task information block (DSITIB) in Appendix C.

Command processors dispatched by an access method message are the same as other command processors except that the verb is an ACF/VTAM or ACF/TCAM message number. The command processor writer must know the exact wording of the message and where the resource name exists syntactically (which PDB element would contain the resource name).

When an ACF/VTAM message requiring a reply is received, the first PDB syntactic element contains the reply ID, not the verb. The verb is the second element because ACF/VTAM sends the message that way. The first syntactic element, L*nn* or P*nn* (where *nn* is a 2-digit number), is 3 bytes long and is followed by a blank delimiter. The second syntactic element is 7 characters long in OS/VS and starts with the characters IST, followed by three numeric characters, followed by an A. In VSE, the ACF/VTAM message number starts with a 5, followed by a letter from A to K, followed by two numeric characters, followed by an A.

If an ACF/VTAM or ACF/TCAM message does not require a reply, the message number (as above) occurs first and there is no reply ID.

Note: If the ACF/VTAM MSGMOD facility is in effect, the 5-character module identifier is removed and saved in the TIBMMD field of DSITIB.

Immediate Command Processors

An immediate command processor performs its work as soon as the command is entered by the operator, regardless of any other command currently running. For example, the RESET command halts an executing regular command. GO and CANCEL control command lists (all command lists are regular commands). AUTOWRAP, CLEAR key, and no-data-enter control the display screen. An immediate command may execute under an OST or NNT.

While an immediate command processor is running, the subtask cannot be interrupted, as only one interrupt-originated (IRB) exit runs at a time. Regular commands can be preempted by immediate commands. Immediate commands are executed serially.

Immediate command processors are called as subroutines of the NCCF receive exit routine (DSIRCV). While immediate command processors are serialized for one subtask, multiple receive exits (hence multiple immediate commands) can be executing simultaneously (especially in a multiprocessor system). Therefore, immediate command processor modules must be coded as reentrant. The Compare and Swap instruction should be used for referring to or modifying fields shared across subtasks.

If the queue option is used (Q=YES) with the DSIGET and DSIFRE macro instructions in an immediate command processor, EXIT=YES must also be coded.

When presenting data to the display screen, DSIPSS must have the TYPE=IMMED operand specified. Output is limited to one line of 71 characters, on the line immediately preceding the input area. If more than 71 characters of data are required (and not required immediately), DSIMQS may be used to queue the data to the subtask mainline processor. The data is displayed when the current command processor, if any, completes processing and returns control to the mainline processor.

Both Regular and Immediate Command Processors

A "both" command processor can run as an immediate command processor, but can also be included in a command list, called by a message number, and received from another NCCF for execution in this domain. This command processor must check the PDBIMMED flag and process differently depending on whether the command processor is running as an immediate or as a regular command processor. See "Parse Descriptor Block" in Chapter 3.

Command Processors Executed Under the Primary POI Task (PPT)

For the following conditions, a command processor is executed under the PPT:

- Commands entered in response to an access method message received under the PPT.
- AT and EVERY commands that specify PPT as an operand.
- A command or command list, specified with an NCCFIC statement, that executes as soon as NCCF is initialized.
- System operator MSG and REPLY commands.

There are several restrictions when a command processor is executed under the primary POI task (PPT):

- Immediate commands are not allowed.
- Because no terminal is allocated to the PPT, DSIPSS, issued under the PPT, writes a message to the authorized receiver. If there is no authorized message receiver, the message is sent to the system console operator.
- The task information block (TIB) for the PPT is different from the TIB for the OST or NNT. For all task types, the CBHTYPE fields of the TVB and the TIB indicate the subtask (see the DSICBH control block in Appendix C).

Command Processors Executed Under a Data Services Task (DST)

A command processor executed under a DST may cause itself to be reentered after it completes processing. This feature is helpful for requesting retrieval of multiple records from a data base or for invoking a command when a record is retrieved.

A command processor for execution under a DST must be defined to NCCF on a CMDMDL statement as a data services command processor (DSCP) type D or RD (see *NCCF Installation*). DSCPs are passed a data services request block (DSRB; see Appendix C) that contains information about the progress of the data services request. The address of the DSRB is passed to the DSCP in the CWBDSRB field, described below under "Command Work Block." The following restrictions apply to a DSCP:

- Command lists, immediate commands, and regular commands may not be invoked. Only commands defined as D or RD are allowed.

- There is no terminal associated with a DST, so the DSIPSS macro instruction may not be used. If DSIPSS is issued, a code of 20 is returned.

The task information block for the DST differs from the TIB for other subtasks in having the DSITID extension. This extension is not contiguous with the TIB as in other subtasks; its address is in the TIBOSEXT field.

Figure 4-1 shows one way in which data services requests can be structured. This design includes:

- A "front-end" regular command processor Ⓐ that checks command syntax and operands. This command processor builds a command buffer (IFR code 3) and issues the DSIMQS macro instruction to pass the command to the DSCP.

- The DSCP Ⓑ executes the command and interacts with the VSAM data base (DSIZVSMS macro instruction), the CNM interface (DSIZCSMS macro instruction), or both. The DSCP can then accumulate messages and data for the originating subtask. If the programmer wants to have individual messages displayed on the screen of the originating subtask, the DSCP issues DSIMQS to send the data directly to the terminal.

- If the programmer wants to have the DSCP's data formatted with full-line and full-screen presentation services, a presentation services command processor (PSCP) can be coded Ⓒ. The PSCP accumulates data from the DSCP in a buffer. When the PSCP has enough data for a full screen at the originating subtask, it issues DSIPSS to send the data. If in full-line mode, the PSCP must issue DSIPSS OPTION=LAST before returning control to NCCF or an error condition occurs. The DSCP formats its data in a buffer with an IFR code 3 and uses the DSIMQS macro instruction to send the data to the PSCP.

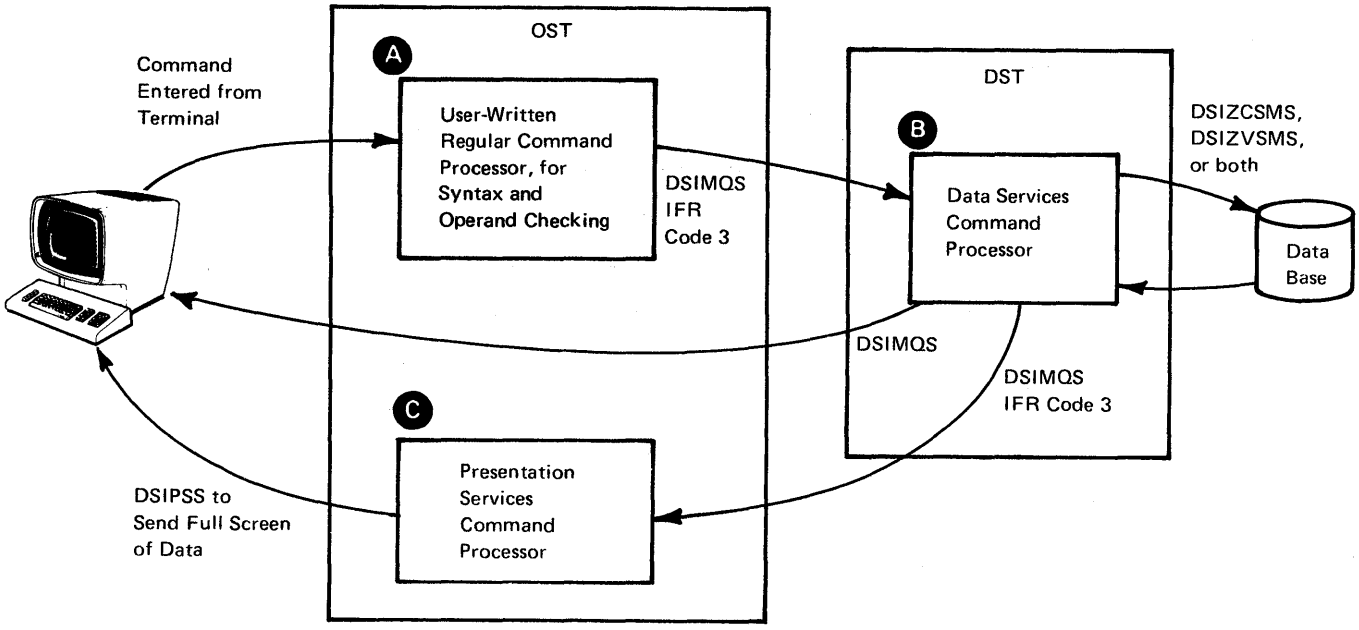


Figure 4-1. Example of Program Design for Data Services Requests

An operator may have any number of pending DST requests. Active DST requests may be listed using the LIST DST command and purged using the PURGE DST command.

Control Block Considerations

The control blocks that are passed to a command processor are shown in Figure 4-2. The control blocks described below are of particular importance to a command processor. You should also be familiar with these control blocks described in Chapter 3: SWB, BUFHDR, IFR, and PDB. In the control block discussions that follow, the sequence of fields may not directly correspond to the field sequence in the actual DSECT. Appendix C contains the control block listings.

Command Work Block (CWB)

The command work block (CWB) contains the command processor parameters, a save area, and a work area. Its fields are described below.

Field	Description
CWBCBH	Is the standard NCCF control block header (DSICBH). It shows the type and length of the CWB, and contains a byte used by DSILCM (locate and control blocks) to indicate whether the control block is currently in use. This byte is only set when a CWB is obtained with the DSILCS macro. It should not be modified by the user.
CWBSAVEA	Is a 72-byte save area that may be used for the command processor.

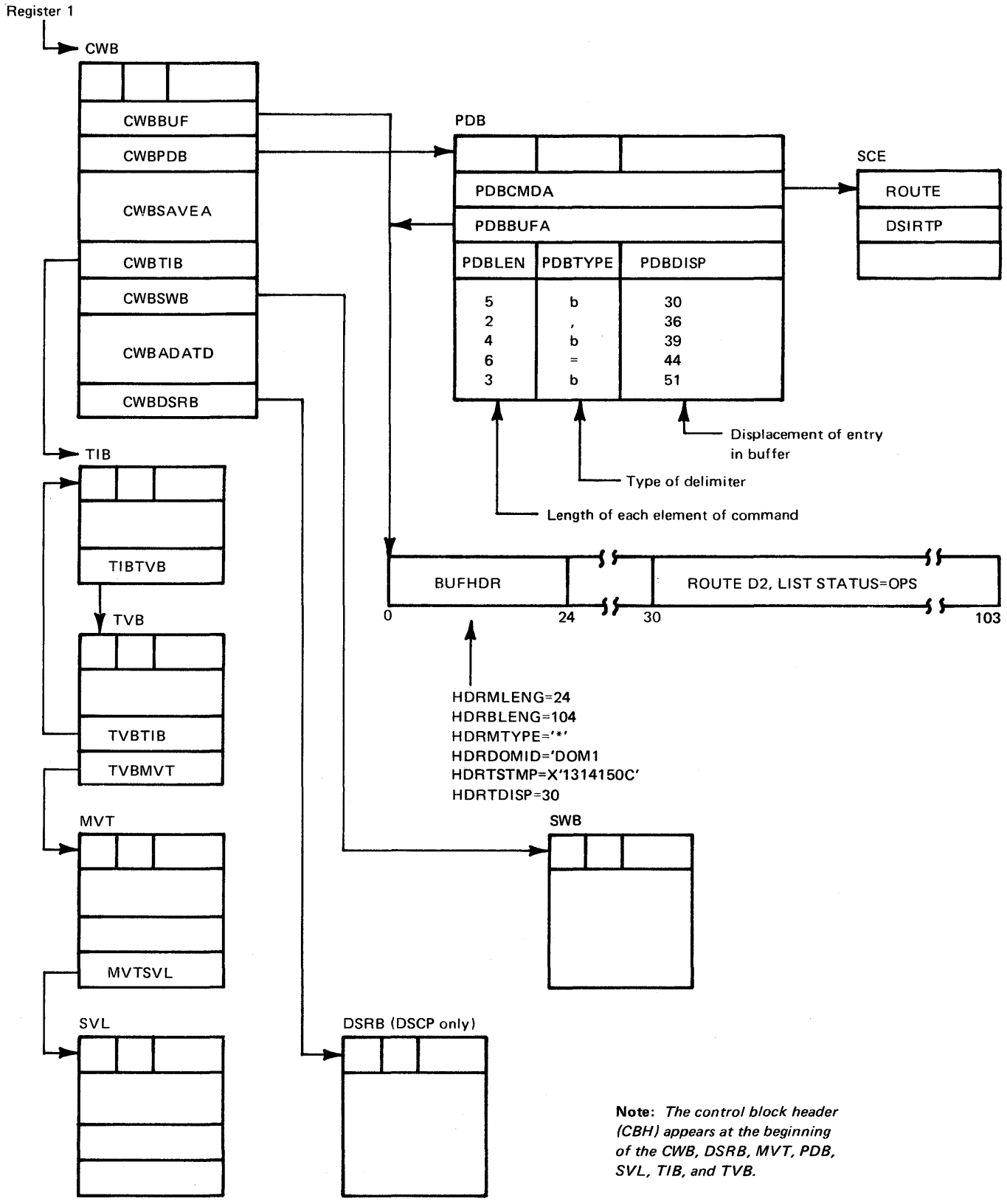


Figure 4-2. Command Processor Input Parameter Control Blocks

Field	Description
CWBPARMS	Is a command processor parameter area. Its subfields are: CWBBUF, CWBPDB, and CWBSWB.
CWBBUF	Points to a buffer containing an NCCF buffer header (BUFHDR) and the command text.
CWBPDB	Contains a pointer to a parse descriptor block (DSIPDB), which is described below. The PDB contains parse information for the command pointed to by CWBBUF. If a special type of parse is required, the PDB may be reused by the command processor.
CWBSWB	Points to a service work block (SWB) that the command processor may use or pass as a parameter to NCCF service macros or modules. NCCF service macros build parameter lists in the SWB for the service modules. The SWB also contains a TIB pointer, a 256-word area, and a 72-byte save area for use by the service routine. SWBs may be reused, without reinitialization (service routines or macros only need the CBH and the TIB address).
CWBNEXT	Is a reserved field.
CWBTIB	Contains the address of the task information block (TIB) for the subtask. The TIB and the task vector block (TVB), which is pointed to by the TIB field TIBTVB, contain all the information relating to the subtask under which the command is running, such as the operator ID, the task type, and the task status. The TVB in turn points to the main vector table (MVT), which contains information of a global nature, such as the domain name, the status of NCCF, (CLOSE NORMAL has been issued, for example), pointers to the other global tables, and to the other subtasks. One pointer contains the address of the service vector list (SVL), which contains all the addresses of the service routines (except the address of presentation services). NCCF service macros expand to refer to both the MVT and the SVL. NCCF requires a USING statement for DSIMVT (the DSECT for the MVT) prior to coding most NCCF macros (see Chapter 3, "Main Vector Table Addressability")
CWBADATD	Is a 256-byte work area for the command processor. If more storage is required, the command processor must obtain it with the DSIGET macro and release it with the DSIFRE macro. Command processors must free any storage obtained.
CWBDSRB	Is used only by data services command processors (DSCPs). The data services task (DST) initializes this field with the address of the data services request block (DSRB). This field should contain zero for all other command processor types.

System Command Entry (SCE)

The PDBCMDA field in the parse descriptor block contains the address of the system command entry (SCE), which contains information about the command. The DSICES macro instruction uses the SCE to find the command processor for a verb or to find the verb itself (if the search is by module name). The address of the SCE is also used as input to the DSIPAS macro instruction, described in Chapter 2.

Field	Description
SCEVERB	Contains the command verb; it is 1-to-8 bytes, left-justified and padded with blanks.
SCELNAME	Contains the load module or phase name of the command processor to be called for the verb.
SCECADDR	Contains the address of the command processor's entry point.

Data Services Request Block (DSRB)

The data services request block (DSRB) is the method of communication between the NCCF data services task (DST) and a data services command processor (DSCP). It contains information for the DSCP and work space for the I/O routines. The fields described below are those of interest to a DSCP programmer.

Field	Description
DSRBCBH	Is a standard NCCF control block header.
DSRBNXTV	Contains the address of the next DSRB in the chain; this field is reserved for DST use.
DSRBVECB	Is reserved for DST use as an event control block (ECB) when requesting VSAM I/O.
DSRBVRPL	Contains the address of the RPL that the DST uses for VSAM I/O. This field is reserved for DST use.
DSRBCUSB	Contains the address of an NCCF buffer used by the CNM interface for unsolicited data. This field is only used when the DSRB function code (DSRBFNCD) indicates that unsolicited data has been received. The buffer contains a BUFHDR and the data length is in the HDRMLENG field of BUFHDR.
DSRBFLG	Contains the flag settings described below. The bits may be examined but not changed.
DSRBTYPE	1 indicates that the DSRB is for unsolicited CNM data; 0 indicates that the DSRB is for VSAM or CNM solicited data traffic.

Field	Description
DSRBACTV	1 indicates that there is an active transaction using this DSRB. A transaction is defined as a request from the time of its first arrival at the DSCP to the last exit of the DSCP. When a transaction ends, the DSRB is available for reassignment to another transaction, for the same or another user.
DSRBINUS	1 indicates that either the VSAM or CNM interface service routine has an active request using this DSRB. DSRBINUS should not be on when DSRBACTV is off.
DSRBRSMV	Is a reserved field.
DSRBRADD	Is reserved for DST use.
DSRBOID	Is an 8-byte field containing the ID of the operator that initiated the transaction.
DSRBTIB	Contains the address of the DST task information block (TIB).
DSRBUSER	Is a field available for user purposes. If this field is used for a storage address, the DST does not free the storage. However, storage may be allocated using the DSIGET Q=YES option and the storage may be freed as with any subtask using Q=YES. If not freed, the storage remains allocated until the subtask terminates.
DSRBVACB	Contains the address of the VSAM ACB used for disk I/O. This field is reserved for DST use.
DSRBVDAD	Is the field in which the VSAM service routines keep the data buffer address while a request is being processed. The buffer must have a BUFHDR that is filled in. For a VSAM GET or a CNM input request, HDRMLENG contains the length of the data actually retrieved. Data is truncated, in the event of overflow.
DSRBUBUF	Contains the address of the original command that was sent to the DST. This field is unchanged during the data services transaction. This buffer contains a BUFHDR and the HDRMCEXT extension. It also has an X'0003' IFRCODE and HDRTYPEI (see "Internal Function Request in Chapter 3).
DSRBPRID	Is a halfword field that contains a correlation identifier for use by the CNM interface.
DSRBINPT	Is the address of the CNM interface input buffer.

Field	Description
DSRBRCMA and DSRBRCMI	Are fields that contain the return codes for a completed request. They are set after the request is completed but before the DSCP is reinvoked for request completion. The major return code (DSRBRCMA) value is further explained by the minor return code (DSRBRCMI). The symbolic values of the return codes are in the DSIDSRB DSECT. The values are discussed later in this chapter under "Output."
DSRBFNCD	Is a 1-byte field that contains a function code indicating the reason that the command processor was called.
DSRBFNRM	Indicates that this is the first invocation of the command processor, as the result of a message received from another subtask.
DSRBFUNS	Indicates that the command processor was called to handle unsolicited CNM data.
DSRBFSOL	Indicates that the data was requested from the CNM interface.
DSRBFVSM	Indicates that a VSAM I/O request has completed.

Invoking a Command Processor

The following are required to invoke a command processor:

- A CWB
- An SWB
- A command buffer
- A PDB
- A save area
- Registers 1, 13, 14, and 15

Obtaining a Command Work Block (CWB)

A command processor requires a command work block (CWB) for use as a parameter list, a save area, and a work area.

A CWB may be preallocated (and reused) or may be obtained by issuing the DSILCS macro. Before calling the command processor, the TIB address must be stored in the CWBTIB field.

Obtaining a Service Work Block (SWB)

The invoker of a command processor must provide an SWB. The SWB may be preallocated, obtained with the DSILCS macro, or may be one the invoker was passed. This control block must also have its SWBTIB field pointing to the TIB. The SWB address must be stored in CWBSWB.

Building a Command Buffer

Each command is invoked with a command buffer containing a verb and optional operands. The verb is prefixed with the buffer header (BUFHDR). Each BUFHDR field must be initialized except the message command extension HDRMCEXT. The address of this command buffer is stored in CWBBUF. For details on the buffer header, see Chapter 3.

Obtaining a Parse Descriptor Block (PDB) and Parsing the Command

The invoker must obtain storage for a PDB to parse the command for the command processor to be called. The size of the storage for the PDB may be obtained by issuing the DSIPRS macro with the PDBSIZE option. The usual sizeis 160 bytes. After the storage is obtained (from preallocated storage or with DSIGET), the address is stored in the CWBPDB field. The control block header (CBH) is built and the first byte is set to the value defined by symbol CBHPDB. The second byte is zeroed and the PDB length is stored in the third and fourth bytes prior to invoking the DSIPRS macro. Issuing DSIPRS fills in the PDB including the PDBBUFA pointer to the command buffer, the parse elements, and the number of parse elements. For details on the parse descriptor block, see Chapter 3.

Looking Up the Command Processor Address

After the command is parsed, the command must be found in the NCCF system command table (DSISCT). The command may be looked up in one of three ways:

With the parsed command in the PDB

Without prior parsing

By command processor module name (the module name is known but the verb name may change, as in a synonym)

The DSICES macro invokes the command search routine (DSICAI) to locate the appropriate position in the SCT. The position is returned in an area passed on the DSICES macro as the SCTADDR parameter. This address points to an SCT entry (SCE). The SCE address must be stored in the PDBCMDA field. The area returned is mapped by DSISCE.

When the DSICES macro returns to the caller, the return code indicates whether the command is immediate, regular, or both. The caller must set the PDBIMMED flag according to the DSICES return codes and the invoker's environment (if running under control of the receive exit routine as an immediate command processor and the return code was "immediate" or "both").

Calling the Command Processor

Register 1 must point to the CWB (which now in turn points to the PDB, SWB, TIB, and the command buffer). Register 13 must point to a save area (where it is probably already set, because a save area is required for the service macros). Register 15 must contain the command processor's entry point address (found in DSISCE) and register 14 must have the return point address. The command processor entry point address is stored in the SCECADDR field of the SCE entry pointed to by the PDBCMDA field.

Initial DSCP Invocation

When a DST initially invokes a data services command processor (DSCP), the DST passes the address of a DSRB in the CWBDSRB field, as described earlier in this chapter. The DSRBFNCD field is set to 1 (DSRBFNRM) and the fields described under "Data Services Request Block" are also set. If the DSCP issues a data services request (DSIZVSMS or DSIZCSMS macro instruction) that is accepted, the same DSCP is reentered after the request has been completed. Figure 4-3 shows an example of the processing logic for a DSCP.

Passing a Command to Another Subtask in the Same Domain

NCCF data services often require passing a command to another subtask (for example, from an OST or NNT to the DST). For passing commands from one subtask to another in the same domain, an internal function request (IFR; described in Chapter 3) is placed in front of the data. The HDRMTYPE field is set to HDRTYPEI and the IFRCODE is set to X'0003' in the text area of the buffer (IFRCODE=IRFCODCR). HDRTDISP must be set to the displacement of IFRCODE. A command and its operands follow IFRCODE. The HDRMLENG field is set to the length of the command and its operands plus 2 bytes for the IFR. The DSIMQS macro instruction is issued to transfer the IFR to the destination subtask, as described under the DSIMQS macro instruction in this chapter.

The receiving subtask removes the IFR by setting HDRTDISP to the command verb and subtracting the length of the IFR (2) from HDRMLENG. HDRMTYPE remains HDRTYPEI. The address of the BUFHDR plus the HDRTDISP value equals the location of the command text.

The HDRSENDR field of BUFHDR contains the operator ID of the sender of the buffer. To return error messages or data to the originating subtask, a message may be built and sent with DSIMQS using HDRSENDR as the destination.

For any buffer that is not an IFR (HDRMTYPE=HDRYPTEI), the destination subtask message receiver issues DSIPSS TYPE=OUTPUT to display the message on the operator's screen. Another method of returning data to an originating subtask is to build an IFR and issue DSIMQS to schedule a presentation services command processor (PSCP) in the originating subtask. The PSCP receives control in the same manner as the original command processor.

Forwarding a Command to Another Domain for Execution

A command can be forwarded to another domain for execution in two ways:

Build a buffer, PDB, and CWB and call the ROUTE command, or

Build a buffer as explained earlier and issue the DSIPSS macro with TYPE=XSEND to transmit the command to the NNT (NCCF-to-NCCF) task in the other domain. The command is executed in the NNT as if it were entered from a terminal in that domain.

Data can be returned to an originating domain by issuing DSIPSS TYPE=OUTPUT for any HDRMTYPE except HDRTYPEX.

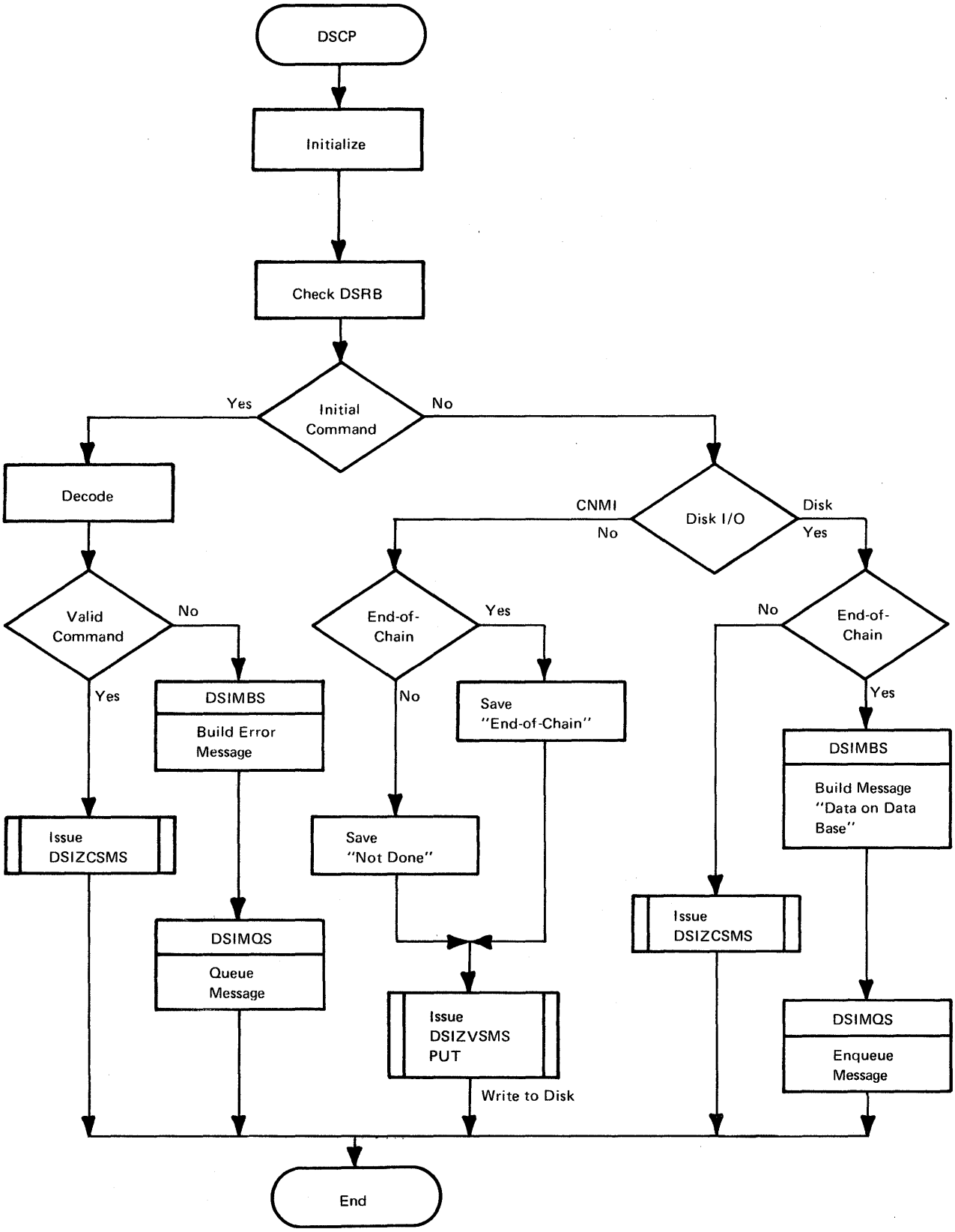


Figure 4-3. Example of DSCP Processing Logic

Returning a Command to Another Domain for Output

For standard NCCF mode, an NNT issues DSIPSS TYPE=OUTPUT (for any HDRMTYPE except HDRTYPEX) to return a message for output to an OST in another domain. If data formatting (for example, a full-screen display) is required, the NNT builds a buffer with HDRMTYPE=HDRTYPEX and a command in the buffer text. The command verb identifies a user-defined presentation services command processor (PSCP). The verb must be 8 or fewer characters long, delimited from the rest of the buffer by a blank, and defined in the receiving domain. When the OST's cross-domain message receiver receives the command buffer, the OST calls the PSCP for the command. The PSCP is invoked as described under "Invoking a Command Processor" earlier in this chapter.

Passing Commands to the Access Method

To pass a command buffer to ACF/VTAM while executing under an OST, NNT, or PPT, the user builds a command buffer as described under "Buffer Header" in Chapter 3. The user calls the IBM-supplied command processor, which is either DSIVTP if it is an ACF/VTAM VARY, MODIFY, or DISPLAY command, or DSIREP for a REPLY command.

A similar procedure applies to ACF/TCAM operator commands. The user builds a buffer header and then calls DSITOCP, which passes the command to ACF/TCAM for execution.

Output

As shown in Figure 4-4, command processor return codes depend on the operating environment.

Regular Commands

Regular command return codes have meaning in only two situations:

When returning from a command processor invoked due to terminal input

When one command processor returns to another command processor (a command processor used as a subroutine)

The return codes are defined in the following categories:

RC = 0: command processor completed successfully

RC > 0: command processor did not complete successfully

RC < 0: special conditions defined by NCCF

For terminal-originated regular commands, see events 4, 5, and 6 in Figure 4-4 for the meanings of the return code categories. For a command processor used as a subroutine, the specific return codes may be defined as required within the categories.

Event	Ready Indicator A	Immediate Message Area B	Unlock Keyboard C	Command Input Area D
1- Immediate command completion with return code = 0	N/A	Erase	If a regular command is not running, unlock the keyboard, else leave locked	Erase
2- Immediate command completion with return code > 0	N/A	Remains as is	Same as 1C	Remains as is
3- Immediate command completion with return code < 0	N/A	Remains as is	Same as 1C	Erase
4- Regular command completion with return code = 0	Set ? ? ? on	N/A	Unlock keyboard	Erase
5- Regular command completion with return code > 0	Set ? ? ? on	N/A	Unlock keyboard	Remains as is
6- Regular command completion with return code < 0	Set ? ? ? on	N/A	Unlock keyboard	Depends on Value: -1 Remains as is -2 Remains as is -3 Remains as is -4 Last regular command is written to command input area

Figure 4-4. Effect of Command Processor Return Codes for Terminal-Originated Commands

Immediate Commands

Immediate commands have three types of return codes. As with regular commands, specific return codes are not defined. (All immediate commands originate with terminal input.)

RC < 0: command is successful - the input area is erased but *not* the immediate message area.

RC = 0: command is successful - both the input area *and* the immediate message area are erased.

RC > 0: command is unsuccessful - neither the input area nor the immediate message area is erased.

The immediate return codes are shown in Events 1, 2, and 3 of Figure 4-4.

Completion of a VSAM I/O Request

After a DSIZVSMS macro instruction completes processing, NCCF initializes the DSRB and sets the following fields:

- DSRBVRPL contains the address of the VSAM RPL that was used for the I/O.
- DSRBVACB contains the address of the VSAM ACB for the DST.

- DSRBVDAD contains the address of the VSAM I/O buffer, with a standard BUFHDR. For GET requests, the BUFHDR HDRMLENG field indicates the length of the data read. HDRTDISP contains the offset to the data.
- DSRBVKEY contains the address of the key in the DSRBVDAD buffer.
- DSRBVKLN contains the key length.
- DSRBVRTP indicates the type of request just completed:

  1. DSRBVGET (VSAM GET)
  2. DSRBVPUT (VSAM PUT)
  3. DSRBVPNT (VSAM POINT)
  4. DSRBVERS (VSAM ERASE)
  5. DSRBVNRQ (VSAM ENDREQ)

The return codes on reentry to the DSCP are as follows:

DSRBRCMA	DSRBRCMI	Explanation
00	00	Successful completion.
00	16	User exit processing of VSAM input has rejected the input. HDRMLENG has been set to zero.
00	24	Data truncated because user exit returned data longer than NCCF buffer on RC = USERSW (see the description of DSIUSE in Appendix C). HDRMLENG set to truncated length.
00	28	Invalid return code from user exit.
08	VSAM RPL feedback	VSAM logical error, indicated in DSRBRCMI. See VSAM manuals.
12	VSAM RPL feedback	VSAM physical error, indicated in DSRBRCMI. See VSAM manuals.

Completion of a CNM I/O Request

When a DSIZCSMS macro instruction completes processing, the DSRB indicates the completion. A Deliver RU has been received at the CNM interface. DSRBINPT contains the address of the buffer that contains the Deliver RU or negative response. The return codes on reentry to the DSCP are as follows:

DSRBRCMA	DSRBRCMI	Explanation
00	00	Successful completion.
00	04	Negative response was received. DSRBINPT contains the address of the negative response.
00	08	There was not enough NCCF storage to process the request.

DSRBRCMA	DSRBRCMI	Explanation
00	16	The user exit rejected the Deliver RU. The DSCP sets HDRMLENG to zero.
00	20	The data has been truncated. The length of the Deliver RU was greater than the buffer. HDRMLENG is set to the truncated length.
00	24	The data was truncated after the user exit returned with a return code of USERSWAP (see DSIUSE in Appendix C). The DSCP sets HDRMLENG to the truncated length.
00	28	The access method rejected the request.

Completion of Receipt of Unsolicited CNM Data

The command processor that is defined as the unsolicited input DSCP receives control when the network presents an unsolicited Deliver RU. (This DSCP is defined with the UNSOL operand of the DSTINIT definition statement as described in *NCCF Installation*.) When this command processor receives control:

- DSRBUBUF contains zero because there is no command.
- DSRBCUSB contains the address of the buffer containing the unsolicited Deliver RU. The RU starts at the offset specified in HDRTDISP and the RU length is in HDRMLENG.

The return codes on reentry to the DSCP are as follows:

DSRBRCMA	DSRBRCMI	Explanation
00	00	Successful completion.
00	16	The user exit rejected the Deliver RU. HDRMLENG has been set to zero.
00	20	Data truncation has occurred. The length of the Deliver RU was greater than the buffer. HDRMLENG is set to the truncated length.
00	24	Data truncation occurred after the user exit returned with a return code of USERSWAP. HDRMLENG is set to the truncated length.

Full-Line Command Processor Considerations

Full-line presentation services can be used to send a full screen of 80-byte messages to the operator from a subtask other than an OST, such as from an NCCF-to-NCCF task (NNT). This full-screen facility, known as title-line output, allows you to send a number of messages, one right after another, and have them presented on the screen in a tabular format, with optional title lines.

Figure 4-5 shows an example of title-line output. The message lines are displayed below fixed title lines. The data lines will wrap around until all the data is displayed, but the title lines will stay at the top of the screen.

To use title-line output, format and send the message buffer as follows:

1. Set the HDRMTYPE field to HDRTYPEL ('=').

2. If you want to use title lines, mark the title line or lines by setting the bits in HDRIND to title label (HDRLNLBL). There can be from 1 to 5 title lines (1 to 4 lines if the first title line have data in columns 70 to 80).

   Note: If the first title line has data in columns 70 to 80, a maximum of 4 title lines may be used).

   If you do not wish to have title lines, omit this step.

3. Mark the data lines by setting the bits in HDRIND to data line (HDRLNDAT).

4. Mark the last line of data by setting the bits in HDRIND to data/end (HDRLNEND).

5. From an NNT, issue DSIPSS TYPE=OUTPUT using OPTIONS=MSG for each line sent. From a subtask other than NNT, issue the DSIMQS macro to the OST that is to receive the output.

NCCF Title-Line Processing

At the operator station task, NCCF groups all the full-line buffers until a buffer marked as data end (HDRLNEND) is received. The title lines or, if no title lines are present, the first message line, is sent to the top of the screen, directly under the NCCF title line. Each data line is then shown one at a time. When the bottom of the screen is reached, the screen is locked, unless AUTOWRAP FULL has been specified. When the screen is cleared, the title lines (if present) are repositioned at the top of the screen, followed by the next data lines. This process continues until all the messages have been shown.

NETWORK COMMUNICATIONS CONTROL FACILITY mm/dd/yy hh:mm:ss

NCP	LINE	PU/CLUSTER	LU/TERMINAL	TYPE	(NCCF1) LOCATION
NCPA				3705	MACH. ROOM
NCPA	A01			SDLC	SATTELLITE
NCPA	A01	A01A		3274	ANCHORAGE
NCPA	A01	A01A	A01A01	3278	ANCHORAGE
NCPA	A01	A01A	A01A02	3278	ANCHORAGE
NCPA	A01	A01A	A01A03	3278	ANCHORAGE
NCPA	A01	A01B		3274	NOME
NCPA	A01	A01B	A01B01	3278	NOME
NCPA	A01	A01B	A01B02	3278	NOME
NCPA	A01	A01B	A01B03	3278	NOME
•	•	•	•	•	•
•	•	•	•	•	•
•	•	•	•	•	•

Figure 4-5. Example of Full-Line Title-Line Output

Coding Requirements

If possible, NCCF uses screen columns 70 to 80 of the first data line for the domain identifier. If the first line contains nonblank characters in these columns, NCCF will generate a blank line and add the domain identifier to this line. In title-line output, this extra line is treated as a title line.

At least one character must be in each buffer record sent to the screen. If you wish to print a blank line, place a blank character (X'40') in the buffer. NCCF supports full-line messages up to 80 characters. Any data line longer than 80 characters is truncated.

Full-Screen Command Processor Considerations

A full-screen command processor displays a full screen of data to the NCCF operator. This screen of data is sent using the DSIPSS macro instruction under an operator station task (OST).

Types of Full-Screen Command Processors

There are two types of full-screen command processors: asynchronous and synchronous.

An asynchronous full-screen command processor allows several full screens to be displayed in sequence before returning to standard NCCF mode. Once a full-screen mode has been started, further terminal input is treated as input to the asynchronous full-screen processor until this command processor is ended. An asynchronous command processor can control when NCCF messages and commands are able to interrupt full-screen processing.

A synchronous full-screen command processor shows a full-screen panel and waits for operator input. After the operator responds to the full-screen panel, further input is considered as NCCF input and standard NCCF mode is resumed. To maintain full-screen mode, another full-screen panel can be sent to the operator.

Operations of a Full-Screen Command Processor

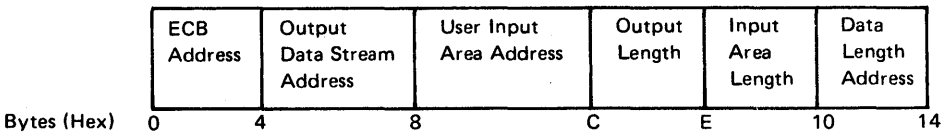
A full-screen command processor is executed as a regular command processor under the operator station task (OST) (see "Operating Environment"). On initial entry, a full-screen processor issues the DSIGET macro instruction, which obtains storage to keep track of the full-screen process. (This allows the full-screen processor to resume control after an interruption.) The full-screen command processor then uses a series of DSIPSS macro instructions to prompt the operator to enter full-screen data. The input and output data streams are 3270 data streams that NCCF does not modify.

Asynchronous Full-Screen Command Processors

For asynchronous full-screen command processors, a full screen of data is sent to the NCCF display terminal using the DSIPSS macro instruction with TYPE=ASYPANEL. The data from an asynchronous full-screen panel is read directly by NCCF into the user's buffer area, and an event control block (ECB) is posted when the data has been read. After the ECB is posted, the command can process the input and issue more full-screen panels. While the asynchronous full-screen command processor is running, input to the NCCF terminal is treated as input to the command processor and not as a direct input to NCCF.

Asynchronous Full-Screen Command Processor Parameter List

When the asynchronous command processor is invoked, it reads and writes to the terminal using DSIPSS TYPE=ASYPANEL. A 20-byte parameter list pointed to by the PANEL operand of DSIPSS. The format of this list is shown below:



If asynchronous full-screen output is requested, the output data stream address field contains the address of a 3270 data stream including a command, WCC, and data orders to be written to the terminal. The output length field indicates the length, in bytes, of the 3270 data stream (32,767 bytes maximum). The ECB address, input area length, user input area address, and data length address are not used if only output is requested.

To read asynchronous full-screen input from the terminal, the ECB address area contains the address of an event control block to be posted when the asynchronous input is received. The user input area address contains the address of a user area into which a full-screen panel data is read. (If the length of the data being read is greater than the user input area, the data is truncated.) The input area length field indicates the length of the input data area in bytes (32,767 bytes maximum). The data length address field contains the address of a halfword field that is set to the amount of data actually read when the ECB is posted.

Processing Asynchronous Full-Screen Input

The DSIPSS macro with TYPE=PSSWAIT allows the full-screen command processor to wait on both its own list of events and on a list of NCCF events that should be allowed to interrupt the command processor (such as important messages). The command processor is able to test the return code after the wait to determine if its own ECB or one of NCCF's ECBs has been posted. If the return code shows that an NCCF event had completed, the command may return to NCCF to allow the processing of the event to occur. If the panel ECB is posted, the command processor can process the input in the buffer. In this manner, the command processor has complete control of the screen format and can return to NCCF after saving the screen status so processing can resume later.

Testing if NCCF Events have Occurred

The DSIPSS macro with the TYPE=TESTWAIT allows the command processor to test if an NCCF event has already been posted. This option is useful to do before issuing DSIPSS TYPE=ASYPANEL to avoid doing input or output when a NCCF command is already posted. This option allows early detection of interruptions and allows return to NCCF with a minimum of screen interruptions.

Noninterruptible Command Processors

If a noninterruptible asynchronous full-screen command processor is desired, the command processor can wait on its own list of ECBs and not use TYPE=PSSWAIT. This allows the command processor to ignore any NCCF interruptions. In this case, it is strongly recommended that the command processor include the OST termination ECB, which is located in the TVBTECB field of the DSITVB (see Appendix C). This field allows the command processor to be notified of any major condition in which the command processor should clean up and exit.

Ending an Asynchronous Full-Screen Command Processor

When the full-screen command processor has completed, the DSIPSS macro with TYPE=OUTPUT or TYPE=IMMED should be issued to restore standard NCCF mode. Input to the terminal is now treated as input to NCCF rather than as input to the command processor.

Canceling an Asynchronous Full-Screen Command Processor

The DSIPSS macro with TYPE=CANCEL allows the coder to change characteristics of the asynchronous full-screen command processor, such as the input area length or the ECB address, without returning control to standard NCCF mode. TYPE=CANCEL can be issued whether or not a DSIPSS TYPE=ASYPANEL is active or the input from TYPE=ASYPANEL has been posted as complete. This is sometimes necessary since there is no way to guarantee that the operator will ever enter data to any given panel.

For More Information

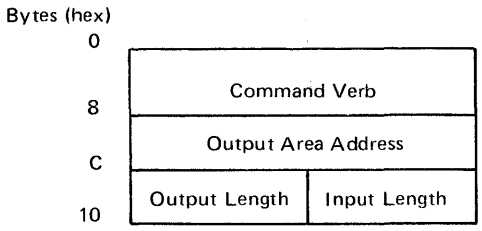
See the section titled "General Guidelines" later in this chapter.

Synchronous Full-Screen Command Processors

For synchronous full-screen command processors, a full screen of data is sent to the NCCF display terminal using the DSIPSS macro instruction with TYPE=PANEL. The full screen of data is displayed. After the operator responds to this full-screen panel, either another full-screen panel is sent or further operator input is considered as NCCF input and standard NCCF mode is required.

Synchronous Full-Screen Command Processor Parameter List

When the synchronous command processor is invoked, it reads and writes to the screen using DSIPSS TYPE=PANEL. A 16-byte parameter list pointed to by the PANEL operand of DSIPSS. The format of this parameter list is shown below:



The command verb is a verb that is defined as a regular, both, or RD command processor in the NCCF CMDMDL definition statement. The verb is 1 to 8 characters, padded to the right with blanks. When the input length is zero, the command verb is not required; bytes 0-7 of the parameter list are ignored and considered reserved.

The output area address is the address of a 3270 data stream conforming to ACF/VTAM requirements for 3270 record-mode data, and including a command code, WCC, orders and data. (See Figure 4-6.)

The output length is the number of bytes in the 3270 data stream. For read-only requests, the output length is set to zero. The maximum output length is 32,767 bytes (X'7FFF').

The input length is the maximum input data to be expected by NCCF. Data exceeding this length is truncated. NCCF acquires and later frees the required amount of virtual storage. If storage is not available, the DSIPSS PANEL request fails with a return code of 12 (PSMNOSTG). To write to a terminal without reading input and without scheduling a command to process input, set the input length to 0. The maximum input length is 32,767 bytes (X'7FFF').

```
DC    X'F5'             ERASE/WRITE COMMAND CODE
DC    X'C3'             WCC=RESET KEYBOARD AND MDTS
DC    X'I14040'         SBA ROW 1 COLUMN 1
DC    X'1DF0'           START FIELD--PROTECTED, LOW
DC    C'ENTER DATA'     MESSAGE TEXT
DC    X'1D40'           START FIELD--UNPROTECTED, LOW
DC    X'13'             INSERT CURSOR IN UNPROTECTED AREA
```

Figure 4-6. Sample 3270 Data Stream

Processing Synchronous Input

The input data for a full-screen processor is different from the input data for standard NCCF commands. The format of the full-screen input buffer is shown below:

BUFHDR	Command Verb	b	3270 Data

The full-screen command receives the same input as any other command: register 1 points to a CWB, which in turn points to the TIB, SWB, the command buffer, and a PDB containing parse data relevant to the buffer. Since the 3270 data in the command is not translated or edited in any way, the parse data may not be meaningful. The 3270 data stream contains the AID byte, buffer addresses, SBA characters, and data.

Establishing a Full-Screen Subroutine

Since a synchronous full-screen command processing must return to standard NCCF mode to process any commands, it is recommended that a subroutine be used to handle full-screen input and output. The subroutine can then suspend the full-screen command processor without informing the main screen processor. To do this, save the registers at the time of entry into the subroutine. Next, restore the registers that the command processor was originally called with and return to the OST. When the command is redriven because of the data entered at the terminal, reestablish the environment that was saved when the input/output subroutine was entered and return to the requester of the input. You may also want to have the reshow command processed by this full-screen subroutine. Appendix F shows an example of a full-screen command processor subroutine.

Ending a Synchronous Full-Screen Command Processor

When the synchronous full-screen command processor has completed and is about to exit for the last time, it is recommended that the DSIPSS macro with TYPE=OUTPUT or TYPE=IMMED be issued to automatically restore the standard NCCF panel. If this is not done, the operator must press the CLEAR key to return to the standard NCCF panel.

General Guidelines

The following guidelines should be followed for both asynchronous and synchronous full-screen command processors.

Screen Formatting for the 3270 Data Stream

Since the full-screen command processor is responsible for the 3270 data stream, the processor should do one of the following:

- Issue the DSIPSS macro with TYPE=SCRSIZE to find the presentation space dimensions. If the input data is larger than 24 by 80 bytes, issue the 3270 Erase/Write Alternate command.
- Issue the 3720 Erase/Write command and use the default 24 by 80 byte screen size.

Note: When writing data to a full-screen processor, avoid sequences of a read followed by another read. This combination leaves the 3270 Input Inhibited indicator set, and the operator has to press the RESET key before entering data. It is better to follow the first read with a write/read where the output data is a WCC that unlocks the keyboard and, optionally, resets the Modified Data tags.

The Escape Key

In synchronous full-screen command processors, the operator may temporarily suspend full-screen processing and escape to standard NCCF by pressing the CLEAR key and requesting options 1 or 3 of the DSI817A options menu (see *NCCF Terminal Use*). Synchronous full-screen command processors should define a key to allow the operator to perform this escape function.

The Reshow Option

After suspending full-screen processing using the escape key, the operator may wish to resume to full-screen processing again. It is recommended that every full-screen processor define a command option to reconstruct the last full-screen panel and continue processing from where the full-screen process was interrupted.

The Reshow Key

A full-screen command processor should define a key, such as the PA2 key, to redisplay the last screen shown during full-screen processing. The reshow key might be necessary if there are two full-screen processors running alternately. In this case, it is possible for data from both of the processors to be written to the same screen panel. A reshow key helps the operator avoid this problem. The reshow key is useful if the operator has errors in the input data and wishes to erase the data and start over, or if the operator accidentally hits the ERASE INPUT key and erases good data. The full-screen command processor refreshes the screen in response to the reshow key.

Logging Full-Screen Input/Output

NCCF does not automatically log full-screen input and output. However, it is recommended that a full-screen application program use the DSIWLS macro instruction to log pertinent data.

DSIPSS Return Code from a Full-Screen Command Processor

The possible return codes to a full-screen command processor from DSIPSS are described under the DSIPSS macro instruction. A nonzero return code shows that no input command is scheduled. Any cleanup required should be done before returning to NCCF. For synchronous full-screen command processors, the TVBRESET and TVBPNMOD fields in DSITVB can help interpret several of the DSIPSS return codes (see Figure 4-7). The TVBRESET bit is set on by the RESET NORMAL command or by option 3 or the DSI817A options menu. The TVBPNMOD bit is set on if the full-screen command processor is interrupted by either the CLEAR key or an event that causes NCCF to reformat the screen in standard NCCF mode. Whenever a command processor builds a data stream for panel mode, the TVBPNMOD bit should be tested.

Note: If you write a full-screen command processor that gets control in the OST from a cross-domain message, set the TVBPNMOD bit on before exiting. This ensures that TVBPNMOD is set for full-screen commands issued from a terminal.

Return Code	TVBRESET	TVBPNMOD	Meaning
0	OFF	OFF	Input data was scheduled. Processing continues.
0	OFF	ON	Input data was scheduled. The screen was modified after processing completed.
0	ON	OFF	Input data was scheduled; a reset was subsequently requested. The input command will be executed. Delay the reset until the input command is running. Since TVBRESET may be off by that time, it is recommended that another bit be set, indicating the reset. The screen has not yet been modified.
0	ON	ON	Input data was scheduled; a reset was subsequently requested. The input command will be executed. Delay the reset until the input command is running. Since TVBRESET may be off by that time, it is recommended that another bit is set, indicating the reset. The screen has been modified.
32	OFF	ON	Operator requested escape to standard NCCF mode (option 1 of DSI817A). The command processor should exit to NCCF and expect the operator to request a redisplay of the panel later. Input has not been scheduled.
32	ON	ON	Operator requested escape to standard NCCF mode (option 1 of DSI817A). The command has been reset. The full-screen command processor should exit to NCCF and expect the operator to request a redisplay of the panel later. Input has not been scheduled.
44	ON	ON	No input data was scheduled, because the operator requested a reset (option 3 of DSI817A).

Figure 4-7. Interpreting the TVBRESET and TVBPNMOD Bits

Chapter 5. Exit Routines

This chapter describes the exit routines available in NCCF. It includes a summary of the exit routines and instructions for coding and installing them. A sample user-written exit routine is shown at the end of the chapter.

NCCF provides service facilities that may be used by user-written exit routines. These facilities and the macro instructions that call them are discussed in detail in Chapter 3.

What Can NCCF Exit Routines Do?

NCCF exit routines allow the user to edit data flowing to or from NCCF. The exit routine can be used to:

- Change commands or messages
- Delete unnecessary messages
- Summarize NCCF events or data
- Handle a specific event in a way different from NCCF processing
- Count or summarize specified information automatically
- Automate processes based on information from the access method

Each exit routine is designed to handle a particular type of event. When such an event occurs, NCCF passes control to the appropriate exit routine for processing, and then control is returned to NCCF. Unlike a command processor which is invoked to perform a particular service or to handle a specific message, an exit routine is used to screen all messages that fall into one of the exit routine categories. Exit routines are particularly helpful when they handle an event that occurs frequently.

Exit routines must be reentrant, written in assembler language, assembled, and link-edited into phases or load modules specified as DSIEXnn or the name on the XIT-- operand of DSTINIT. (See "Installation" later in this chapter.)

It is not necessary for you to code all of the exit routines available. If you do not code an exit routine, NCCF uses a default exit routine and no processing changes are made.

Overview of NCCF Exit Routines

This section describes each of the NCCF exit routines in detail, including coding requirements and examples of use. Figure 5-1 shows the interfaces between the exit routines and NCCF, the CNM interface, and the access method. Figure 5-2 shows the subtask environment of each of the exit routines. NCCF modules associated with specific exit routines are described in *NCCF Logic*.

There are two types of exit routines: NCCF exit routines (DSIEX01 to DSIEX15), and data services task exit routines (XIT--). Data services task exit routines are invoked by the NCCF data services task (DST). DST exit routines are also defined differently than the NCCF exit routines (see "Installation" later in this chapter).

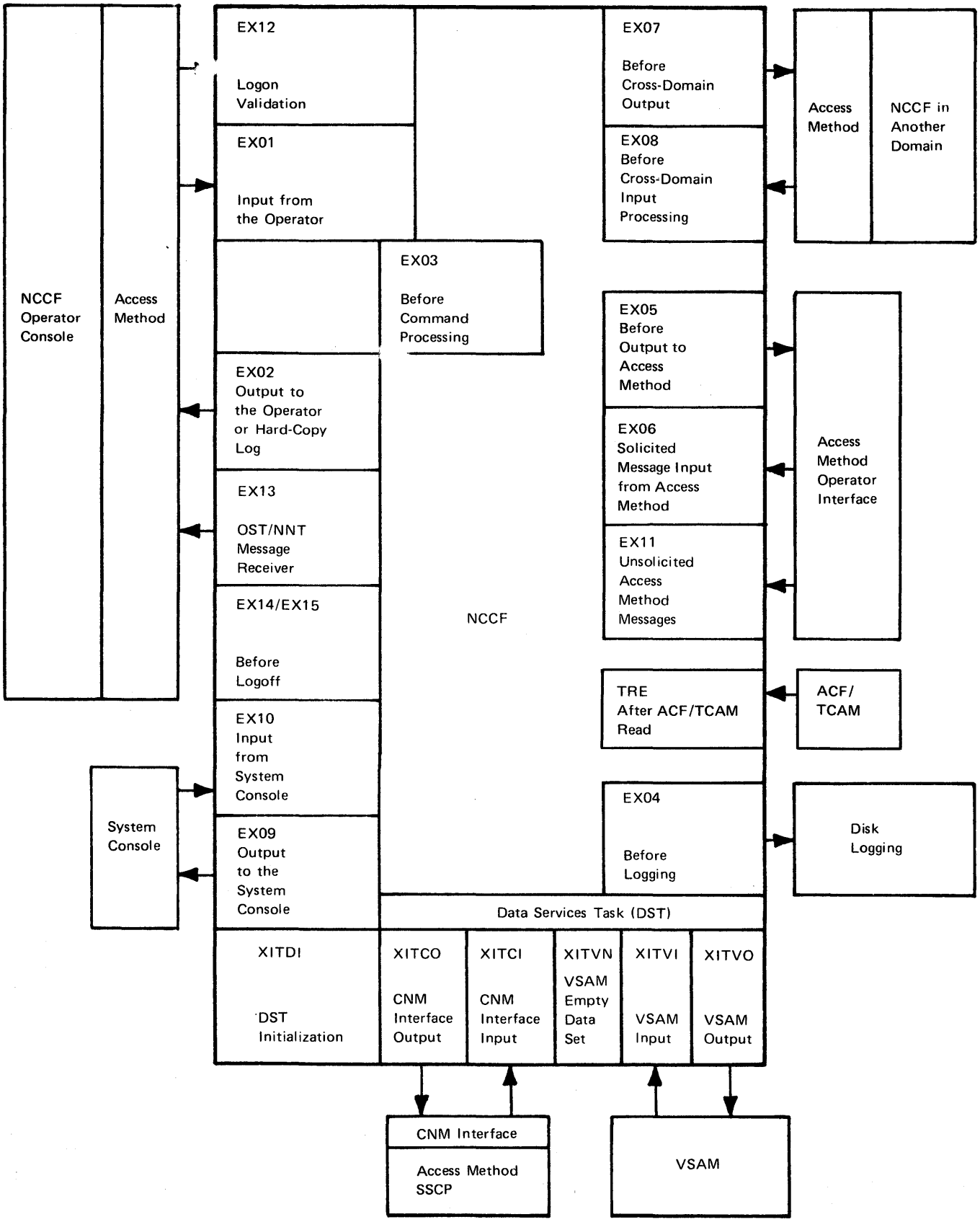


Figure 5-1. NCCF Exit Routine Interfaces

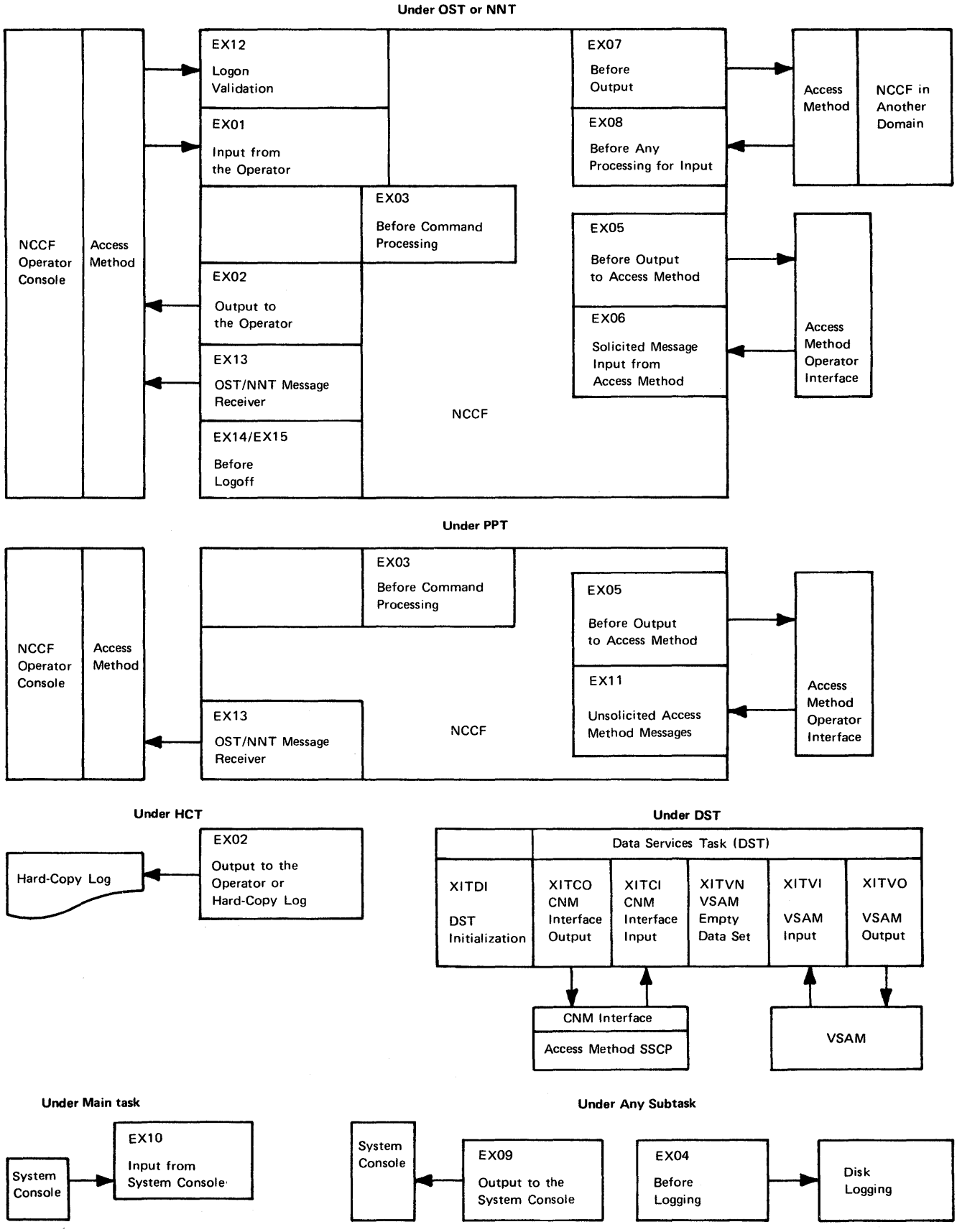


Figure 5-2. Environment of NCCF Exit Routines

The chart below summarizes the exit routines in NCCF. Each of the exit routines are then discussed in detail.

Exit	Description	Subtask Environment
DSIEX01	Input from the operator	OST, NNT
DSIEX02	Output to the operator	OST, NNT, HCT
DSIEX03	Input before command processing	OST, NNT, PPT
DSIEX04	Log output	Any subtask
DSIEX05	Before output to the access method	OST, NNT, PPT
DSIEX06	Solicited message input from the access method	OST, NNT
DSIEX07	Before cross-domain output	OST, NNT
DSIEX08	Before cross-domain input processing	OST, NNT
DSIEX09	Output to the system console	Any subtask
DSIEX10	Input from the system console	Main task
DSIEX11	Unsolicited access method messages	PPT
DSIEX12	Logon validation	OST, NNT
DSIEX13	OST/NNT message receiver	OST, NNT, PPT
DSIEX14	Before logoff	OST, NNT
DSIEX15	Before VSE logoff	OST, NNT
XITDI	DST initialization	DST
XITCO	CNM interface output	DST
XITCI	CNM interface input	DST
XITVN	VSAM empty data set	DST
XITVI	VSAM input	DST
XITVO	VSAM output	DST
DSITRE	After ACF/TCAM read	TCT

DSIEX01: Input from the Operator

Description: DSIEX01 is invoked from the NCCF receive exit routine (DSIRCV) for standard NCCF mode input from the operator terminal or from a cross-domain session. When input is routed to a terminal in another domain (OST-to-NNT), DSIEX01 is invoked under the NCCF-to-NCCF task (NNT).

The DSIEX01 exit routine is executed after device-dependent processing occurs but before syntax or command verbs are analyzed. The message has not yet been logged.

DSIEX01 executes asynchronously, and may interrupt NCCF processing of regular commands.

Example of Use: DSIEX01 might be used to call a command list with a program function key to handle input data from the operator (see "Sample User-Written Exit Routine" later in this chapter).

Coding Requirements: In DSIEX01, messages may be sent to the immediate message area of the operator's screen only. If DSIPSS is used, only TYPE=IMMED is allowed.

Avoid coding a WAIT in this exit routine. Do not use any disk services macros in this exit routine.

DSIEX02: Output to the Operator

Description: DSIEX02 is invoked for standard NCCF output to an operator terminal (DSIPSS TYPE=OUTPUT, DSIPSS TYPE=IMMED). This exit routine is executed before device-dependent output is inserted. The data has not yet been logged.

DSIEX02 may executes asynchronously, and may interrupt NCCF processing of regular commands. DSIEX02 executes asynchronously if it is invoked with DSIPSS TYPE=IMMED, from DSIEX01, or from the NCCF receive exit routine, DSIRCV.

DSIEX02 may also be invoked under the hard-copy task (HCT). In this case, the exit is synchronous and can not be interrupted by other network events. You can determine if the exit routine is running under the HCT by checking the CBHTYPE field of DSITIB or DSITVB for a value of X'03' (see the mapping of DSICBH in Appendix C).

Example of Use: Since the message has been formatted but not yet logged, DSIEX02 can be used to examine the message type and the message text. A substitute message may be supplied, or the message may be deleted entirely.

It is possible to use DSIEX02 to log a message but not display it on the screen. DSIEX02 can issue DSIWLS to log the message and then issue return code 4 to stop processing before the message is displayed.

Coding Requirements: This exit routine should check the TVBINXIT bit in DSITVB to determine for which DSIPSS option the exit was invoked. If the bit is not on, the exit was invoked for DSIPSS TYPE=OUTPUT. If the bit is on, the exit was invoked for DSIPSS TYPE=IMMED and all restrictions on DSIEX01 apply to this exit routine.

Do not use the DSIPSS macro when coding this exit routine. If a message is required, use DSIMQS to queue the message to the subtask. The message receiver will call DSIEX13 and then write the message to the operator's terminal output area.

DSIEX02 does not receive a parse descriptor block (PDB) because NCCF does not check the syntax of messages being sent to a terminal. If you wish to parse the messages in DSIEX02, use the DSIGET macro to obtain storage, then format a PDB control block header in the first 4 bytes, and issue the DSIPRS macro instruction to parse the output buffer. Use the DSIFRE macro to free the storage.

DSIEX03: Input Before Command Processing

Description: DSIEX03 is invoked when NCCF is about to execute a command that was not directly received from a terminal. This exit routine is similar to DSIEX01, but it is for internally generated commands, such as a command:

- In a command list
- Received from another subtask
- Representing an ACF/VTAM or ACF/TCAM message

- Starting the hard-copy log at logon
- Used as the NCCF initial command

Before execution, commands are passed to either DSIEX01 or DSIEX03, but not to both.

Example of Use: DSIEX03 can be used for special command checking, such as counting the number of times a certain command is entered.

Coding Requirements: There are no special coding requirements for DSIEX03.

DSIEX04: Log Output

Description: DSIEX04 is invoked during the logging process. It applies to messages logged on both disk and hard-copy. This exit routine is within log services but is executed before the message is reformatted and sent to the log.

Example of Use: DSIEX04 may be used to edit information sent to the disk or hard-copy logs. Certain messages may be sent to one log and not the other, or not logged at all.

Coding Requirements: Do not use the DSIWCS, DSIWLS, or DSIPSS macros in this exit.

DSIEX04 is not restricted to a particular subtask, but can run under any subtask that issues the DSIWLS macro instruction. For this reason, be sure that any service facilities you request are supported by the task under which you are running. For example, VSAM services may only be used under the data services task.

DSIEX04 does not receive a parse descriptor block (PDB) because NCCF does not check the syntax of messages being sent to a log. If you wish to parse the messages in DSIEX04, use the DSIGET macro to obtain storage, then format a PDB control block header in the first 4 bytes, and issue the DSIPRS macro instruction to parse the output buffer. Use the DSIFRE macro to free the storage.

DSIEX05: Before Output to the Access Method

Description: DSIEX05 invoked when a command is about to be passed to ACF/TCAM or ACF/VTAM. Any domain qualifiers have been removed, and all span checking has been completed.

Example of Use: You might use DSIEX05 to check if an operator has the authority to issue a specific command.

Coding Requirements: There are no special coding requirements for DSIEX05.

DSIEX06: Solicited Message Input from the Access Method

Description: DSIEX06 is invoked when a solicited ACF/VTAM or ACF/TCAM message is received (solicited messages are messages generated in response to an operator command). No processing has been done on the message yet, and the message has not been logged.

Example of Use: DSIEX06 could be used to change the message number or text of an access method message, or to process access method messages in a special way.

Coding Requirements: There are no special coding requirements for DSIEX06.

DSIEX07: Before Cross-Domain Output

Description: DSIEX07 is invoked before output is sent to a cross-domain operator station task in another NCCF (DSIPSS TYPE=XSEND). The output has not yet been formatted and transmitted.

Example of Use: DSIEX07 might be used to monitor cross-domain traffic over the network.

Coding Requirements: Do not use DSIPSS TYPE=XSEND in this exit routine. Avoid issuing commands that route a commmand to another domain, such as ROUTE, DISPLAY, or VARY. (These commands may be queued for execution by building an IFR type 3 and issuing the DSIMQS macro instruction, if desired.)

DSIEX07 does not receive a PDB; the cross-domain NCCF will parse the messages after they are received. If you wish to parse the messages in DSIEX07, you may use the DSIGET macro instruction to obtain storage, then format a PDB control block header in the first 4 bytes, and issue the DSIPRS macro to parse the output buffer. Use the DSIFRE macro instruction to free the storage.

DSIEX08: Before Cross-Domain Input Processing

Description: DSIEX08 is invoked when input is received from a cross-domain operator station task in another domain. This exit routine handles responses from previously sent messages. DSIEX08 is not invoked if the input from the other domain is a command; in this case, DSIEX03 is used. No processing has been done on the message yet, and the message has not been logged.

Example of Use: You might code DSIEX08 to check whether the OST subtask is in a pause state when message DSI809A prompts the user for cross-domain logon data. If so, DSIEX08 could post the GO-CANCEL ECB to simulate an operator entering the GO command.

Coding Requirements: There are no special coding requirements for DSIEX08.

DSIEX09: Output to the System Console

Description: DSIEX09 is invoked when a message is written to the system console operator using the DSIWCS macro. The message has not been formatted for transmission.

Example of Use: This exit routine may be used to edit messages sent to the system console.

Coding Requirements: Do not use the DSIWCS or DSIMQS macros in DSIEX09. If you need to send a message to the system console from this exit routine, use system macros.

DSIEX09 does not receive a PDB. If you wish to parse the messages in DSIEX09, use the DSIGET macro to obtain storage, then format a PDB control block header in the first 4 bytes, and issue the DSIPRS macro to parse the output buffer. Use the DSIFRE macro to free the storage.

DSIEX10: Input from the System Console

Description: DSIEX10 is invoked when input is received from the system console operator. The exit is invoked after the complete message is available, but before it is interpreted for execution. The message has not been logged.

Example of Use: You could use DSIEX10 to allow the system console operator to enter command abbreviations and synonyms. These could then be expanded in the exit routine.

Coding Requirements: DSIEX10 is called from the NCCF main task, not from a subtask.

DSIEX10 does not receive a PDB. If you wish to parse the messages in DSIEX10, use the DSIGET macro to obtain storage, then format a PDB control block header in the first 4 bytes, and issue the DSIPRS macro to parse the output buffer. Use the DSIFRE macro to free the storage.

DSIEX11: Unsolicited Access Method Messages

Description: DSIEX11 is invoked from the primary POI receiver for unsolicited messages from ACF/VTAM or ACF/TCAM. This exit routine is invoked before the command verb or the resource name is analyzed. The message has not been logged.

Example of Use: One use of DSIEX11 is to handle all unsolicited messages in a special way, different from normal NCCF processing. DSIEX11 might issue a DSIMQS macro instruction to make a copy of the message buffer before it was processed by NCCF. Or, if you wanted unsolicited messages to be sent to all operators, DSIEX11 might transform the messages into MSG ALL commands.

Coding Requirements: If DSIEX11 calls a command or a command list, the command restrictions for the PPT apply (Appendix A shows which commands can run under the PPT).

DSIEX12: Logon Validation

Description: DSIEX12 is invoked at the completion of the logon process. The logon has been accepted by NCCF. If the exit routine issues a return code of zero, the logon will proceed. If specified, the user's hardcopy log is started and the NCCF initial command is executed. If the return code is nonzero, the operator is logged off.

Example of Use: DSIEX12 might be used to do additional checking on user authorization, do user environment customization, or send messages to other operators.

Coding Requirements: There are no special coding requirements for DSIEX12.

DSIEX13: OST/NNT Message Receiver

Description: DSIEX13 is invoked within the message receiver for subtask-to-subtask communication. This exit routine is invoked when either a message buffer or an internal function request (IFR) type 8 is received through the DSIMQS macro.

A message buffer is any buffer that does not have a HDRMTYPE of "I" (internal function request). When DSIEX13 returns, these buffers are written to the operator terminal with DSIPSS TYPE=OUTPUT, unless return code 4 is issued. The messages are logged after exit routine DSIEX02 is invoked.

IFR type 8 is an internal function request reserved for definition by the user. An IFR type 8 is not written to the operator terminal.

Example of Use: If you wish to initiate a user function with a buffer, you might choose to use IFR type 8 in conjunction with DSIEX13. IFR type 8 could be further subdivided by providing a unique value in the first two bytes of each IFR type 8 buffer.

Coding Requirements: There are no special coding requirements for DSIEX13.

DSIEX14: Before Logoff

Description: DSIEX14 is invoked when an OST or NNT subtask is about to terminate normally (not abend). This exit routine may be invoked for several reasons, including:

- LOGOFF is entered at the operator's terminal.
- Subtask LOSTERM exit is driven (ACF/VTAM).
- The subtask is posted to terminate.

The subtask cannot communicate with the operator's terminal at this point; however, it is possible to issue the DSIWCS macro to write to the system console and the DSIWLS macro to write entries to the log.

Example of Use: DSIEX14 could be coded to save accounting information, free user-obtained storage, or update tables.

Coding Requirements: DSIEX14 does not receive an input buffer or PDB because there is no buffer associated with logoff processing. The return code from this exit routine is ignored.

DSIEX15: Before Logoff with MVX/OCCF or VSE/OCCF

Description: DSIEX15 is only invoked if NCCF is running as a subtask of MVS/OCCF, or VSE/OCCF, separately orderable IBM program products. DSIEX15 is provided with MVS/OCCF and VSE/OCCF, so the user does not code this exit routine.

DSIEX15 is invoked during primary session termination processing. This exit routine is called immediately before DSIEX14 and is passed the same parameters. DSIEX15 is invoked after the logoff has been accepted but before cleanup of the

work area. The exit routine notifies MVS/OCCF or VSE/OCCF when the subtask through which it is communicating with NCCF has been terminated.

Example of Use: This exit routine is used by VSE/OCCF only.

Coding Requirements: DSIEX15 should not be coded by the user.

XITDI: Data Services Task (DST) Initialization

Description: XITDI is invoked for each statement read by the DST during initialization. When end-of-file is reached, this exit routine is entered with two DSIUSE fields, USERMSG and USERPDB, set to zero to indicate that there is no more data.

Example of Use: XITDI can be added to the DST initialization deck to provide user initialization values to this exit routine. After processing this statement, the exit routine can prevent the DST from scanning the statement by setting return code 4, USERDROP.

Coding Requirements: When invoked for an end-of-file situation, A nonzero return code in register 15 indicates to the DST that it should terminate.

XITDI is restricted to the service facilities available to DST subtasks.

Note: If all initialization data is to be processed by the exit routine specified as XITDI, the user must specify the DST initialization statement that specifies XITDI as the first statement in the DST initialization member.

XITCO: CNM Interface Output

Description: XITCO is invoked by the data services task (DST) prior to a request for CNM interface output.

Example of Use: This exit routine allows the user to modify the request for CNM data (Forward RU).

Coding Requirements: The exit routine is restricted to the service facilities available to DST subtasks.

If a substitute buffer is specified with return code 8 and register 0, the data must be a valid SNA request unit (RU).

XITCI: CNM Interface Input

Description: XITCI is invoked by the DST after CNM data is received.

Example of Use: This exit routine allows the user to modify CMN interface input data (Deliver RU).

Coding Requirements: The exit routine is restricted to the service facilities available to DST subtasks.

Any output from this exit routine must be in the form of a valid SNA request unit (RU).

XITVN: VSAM Empty Data Set

Description: XITVN is invoked if the DST encounters a VSAM open failure because of an empty data set or file.

Example of Use: This exit routine allows the user to supply a record to be placed into the empty data set. For NPDA and the NCCF VSAM log, each of which run under a DST, an XITVN exit routine is supplied with the program product. You should code this exit routine if you wish to run your own VSAM DST.

Coding Requirements: If you are using the NPDA or NCCF VSAM log DST, you should not code this exit routine.

The exit routine should return with return code 8, and register 0 pointing to a buffer containing the record that will be used to initialize the VSAM data set or file. A return code other than 8 will cause the DST to terminate.

The exit routine is restricted to the service facilities available to DST subtasks.

XITVI: VSAM Input

Description: XITVI is invoked by the DST after a VSAM GET macro has been issued. The record has been read from the VSAM data base but has not yet been passed to the requesting data services command processor.

Example of Use: This exit routine allows the user to modify the record after it has been retrieved from a VSAM data set or file, and before the data is passed to the data services command processor.

Coding Requirements: The exit routine is restricted to the service facilities available to DST subtasks.

XITVO: VSAM Output

Description: XITVO is invoked by the DST immediately before the record is written to the VSAM data base.

Example of Use: This exit routine allows the user to modify the record before it has been written to the VSAM data set or file.

Coding Requirements: The exit routine is restricted to the service facilities available to DST subtasks.

DSITRE: ACF/TCAM Read

Description: This exit routine is called after the check for read completion and before passing the data to NCCF subtasks for processing.

Example of Use: This exit routine allows the user to modify an ACF/TCAM record within NCCF, if the user does not choose to do it in a message handler. For example, either status messages received from BSC devices (such as intervention required) can be discarded (using the USERDROP code from DSIUSE) or a logoff can be forced by substituting a buffer with a logoff command (using USERSWAP code from DSIUSE).

Coding Requirements: Care must be used when inspecting or modifying the buffer. Five types of messages are received from ACF/TCAM:

- Screen-size request
- Cross-domain data
- Operator data
- Terminal data
- Logon request

See Figure 5-3. All five messages begin with the standard NCCF header (BUFHDR). Only the HDRMLENG, HDRBLENG and HDRTDISP fields are valid. HDRTDISP is the displacement from the start of the buffer to the beginning of the ACF/TCAM message. HDRMLENG contains the length of the ACF/TCAM message.

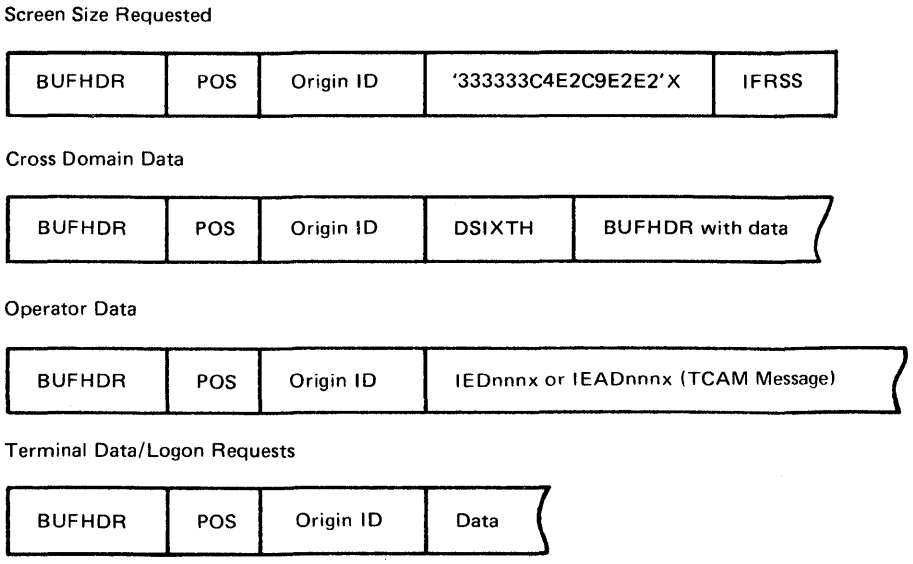


Figure 5-3. Message Formats for DSITRE: ACF/TCAM Read

All ACF/TCAM messages begin with a 1-byte position value followed by an 8-byte origin ID. The data following the origin ID varies by data type.

For screen-size requests, an 8-byte constant ('333333C4E2C9E2E2'X) is followed by an IFRSS (see DSIIFR).

For cross-domain data, it is an XTH control block followed by a NCCF header and data (see DSIXTH).

For operator data, it is an ACF/TCAM network message number in the IEDnnnx or IEAnnnx, followed by the message text. IED and IEA are component *ids, nnn* is the message number, and *x* is the action code (A, I, or E).

Terminal data and logon requests have no special format.

Installation

Exit routines should be written in assembler language, assembled, and link-edited into phases or load modules. These phases or load modules are made available to NCCF by being included in the NCCF library during installation. Only one phase or load module is permitted for each exit routine, and conditional selection at exit time is not allowed.

Exit routines DSIEX01 through DSIEX15 are loaded at NCCF start time by their assigned names. DST user exits (XIT--) are defined to the data services task by the XITxx operands of the DSTINIT statement. Each DST may have different user exit routines. When the DST is started, it loads the exit routines.

If you do not code one of the exit routines, NCCF supplies a default exit routine and processing continues normally. The following message is issued each time NCCF uses a default exit routine:

```
DSI090I  LOAD FAILED FOR NCCF MODULE exitname
```

This message is for your information only; processing will not be affected. If you wish to avoid receiving this message, code any unused exit routines as follows:

```
exitname  CSECT
          SR    15,15
          BR    14
          END
```

Then, link-edit this code into the NCCF load library.

Coding Guidelines

If you intend to write exit routines, you should be familiar with the NCCF service facilities and macro instructions described in Chapter 3.

The following guidelines should be followed in coding exit routines:

- Make all exit routines reentrant.
- Save registers at entry to the exit routine and restore them before returning control to NCCF.
- Avoid wait states. The DSIWAT macro instruction must not be issued in exits DSIEX01 and DSIEX02.

- Do not rely on the contents of registers 0 and 2 through 12 for constant values. Register assignments may vary from exit to exit or from one program release to another.
- NCCF uses registers 0, 1, 14, and 15 for macro instruction expansion.
- Register 13 should always point to a standard 72-byte save area.
- Do not return control to any location in the NCCF program other than that specified by register 14.
- If you are rewording a full-line message, do not change the HDRMTYPE or HDRIND fields in the NCCF buffer header. If you are deleting a full-line message, delete each section of the message on successive exit routine calls. Be careful not to delete a CONTROL, LABEL, or END line unless you are deleting the whole message. You can tell if a message is a full-line message by checking the HDRMTYPE field of the NCCF buffer header (BUFHDR section of DSITIB). HDRTYPEJ, HDRTYPEK, and HDRTYPEL are full-line messages. For information on user-written full-line messages, see the section titled "Full-Line Command Processor Considerations."

Input Parameters

Registers

Standard CALL and RETURN sequences transfer control to and from exit routines. Upon entry to the exit routine, registers contain the following information:

Register	Contents
1	Address of the user exit parameter list (DSIUSE). This parameter list is described in detail later.
13	Address of a standard 72-byte NCCF save area used to store the caller's registers.
14	Return address of the NCCF program.
15	Entry address of the exit routine.
0,2--12	Unspecified.

Control Block Considerations

If you use NCCF service facilities in an exit routine, you must include some control block DSECTs in the exit routine. This can be done using the DSICBS macro instruction (see Chapter 3). The control blocks needed depend on what services your exit routine invokes; however, you will want to include at least DSIMVT, DSIUSE, and DSISWB. DSIUSE DSISWB are described below.

User Exit Parameter List (DSIUSE)

The user exit parameter list (DSIUSE) contains addresses for the following: the buffer containing the message, the LU name associated with the message, the operator identification, and control blocks DSISWB, DSITVB, and DSIPDB. An extension to DSIUSE is present for DSIEX12 and the DST exit routines involved with input/output (XITCO, XITCI, XITVN, XITVI, XITVO). For DSIEX12, the password, hard-copy printer name, and profile name are given. For the DST exit routines, the address of DSIDSRB is given. Refer to *Appendix C* for the location of each of these fields.

Field	Description
USERCBH	Is a standard NCCF control block header. The second byte USERCODE, indicates what exit routine is being invoked.
USERMSG	Points to a buffer in standard NCCF buffer format, consisting of a buffer header (BUFHDR) followed by text. For input-type exists, device-dependencies have been removed. For input from an operator terminal, substitution for the AGAIN command has not occurred. This buffer should not be changed, but may be referenced. In exit routines DSIEX14, DSIEX15, XITDI for end-of-file, and XITVN, this field is set to zero. In DSIEX04, the buffer is in the format set up by the caller. It has not yet been reformatted for the NCCF log.
USERLU	Points to an 8-byte area that contains the logical unit name related to the subtask in control, as follows: For an OST, the node name of the operator's terminal. For an NNT, the APPL name of the OST that issued the START DOMAIN command (NCCFID DOMAINID appended with 3-digit number). For a PPT, the NCCFID DOMAINID parameter appended with the characters "PPT". For an HCT, the node name of the hardcopy printer. For a DST, the name from the TSKID operand of the TASK definition statement. If the main task is in control, this 8-byte area contains the characters "SYSOP".
USEROPID	Points to an 8-byte area that contains a name related to the subtask in control, as follows: For an OST or NNT, the operator's identifier.

Field	Description
	For a PPT, the NCCFID DOMAINID parameter appended with the characters "PPT". For an HCT, the address of the node name of the hardcopy printer. If the main task is in control, this 8-byte area contains the characters "SYSOP".
USERSWB	Points to a service work block (SWB) that may be used by the exit routine to request services from NCCF or as a work area. If necessary, another SWB may be obtained by using the DSILCS macro (see the description of SWB below).
USERTVB	Points to the task vector block (TVB). The TVB contains information about the subtask under which the exit routine was invoked. The TVB is also used to obtain the addresses of the TIB, MVT, and SVL (through the MVT).
USERPDB	Points to a parse descriptor block (PDB) or contains 0. The PDB contains parse data relating to the buffer pointed to by USERMSG. For exit routines DSIEX02, DSIEX04, DSIEX07, DSIEX09, DSIEX10, DSIEX14, DSIEX15, DSITRE, and XITDI for end-of-file, this field contains 0; A PDB is not available when calling these exit routines.
USERLGON	Extension for DSIEX12 and the DST exit routines. If present, this extension contains the following fields:
USERPSWD	For DSIEX12 only, contains the password entered by the operator during logon. If OPTIONS VERIFY=MINIMAL is specified, this field contains blanks. For exit routines other than DSIEX12, this field is not initialized.
USERHCPY	For DSIEX12 only, contains the name of the hard-copy printer used by the operator for this session. If no hard-copy is used or if OPTIONS VERIFY=MINIMAL is specified, this field contains blanks. For exit routines other than DSIEX12, this field is not initialized.
USERPROF	For DSIEX12 only, contains the name of the profile used for this session. If OPTIONS VERIFY=MINIMAL is specified, this field contains blanks. For exit routines other than DSIEX12, this field is not initialized.
USEDSRB	For DST exit routines XITVN, XITVI, XITVO, XITCI, and XITCO, points to the DSRB associated with the DST input/output request. For other exit routines this field is not initialized.

Service Work Block (SWB)

The service work block (SWB) contains the parameter list for most of the NCCF service facilities that are used in an exit routine. The USERSWB field of DSIUSE points to a SWB that can by used to request these service facilities. Remember that all exit routines that use NCCF service facilities must have addressability to the main vector table (MVT). See Chapter 3 for more information.

If you decide to use the SWB pointed to by DSIUSE as a work area, you can obtain another SWB with the DSILCS macro instruction when requesting NCCF services. The DSILCS macro might be coded as follows:

```
DSILCS    CBADDR=(R2),SWB=GET
```

If you use the DSILCS macro instruction to obtain another DSISWB, be sure to initialize the SWBTIB field of the SWB with the address of the caller's TIB before you request NCCF services.

When the exit routine no longer requires the SWB obtained using DSILCS, the SWB should be freed. To free the SWB in the example above, you would code:

```
DSILCS    CBADDR=(R2),SWB=FREE
```

Note: If you use an SWB as a work area, be careful not to overlay the SWBTIB or SWBCBH fields because these fields are not reinitialized by NCCF. If you must change either of these fields, reinitialize them before returning control to NCCF.

Output Parameters

When an exit routine returns control to NCCF, the register contents should be as follows:

Register	Contents
0	Unchanged, unless return code 8 is received in register 15.
1-14	Unchanged.
15	A return code (see Figure 5-4 for a list of valid return codes). The return code is not examined by DSIEX14.

The parameter list should be unchanged with the exception of the work area.

Exit Routine	Return Code	Symbol	Meaning
All but DSIEX14	0	USERASIS	Use the message as presented to the exit routine. For DSIEX12, allow the logon.
	4	USERDROP	Delete the message; do not process it further. For DSIEX12, reject the logon.
All but DSIEX12 and DSIEX14	8	USERSWAP	A message has been substituted for the message presented to the exit routine. The address of buffer containing the new message is in register 0. When using this return code, follow these restrictions: • The message cannot be longer than the data portion of the original buffer. You can calculate the length of the buffer area by subtracting HDRTDISP from HDRBLENG in the BUFHDR section of DSITIB. • To ensure that the user buffer will be freed, either (1) build the buffer in the SWBPLIST or SWBDATD fields of DSISWB or (2) acquire the buffer at logon in DSIEX12 by using user fields such as TIBUFLD, TVBUFLD, or MVTUFLD and free the buffer with DSIEX14 during logoff.
DSIEX04 Only	12	USERLOG	Write the message to the disk log only.
	16	USERLOGR	Write the substituted message to the disk log only. The address of the buffer containing the new message is in register 0.
	20	USERHCL	Write the message to the hard-copy log only.
	24	USERHCLR	Write the substituted message to the hard-copy log only. The address of the buffer containing the new message is in register 0.

Figure 5-4. Return Codes Set by Exit Routines

Exit Routine Prototype

The following shows the basic structure of an exit routine, including entry, obtaining an SWB freeing the SWB, and exit linkage. This exit routine may be used as a prototype for writing your own exit routines.

```
DSIEXNN  CSECT
         DSICBS DEFER=ALL              INCLUDE CONTROL BLOCKS
         DSICBS DSITIB,DSITVB,DSIMVT,DSISWB,DSIPDB,DSIUSE,DSISVL
         STM    14,12,12(13)           SAVE REGISTERS
         LR     10,15                  SAVE BASE REGISTER
         USING  DSIEXNN,10             REG 10 IS THE BASE
         USING  DSIUSE,1               REG 1 POINTS TO DSIUSE
         L      11,USERSWB             LOAD REG 11 WITH SWB ADDRESS
         USING  DSISWB,11              BASE SWB
         LA     2,SWBSAVEA             GET ADDRESS IF SAVE AREA
         ST     2,8(,13)               SAVE REG 2
         ST     13,4(,2)               SAVE REG 13
         LR     13,2                   REG 13 CONTAINS SAVE AREA ADDR
         LR     9,1                    MOVE DSIUSE ADDRESS
         DROP   1                      DROP ORIGINAL BASE
         USING  DSIUSE,9               REG 9 POINTS TO DSIUSE
         L      12,USERPDB             LOAD REG 12 WITH PDB ADDR
         USING  DSIPDB,12              BASE THE PDB
```

```
         L     8,USERTVB           ADDRESS THE TVB
         USING DSITVB,8            BASE THE TVB
         L     7,TVBMVT            GET THE ADDRESS OF THE MVT
         USING DSIMVT,7            BASE THE MVT
*************************************************************************
*                                                                       *
*     NOW OBTAIN ANOTHER SWB IN ORDER TO ISSUE NCCF SERVICE MACROS      *
*                                                                       *
*************************************************************************
         DSILCS CBADDR=WORKADDR,SWB=GET  GET A NEW SWB
         SPACE 1
* NOTE:  SEE DSISWB DSECT AT THE END OF THE LISTING
         SPACE 1
         LTR   15,15               TEST DSILCS RETURN CODE
         BNZ   ABEND               GO AND ABEND IF OUT OF STORAGE
         L     5,WORKADDR          POINT TO NEW SWB
         L     4,TVBTIB            PUT THE TIB ADDRESS IN REG 4
         ST    4,SWBTIB-DSISWB(,5) STORE THE TIB ADDR IN THE NEW SWB
*************************************************************************
*                                                                       *
*   NOW REGISTER 5 POINTS TO THE NEW SWB. THIS SWB SHOULD BE USED       *
*   FOR ALL SERVICE MACROS IN THIS EXIT.                                *
*                                                                       *
*************************************************************************
*        PUT YOUR USER EXIT CODE HERE                                   *
*         ...                                                           *
*         ...                                                           *
*         ...                                                           *
*         ...                                                           *
*************************************************************************
*                                                                       *
*   NOW THE NEW SWB MUST BE RELEASED BEFORE EXITING.                    *
*                                                                       *
*************************************************************************
RETURN   EQU   *
         DSILCS CBADDR=(5),SWB=FREE   NOW FREE THE GOTTEN SWB
         LTR   15,15               TEST IF DSILCS WAS SUCCESSFUL
         BNZ   ABEND               ABEND IF FAILED TO FREE SWB
         SPACE 1
*   PICK THE EXIT LINKAGE DESIRED FROM THE THREE BELOW:
*   TO PROCESS THE BUFFER ASIS FROM HERE ON, RETURN FROM HERE
ASIS     LA    15,USERASIS         SET AN ASIS RETURN CODE
         L     13,4(,13)           RESTORE CALLER'S SAVE AREA ADDR
         L     14,12(,13)          RESTORE CALLER'S REGISTER 14
         LM    0,12,20(13)         RESTORE CALLER'S REGISTERS 0-15
         BR    14                  RETURN TO CALLER
         SPACE 1
*   TO STOP FURTHER PROCESSING ON THIS BUFFER, RETURN FROM HERE
DROP     LA    15,USERDROP         SET A DROP RETURN CODE
         L     13,4(,13)           RESTORE CALLER'S SAVE AREA ADDR
         L     14,12(,13)          RESTORE CALLER'S REGISTER 14
         LM    0,12,20(13)         RESTORE CALLER'S REGISTERS 0-15
         BR    14                  RETURN TO CALLER
         SPACE 1
*   TO SWAP A BUFFER FOR THE BUFFER PASSED, RETURN FROM HERE
SWAP     LA    15,USERSWAP         SET A SWAP RETURN CODE
         L     0,SWAPBFR           POINT TO THE SWAP BUFFER
         L     13,4(,13)           RESTORE CALLER'S SAVE AREA ADDR
         L     14,12(,13)          RESTORE CALLER'S REGISTER 14
         LM    1,12,24(13)         RESTORE CALLER'S REGISTERS 1-15
         BR    14                  RETURN TO CALLER
         SPACE 1
ABEND    EQU   *
         ABEND 4000                ABEND 4000-4095 RESERVED FOR USER
* IN VSE, USE THE DSIABN MACRO.
         SPACE 1
         DSICBS DEFER=INCLUDE,PRINT=YES,EJECT=YES
DSISWB   DSECT ,                   EXTEND THE SWB DEFINITION
         ORG   SWBADATD            POINT TO 256 BYTE WORK AREA
WORKAREA DS    0CL256              WORKAREA IS 256 BYTES LONG
```

```
WORKADDR DS    A                    ADDRESS OF NEW SWB SAVED HERE
SWAPBFR  DS    A                    ADDRESS OF SUBSTITUTION BUFFER
         SPACE 1
DSIEXNN  CSECT ,                    RESUME CSECT
         END   DSIEXNN              END OF THE USER EXIT
```

Sample User-Written Exit Routine

The following is an example of a user-written exit routine. This DSIEX01 exit routine allows an operator to enter data and press a program function (PF) key that the exit routine interprets and uses to call a command list. The command lists are then defined with names such as $A, $B, and so forth. For example, if an operator enters TASKS and presses PF3, the exit routine changes the percent (%) sign (for the PF) to $, and 3 to C and instructs NCCF to ue the original command in the buffer. The SC then causes NCCF to call command list $C, which can perform the function the user wishes at this point. This exit routine requires inclusion of the following NCCF control blocks: DSICBH, DSIMVT, DSIPDB, DSISWB, DSITIB, DSITVB, and DSIUSE.

```
         TITLE 'DSIEX01 - NCCF TERMINAL INPUT USER EXIT ROUTINE'
*********************************************************************
*                                                                   *
* DSIEX01 - NCCF USER EXIT FOR TERMINAL INPUT                       *
*                                                                   *
* THIS USER EXIT ROUTINE CHECKS TO SEE IF A 3270 PF OR PA KEY WAS   *
* DEPRESSED. IF IT WAS, THE VERB IN THE BUFFER WILL START WITH A '%'. *
* THESE VERBS (COMMANDS) ARE ASSUMED TO BE CLISTS OR COMMANDS       *
* STARTING WITH THE ' ' (X'5B') CHARACTER.  PF KEYS ARE TRANSLATED  *
* TO  A (PF1) TO  X (PF24). PA KEYS ARE TRANSLATED TO 1 (PA1) TO    *
* 3 (PA3).                                                          *
*                                                                   *
* INPUT: R1  = DSIUSE ADDRESS       OUTPUT: REGS SAME AS INPUT EXCEPT *
*        R13 = SAVEAREA ADDRESS               R15 = 0 IF OK         *
*        R14 = RETURN ADDRESS                 R15 = 0 IF UNSUPPORTED *
*        R15 = ENTRY ADDRESS                          KEY WAS PRESSED *
*                                                                   *
*********************************************************************
DSIEX01  CSECT
         DSICBS DSITIB,DSIPDB,DSISWB,DSIUSE,DSIMVT,DSISVL,DSITVB,      *
               DEFER=ALL          INCLUDE CONTROL BLOCKS AT END
         STM   14,12,12(13)       SAVE REGISTERS
         LR    10,15              SET BASE ADDRESS
         USING DSIEX01,10
         LR    7,1                MOVE USER EXIT PARAMETER LIST ADDRESS
         USING DSIUSE,7
         L     11,USERSWB         LOAD SWB REG WITH SWB ADDRESS
         USING DSISWB,11
         LA    2,SWBSAVEA         GET ADDRESS OF SAVEAREA
         ST    2,8(13)
         ST    13,4(2)
         LR    13,2               R13 CONTAINS SAVEAREA ADDRESS
         L     12,USERPDB         LOAD PDB REG WITH PDB ADDRESS
         USING DSIPDB,12
```

```
*********************************************************************
*NOW LOOK AT DATA IN THE INPUT BUFFER
*********************************************************************
         CLI   PDBNOENT+1,X'00'   DATA IN INPUT BUFFER?
         BE    RETURN             IF NOT, GET OUT
         LA    3,PDBTABLE         GET PDB ENTRY FOR CMD VERB
         USING PDBENTRY,3
         CLI   PDBLENG,X'02'      IS LENGTH = 2?
         BNE   RETURN             IF NOT, PFK NOT PRESSED SO LEAVE
         L     5,PDBBUFA          ADDRESS OF COMMAND BUFFER
         AH    5,PDBDISP          ADD DISPLACEMENT TO VERB
*                                 R5 NOW HAS ADDRESS OF CMD VERB
*********************************************************************
*CHECK IF PF/PA KEY WAS USED TO ENTER THIS COMMAND
*********************************************************************
         CLI   0(5),C'%'          IF %, IT WAS ENTER KEY WITH DATA
         BNE   RETURN             BRANCH IF ENTER KEY WITH DATA
*********************************************************************
*PF KEY WAS USED
*********************************************************************
*  PF1 - PF24 MAP TO COMMANDS/CLISTS NAMED  A- X, PA1-PA3 TO  1- 3
*********************************************************************
         TR    1(1,5),TRANTAB     TRANSLATE AID
         MVI   0(5),C' '          CONVERT % TO
         CLI   1(5),C'*'          IS AID INVALID?
         BNE   RETURN             NO
*********************************************************************
*     UNSUPPORTED AID BYTE
*********************************************************************
         L     5,USERMSG          POINT TO BUFFER
         USING BUFHDR,5           COVER BUFFER HDR
         MVI   HDRMTYPE,HDRTYPEU  MAKE A USER MSG
         MVC   HDRTDISP,DISPLACE
         MVC   HDRTEXT(L'AIDMSG),AIDMSG
         MVC   HDRMLENG,MSGLENG
         L     2,USERTVB          POINT TO THE TVB
         L     2,TVBMVT-DSITVB(,2) POINT TO THE MVT
         USING DSIMVT,2
*
         DSICLS SWB=GET,CBADDR=MYSWBPTR
*
         LTR   15,15              WAS SWB GOTTEN?
         BNZ   RETURN             NO - RETURN WITHOUT MSG
         L     15,MYSWBPTR        ADDRESS MY SWB
         MVC   SWBTIB-DSISWB(,15),SWBTIB COPY THE TIB ADDRESS TO MY SWB
*
         DSIPSS TYPE=IMMED,BFR=(5),SWB=MYSWBPTR
*
         DSILCS SWB=FREE,CBADDR=MYSWBPTR
*
ERRRET   L     13,4(13)           ERROR RETURN
         LM    14,12,12(13)
         LA    15,4               RC = DELETE THE MESSAGE/COMMAND
         BR    14
RETURN   EQU   *                  GOOD RETURN
         L     13,4(13)
         LM    14,12,12(13)
         SR    15,15              RC = CONTINUE THE PROCESS
         BR    14
```

```
*
TRANTAB  EQU   *-X'4C'
         DC    C'VWX*'              X'4C'-X'4F'
         DC    16C'*'               X'50'-X'5F'
         DC    11C'*',C'13*2*'      X'60'-X'6F'
         DC    10C'*',C'JKL***'     X'70'-X'7F'
         DC    64C'*'               X'80'-X'BF'
         DC    C'*MNOPQRSTV******'  X'C0'-X'CF'
         DC    32C'*'               X'D0'-X'EF'
         DC    C'*ABCDEFHI'         X'F0'-X'F9'
*
DISPLACE DC    AL2(HDRTEXT-BUFHDR)
AIDMSG   DC    C'USR001I NOT SUPPORTED FOR COMMAND ENTRY'
MSGLENG  DC    AL2(*-AIDMSG)
         DSICBS DEFER=INCLUDE,PRINT=NO
DSISWB   DSECT ,                    RESUME SWB. REDEFINE WORKAREA.
         ORG   SWBADATD
MYSWBPTR DS    A                    POINTER TO MY SWB
DSIEX01  CSECT ,                    RESUME CSECT
         END   DSIEX01
```

Chapter 6. Subtasks

This chapter describes the rules and requirements for writing optional NCCF subtasks. It also describes the control block fields that are of use when coding a subtask. A sample user-written subtask is shown at the end of the chapter.

NCCF provides service facilities that may be used by user-written subtasks. These facilities and the macro instructions that call them are discussed in detail in Chapter 3.

Why Write Your Own Subtask?

Each of the subtasks in NCCF handles a separate function: The OST and NNT control an operator's terminal and cross-domain session. The PPT processes system operator commands, unsolicited access method commands, and timer-initiated commands. The DST provides support to gather, record, and manage data. The HCT controls the hard-copy device. In ACF/TCAM, the TCT acts as the interface between NCCF and the ACF/TCAM application message handler.

You can write your own subtask to provide additional customization of NCCF. For example, you might write a subtask to centralize a process that would be used by several different subtasks, such as access to a data base. You might also write a subtask to process certain types of data or one network management function. The subtask that you write is attached and started by NCCF as an optional subtask.

Defining the Subtask to NCCF

The subtask must be link-edited and stored in an NCCF load library under the name specified on the MOD operand of the TASK definition statement.

You should use the TASK definition statement to define your subtask to NCCF. For example, the following definition statement

```
TASK MOD=USERMOD,TSKID=USERTASK,MEM=USERMEM,PRI=9,INIT=Y
```

indicates that the subtask is in module USERMOD, and has a SUBTASK identification of USERTASK. The dispatching priority is 9, the lowest priority, and the subtask is to be started during NCCF initialization. For a DST, MEM is used as the member name of DSIPARM for additional initialization information. The subtask you write can use the MEM parameter for other functions, for example as DD name, a member name, or an operator identifier. For more information on MEM, see "Reading the Subtask Initialization Deck" later in this chapter.

Subtask Organization

NCCF subtasks are normally divided into three parts: initialization, process, and termination. See Figure 6-1 for an overview. Initialization sets up the processing environment, process performs the subtask functions, and termination cleans up and exits.

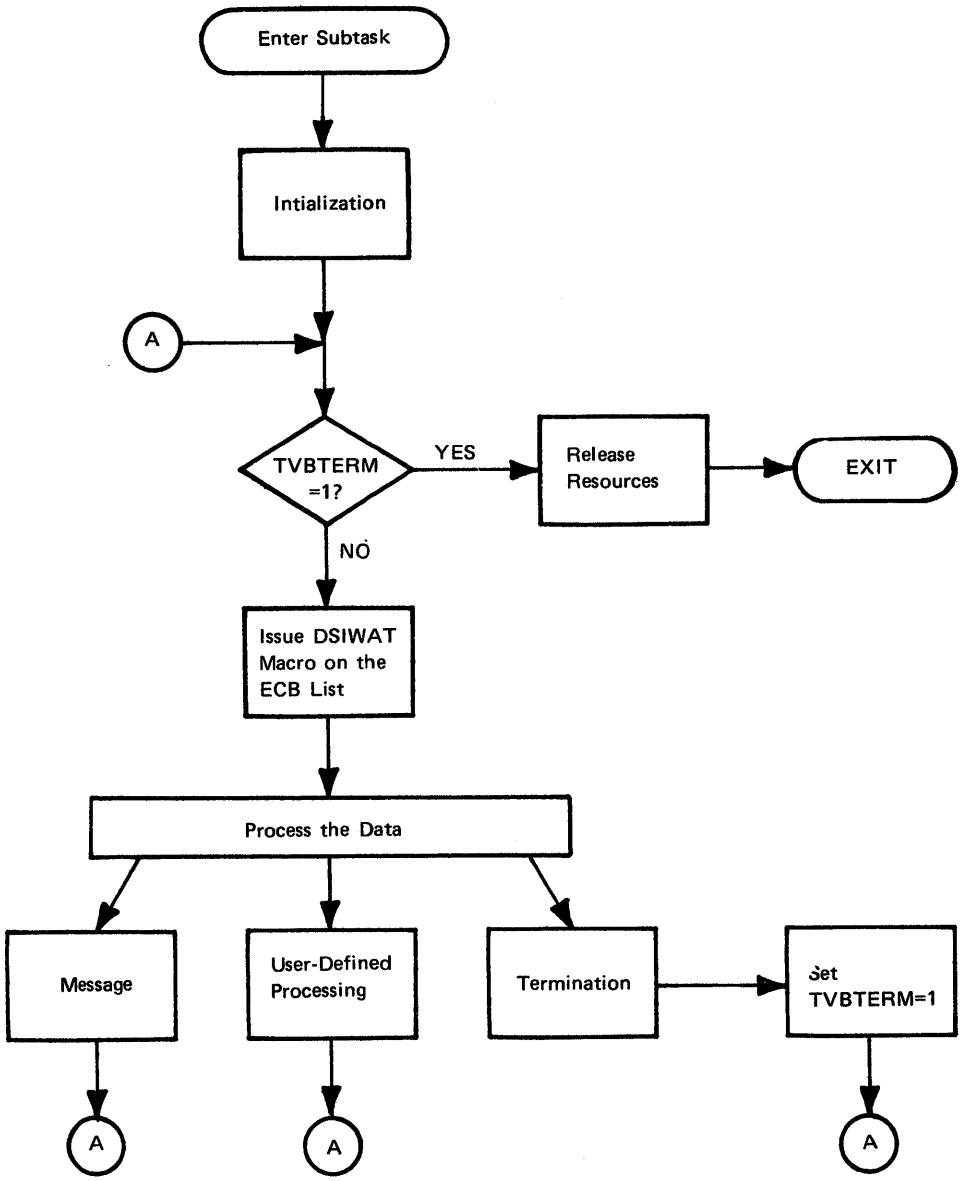


Figure 6-1. Subtask Organization

The basic initialization and termination procedures are standard for all subtasks, and must be followed when you write your optional subtask (See "Requirements," below).

The process section of a subtask usually begins with a DSIWAT (WAIT) macro on an event control block (ECB) list. The contents of the ECB list vary according to the function to be performed; however, all subtask ECB lists must contain the termination ECB, TVBTECB.

In OS/VS, all ECBs are posted using the X'40000000' bit. Both DSIPOS and OS/VS system POST macros use the same bit. In VSE, the DSIPOS macro uses the X'40000000' bit while the VSE system POST macro uses the X'0000800' bit to indicate that the event has been posted. Subtasks running

under VSE should check for the presence of both bits. It is recommended that NCCF subtasks use DSIPOS rather than the system POST macro to post ECBs.

It is the responsibility of the subtask to determine which ECB(s) are posted and take the appropriate action. Before reissuing the DSIWAT macro, the ECBs must be set to zero.

Requirements

This section describes those features that must be provided by a user-written subtask.

Coding Guidelines

When writing subtasks, you should be familiar with the NCCF service facilities and macro instructions described in Chapter 3.

The following guidelines should be followed in coding subtasks:

- Make all subtasks reentrant.
- Save registers at entry to the subtask and restore them before returning control to NCCF.
- NCCF uses registers 0, 1, 14, and 15 for macro instruction expansion.
- Register 13 should always point to a standard 72-byte save area.
- Do not return control to any location in the NCCF program other than that specified by register 14.

Entry and Exit Linkage

When a subtask is attached, the following register contents are provided:

Register 1	Address of task vector block for subtask
Register 13	Save area address
Register 14	Return address
Register 15	Subtask entry point address

The control blocks at entry to an optional subtask are shown in Figure 6-2. From the task vector block (DSITVB) you can obtain the addresses of the task information block (DSITIB) and the main vector table (DSIMVT), which are used by the subtask. DSITIB is pointed to by the TVBTIB and the DSIMVT is pointed to by the TVBMVT in the DSITVB. These control blocks are described in detail at the end of this chapter. The TVBTCB field of DSITVB points to the OS/VS task control block or to the VSE NCCF pseudo TCB (See *NCCF Logic*).

Subtask Attachment

NCCF provides two types of attaches: normal and cleanup. A normal attach is caused by issuing a START TASK command or by specifying INIT=Y on the TASK definition statement. The TVBTERM bit in DSITVB is set to zero.

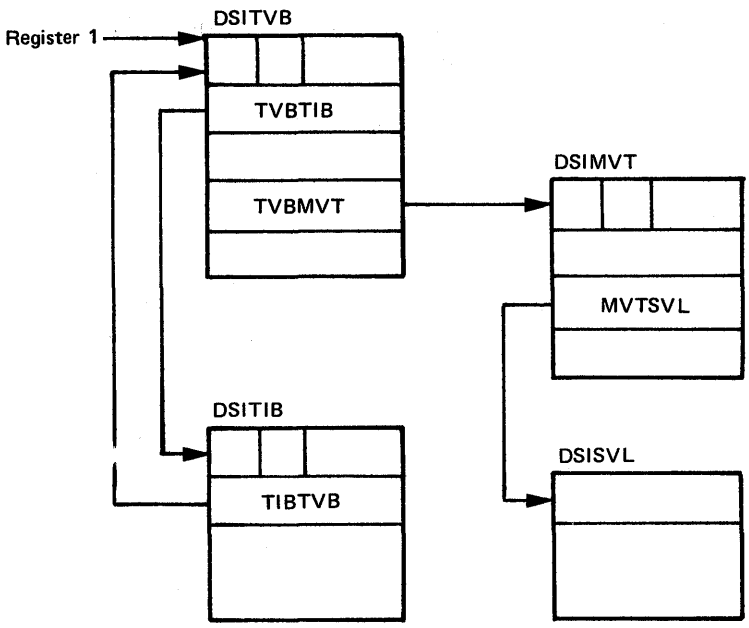


Figure 6-2. Subtask Input Parameter Control Blocks

When a subtask terminates normally, this bit is set to one by the subtask, indicating that the resources allocated by the subtask have been released and now the resources allocated by the main task are to be released.

A cleanup reattach occurs after a subtask has terminated abnormally. The NCCF main task sets the TVBTERM bit to one, and the subtask is reattached. When the subtask gains control, it should free all resources, and then exit normally.

Indicating that the Subtask is Ready

A subtask must indicate that it is ready to operate. After the subtask is initialized and before it starts processing, the subtask must enter a value into the TVBOPID field of the DSITVB. This value is chosen by the subtask, and must be unique in the TVB chain. The most frequently used method is to copy the contents of TVBLUNAM, which is the value of the TSKID operand of the TASK definition statement, into TVBOPID. You may also use a hard-coded value, or another method.

For OS/VS, you should ensure that the TVBOPID value is unique by using the OS/VS ENQ macro and DSILCS, as follows:

1. Issue the ENQ macro:

```
ENQ  (MVTNCCFQ,MVTTVBRN,E,18,STEP)
```

Note: Be sure you have addressability to the MVT before issuing this macro (see Chapter 3).

The ENQ macro prevents other subtasks from changing operator identifiers in the chain.

2. Issue the DSILCS macro, supplying the correct operand values:

```
DSILCS  CBADDR=name,TVB=MVTTVB,OPID=name
```

   The DSILCS macro will attempt to locate a DSITVB containing the specified operator identifier. If the return code is 0, the name cannot be used in TVBOPID because it is already in use. If the return code is 8, the name is unique; place the subtask identifier into TVBOPID and set the TVBACTV bit to 1.

3. Dequeue the TVB chain by issuing:

```
DEQ  (MVTNCCFQ,MVTTVBRN,18,STEP)
```

For VSE, issue the DSILCS macro instruction as shown in step 2, above. The ENQ and DEQ macros are not required.

Subtask Termination

Include the TVBTECB field of DSITVB in the subtask ECB list for each subtask you write. When an NCCF CLOSE NORMAL command is issued, and after all NCCF operators have logged off, the main task posts the TVBTECB bit of the subtask to indicate that subtask termination is requested. When the subtask finds the TVBTECB bit posted, it should:

- Release all resources.
- Set TVBOPID to blanks.
- Set TVBACTV to zero.
- Set TVBTERM to one.
- Reload the registers originally passed, and return to the address specified in register 14.

Optional Facilities

This section describes optional facilities that you may wish to provide in a user-written subtask.

LIST Command

The LIST command may be used to display the status of the subtask from an operator's terminal. For optional subtasks, first a header line is displayed by the LIST command processor. Then the contents of TVBOPID and TVBLUNAM are displayed, along with the subtask type (OPT) and the status of the subtask as determined by the following DSITVB bit fields: TVBRCVRY (in recovery), TVBLGOFF (stopping), TVBACTV (active), TVBLGON (starting), or none of the above (not active). The search order of the bits is the order shown.

The subtask may also create its own status display. The subtask must create its own display when the status changes. Follow these guidelines when creating a status display:

1. NCCF will not print your status display if any of the following are true:
   - The buffer pointer in TIBOSLST is zero.
   - The TVBTIB field of the DSITVB is zero.
   - The TVBTERM bit is set.
   - The TVBACTV bit is zero.
2. The subtask is responsible for keeping the buffer containing status information current. When updating status, set the DSITIB buffer pointer TIBOSLST to zero until the buffer is ready to be displayed.

Queued Storage Management

The DSIGET Q=YES facility is provided to ensure that storage is freed in case of an abnormal termination by recording storage requests. The storage blocks are chained together from one of two queues: TVBEXITQ (if EXIT=YES), and TVBTASKQ (if EXIT=NO). During normal subtask termination, any DSIGET storage remaining should be freed using DSIFRE. To release the storage:

- Check the queue anchor (TVBTASKQ or TVBEXITQ). If the field is zero, there is no storage to be released.
- If the queue anchor is not zero, obtain the storage address from this field.
- Add 16 to address.
- Issue DSIFRE Q=YES specifying the computed address and the size of the storage. The size may be specified as zero (LV=0).
- Continue to check the queue anchor until no storage is left to be released.

For an example of this process, see the section titled "Sample User-Written Subtask."

Reading the Subtask Initialization Deck

You may wish to use the value of the MEM keyword of the TASK definition statement as the 1-to-8 character name of the user-defined initialization data set B book or member name in DSIPARM. This value is found in the

TVBMEMNM field of DSITVB. The subtask is responsible for processing the contents of this book or member. The subtask should issue the following commands to use NCCF disk services to read DSIPARM:

DSIDKS TYPE=CONN,NAME=DSIPARM to connect the subtask to NCCF disk services.

DSIDKS TYPE=FIND,NAME=*TVBMEMNM address* to find the DSIPARM member and read the first record.

DSIDKS TYPE=READ until the end-of-file return code is returned.

DSIDKS TYPE=DISC to disconnect from NCCF disk services.

If the DSIPARM member or book name is not used by the subtask, TVBMEMNM may be used for other purposes, depending on how the MEM keyword of the TASK statement is specified. For example, you may decide to use this field as the DD name to be opened by the subtask, or to specify a default operator to receive messages.

Logging Messages

You may use the DSIWLS macro to write messages to the NCCF disk log from the subtask. See Chapter 3 for more information on DSIWLS. Hardcopy logging may not be started for user-written subtasks.

Issuing Messages

When a subtask starts, it should use the DSIMQS macro to send success or failure messages to the operator who started the subtask. The operator identifier is passed to the subtask in the TIBMSGNM field of DSITIB. If the subtask was started by the main task, TIBMSGNM contains zeros. Once the subtask is operating, further messages are sent to the operator specified by the subtask, which may vary depending on whom you want to receive the messages.

All messages must use standard NCCF buffer format, consisting of a properly initialized buffer header (BUFHDR) including the extension, followed by the message text. The buffer header is described in Chapter 3.

Receiving Messages

The subtask may be coded to receive messages from other subtasks (or from itself) that are sent using the DSIMQS macro.

All of the buffers transfered from one subtask to another with DSIMQS have the same buffer format. The buffer extension contains a queuing pointer field and the operator identifier (TVBOPID) of the sender, which may be used to return data or messages to the sender.

Your subtask may process buffers of any format. The value in the HDRMTYPE field of the buffer header indicates the format and contents of the buffer. When the HDRMTYPE field is set to the character "I" (HDRTYPEI), the buffer is an internal function request (IFR). For an IFR, the two bytes following the HDRMCEXT tell the format of the rest of the buffer.

IFRCODE=3 indicates that the remainder of the buffer is a command (see "Command Processing" later in this chapter.) IFRCODE=8 is reserved for you to define. (The only NCCF subtasks coded to handle IFRTYPE=8 are the OST and the NNT; the IFR is passed to the DSIEX13 exit routine for these two subtasks.)

Because the DSIMQS macro is used to transmit messages to many different subtasks, care must be taken when receiving message buffers. The DSIMQS macro will handle situations such as main line interruption, simultaneous processing in separate subtasks, and parallel processing in multiprocessor environments. To maintain reliability in situations such as these, the subtask should follow these rules when receiving messages:

- Wait until the TVBMECB (DISMQS event control block) has been posted
- Make sure that all previously received buffers have been processed.
- Set TVBMECB to zero.
- Use the assembler Compare and Swap instruction (CS) to obtain the queue of buffers from TVBMPUBQ and place zeros in TVBMPUBQ.
- Reverse the order of the queue to first-in-first-out (FIFO).
- Store the address of the queue of buffers in TVBMPRIQ.

In assembler language, this process would be (assume registers named A, B, and C):

```
         DSIWAT [ECB LIST INCLUDES TVBMECB]
         TM    TVBMECB,X'40'       IS THE DSIMQS ECB POSTED
         BNO   elsewhere           BRANCH IF NOT POSTED
         CLC   TVBMPRIQ,=F'0'      IS PRIVATE ANCHOR EMPTY
         BNZ   some-processing     BRANCH IF BUFFERS ARE STILL ANCHORED
*                                  RETURN TO 'REMOVE' WHEN TVBMPRIQ=0
REMOVE   XC    TVBMECB,TVBMECB     ZERO THE ECB
MSGDEQ   SLR   B,B                 ZERO SWAP REGISTER
         L     A,TVBMPUBQ          LOAD THE COMPARAND REGISTER
         CS    A,B,TVBMPUBQ        CS ZERO ONTO THE QUEUE
         BNE   MSGDEQ              RETRY IF TVBMPUBQ CHANGED
         USING BUFHDR,A            REG A NOW POINTS TO TOP BUFFER
REVERSE  L     C,HDRNEXTM          REVERSE...
         ST    B,HDRNEXTM            THE...
         LR    B,A                     QUEUE...
         LTR   A,C                       ORDER. TEST FOR END OF QUEUE.
         BNZ   REVERSE             BRANCH IF NOT END OF QUEUE
         ST    B,TVBMPRIQ          BASE FIFO QUEUE FROM PRIVATE ANCHOR
```

Freeing DSIMQS Buffers

Buffers transferred with DSIMQS are obtained with DSIGET Q=NO. When your subtask frees these buffers, use DSIFRE Q=NO. During subtask termination, after the TVBOPID field is set to blanks and TVBACTV is set to zeros, the TVBMPUBQ and TVBMPRIQ should be checked for buffers. If a queue is found, the buffers should be processed and freed.

Command Processing

In order to execute commands in your subtask, the command processor has to be designed to run in your optional subtask environment. NCCF command processors are designed to run under NCCF subtasks such as OST, NNT, PPT or DST. You may want to avoid coding a command processor by handling processing with a subroutine. If you want to define commands with CMDMDL statements so that you may use DSICES or DSIPAS macro instructions, the command processor must be defined as TYPE=D or TYPE=RD. Command lists, immediate commands, and regular commands may not be invoked within the subtask.

If an IFR type 3 is received through the DSIMQS macro, the buffer contains a command. To process the buffer, follow this procedure:

1. Add 2 to the HDRTDISP value, and subtract 2 from the HDRMLENG value. This moves the displacement past the IFR so that all commands appear the same to the command processor.
2. Issue the DSIGET macro to obtain a parse descriptor block (PDB), if necessary.
3. Issue the DSIPRS macro to parse the buffer
4. Issue the DSILCS macro to obtain a command work block (CWB)
5. Issue the DSICES macro to look up the command in the NCCF command table.

The command can now be called. When the command returns, issue the DSILCS macro to free the DSICWB, and the DSIFRE macro to free the PDB.

Control Block Considerations

The following control block fields are useful when writing a subtask.

Note: In the control block discussions that follow, the sequence of fields may not directly correspond to the field sequence in the actual DSECT. Appendix C contains the control block listings.

Main Vector Table (MVT)

The main vector table is the main control block for information throughout NCCF. There is one DSIMVT for each NCCF. From a subtask, the DSIMVT can be located through a pointer in the TVB (TVBMVT).

Field	Description
MVTCBH	Is a standard NCCF control block header
MVTDPRAD	(VSE only) Points to the NCCF VSE dispatcher (DSIDPR). This field should be referenced by NCCF service macros only.
MVTSVL	Contains the address of the service vector list (DSISVL) which contains the addresses of the NCCF service routines.

Field	Description
MVTTVB	Contains the address of the first TVB in the TVB chain.
MVTNCCFQ	(OS/VS only) Is the QNAME value for the ENQ and DEQ macros.
MVTTVBRN	(OS/VS only) Is the RNAME value for the ENQ and DEQ macros.
MVTCLOSE	Is a flag bit indicating that the CLOSE NORMAL command has been issued. When this bit is on, no more subtasks are attached and logons are not accepted.
MVTDRTRY	Shows the number of times an input/output operation is to be retried before it is considered a permanent error.
MVTMRC	Contains the MAXABEND definition statement which shows the number of times an operator station task (OST) may abnormally terminate (abend) and be reinstated.
MVTTCNT	Contains the number of TVBs in the TVB chain.
MVTMLGON	Contains the value from MAXLOGON definition statement which specifies the maximum number of times invalid logon information is processed before the session with that terminal ends.
MVTCDSES	Contains the value from the CDMNSESS definition statement which specifies the maximum number of OSTs in other domains that may have sessions at one time with this NCCF. This is the number of TVBs created for NCCF-to-NCCF tasks in the TVB chain.
MVTCURAP	Contains the value from the NCCFID definition statement DOMAINID operand, as follows:
MVTCUPAL	(1 byte) Shows the length of the NCCFID DOMAINID (1-5 characters).
MVTCURAN	(8 bytes) Contains the NCCFID DOMAINID padded with blanks.
MVTGMSG	Points to a buffer containing message DSI073A COMMAND PROCESSOR UNABLE TO BUILD RESPONSE MESSAGE.
MVTTOD	Shows the system time-of-day clock when NCCF was started
MVTUFLD	Is for customer definition and use.
MVTGFMG1	Points to a Write-to-Operator parameter list containing message DSI124I STORAGE REQUEST FAILED FOR NCCF. The message may be used by any WTO macro with MF=E. No additional storage is required. The routing code is (2,11); the descriptor code is 11.

Field	Description
MVTGFMG2	Points to a Write-to-Operator parameter list containing message DSI125I CRITICAL STORAGE SHORTAGE FOR NCCF. The message may be used by any WTO macro with MF=E. No additional storage is required. The routing code is (2,11); the descriptor code is 11.
MVTMETH	Indicates whether the access method is ACF/VTAM (V) or ACF/TCAM (T).
MVTTPROC	(ACF/TCAM only) Contains the value of the ACF/TCAM TPROCESS name (See MVTCURAN).

Task Vector Block (DSITVB)

The task vector block is used by NCCF to represent a subtask. When NCCF is started, one TVB is acquired for each subtask. The TVBs are chained together through the TVBNEXT field, and the beginning of the chain is pointed to by MVTTVB.

Field	Description
TVBCBH	Is a standard NCCF control block header. It contains a CBHTYPE byte used to indicate the subtask type, as follows: X'00' PPT X'01' NNT X'02' OST X'03' HCT X'04' TCT X'05' Optional subtask To distinguish between different types of optional subtasks, examine the TVBMODNM field.
TVBNEXT	Points to the next TVB on the TVB chain. The TVB chain is anchored from MVTTVB.
TVBTIB	Points to the TIB for the subtask.
TVBTCB	Contains the OS/VS task control block (TCB) address or the NCCF VSE pseudo-TCB address for the subtask.
TVBMVT	Points to the DSIMVT.
TVBTECB	Is the event control block (ECB) used to notify the subtask that shutdown is requested as soon as possible. This ECB should be included in every subtask ECB list. A subtask may use this ECB to cause itself to shutdown.
TVBMECB	Is the ECB used to notify the subtask that a message or a queue of messages has been sent using the DSIMQS macro.
TVBMPUBQ	Contains the queue of buffers containing the message sent to the subtask using the DSIMQS macro.
TVBMPRIQ	May be defined by the subtask.

The following bit fields are used by the subtask. Some of these flag bits are defined by the subtask; others are defined by the main task.

Field	Description
TVBIND1	
TVBTERM	1 indicates that normal subtask termination has occurred. The subtask has released all resources. This bit must be supported by the subtask. If the bit is set on by the main task before attaching the subtask, it indicates to the subtask that is has been attached for cleanup. The subtask is to release all resources and return control to the main task with this bit still set.
TVBIND2	
TVBVCLOS	May be defined by the subtask.
TVBIND3	
TVBACTV	1 indicates that the subtask is active. This bit is set by the subtask. While this bit is on, messages may be sent to the subtask using the DSIMQS macro.
TVBLGON	1 indicates that the subtask is starting.
TVBSHUTOFF	1 indicates that the subtask is shutting down upon request.
TVBRESET	1 indicates that regular commands should stop processing immediately. If your subtask does not run under a command processor, you may redefine this flag.
TVBRCVAI	This flag bit may be defined by the subtask. For an OST or NNT, 1 indicates that RECEIVE ANY for cross-domain sessions has been issued.
TVBINXIT	1 indicates that an IRB exit routine is running. This bit is required in VSE.
TVBTCODE	Used for problem analysis. When a subtask terminates, these fields may be set to indicates the reason for the termination, as follows:
TVBMTCOD	Indicates the module that decided to terminate the subtask (for values, see the DSITVB constants shown in Appendix C).
TVBPTCOD	Indicates that the subtask is about to terminate because TVBTECB was posted.
TVBNTCOD	Is a unique number that indicates where in each module the decision to terminate was made.
TVBHCUSE	May be defined by the subtask. For an HCT, this field is used to track how many subtasks are currently using the hardcopy subtask.
TVBLUNAM	Is the value specified in the TSKID operand of the TASK definition statement. This field is initialized before the subtask is attached.

Field	Description
TVBOPID	Is the unique subtask identifier. This name may be the same as TVBLUNAM. It is set up by the subtask when initialization is complete.
TVBUFLD	Is a user field that may be defined by the subtask.
TVBEXITQ	Is a queue of storage obtained in an IRB exit routine (DSIGET Q=YES,EXIT=YES).
TVBTASKQ	Is a queue of storage obtained under mainline processing (DSIGET Q=YES,EXIT=NO).
TVBMODNM	Is the name of the module to be attached as a subtask as specified in the MOD parameter of the TASK definition statement. This field may be used to determine the type of an optional subtask.
TVBMEMNM	Is initialized with the MEM parameter of the TASK definition statement. It may be the name of the member or B book of the DSIPARM data set that contains the initialization parameters for an optional subtask.

Task Information Block (DSITIB)

The task information block is used by NCCF to keep information about an attached subtask. DSITIB is acquired and freed by the main task. The fields described below are those of interest to an optional subtask:

Field	Description
TIBCBH	Is a standard NCCF control block header. The CBHTYPE field is the same as the CBHTYPE for TVB.
TIBTVB	Points Points to the DSITVB. The address of DSIMVT can be obtained from DSITVB; DSIMVT can be used to locate all other NCCF control blocks.
TIBACB	May be defined by the subtask. For NCCF subtasks, this field points to an ACF/VTAM ACB.
TIBEXLST	May be defined by the subtask. For NCCF subtasks, this field is used to locate the ACF/VTAM EXLST.
TIBELT	It is recommended that this field be used to point to the NCCF subtask ECB list.
TIBAPID	May be defined by the subtask. For NCCF subtasks, this field contains the ACF/VTAM application program name for the subtask.
TIBAPWD	May be defined by the subtask. For NCCF subtasks, this field contains the ACF/VTAM password.

Field	Description
TIBAREA1	May be defined by the subtask. For NCCF subtasks, this field is used to point to other control blocks such as CWBs, SWBs, or PDBs.
TIBUFLD	May be defined by the subtask. This field is not referenced or changed by NCCF.
TIBTIFFY	May be defined by the subtask. This field is used by OST and NNT only.
TIBOSEXT	May be used to point to an optional subtask extension to DSITIB. The optional subtask is responsible for freeing any storage pointed to by this area.
TIBOSLST	Is used by the LIST command processor to display the status of an optional subtask.
TIBXECB	May be defined by the subtask. NCCF uses this field as an ECB for cross-domain communication in OST and NNT.
TIBSAVES and TIBSAVEE	Are both 72-byte save areas for the subtask to use.
TIBNDATD and TIBEDATD	Are both 256-byte scratch areas for the subtask to use.
TIBMSGNM	Is the operator identifier of the subtask that issued the START TASK command. If the subtask was started automatically (INIT=YES), the field contains zeros.

Sample User-Written Subtask

The following is an example of a user-written subtask. This subtask is not executable as shown; it is provided as an example only.

```
*********************************************************************************************
*                                                                                           *
*MODULE NAME: SUBTASK                                                                       *
*                                                                                           *
*DESCRIPTIVE NAME: SKELETON NCCF SUBTASK MODULE                                             *
*                                                                                           *
*FUNCTION: TO DEMONSTRATE NCCF SUBTASK PROCEDURES.                                          *
*                                                                                           *
*REGISTER CONVENTIONS: SEE REGISTER EQUATES.                                                *
*                                                                                           *
*MODULE TYPE: MAIN PROGRAM, TO BE ATTACHED BY NCCF.                                         *
*  LANGUAGE: ASSEMBLER                                                                      *
*  MODULE SIZE: SEE ESD IN LISTING                                                          *
*  ATTRIBUTES: REENTRANT                                                                    *
*                                                                                           *
*ENTRY POINT: SUBTASK                                                                       *
*  PURPOSE: TO DO NCCF SUBTASK FUNCTION.                                                    *
*  LINKAGE:                                                                                 *
*    INPUT:                                                                                 *
*      REGISTERS: R1=TVB ADDRESS                                                            *
*                 R13=SAVE AREA ADDRESS                                                     *
*                 R14=RETURN ADDRESS                                                        *
*                 R15=ADDR OF ENTRY POINT 'SUBTASK'                                         *
*      OTHER: TVB FIELDS:                                                                   *
*               TVBTIB   : TIB ADDRESS                                                      *
*               TVBMVT   : MVT ADDRESS                                                      *
*               TVBTERM  : NORMAL/CLEANUP ATTACH FLAG                                       *
*               TVBLUNAM : SUBTASK RESOURCE NAME                                            *
*             TIB FIELDS:                                                                   *
*               TIBMSGNM : OPID OF STARTING OPERATOR OR ZERO                                *
*               TIBTVB   : POINTER TO TVB                                                   *
*             MVT FIELDS:                                                                   *
*               MVTTVB   : TVB CHAIN POINTER                                                *
*               MVTNCCFQ : NCCF ENQ/DEQ QNAME                                               *
*               MVTTVBRN : TVB CHAIN ENQ/DEQ RNAME                                          *
*               MVTSVL   : POINTER TO THE SVL (USED BY NCCF MACROS)                         *
*                                                                                           *
*EXIT NORMAL:                                                                               *
*  PURPOSE: NORMAL END OF SUBTASK.                                                          *
*  LINKAGE: RETURN TO CALLER                                                                *
*  OUTPUT:                                                                                  *
*    REGISTERS:                                                                             *
*      UNCHANGED REGISTERS: ALL REGISTERS EXCEPT R15                                        *
*      OUTPUT REGISTERS: R15 CONTAINS A RETURN CODE 0.                                      *
*    OTHER: TVB AND TIB ARE INTACT.                                                         *
*                                                                                           *
*EXIT ERROR: NONE                                                                           *
*                                                                                           *
*CONTROL BLOCKS:                                                                            *
*    NCCF CONTROL BLOCKS: DSICBH                                                            *
*                         DSIMVT                                                            *
*                         DSISVL                                                            *
*                         DSISWB                                                            *
*                         DSITIB                                                            *
*                         DSITVB                                                            *
*    INTERNAL CONTROL BLOCKS: DATD, THE TIB WORK AREA DSECT.                                *
*                                                                                           *
```

```
*MACROS ISSUED:                                                               *
*  SYSTEM MACROS: DEQ                                                         *
*                 ENQ                                                         *
*  NCCF MACROS: DSICBS                                                        *
*               DSIFRE                                                       *
*               DSILCS                                                       *
*               DSIMBS                                                       *
*               DSIMQS                                                       *
*               DSIWAT                                                       *
*                                                                            *
*MESSAGES ISSUED:                                                            *
*   DSI068I USER &1 ALREADY LOGGED ON.                                       *
*                                                                            *
******************************************************************************
```

```
SUBTASK  CSECT ,
         DSICBS DSITVB,DSITIB,DSIMVT,DSISVL,DEFER=ALL



*****************************************************************************************
*   TASKINIT:                                                                           *
*             FUNCTION IS TO HANDLE SUBTASK INITIALIZATION.                             *
*             PERFORM ENTRY LINKAGE.                                                    *
*             INITIALIZE THE ECB-S AND ECBLIST.                                         *
*             ISSUE DSILCS TO GET AN SWB.                                               *
*             IF UNSUCCESSFUL                                                           *
*               THEN                                                                    *
*                 SET THE TVB TERMINATION BIT.                                          *
*             ENDIF.                                                                    *
*             CLEAR THE ERROR MESSAGE FLAG.                                             *
*             IF THE SUBTASK IS NOT TERMINATING                                         *
*               THEN                                                                    *
*                 ISSUE ENQ TO LOCK THE TVB CHAIN.                                      *
*                 ISSUE DSILCS TO SCAN FOR THIS SUBTASK-S NAME.                         *
*                 IF THE NAME WAS FOUND                                                 *
*                   THEN (THERE IS A DUPLICATE)                                         *
*                     SET AN ERROR MESSAGE FLAG.                                        *
*                   ELSE (THE NAME IS UNIQUE)                                           *
*                     MOVE THE SUBTASK NAME TO THE OPERATOR ID FIELD.                   *
*                     SET THE ACTIVE SUBTASK FLAG.                                      *
*                 ENDIF (THE NAME WAS FOUND).                                           *
*                 ISSUE DEQ TO UNLOCK THE TVB CHAIN.                                    *
*               ELSE (THE SUBTASK IS TERMINATING, DO NOTHING).                          *
*             ENDIF (THE SUBTASK IS NOT TERMINATING).                                   *
*             IF THE ERROR MESSAGE FLAG WAS SET                                         *
*               THEN                                                                    *
*                 ISSUE DSIMBS TO BUILD MESSAGE DSI068I, USER IS                        *
*                     ALREADY LOGGED ON.                                                *
*                 ISSUE DSIMQS TO SEND THE MESSAGE TO THE OPERATOR                      *
*                     WHO STARTED THIS SUBTASK.                                         *
*                 IF THE DSIMQS FAILED                                                  *
*                   THEN                                                                *
*                     ISSUE DSIMQS TO SEND THE MESSAGE TO THE                           *
*                         AUTHORIZED RECEIVER.                                          *
*                   ELSE (THE MESSAGE WAS SUCCESSFULLY SENT).                           *
*                 ENDIF (DSIMQS FAILED).                                                *
*               ELSE (ERROR MESSAGE FLAG WAS NOT SET).                                  *
*             ENDIF (ERROR MESSAGE FLAG WAS SET).                                       *
*             END TASKINIT.                                                             *
*****************************************************************************************
```

```
*********************************************************************************************
*              ENTRY LINKAGE                                                                *
*********************************************************************************************
SUBTASK  CSECT ,
         USING *,R15
         B     PROLOG
         DC    C'SUBTASK &SYSDATE'
         DROP  R15
PROLOG   STM   R14,R12,12(R13)
         BALR  R12,0
PSTART   DS    0H
         USING PSTART,R12
         LR    TVBPTR,R01              BASE THE TVB
         USING DSITVB,TVBPTR
         L     TIBPTR,TVBTIB           SAVE INPUT PARAMETER
         USING DSITIB,TIBPTR           BASE THE TIB
         LA    R10,TIBNDATD            POINT TO NORMAL PROC WORK AREA
         USING DATD,R10                BASE SUBTASK WORK AREA
         LR    R14,R13                 SAVE CALLER-S REG 13
         LA    R13,TIBSAVES            POINT TO MY SAVEAREA
         ST    R14,SAVAREA+4(,R13)     POINT MINE TO CALLER-S
         ST    R13,SAVAREA+8(,R14)     POINT CALLER-S TO MINE
         ST    TVBPTR,SAVAREA(,R13)    TVB ADDR IN 1ST WORD OF MY S.A.

         L     MVTPTR,TVBMVT           BASE THE MVT
         USING DSIMVT,MVTPTR


*********************************************************************************************
*   INITITALIZE THE ECB LIST                                                                *
*********************************************************************************************
         SLR   R02,R02                 ZERO WORK REG
         ST    R02,TVBTECB             ZERO TERMINATION ECB
         ST    R02,TVBMECB             ZERO MESSAGE RECEIVER ECB
         ST    R02,USERECB             ZERO USER ECB

         LA    R02,TVBTECB             INITIALIZE...
         ST    R02,ECBLIST
         LA    R02,TVBMECB                  THE...
         ST    R02,ECBLIST+4
         LA    R02,USERECB                       ECBLIST
         O     R02,ENDOLIST            MARK AS END OF ECBLIST
         ST    R02,ECBLIST+8
```

```
*********************************************************************************
*    GET AND INITIALIZE AN SWB                                                  *
*********************************************************************************

         DSILCS CBADDR=TIBNPSWB,SWB=GET   GET AN SWB

         LTR   R15,R15                  WAS ONE GOTTEN?
         BNZ   NOSWB                    NO, BRING DOWN THE SUBTASK
         L     R02,TIBNPSWB             SET THE TIB ADDRESS IN THE SWB
         USING DSISWB,R02               BASE THE SWB
         ST    TIBPTR,SWBTIB            STORE THE TIB ADDRESS
         DROP  R02                      DROP SWB COVER
         B     ENDSWB                   CONTINUE

NOSWB    OI    TVBIND1,TVBTERM          TERMINATE THE SUBTASK

ENDSWB   DS    0H                       END OF SWB INITIALIZATION


*********************************************************************************
*   IF INITIALIZATION IS SUCCESSFUL, MARK THE SUBTASK ACTIVE                    *
*********************************************************************************

         TM    TVBIND1,TVBTERM          IF THE SUBTASK IS NOT TERMINATING
         BNZ   ENDINIT                  BRANCH IF TERMINATING


*------> LOCK THE TVB CHAIN WHILE ADDING THIS SUBTASK ID
         MVC   ENQWORK(ENQLN),ENQLIST   MOVE LIST FORM TO WORK AREA
         ENQ   (MVTNCCFQ,MVTTVBRN,E,18,STEP),MF=(E,ENQWORK)

         SLR   ERMSGNO,ERMSGNO          CLEAR THE ERROR MSG NUMBER
         DSILCS OPID=TVBLUNAM,          SEARCH FOR THIS SUBTASK-S NAME   *
               TVB=MVTTVB,              STARTING AT THE TOP OF THE CHAIN *
               CBADDR=DUPTVB            PUT THE ADDRESS HERE

         LTR   R15,R15                  IF A TVB WAS FOUND, THAT IS BAD
         BNZ   UNIQUE                   BRANCH IF UNIQUE

DUPNAME  DS    0H                       OTHERWISE
         LA    ERMSGNO,68               USER ALREADY USING THIS NAME
         B     UNLOCK                   UNLOCK TVB CHAIN AND EXIT

UNIQUE   DS    0H
         MVC   TVBOPID,TVBLUNAM         PUT THE USERID IN TVB
         OI    TVBIND3,TVBACTV          MARK TVB AS ACTIVE
UNLOCK   DS    0H

*------> UNLOCK THE TVB CHAIN
         MVC   ENQWORK(DEQLN),DEQLIST   COPY THE DEQ PARMLIST
         DEQ   (MVTNCCFQ,MVTTVBRN,18,STEP),MF=(E,ENQWORK)
```

```
*------> IF AN ERROR WAS DETECTED WHILE LOCKED, PUT A MESSAGE OUT NOW
         LTR   ERMSGNO,ERMSGNO        WAS AN ERROR DETECTED?
         BZ    ENDINIT                NO, CONTINUE
         OI    TVBIND1,TVBTERM        TERMINATE THE SUBTASK
         LA    R02,BUFFER             POINT TO ERROR MSG BUFFER
         USING BUFHDR,R02             SET TEMPORARY BASE
         LA    R14,L'BUFFER           GET THE BUFFER LENGTH
         STH   R14,HDRBLENG           SET THE BUFFER LENGTH
         MVI   HDRMTYPE,HDRTYPEU      SET BUFFER TYPE = USER MSG
         LA    R14,BUFHDRND-BUFHDR    GET OFFSET TO TEXT
         STH   R14,HDRTDISP           SET TEXT DISPLACEMENT

         DSIMBS MID=068,BFR=(R02),SWB=TIBNPSWB   BUILD MSG DSI068I

         DSIMQS TASKID=TIBMSGNM,BFR=(R02),SWB=TIBNPSWB   SEND TO OPER

         LTR   R15,R15                WAS THE STARTING OPERATOR THERE?
         BZ    ENDINIT                YES, CONTINUE TERMINATING

         DSIMQS AUTHRCV=YES,BFR=(R02),SWB=TIBNPSWB  SEND IT TO AUTHRCV

         DROP  R02                    DROP BUFHDR COVER
ENDINIT  DS    0H
```

```
*********************************************************************************
*   ECBPROCR:                                                                   *
*             FUNCTION IS TO WAIT ON THE ECB-S AND SERVICE THOSE                *
*                THAT ARE POSTED.                                               *
*             DO WHILE THE TVB TERMINATION BIT IS NOT SET.                      *
*               ISSUE DSIWAT TO WAIT ON THE ECB LIST.                           *
*               IF THE TVB TERMINATION ECB IS POSTED                            *
*                 THEN                                                          *
*                   SET THE TVB TERMINATION BIT.                                *
*               ENDIF (TERMINATION ECB POSTED).                                 *
*               IF THE DSIMQS MESSAGE ECB IS POSTED                             *
*                 THEN                                                          *
*                   CALL MSGPROCR TO INSURE THE TVB PRIVATE QUEUE               *
*                       IS CLEAR.                                               *
*                   CLEAR THE TVB MESSAGE ECB.                                  *
*                   DEQUEUE THE MESSAGE BUFFER QUEUE FROM THE                   *
*                       TVB MESSAGE PUBLIC QUEUE WITH COMPARE                   *
*                       AND SWAP.                                               *
*                   REVERSE THE ORDER OF THE DEQUEUED CHAIN OF                  *
*                       BUFFERS FROM LIFO TO FIFO.                              *
*                   ANCHOR THE FIFO QUEUE ON THE TVB PRIVATE                    *
*                       MESSAGE QUEUE.                                          *
*                   CALL MSGPROCR TO PROCESS THE PRIVATE QUEUE.                 *
*               ENDIF (THE MESSAGE ECB IS POSTED).                              *
*               IF THE USER ECB IS POSTED                                       *
*                 THEN                                                          *
*                   DO USER PROCESSING.                                         *
*               ENDIF (THE USER ECB IS POSTED).                                 *
*             ENDDO (WHILE TVB TERMINATION BIT IS NOT SET).                     *
*             END ECBPROCR.                                                     *
*********************************************************************************
```

```
ECBPROCR DS    0H
         B     LOOPTEST              LOOP UNTIL SHUTDOWN

LOOPTOP  DS    0H
         DSIWAT ECBLIST=ECBLIST      WAIT FOR SOMETHING TO DO

*********************************************************************************
*    DETERMINE WHICH ECB WAS POSTED.                                            *
*********************************************************************************

*********************************************************************************
*    TEST THE TERMINATION ECB IN THE TVB                                        *
*********************************************************************************
TESTTERM TM    TVBTECB,TIBECBPO      IF TERMINATION IS POSTED
         BNO   TESTMQS               NO, TEST FOR MESSAGE RECEIVED
         OI    TVBIND1,TVBTERM       TERMINATE THE SUBTASK
         B     LOOPTEST              GO TO THE BOTTOM OF THE LOOP
```

```
*****************************************************************************
*     TEST THE DSIMQS MESSAGE RECEIVED ECB IN THE TVB                       *
*****************************************************************************
TESTMQS  TM     TVBMECB,TIBECBPO        IF DSIMQS INPUT ECB IS POSTED
         BNO    TESTUSER                NO, SEE IF USER ECB IS POSTED
         BAL    R14,MSGPROCR            CALL MSGPROCR
REMOVE   XC     TVBMECB,TVBMECB         ZERO THE ECB
MSGDEQ   SLR    R00,R00                 ZERO SWAP REGISTER
         L      R02,TVBMPUBQ            LOAD THE COMPARAND REGISTER
         CS     R02,R00,TVBMPUBQ        CS ZERO ONTO THE QUEUE
         BNE    MSGDEQ                  RETRY IF TVBMPUBQ CHANGED
         USING  BUFHDR,R02              REG R02 NOW POINTS TO TOP BUFFER
REVERSE  L      R01,HDRNEXTM            REVERSE...
         ST     R00,HDRNEXTM             THE...
         LR     R00,R02                   QUEUE...
         LTR    R02,R01                    ORDER. TEST FOR END OF QUEUE.
         BNZ    REVERSE                 BRANCH IF NOT END OF QUEUE
         ST     R00,TVBMPRIQ            POINT PRIVATE ANCHOR AT FIFO QUE
         DROP   R02                     DROP BUFHDR COVER
         BAL    R14,MSGPROCR            CALL MSGPROCR TO CLEAN QUEUE


*****************************************************************************
*     TEST THE USER ECB                                                     *
*****************************************************************************
TESTUSER TM     USERECB,TIBECBPO        IF USERECB IS POSTED
         BNO    LOOPTEST                NO, RESTART LOOP
         XC     USERECB,USERECB         CLEAR THE POSTED ECB


*****************************************************************************
         NOP    0                       *     ADD USER CODE HERE          *
*****************************************************************************


*****************************************************************************
*     TEST FOR TERMINATION REQUESTED                                        *
*****************************************************************************
LOOPTEST TM     TVBIND1,TVBTERM         IS THE SUBTASK TERMINATING?
         BZ     LOOPTOP                 NO, LOOP TO TOP AND WAIT
```

```
*********************************************************************************
*   TASKTERM:                                                                   *
*             FUNCTION IS TO CLEAN UP AND PREPARE TO RETURN TO THE              *
*                  MAIN TASK.                                                   *
*             LOCK THE TVB CHAIN.                                               *
*             BLANK THE TVB USERID FIELD AND ACTIVE SUBTASK FLAG.               *
*             UNLOCK THE TVB CHAIN.                                             *
*             CALL MSGPROCR ROUTINE TO PURGE MSGS ON THE                        *
*                 PRIVATE QUEUE.                                                *
*             DEQUEUE ALL MESSAGES ON THE PUBLIC MESSAGE QUEUE                  *
*                  WITH COMPARE AND SWAP.                                       *
*             ANCHOR THE MESSAGES ON THE PRIVATE BUFFER QUEUE.                  *
*             CALL MSGPROCR ROUTINE TO PURGE MSGS ON THE                        *
*                 PRIVATE QUEUE.                                                *
*             ISSUE DSIFRE FOR ALL EXIT QUEUED STORAGE.                         *
*             ISSUE DSIFRE FOR ALL TASK QUEUED STORAGE.                         *
*             SET THE RETURN CODE FOR THE MAIN TASK INTO REG 15.                *
*             END TASKTERM.                                                     *
*********************************************************************************


*-------> LOCK THE TVBCHAIN
          MVC   ENQWORK(ENQLN),ENQLIST
          ENQ   (MVTNCCFQ,MVTTVBRN,E,18,STEP),MF=(E,ENQWORK)

          MVC   TVBOPID,BLANKS          BLANK OUT USERID
          NI    TVBIND3,TVBACTV         INDICATE TVB NOT ACTIVE

*-------> UNLOCK THE TVBCHAIN
          MVC   ENQWORK(DEQLN),DEQLIST
          DEQ   (MVTNCCFQ,MVTTVBRN,18,STEP),MF=(E,ENQWORK)

          BAL   R14,MSGPROCR            CALL MSGPROCR TO CLEAN QUEUE
TERMMSG   SLR   R00,R00                 CLEAR REG FOR COMP & SWAP
          L     R02,TVBMPUBQ
          CS    R02,R00,TVBMPUBQ        DEQ ALL MSG BUFFERS
          BNE   TERMMSG                 IF SOMETHING CHANGED, RETRY IT
          ST    R02,TVBMPRIQ            SAVE THE BUFFER STRING.
          BAL   R14,MSGPROCR            CALL MSGPROCR TO CLEAN QUEUE



          OC    TIBNPSWB,TIBNPSWB       IF AN SWB WAS GOTTEN
          BZ    FREEQSTG
          DSILCS CBADDR=TIBNPSWB,SWB=FREE  FREE THE SWB

          XC    TIBNPSWB,TIBNPSWB       CLEAR THE SWB POINTER
```

```
*********************************************************************************************
*        FREE STORAGE GOTTEN WITH DSIGET Q=YES                                              *
*********************************************************************************************

*------>  FREE ALL EXIT QUEUED STORAGE
FREEQSTG DS    0H
         B     FREEXIT1                 TEST FOR ZERO QUEUE
FREEXIT  DS    0H                       QUEUE NOT ZERO
         LA    R02,16                   BLOCK ADDRESS STARTS AT 16 PAST
         AL    R02,TVBEXITQ                THE QUEUE ANCHOR VALUE
         DSIFRE A=(R02),                FREE BLOCK ADDRESSED BY R02      *
               LV=0,                    DSIFRE KNOWS THE LENGTH          *
               SP=0,                    DSIFRE KNOWS THE SUBPOOL         *
               EXIT=YES,                STORAGE GOTTEN IN AN IRB EXIT    *
               TASKA=(TVBPTR),          TVB POINTER                      *
               LISTA=FRELST,            DSIFRE WORKAREA                  *
               Q=YES                    FREE QUEUED STORAGE

FREEXIT1 DS    0H
         OC    TVBEXITQ,TVBEXITQ        IS THERE ANYTHING ON THE QUEUE?
         BNZ   FREEXIT                  YES, LOOP UNTIL ZERO



*------>  FREE ALL TASK QUEUED STORAGE
         B     FREETSK1                 TEST FOR ZERO QUEUE
FREETSK  DS    0H                       QUEUE NOT ZERO
         LA    R02,16                   BLOCK ADDRESS STARTS AT 16 PAST
         AL    R02,TVBTASKQ                THE QUEUE ANCHOR VALUE
         DSIFRE A=(R02),                FREE BLOCK ADDRESSED BY R02      *
               LV=0,                    DSIFRE KNOWS THE LENGTH          *
               SP=0,                    DSIFRE KNOWS THE SUBPOOL         *
               EXIT=NO,                 STORAGE GOTTEN IN MAIN LINE CODE *
               TASKA=(TVBPTR),          TVB POINTER                      *
               LISTA=FRELST,            DSIFRE WORKAREA                  *
               Q=YES                    FREE QUEUED STORAGE

FREETSK1 DS    0H
         OC    TVBTASKQ,TVBTASKQ        IS THERE ANYTHING ON THE QUEUE?
         BNE   FREETSK                  YES, LOOP UNTIL ZERO



*********************************************************************************************
*        EXIT THE SUBTASK                                                                   *
*********************************************************************************************

         L     R13,SAVEAREA+4(,R13)     RESTORE SAVEAREA ADDR
         SR    R15,R15                  SET A ZERO RETURN CODE
         L     R14,12(,R13)             RETURN TO THE OPERATING SYSTEM
         LM    R00,R12,20(R13)
         BR    R14
```

```
*****************************************************************************
*   MSGPROCR: SUBROUTINE.                                                   *
*             FUNCTION IS TO PROCESS MESSAGE BUFFERS SENT VIA DSIMQS        *
*                 AND FREEMAIN THE BUFFERS.                                 *
*             SAVE THE CALLER-S REGISTERS.                                  *
*             DO WHILE THERE ARE MESSAGES ON THE PRIVATE QUEUE.             *
*               IF THE SUBTASK IS NOT TERMINATING                           *
*                 THEN                                                      *
*                   DO YOUR THING WITH THE BUFFER HERE.                     *
*                 ELSE (TERMINATING, DO NOT PROCESS).                       *
*               ENDIF.                                                      *
*               ISSUE DSIFRE TO FREEMAIN THE MESSAGE BUFFER.                *
*             ENDDO (DO WHILE MESSAGES EXIST).                              *
*             RESTORE THE CALLER-S REGISTERS.                               *
*             RETURN TO CALLER.                                             *
*             END MSGPROCR.                                                 *
*****************************************************************************


MSGPROCR DS    0H
         STM   R14,R01,SUBRSAVE       SAVE REGS

*-------> DO WHILE THERE ARE QUEUED MSGS
         B     MSGLOOP1
MSGLOOP  DS    0H
         TM    TVBIND1,TVBTERM        IS THE SUBTASK TERMINATING?
         BO    MSGFREE                YES, ONLY FREE THE BUFFER

*****************************************************************************
*   IF YOU WANT TO DO SOMETHING WITH THE MESSAGE BUFFER, DO IT HERE!  *
*   NOTE: THE TOP BUFFER ON THE QUEUE IS LOCATED BY TVBMPRIQ.         *
*****************************************************************************

MSGFREE  L     R02,TVBMPRIQ           POINT TO THE FIRST BUFFER
         L     R08,HDRNEXTM-BUFHDR(,R02)  ANCHOR THE NEXT MESSAGE
         ST    R08,TVBMPRIQ
         LH    R09,HDRBLENG-BUFHDR(,R02) GET THE BUFFER LENGTH
         DSIFRE R,A=(R02),            FREE BUFFER SENT VIA DSIMQS      *
               LV=(R09),              LENGTH FROM BUFHDR IS IN R09     *
               SP=0                   DSIMQS BUFFERS ARE IN SUBPOOL 0

MSGLOOP1 OC    TVBMPRIQ,TVBMPRIQ      IS ANYTHING LEFT ON THE QUEUE
         BNZ   MSGLOOP                YES, LOOP UNTIL DONE



         LM    R14,R01,SUBRSAVE       RESTORE REGS
         BR    R14                    RETURN TO CALLER

*-------> END   MSGPROCR
```

```
*********************************************************************************************
*                                  CONSTANTS                                                *
*********************************************************************************************
          DS    0F
ENDOLIST  DC    XL4'80000000'           END OF ECBLIST FLAG




BLANKS    DC    CL8'                    8 BLANKS
ENQLIST   ENQ   (,,,,),MF=L             LIST FORM ENQ
ENQLN     EQU   *-ENQLIST               LENGTH OF LIST FORM ENQ
DEQLIST   DEQ   (,,,),MF=L              LIST FORM DEQ
DEQLN     EQU   *-DEQLIST               LENGTH OF LIST FORM DEQ



SAVAREA   EQU   0                       OFFSET TO SAVEAREA



*********************************************************************************************
*                  REGISTER EQUATES                                                         *
*********************************************************************************************
R00       EQU   00                       EQUATES FOR REGISTERS 0-15
R01       EQU   01
R02       EQU   02
R03       EQU   03
R04       EQU   04
R05       EQU   05
R06       EQU   06
R07       EQU   07
R08       EQU   08
R09       EQU   09
R10       EQU   10
R11       EQU   11
R12       EQU   12
R13       EQU   13
R14       EQU   14
R15       EQU   15
TVBPTR    EQU   R03
MVTPTR    EQU   R04
TIBPTR    EQU   R05
ERMSGNO   EQU   R06
```

```
*********************************************************************************************
*                TIB WORKAREA OVERLAY                                                       *
*********************************************************************************************
DATD      DSECT
USERECB   DS    F                       USER ECB (USE NOT SHOWN HERE)
ECBLIST   DS    3A                      ECB LIST FOR THIS SUBTASK
BUFFER    DS    0CL132                  ERROR MESSAGE BUFFER
ENQWORK   DS    CL12                    ENQ/DEQ WORK AREA (WORD BOUNDARY)
FRELST    DS    CL14                    DSIFRE WORK AREA  (WORD BOUNDARY)
DUPTVB    DS    A                       WORD FOR TVB SEARCH
SUBRSAVE  DS    4A                      MSGPROCR SAVEAREA FOR R14-R01


*********************************************************************************************
*                INCLUDE NCCF CONTROL BLOCK DSECTS                                          *
*********************************************************************************************

SUBTASK   CSECT
          DSICBS DEFER=INCLUDE
SUBTASK   CSECT
          END    SUBTASK
```

Appendix A. Command Summary

This appendix summarizes all of the commands related to NCCF. The first chart shows commands related to NCCF under ACF/VTAM and ACF/VTAME. The second chart shows commands related to NCCF under ACF/TCAM. Commands are listed in alphabetical order.

The second column indicates whether the command is regular (R), immediate (I), both regular and immediate (B), regular and data services (RD), or not applicable for the command type(N).

The third column indicates whether or not the command can be restricted by span of control.

The next five columns indicate the places from which the command may be issued. The column headings are as follows:

Terminal.	The operator may issue the command at the physical operator station.
Command list.	The command may be used in a command list.
System console.	The command may be entered from the system console to NCCF.
Command processor.	The command may be issued by a command processor to NCCF.
PPT	The command may be executed under the primary POI task.

The numbers in the columns refer to the notes that follow the table.

ACF/VTAM and NCCF Command Summary

Command	Type	SPAN Applies	Issued From				
			Terminal	Command List	System Console	Command Processor	PPT
AGAIN	1	NO	YES	NO	NO	NO	NO
AT	R	2	YES	YES	NO	NO	YES
AUTOWRAP	B	NO	YES	YES	NO	NO	NO
BGNSESS*	R	NO	YES	YES	NO	YES	NO
CANCEL	B	NO	YES	YES	NO	NO	NO
CLEAR key	B	NO	YES	YES	NO	NO	NO
clistname	R	2	YES	YES	NO	YES	YES
CLOSE	B	NO	YES	YES	YES	YES	NO
DISPLAY (ACF/VTAM)	R	YES	YES	YES	3	YES	YES
ENDSESS*	R	NO	YES	YES	NO	YES	NO
EVERY	R	2	YES	YES	NO	NO	YES
GO	B	NO	YES	YES	NO	NO	NO
HALT (ACF/VTAM)	N	NO	NO	NO	3	NO	NO
INPUT	R	NO	YES	YES	NO	NO	YES
LIST	R	NO	YES	YES	NO	YES	NO
LISTSESS*	R	NO	YES	YES	NO	YES	NO
LOGOFF	R	NO	YES	NO	NO	NO	NO
LOGOFF (ACF/VTAM)	N	NO	6	NO	NO	NO	NO
LOGON (ACF/VTAM)	N	NO	7	NO	NO	NO	NO
MODIFY (ACF/VTAM)	R	YES	YES	YES	3	YES	NO
MOVE	R	NO	YES	YES	NO	YES	NO
MSG	R	NO	YES	YES	YES	YES	YES
no data enter	B	NO	YES	YES	NO	NO	NO
PURGE	R	2	YES	YES	YES	NO	YES
REPLY (ACF/VTAM)	R	NO	YES	YES	8	YES	NO
RESET	B	NO	YES	YES	NO	YES	NO
ROUTE	R	NO	YES	YES	NO	YES	NO
RTRNSESS*	R	NO	YES	YES	NO	YES	NO
SENDSESS*	R	NO	YES	YES	NO	YES	NO
START (ACF/VTAM)	N	NO	NO	NO	3	NO	NO
START	R	9	YES	YES	NO	YES	NO
STOP	R	9	YES	YES	NO	YES	NO
SWITCH	RD	NO	YES	YES	NO	YES	NO
user command	11	NO	12	12	NO	YES	12
VARY (ACF/VTAM)	R	YES	YES	YES	3	YES	YES

*Applies only to the Terminal Access Facility.

ACF/TCAM and NCCF Command Summary

Command	Type	SPAN Applies	Issued From				
			Terminal	Command List	System Console	Command Processor	PPT
AGAIN	1	NO	YES	NO	NO	NO	NO
ALTER (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
AT	R	NO	YES	YES	NO	NO	YES
AUTO (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
AUTOWRAP	B	NO	YES	YES	NO	NO	NO
BGNSESS**	R	NO	YES	YES	NO	YES	NO
CANCEL	B	NO	YES	YES	NO	NO	NO
CLEAR key	B	NO	YES	YES	NO	NO	NO
clistname	R	NO	YES	YES	NO	YES	YES
CLOSE	B	NO	YES	YES	YES	YES	NO
CLOSE (ACF/TCAM)	R	NO	YES	YES	3	YES	YES
COM (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
COMMAND (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
CONTACT (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
DATA (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
DISPLAY (ACF/TCAM)	R	NO	YES	YES	3	YES	YES
ENDSESS**	R	NO	YES	YES	NO	YES	NO
EVERY	R	NO	YES	YES	NO	NO	YES
GO	B	NO	YES	NO	NO	NO	NO
HALT (ACF/TCAM)	N	NO	4	4	3,4	4	NO
HOLD (ACF/TCAM)	R	NO	YES	YES	3	YES	YES
INIT	R	NO	YES	YES	NO	YES	NO
INITS (ACF/TCAM)	N	NO	5	NO	NO	NO	NO
INPUT	R	NO	YES	YES	NO	NO	YES
LIST	R	NO	YES	YES	NO	YES	NO
LISTSESS**	R	NO	YES	YES	NO	YES	NO
LOGOFF	R	NO	YES	NO	NO	NO	NO
MANUAL (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
MODIFY (ACF/TCAM)	R	NO	YES	YES	3	YES	NO
MOVE	R	NO	YES	YES	NO	YES	NO
MSG	R	NO	YES	YES	YES	YES	YES
NET (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
NETWORK (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
no data enter	I	NO	YES	NO	NO	NO	NO
OFFLN (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
ONLN (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
PAUSE	R	NO	NO	YES	NO	YES	NO
PIUT (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
PURGE	R	NO	YES	YES	YES	NO	YES
QUEUE (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
RELEASE (ACF/TCAM)	R	NO	YES	YES	3	YES	YES
RESEND (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
RESET	B	NO	YES	YES	NO	YES	NO
ROUTE	R	NO	YES	YES	NO	YES	NO
RTRNSESS**	R	NO	YES	YES	NO	YES	NO
SEND (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
SENDSESS**	R	NO	YES	YES	NO	YES	NO
SETSQ (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
SHUTDOWN	R	NO	YES	YES	NO	YES	NO
START	R	NO	YES	YES	NO	YES	NO
START (ACF/TCAM)	R	NO	YES	YES	3	YES	YES
STOP	R	NO	YES	YES	NO	YES	NO
STOP (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
SWITCH	RD	NO	YES	YES	NO	YES	NO
TERM	R	NO	YES	YES	NO	YES	NO
TERMS (ACF/TCAM)*	N	NO	10	NO	NO	NO	NO
UNITS (ACF/TCAM)*	R	NO	YES	YES	3	YES	YES
user command	11	NO	12	12	NO	YES	12
VARY (ACF/TCAM)	R	NO	YES	YES	3	YES	YES

*Applies to extended operator control.
**Applies only to the Terminal Access Facility.

Notes:

1. *Only regular and RD commands may be repeated.*
2. *The generated commands are checked; the original command is not.*
3. *Goes directly to the access method, not to NCCF.*
4. *Terminates ACF/TCAM, which causes NCCF to abend.*
5. *Establishes an SNA session between the terminal and the device message handler (ACF/TCAM).*
6. *Data is entered from the terminal, but is received by the access method and becomes available to NCCF only through the facilities of the LOGON exit routine (ACF/VTAM).*
7. *Data is entered from the terminal but is received by ACF/VTAM and results in entry to the NCCF LOSTERM exit routine.*
8. *Goes either directly to ACF/VTAM or to NCCF.*
9. *Applies to START | STOP { RESOURCE / SPAN } only.*
10. *Breaks the SNA session between the SNA device and the device message handler (ACF/TCAM).*
11. *Defined by user as R, I, B, D, or RD.*
12. *Depends on user definition.*

Appendix B. NCCF Log and Hard-Copy Log

NCCF Log

NCCF provides a means of recording on a disk all messages and commands that are received or sent. The write-log routine (DSIWLM) records the information in the order that is received.

Two VSAM disk log data sets may be defined, a required primary data set and an optional secondary data set. The NCCF disk log is maintained on the primary VSAM file, which is opened for output at initialization. The user must have INIT=Y in the DSTINIT statement for DSILOG, or must issue START TASK=DSILOG to initiate logging. When the end of the primary file is reached, logging is automatically switched to a secondary file, if one has been defined. The user can also control switching from one disk log data set to the other by means of the SWITCH command. (See *NCCF Terminal Use* for more details on this action.) When logging is switched, the primary file is closed and may be printed in batch mode while NCCF continues logging on the secondary file. The file may be printed using the NCCF utility program (DSIPRT), system utilities, or user-written programs. If the end of the secondary file is reached before the primary file is printed, logging stops.

If I/O errors occur, logging is terminated and the authorized operator is notified. If the user is currently logging on the primary data set, and the secondary is specified, NCCF automatically switches to the secondary data set. If the user is logging on the secondary data set, or no secondary is specified, the data set is closed and logging terminates. During NCCF execution, if storage is not available for logging routines, records are not sent to the log. The authorized operator is notified of the storage shortage.

For information on how to format and print the NCCF log, see Appendix B of *NCCF Installation.*

Figure B-1 shows the format of each disk record in the log. An example of the printed NCCF log is shown in Figure B-2.

NCCF Hard-Copy Log

The NCCF hard-copy log is printed by the hard-copy log task (DSIHCT). Entries are printed in the sequence that the hard-copy task receives them. Each entry is preceded by the task identification associated with the message (the operator identification for operator station tasks). Figure B-3 shows an example of the hard-copy log.

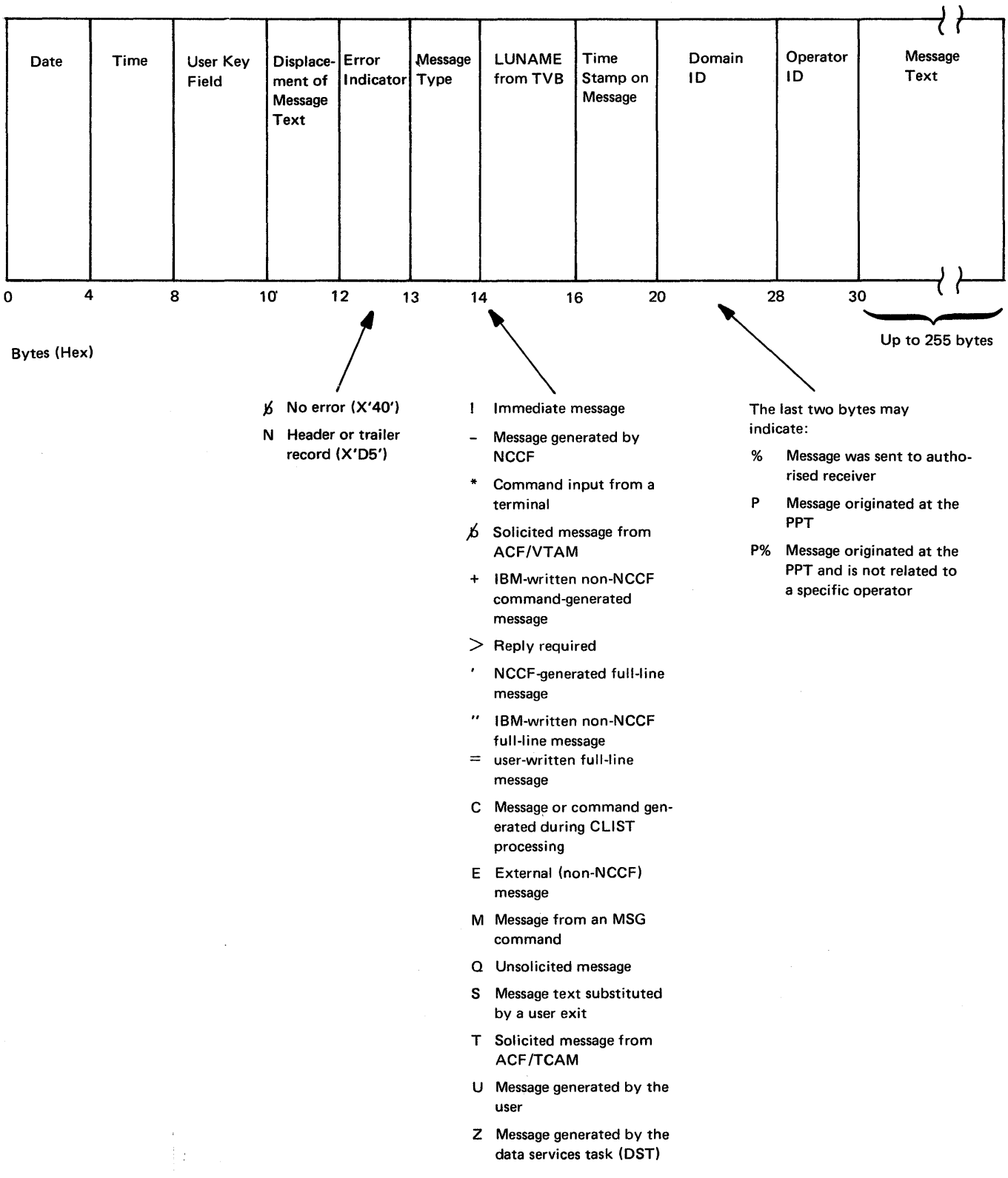


Figure B-1. NCCF Log File Record Format

```
          NETWORK COMMUNICATIONS CONTROL FACILITY PRINT LOG UTILITY                    09/18/80              1

        09/18/80 PARIS     N 10:37:29   NETWORK COMMUNICATIONS CONTROL FACILITY DISK LOC
FRED                         10:37:31 - DSI546I DSILOG  : PRIMARY VSAM DATA SET IS NOW ACTIVE
                         %   10:37:46 Z DSI556I DSILOG  : VSAM DATASET 'CLOSE' COMPLETED, DDNAME = 'NCCFSEC ' RETURN
                                        CODE = '00', ACB ERROR FIELD = '00'
                             10:37:49 * LOGOFF
                             10:37:51 - DSI081I OPERATOR FRED     LOGGED OFF TERMINAL L3270D
                             10:38:51 - DSI020I OPERATOR FRED     LOGGED ON FROM TERMINAL L3270D   USING
                                        PROFILE(VTAMSTAR),HCL(        )
                         %   10:38:52 - DSI020I OPERATOR FRED     LOGGED ON FROM TERMINAL L3270D   USING
                                        PROFILE(VTAMSTAR),HCL(        )
                             10:39:13 * AUTOWRAP
                             10:39:14 ! DSI082I AUTOWRAP STARTED
                             10:40:04 * AT 10:41,ID=FRED1,MSG FRED,PLEASE START CROSS DOMAIN SESSION TO DOMAIN DSIMN.
                             10:40:10 - DSI201I TIMER REQUEST SCHEDULED FOR EXECUTION - ID=FRED1
                             10:40:45 * MSG ALL,THIS IS OPERATOR FRED LOGGED ON.

                             10:40:46 M DSI039I MSG FROM FRED    : THIS IS OPERATOR FRED LOGGED ON.
                             10:40:48 - DSI001I MESSAGE SENT TO ALL
                             10:40:48 M DSI039I MSG FROM FRED    : THIS IS OPERATOR FRED LOGGED ON.
                             10:40:57 * L S=OPS
                             10:41:01 - OPERATOR: FRED     TERM: L3270D   STATUS: ACTIVE
                             10:41:01 - END OF STATUS DISPLAY
                             10:41:00 - DSI208I TIME EXPIRATION - ID= FRED1    - CMD= MSG FRED,PLEASE START CROSS DOMAIN
                                         SESSION TO DOMAIN DSIMN.
                             10:41:01 - DSI001I MESSAGE SENT TO FRED
                             10:41:01 M DSI039I MSG FROM FRED    : PLEASE START CROSS DOMAIN SESSION TO DOMAIN DSIMN.
                             10:41:19 * START DOMAIN=DSIMN
                             10:41:21 - DSI033I DSIMN    SESSION STARTING FOR FRED
                   DSIMN     10:42:50 - DSI809A PLEASE ROUTE OPID,PASSWORD,PROFILE,HARDCOPY,INITIAL CMD
                             10:47:55 - DSI810I NCCF ACF/VTAM READY
                             10:48:03 - DSI020I OPERATOR FRED     LOGGED ON FROM TERMINAL PARIS000 USING
                                        PROFILE(VTAMPLUS),HCL(        )
                   PARIS     10:49:44 * EVERY 5,PPT,LIST STATUS=OPS
                             10:49:48 - DSI201I TIMER REQUEST SCHEDULED FOR EXECUTION - ID=SYS00001
                             10:50:27 * L TIMER=ALL
                         P   10:50:34 - DISPLAY OF OUTSTANDING TIMER REQUESTS
                         P   10:50:34 - 000  TIMER ELEMENT(S) FOUND FOR FRED
                         P   10:50:34 - END OF DISPLAY
                             10:51:23 * AT 10:55,ID=FRED2,DISPLAY NET,ID=FPU3276
                             10:51:25 - DSI201I TIMER REQUEST SCHEDULED FOR EXECUTION - ID=FRED2
                             10:51:32 * L TIMER=ALL
                         P   10:51:34 - DISPLAY OF OUTSTANDING TIMER REQUESTS
                         P   10:51:34 - TYPE: AT      TIME: 10:55   ID: FRED2
                         P   10:51:34 -   COMMAND: DISPLAY NET,ID=FPU3276
                         P   10:51:34 - 001  TIMER ELEMENT(S) FOUND FOR FRED
                         P   10:51:34 - END OF DISPLAY
                             10:51:43 * L TIMER=ALL,OP=PPT
                         P   10:51:45 - DISPLAY OF OUTSTANDING TIMER REQUESTS
                         P   10:51:45 - TYPE: EVERY   TIME: 10:54   ID: SYS00001   PPT INTERVAL: 00:05
                         P   10:51:45 -   COMMAND: LIST STATUS=OPS
                         P   10:51:45 - 001  TIMER ELEMENT(S) FOUND FOR PPT
                         P   10:51:45 - END OF DISPLAY
                             10:53:10 * ROUTE DSIMN,AT 10:56,ID=DSIMNAT,L S=OPS
                   DSIMN     10:53:19 * AT 10:56,ID=DSIMNAT,L S=OPS
                             10:53:24 - DSI201I TIMER REQUEST SCHEDULED FOR EXECUTION - ID=DSIMNAT
                             10:53:24 - DSI810I NCCF ACF/VTAM READY
```

Figure B-2 (Part 1 of 2). Example of NCCF Log

```
FRED                PARIS P%   10:54:49 - DSI208I TIME EXPIRATION - ID= SYS00001 - CMD= LIST STATUS=OPS
                         P%    10:54:49 - OPERATOR: FRED     TERM: L3270D   STATUS: ACTIVE
                         P%    10:54:51 - END OF STATUS DISPLAY
                               10:55:00 - DSI208I TIME EXPIRATION - ID= FRED2    - CMD= DISPLAY NET,ID=FPU3276
                               10:55:04   IST097I  DISPLAY  ACCEPTED
                               10:55:09   IST075I  VTAM DISPLAY - NODE TYPE= PHYSICAL UNIT
                               10:55:09   IST486I  NAME = FPU3276  , STATUS = NEVAC
                               10:55:09   IST081I  LINE NAME= FSDLC26  , LINE GROUP= FGROUP20 , MAJNOD= NCP6C75
                               10:55:09   IST654I  I/O TRACE= OFF ,BUFFER TRACE= OFF
                               10:55:09   IST314I  END
                    DSIMN      10:57:02 - DSI208I TIME EXPIRATION - ID= DSIMNAT  - CMD= L S=OPS
                               10:57:09 - OPERATOR: PETE     TERM: L3270A   STATUS: ACTIVE
                               10:57:09 - END OF STATUS DISPLAY
                    PARIS      10:57:29 * PURGE TIMER=ALL,OP=PPT
                          P    10:57:34 - DSI205I 001 TIMER ELEMENTS PURGED - OP= PARISPPT
                               10:58:08 * PURGE TIMER=ALL
                          P    10:58:11 - DSI205I 000 TIMER ELEMENTS PURGED - OP= FRED
                               10:58:34 * LOGOFF
                               10:58:35 - DSI081I OPERATOR FRED     LOGGED OFF TERMINAL L3270D
BVD                            11:00:25 - DSI020I OPERATOR BVD      LOGGED ON FROM TERMINAL L3270D   USING
                                 PROFILE(PROFAUTH),HCL(       )
                           %   11:00:25 - DSI020I OPERATOR BVD      LOGGED ON FROM TERMINAL L3270D   USING
                                 PROFILE(PROFAUTH),HCL(       )
                               11:01:45 * SWITCH DSILOG,S
 END OF PRINT FOR NCCF LOG
```

Figure B-2 (Part 2 of 2). Example of NCCF Log

```
09/23/80 09:15:08

-  09:15:08  DSIM2     PETE         DSI033I L3284A   SESSION STARTING FOR PETE
-  09:15:08  DSIM2     PETE         DSI020I OPERATOR PETE     LOGGED ON FROM TERMINAL L3270A   USING
                                    PROFILE(TCAMSTAR),HCL(L3284A  )
-  09:15:08  DSIM2     PETE         DSI020I OPERATOR PETE     LOGGED ON FROM TERMINAL L3270A   USING
                                    PROFILE(TCAMSTAR),HCL(L3284A  )
*  09:15:14  DSIM2     PETE         AUTOWRAP
!  09:15:14  DSIM2     PETE         DSI082I AUTOWRAP STARTED
*  09:15:28  DSIM2     PETE         L S=OPS
-  09:15:29  DSIM2     PETE         OPERATOR: PETE     TERM: L3270A   STATUS: ACTIVE
-  09:15:29  DSIM2     PETE         END OF STATUS DISPLAY
-  09:15:42  DSIM2     PETE         DSI208I TIME EXPIRATION - ID= SYS00001 - CMD= MSG ALL,THIS IS FROM NCCFIC
                                    EVERY 5(PPT)(-------------EV5P
M  09:15:42  DSIM2     PETE         DSI039I MSG FROM DSIM2PPT: THIS IS FROM NCCFIC EVERY
                                    5(PPT)(-------------EV5P
-  09:15:42  DSIM2     PETE         DSI001I MESSAGE SENT TO ALL
*  09:16:04  DSIM2     PETE         PURGE TIMER=ALL,OP=PPT
-  09:16:04  DSIM2     PETE         DSI205I 004 TIMER ELEMENTS PURGED - OP= DSIM2PPT
*  09:17:39  DSIM2     PETE         D TP.ACT.NTCAM2
T  09:17:40  DSIM2     PETE         IED036I NTCAM2   ACTIVE=FGROUP20.006
*  09:17:56  DSIM2     PETE         D TP.TERM.L3270A
T  09:17:56  DSIM2     PETE         IED033I L3270A   STATUS= SNGLTRM  OPTFLDS SCNDARY INTENSE= NO
                                    IN-SEQ=0001, OUT-SEQ=0045
*  09:18:14  DSIM2     PETE         DOMAINS
C  09:18:15  DSIM2     PETE         PARIS DOMAIN IS INACTIVE. TO START REPLY'GO START
C  09:18:15  DSIM2     PETE         TO CONTINUE ON REPLY'GO'.
*  09:18:21  DSIM2     PETE         GO
C  09:18:21  DSIM2     PETE         DSIMN DOMAIN IS INACTIVE. TO START REPLY'GO START
C  09:18:21  DSIM2     PETE         TO CONTINUE ON REPLY'GO'.
*  09:18:27  DSIM2     PETE         GO
*  09:27:26  DSIM2     PETE         ASSIGN 87654321 + 4567 - 321 + -34
C  09:27:27  DSIM2     PETE         ** 87654321 + 4567 - 321 + -34 = 87658533
*  09:34:13  DSIM2     PETE         DOMAIN
C  09:34:15  DSIM2     PETE         THE DOMAIN NAME IS DSIM2
*  09:35:28  DSIM2     PETE         AT 9:37,ID=AT937,L S=OPT
-  09:35:28  DSIM2     PETE         DSI201I TIMER REQUEST SCHEDULED FOR EXECUTION - ID=AT937
*  09:36:02  DSIM2     PETE         EVERY 5,PPT,REQMS FPU3276
-  09:36:03  DSIM2     PETE         DSI201I TIMER REQUEST SCHEDULED FOR EXECUTION - ID=SYS00003
*  09:36:11  DSIM2     PETE         LIST TIMER=ALL,OP=ALL
-  09:36:11  DSIM2     PETE         DISPLAY OF OUTSTANDING TIMER REQUESTS
-  09:36:12  DSIM2     PETE         TYPE: AT      TIME: 09:37   ID: AT937
-  09:36:12  DSIM2     PETE           COMMAND: L S=OPT
-  09:36:12  DSIM2     PETE           OP: PETE
-  09:36:12  DSIM2     PETE         TYPE: EVERY   TIME: 09:41   ID: SYS00003   PPT INTERVAL: 00:05
-  09:36:12  DSIM2     PETE           COMMAND: REQMS FPU3276
-  09:36:12  DSIM2     PETE           OP: PPT(PETE)
-  09:36:13  DSIM2     PETE         002  TIMER ELEMENT(S) FOUND FOR ALL
-  09:36:13  DSIM2     PETE         END OF DISPLAY
-  09:37:01  DSIM2     PETE         DSI208I TIME EXPIRATION - ID= AT937    - CMD= L S=OPT
-  09:37:02  DSIM2     PETE         NAME: DSILOG   TASKNAME: DSILOG   STATUS: ACTIVE
-  09:37:02  DSIM2     PETE         NAME: BNHDSERV TASKNAME: BNHDSERV STATUS: ACTIVE
-  09:37:02  DSIM2     PETE         NAME: BNHLNPDA TASKNAME: BNHLNPDA STATUS: ACTIVE
-  09:37:02  DSIM2     PETE         NAME: BNHEDCBA TASKNAME: BNHEDCBA STATUS: ACTIVE
-  09:37:02  DSIM2     PETE         END OF STATUS DISPLAY
-  09:38:14  DSIM2     FRED         DSI033I L3284A   SESSION STARTING FOR FRED
-  09:38:14  DSIM2     FRED         DSI020I OPERATOR FRED     LOGGED ON FROM TERMINAL L3270C   USING
                                    PROFILE(TCAMPROF),HCL(L3284A  )
-  09:38:14  DSIM2     FRED         DSI164I PROFILE TCAMPROF SPECIFIES FEATURES NOT SUPPORTED - PARAMETERS
                                    IGNORED.
-  09:38:14  DSIM2     PETE         DSI020I OPERATOR FRED     LOGGED ON FROM TERMINAL L3270C   USING
                                    PROFILE(TCAMPROF),HCL(L3284A  )
*  09:39:49  DSIM2     FRED         EVERY 10,REQMS FPU3274
-  09:39:50  DSIM2     FRED         DSI201I TIMER REQUEST SCHEDULED FOR EXECUTION - ID=SYS00004
*  09:40:21  DSIM2     PETE         AT 9:42,ID=PETE1,MSG SYSOP,PLEASE DISABLE CHANNEL 0C1.
-  09:40:22  DSIM2     PETE         DSI201I TIMER REQUEST SCHEDULED FOR EXECUTION - ID=PETE1
*  09:40:37  DSIM2     FRED         LIST TIMER=ALL,OP=PETE
-  09:40:37  DSIM2     FRED         DSI213I ACCESS NOT AUTHORIZED: OP NOT IN OPERATOR'S SCOPE
*  09:40:52  DSIM2     FRED         LIST TIMER=ALL
-  09:40:55  DSIM2     FRED         DISPLAY OF OUTSTANDING TIMER REQUESTS
-  09:40:55  DSIM2     FRED         TYPE: EVERY   TIME: 09:49   ID: SYS00004       INTERVAL: 00:10
-  09:40:55  DSIM2     FRED           COMMAND: REQMS FPU3274
-  09:40:55  DSIM2     FRED         001  TIMER ELEMENT(S) FOUND FOR FRED
-  09:40:55  DSIM2     FRED         END OF DISPLAY
-  09:41:03  DSIM2     PETE         DSI208I TIME EXPIRATION - ID= SYS00003 - CMD= REQMS FPU3276
-  09:42:03  DSIM2     PETE         DSI208I TIME EXPIRATION - ID= PETE1    - CMD= MSG SYSOP,PLEASE DISABLE
                                    CHANNEL 0C1.
M  09:42:06  DSIM2     PETE         DSI039I MSG FROM PETE    : PLEASE DISABLE CHANNEL 0C1.
-  09:42:06  DSIM2     PETE         DSI001I MESSAGE SENT TO SYSOP
*  09:42:10  DSIM2     FRED         MSG PETE,ARE YOU USING THE 3705-6 0C0?
M  09:42:10  DSIM2     PETE         DSI039I MSG FROM FRED    : ARE YOU USING THE 3705-6 0C0?
-  09:42:11  DSIM2     FRED         DSI001I MESSAGE SENT TO PETE
*  09:42:25  DSIM2     PETE         MSG FRED,YES I AM
-  09:42:26  DSIM2     PETE         DSI001I MESSAGE SENT TO FRED
M  09:42:26  DSIM2     FRED         DSI039I MSG FROM PETE    : YES I AM
*  09:42:46  DSIM2     FRED         PURGE TIMER=ALL
-  09:42:47  DSIM2     FRED         DSI205I 001 TIMER ELEMENTS PURGED - OP= FRED
*  09:42:54  DSIM2     FRED         LOGOFF
-  09:42:55  DSIM2     FRED         DSI081I OPERATOR FRED     LOGGED OFF TERMINAL L3270C
-  09:43:05  DSIM2     PETE         DSI081I OPERATOR FRED     LOGGED OFF TERMINAL L3270C
-  09:43:06  DSIM2                  DSI056I L3284A   SESSION STOPPING FOR FRED
*  09:44:29  DSIM2     PETE         LOGOFF
-  09:44:29  DSIM2     PETE         DSI081I OPERATOR PETE     LOGGED OFF TERMINAL L3270A
-  09:44:30  DSIM2                  DSI056I L3284A   SESSION STOPPING FOR PETE
-  09:44:30  DSIM2     L3284A       DSI104I *****     END      OF      NCCF      HARDCOPY      LOG       ****
```

Figure B-3. Example of Hard-Copy Log

Appendix C. NCCF Controls Blocks

This appendix describes some of the control blocks used by NCCF command processors and service routines. The complete description of all NCCF control blocks is in *NCCF Logic,* LY38-3010.

How to Read Data Maps

The data map descriptions in this manual consist of four sections:

- A reference list precedes the diagram of the data area. This list may contain some or all of the following items:

DSIname:	Functional description
BOUNDARY:	Byte, halfword, fullword, or doubleword
LENGTH:	Decimal and hexadecimal value in bytes
POINTED TO BY:	Control blocks that contain pointers to this data area
INCLUDED BLOCKS:	Control blocks embedded within this data area
NOTE:	Additional information

- A diagram of the control block follows the reference list. This diagram shows the offsets, type, length, name, and description of each field in the control block. The offsets are the decimal and hexadecimal displacements of the fields. The type indicator tells whether the field is a pointer, character string, bit string, or the first field of a structure. The length of each field is shown in bytes. Variable-length fields are shown with a length of zero. These variable-length fields contain appendages that may end on a boundary other than the one specified. Most of the control block fields contain the name of the field and a description of the function of the field.
- A cross-reference list immediately follows the data map diagram. This list contains, in alphabetic order, the symbolic names of the fields that define storage in the control block. Each field name is followed by its decimal and hexadecimal displacement within the control block. Bit names are followed by a decimal displacement and a hexadecimal value representation.
- A list of constant fields in the control block, if any, follows the alphabetic cross-reference list. The constants are listed by name, value, and meaning. The constant values may be in decimal, hexadecimal (X), binary (B), or character representations. A constant or group of constants that define values for a particular field in the control block is identified with a subheading that specifies the primary field name.

CBH

DSICBH: MAPS THE CONTROL BLOCK HEADER WHICH APPEARS AS THE FIRST FIELD IN MOST NCCF CONTROL BLOCKS
BOUNDARY: DOUBLEWORD
LENGTH: 4 BYTES
LOCATION: THE FIRST FIELD IN MOST NCCF CONTROL BLOCKS

OFFSETS		TYPE	LENGTH	NAME	DESCRIPTION
0	(0)	STRUCTURE	4	DSICBH	CONTROL BLOCK HEADER
0	(0)	BITSTRING	1	CBHID	CONTROL BLOCK ID (UNIQUE FOR EACH CONTROL BLOCK)
1	(1)	BITSTRING	1	CBHTYPE	CONTROL BLOCK TYPE (APPLIES TO DSITVB AND DSITIB ONLY
2	(2)	SIGNED	2	CBHLENG	CONTROL BLOCK LENGTH (INCLUDING THIS HEADER). LENGTH BASED ON CBHID AND CBHTYPE FIELDS.

CROSS REFERENCE

CBHID	0	(0)
CBHLENG	2	(2)
CBHTYPE	1	(1)
DSICBH	0	(0)

Constants in DSICBH

NAME	VALUE	MEANING
Masks For Setting and Testing CBHID Field (Bit 8)		
CBHMVT	X'F1'	MVT CTL BLK ID VALUE
CBHTVB	X'F2'	TVB CTL BLK ID VALUE
CBHTIB	X'F3'	TIB CTL BLK ID VALUE
CBHTID	X'F4'	TID CTL BLK ID VALUE
CBHSNT	X'C1'	SNT CTL BLK ID VALUE
CBHART	X'C2'	ART CTL BLK ID VALUE
CBHDQT	X'C3'	DQT CTL BLK ID VALUE
CBHSCT	X'C4'	SCT CTL BLK ID VALUE
CBHOIT	X'C5'	OIT CTL BLK ID VALUE
CBHDDT	X'C6'	DDT CTL BLK ID VALUE
CBHSVL	X'C8'	SVL CTL BLK ID VALUE
CBHCWB	X'C9'	CWB CTL BLK ID VALUE
CBHSWB	X'D1'	SWB CTL BLK ID VALUE
CBHDSB	X'D2'	DSB CTL BLK ID VALUE
CBHDCT	X'D3'	DCT CTL BLK ID VALUE
CBHSCB	X'D4'	SCB CTL BLK ID VALUE
CBHPDB	X'D5'	PDB CTL BLK ID VALUE
CBHNAT	X'D6'	NAT CTL BLK ID VALUE
CBHSAT	X'D7'	SAT CTL BLK ID VALUE
CBHCLB	X'D8'	CLB CTL BLK ID VALUE
CBHUSE	X'D9'	USE CTL BLK ID VALUE
CBHSSB	X'E2'	SSB CTL BLK ID VALUE
CBHDSRB	X'E3'	DSRB CTL BLK ID VALUE
CBHTIQ	X'E4'	TIQ CBH ID
CBHNDT	X'E5'	NDT CTL BLK VALUE
CBHNMB	X'E6'	NMB CTL BLK ID VALUE
CBHSWA	X'E7'	SWB CTL BLK ID VALUE
CBHCDB	X'E8'	CDB CTL BLK
CBHLUTSK	X'4A'	LU TASK BLOCK ID
CBHGRTTB	X'4C'	GLOBAL CNM ROUTING TABLE
CBHCDNID	X'4D'	CDRM-TO-NCCFID TABLE
Constants for Setting/Testing CHBTYPE Field (Bit 8)		
CBHPPT	X'00'	DSIPPT CTL BLK TYPE VALUE
CBHNNT	X'01'	DSINNT CTL BLK TYPE VALUE
CBHOST	X'02'	DSIOST CTL BLK TYPE VALUE
CBHHCT	X'03'	DSIHCT CTL BLK TYPE VALUE
CBHTCT	X'04'	DSITCT CTL BLK VALUE
CBHOPTSK	X'05'	OPTIONAL TASK CTL BLK TYPE VALUE
CBHMNT	X'06'	MAIN TASK CTL BLK TYPE VALUE
CBHACT	X'00'	DSISWB/CWB CTL BLK TYPE VALUE
CBHINACT	X'FF'	DSISWB/CWB CTL BLK TYPE VALUE
CBHTIQS	X'E2'	SEND TYPE TIQ
CBHTIQR	X'D9'	RECEIVE TYPE TIQ
CBHTIQL	X'D3'	LOGON TYPE TIQ
CBHTIQO	X'D6'	OP QUEUE TYPE TIQ
CBHTIQX	X'E7'	CROSS DOMAIN TIQ
CBHTIQC	X'C3'	CSMI TYPE TIQ
CBHCDADD	X'C1'	ADDITIONAL CDB
CBHCDMN	X'D4'	MAIN CLB
CBHLUSSB	X'E2'	LU TASK LUSSB BLK
CBHLUTVT	X'E3'	LU TASK LUTVT BLK

CWB

DSICWB: MAPS THE COMMAND WORK BLOCK
BOUNDARY: DOUBLEWORD
LENGTH: 364 BYTES (X'16C')
POINTED TO BY: TIB (TIBNCCWB) NORMAL COMMAND
(TIBICCWB) IMMEDIATE COMMAND
(TIBMRCCWB) RECEIVED COMMAND
MVT (MVTLCWB) FIRST CWB ON CHAIN
INCLUDED BLOCKS: CBH (CWBCBH)

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
0 (0)	STRUCTURE	364	DSICWB	COMMAND WORK BLOCK
0 (0)	CHARACTER	4	CWBCBH	CONTROL BLOCK HEADER
4 (4)	A-ADDRESS	72	CWBSAVEA	SAVEAREA USED TO CALL A SERVICE ROUTINE OR ANOTHER COMMAND PROCESSOR
76 (4C)	CHARACTER	12	CWBPARMS	COMAND PROCESSOR PARAMETERS
76 (4C)	A-ADDRESS	4	CWBBUF	POINTER TO COMMAND BUFFER
80 (50)	A-ADDRESS	4	CWBPDB	POINTER TO PARSED COMMAND
84 (54)	A-ADDRESS	4	CWBSWB	POINTER TO SWB FOR USE BY THE COMMAND PROCESSOR
88 (58)	A-ADDRESS	4	CWBNEXT	ADDR OF NEXT CWB ON THE CHAIN
92 (5C)	A-ADDRESS	4	CWBTIB	ADDR OF CALLERS TIB
96 (60)	CHARACTER	256	CWBADATD	AUTOMATIC WORKAREA
352 (160)	A-ADDRESS	4	CWBDSRB	POINTER TO THE DSRB
356 (164)	A-ADDRESS	4		RESERVED
360 (168)	A-ADDRESS	4		RESERVED

LOGON AUTHORIZATION WORKING STORAGE

96 (60)	STRUCTURE	256	CWBLAPRM	LOGON ATHORIZATION PARM LIST
96 (60)	CHARACTER	8	CWBLAUSR	USER ID IF LOGON IS AUTHORIZED
104 (68)	A-ADDRESS	4	CWBLASTG	PTR TO 1ST PIECE OF WORKING STORAGE
108 (6C)	A-ADDRESS	8	CWBLASPN	PTRS TO SPAN/ISPAN SKELETONS
116 (74)	A-ADDRESS	4	CWBLADOM	PTR TO DOMAINS SKELETON
120 (78)	SIGNED	4	CWBLASCT	SPAN/ISPAN SKELETON COUNT
124 (7C)	SIGNED	4	CWBLAOIT	OIT ENTRY INDEX NUMBER
128 (80)	SIGNED	2	CWBLADCT	DOMAINS SKELETON COUNT
130 (82)	CHARACTER	1	CWBLAFLG	INDICATOR FLAGS
1...			CWBLAUTH	AUTH YES INDICATED
.1..			CWBLAGBL	GLOBAL AUTHORITY
131 (83)	CHARACTER	8	CWBLAHCY	HARDCOPY LUNAME
139 (8B)	CHARACTER	8	CWBLAPRF	PROFILE NAME
147 (93)	CHARACTER	205	CWBLAWRK	REMAINING WORK AREA
0 (0)	STRUCTURE	8	CWBLASTO	LAC STORAGE BLOCK HEADER

OFFSETS		TYPE	LENGTH	NAME	DESCRIPTION
0	(0)	A-ADDRESS	4	CWBLACHN	NEXT PIECE OF STORAGE
4	(4)	SIGNED	4	CWBLALEN	LENGTH OF STORAGE PIECE
0	(0)	STRUCTURE	8	CWBLASEN	SPAN/ISPAN SKELETON ENTRY
0	(0)	A-ADDRESS	4	CWBLASCH	PTR TO NEXT ENTRY OF SAME TYPE
4	(4)	SIGNED	2	CWBLASIN	SNT INDEX VALUE
6	(6)	SIGNED	2		ALIGNMENT
0	(0)	STRUCTURE	12	CWBLADEN	DOMAINS SKELETONS ENTRY
0	(0)	A-ADDRESS	4	CWBLADCH	PTR TO NEXT DOMAINS SKELETON
4	(4)	CHARACTER	8	CWBLADNM	DOMAIN NAME

CROSS REFERENCE

CWBADATD	96 (60)	CWBLASCH	0 (0)
CWBBUF	76 (4C)	CWBLASCT	120 (78)
CWBCBH	0 (0)	CWBLASEN	0 (0)
CWBDSRB	352(160)	CWBLASIN	4 (4)
CWBLACHN	0 (0)	CWBLASPN	108 (6C)
CWBLADCH	0 (0)	CWBLASTG	104 (68)
CWBLADCT	128 (80)	CWBLASTO	0 (0)
CWBLADEN	0 (0)	CWBLAUSR	96 (60)
CWBLADNM	4 (4)	CWBLAUTH	130 X'80'
CWBLADOM	116 (74)	CWBLAWRK	147 (93)
CWBLAFLG	130 (82)	CWBNEXT	88 (58)
CWBLAGBL	130 X'40'	CWBPARMS	76 (4C)
CWBLAHCY	131 (83)	CWBPDB	80 (50)
CWBLALEN	4 (4)	CWBSAVEA	4 (4)
CWBLAOIT	124 (7C)	CWBSWB	84 (54)
CWBLAPRF	139 (8B)	CWBTIB	92 (5C)
CWBLAPRM	96 (60)	DSICWB	0 (0)

Constants in DSICWB

NAME	VALUE	MEANING
ACF/VTAM Command Processor--DSIVTP Constants and Return Codes (Fullword)		
VTPGOOD	0	COMMAND EXECUTED SUCCESSFULLY
VTPBAD	8	COMMAND FAILED-- ERROR MSG ISSUED
VTPABORT	12	COMMAND FAILED-- NO ERROR MSG ISSUED
START Command Processor--DSISRP Constants and Return Codes (Fullword)		
SRPGOOD	0	COMMAND EXECUTED SUCCESSFULLY
SRPBAD	8	COMMAND FAILED-- ERROR MSG ISSUED
ROUTE Command Processor--DSIRTP Constants and Return Codes (Fullword)		
RTPGOOD	0	COMMAND EXECUTED SUCCESSFULLY
RTPBAD	8	COMMAND FAILED-- ERROR MSG ISSUED

DSB

DSIDSB: MAPS THE DATA SERVICE BLOCK USED BY NCCF DATA SET SERVICE ROUTINES
BOUNDARY: DOUBLEWORD
LENGTH: 50 BYTES (X'32')
POINTED TO BY: SWB (DKSDSB)
CLB (CLBDSB)
INCLUDED BLOCKS: CBH (DSBCBH)

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
0 (0)	STRUCTURE	50	DSIDSB	DATA SERVICE BLOCK
0 (0)	CHARACTER	4	DSBCBH	CONTROL BLOCK HEADER
4 (4)	A-ADDRESS	4	DSBNEXT	CHAIN PTR TO NEXT DSB
8 (8)	A-ADDRESS	4	DSBFILE	ADDR OF THE DCB/DTF
12 (C)	A-ADDRESS	4	DSBBUFF	ADDR OF I/O BUFFER
16 (10)	A-ADDRESS	4	DSBREC	ADDR OF LOGICAL RECORD
20 (14)	CHARACTER	12	DSBBLOCK	VSE ADDR OF LAST RECORD READ
20 (14)	SIGNED	4	DSBTTR	OS/VS REL BLOCK ADDR OF LAST RECORD
24 (18)	UNSIGNED	1	DSBLOGRE	OS/VS LOGICAL RECORD FOR BLOCKED RECORDS
25 (19)	CHARACTER	1		RESERVED
26 (1A)	UNSIGNED	2	DSBLOGRC	SAVE AREA FOR LOG RECORD COUNT
28 (1C)	CHARACTER	4		RESERVED
32 (20)	SIGNED	4	DSBCUREC	CURRENT PHYSICAL RECORD IN BUFFER
36 (24)	CHARACTER	8	DSBMEMBR	DATA SET MEMBER NAME
44 (2C)	SIGNED	2	DSBIOSZ	I/O BUFFER SIZE
46 (2E)	SIGNED	2	DSBRECNT	LOGICAL RECORD COUNT IN CALLERS BUFFER TO PROCESS
48 (30)	BITSTRING	2	DSBFLGS	INDICATOR FLAGS
1...			DSBIOERR	I/O ERROR ON READ OPERATION
.1..			DSBFND	MEMBER FOUND IN PDS
..1.			DSBEOD	END OF DATA SET ON THIS MEMBER

CROSS REFERENCE

DSBBLOCK	20 (14)
DSBBUFF	12 (C)
DSBCBH	0 (0)
DSBCUREC	32 (20)
DSBEOD	48 X'20'
DSBFILE	8 (8)
DSBFLGS	48 (30)
DSBFND	48 X'40'
DSBIOERR	48 X'80'
DSBIOSZ	44 (2C)
DSBLOGRC	26 (1A)
DSBLOGRE	24 (18)
DSBMEMBR	36 (24)
DSBNEXT	4 (4)
DSBREC	16 (10)
DSBRECNT	46 (2E)
DSBTTR	20 (14)
DSIDSB	0 (0)

DSRB

DSIDSRB: MAPS THE DATA SERVICES REQUEST BLOCK FOR THE DATA SERVICES TASK
BOUNDARY: DOUBLEWORD
LENGTH: 163 BYTES (X'A3')
POINTED TO BY: CWB (CWBDSRB)
TID (TIDDSRB) ANCHOR FOR CHAIN
SWB (SWBCSDRB)
INCLUDED BLOCKS: CBH (DSRBCBH)

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
0 (0)	STRUCTURE	163	DSIDSRB	DATA SERVICES REQUEST BLOCK
0 (0)	CHARACTER	1	DSRBCBH	NCCF CTL BLK HDR
4 (4)	A-ADDRESS	4	DSRBNXTV	ADDR OF NEXT DSRB IN THE CHAIN
8 (8)	A-ADDRESS	4	DSRBVECB	VSAM ECB
12 (C)	A-ADDRESS	4	DSRBVRPL	ADDR OF VSAM RPL
16 (10)	A-ADDRESS	4	DSRBCUSB	ADDR PREALLOCATED CSMI UNSOLICITED RESPONSE BUFFER
20 (14)	A-ADDRESS	4		RESERVED CRITICAL FIELD
24 (18)	A-ADDRESS	4		RESERVED CRITICAL FIELD
28 (1C)	A-ADDRESS	4		RESERVED CRITICAL FIELD
32 (20)	A-ADDRESS	4		RESERVED CRITICAL FIELD
36 (24)	BITSTRING	2	DSRBFLG	INDICATOR FLAGS
1...			DSRBTYPE	1 = UNSOLICITED 0 = SOLICITED
.1..			DSRBACTV	1 = IN USE BY A DSCP 0 = AVAILABLE
..1.			DSRBINUS	1 = IN USE BY CSMI/VSAM 0 = AVAILABLE FOR CSMI/VSAM
...1			DSRBVRDV	1 = VSAM REDRIVE OPERATION 0 = VSAM NORMAL OPERATION
.... 1...			DSRBVRSM	VSAM NOT FIRST TIME SWITCH
.... .1..			DSRBCRSM	CSMI NOT FIRST TIME SWITCH
.... ..1.			DSRBVSAM	LAST MACRO WAS DSIZVSMS
....1			DSRBCNMI	LAST MACRO WAS DSIZCSMS
38 (26)	SIGNED	2		RESERVED
40 (28)	A-ADDRESS	4	DSRBTIB	ADDR OF TIB OF DATA SERVICES TASK
44 (2C)	SIGNED	4		RESERVED
48 (30)	SIGNED	4		RESERVED
52 (34)	A-ADDRESS	4		RESERVED CRITICAL FIELD
56 (38)	A-ADDRESS	4		RESERVED CRITICAL FIELD
60 (3C)	CHARACTER	103	DSRBLOCK	CODE ZEROED OUT DURING CLEAN UP
60 (3C)	CHARACTER	8	DSRBRSMV	RESUME VERB (COMMAND PROCESSOR)
68 (44)	A-ADDRESS	4	DSRBRADD	ADDR OF RESUME COMMAND PROCESSOR

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
72	(48) CHARACTER	8	DSRBOID	OPERATOR ID THAT INVOKED PSCP
80	(50) A-ADDRESS	4	DSRBDSCP	ADDR OF ORIGINAL DSCP
84	(54) A-ADDRESS	4	DSRBUSER	USER FIELD
88	(58) A-ADDRESS	4	DSRBVACB	ADDR OF VSAM ACB
92	(5C) A-ADDRESS	4	DSRBVDAD	ADDR VSAM USER DATA AREA
96	(60) A-ADDRESS	4	DSRBVKEY	ADDR OF VSAM KEY
100	(64) SIGNED	2	DSRBVDLN	LENGTH OF VSAM USER DATA AREA
102	(66) SIGNED	2	DSRBVKLN	LENGTH OF VSAM KEY
104	(68) UNSIGNED	4	DSRBVOPT	VSAM OPTION INDICATORS
104	(68) BITSTRING	1	DSRVOPT1	
	1...		DSRVSEQ	SEQUENTIAL ACCESS
	.1..		DSRVDIR	DIRECT ACCESS
	..1.		DSRVSKP	SKIP SEQUENTIAL ACCESS
	...1		DSRVARD	USER ARGUMENT
	.... 1...		DSRVLRD	LAST RECORD
	.... .1..		DSRVFWD	FORWARD PROCESSING
	.... ..1.		DSRVBWD	BACKWARD PROCESSING
	1		DSRVNUP	RECORD NOT TO BE UPDATED
105	(69) BITSTRING	1	DSRVOPT2	
	1...		DSRVNSP	REMEMBER POSITION
	.1..		DSRVUPD	UPDATE RECORD
	..1.		DSRVKEQ	KEY = FOR ARGUMENT
	...1		DSRVKGE	KEY GREATER THAN OR EQUAL TO
	.... 1...		DSRVFKS	FULL KEY FOR ARGUMENT
	.... .1..		DSRVGEN	GENERIC KEY FOR ARGUMENT
108	(6C) SIGNED	4		RESERVED
112	(70) CHARACTER	1		RESERVED
113	(71) CHARACTER	1	DSRBVRTP	VSAM REQUEST TYPE
114	(72) SIGNED	2		RESERVED
116	(74) SIGNED	4	DSRBVRCT	VSAM LOGICAL ERROR RETRY COUNTER
120	(78) UNSIGNED	2	DSRBCCOR	NEG RESPONSE CORRELATION VALUE
122	(7A) CHARACTER	2		RESERVED
124	(7C) A-ADDRESS	4	DSRBUBUF	ADDR OF INPUT BUFFER QUEUED BY DSCP
128	(80) SIGNED	2	DSRBBUFL	LENGTH OF DSCP INPUT BUFFER
130	(82) SIGNED	2	DSRBPRID	USER CORRELATION VALUE
132	(84) SIGNED	4		RESERVED
136	(88) A-ADDRESS	4	DSRBINPT	ADDR OF CSMI INPUT BUFFER
140	(8C) SIGNED	2	DSRBILEN	LENGTH OF CSMI INPUT BUFFER
142	(8E) SIGNED	2		RESERVED
144	(90) SIGNED	4	DSRBRCMA	MAJOR RETURN CODE VALUE
148	(94) SIGNED	4	DSRBRCMI	MINOR RETURN CODE VALUE
152	(98) SIGNED	4	DSRBDSRC	VSE I/O COMPLETE RETURN CODE
156	(9C) CHARACTER	1	DSRBFNCD	DSRB FUNCTION INDICATION CODE
157	(9D) CHARACTER	1		RESERVED
158	(9E) CHARACTER	3	DSRBREQN	REQUEST SERIAL NO PACKED
161	(A1) CHARACTER	2	DSRBSTEP	STEP SERIAL NO IN PACKED

CROSS REFERENCE

DSIDSRB	0	(0)
DSRBACTV	36	X'40'
DSRBBUFL	128	(80)
DSRBCBH	0	(0)
DSRBCCOR	120	(78)
DSRBCNMI	36	X'01'
DSRBCRSM	36	X'04'
DSRBCUSB	16	(10)
DSRBDSCP	80	(50)
DSRBDSRC	152	(98)
DSRBFLG	36	(24)
DSRBFNCD	156	(9C)
DSRBILEN	140	(8C)
DSRBINPT	136	(88)
DSRBINUS	36	X'20'
DSRBLOCK	60	(3C)
DSRBNXTV	4	(4)
DSRBOID	72	(48)
DSRBPRID	130	(82)
DSRBRADD	68	(44)
DSRBRCMA	144	(90)
DSRBRCMI	148	(94)
DSRBREQN	158	(9E)
DSRBRSMV	60	(3C)
DSRBSTEP	161	(A1)
DSRBTIB	40	(28)
DSRBTYPE	36	X'80'
DSRBUBUF	124	(7C)
DSRBUSER	84	(54)
DSRBVACB	88	(58)
DSRBVDAD	92	(5C)
DSRBVDLN	100	(64)
DSRBVECB	8	(8)
DSRBVKEY	96	(60)
DSRBVKLN	102	(66)
DSRBVOPT	104	(68)
DSRBVRCT	116	(74)
DSRBVRDV	36	X'10'
DSRBVRPL	12	(C)
DSRBVRSM	36	X'08'
DSRBVRTP	113	(71)
DSRBVSAM	36	X'02'
DSRVARD	104	X'10'
DSRVBWD	104	X'02'
DSRVDIR	104	X'40'
DSRVFKS	105	X'08'
DSRVFWD	104	X'04'
DSRVGEN	105	X'04'
DSRVKEQ	105	X'20'
DSRVKGE	105	X'10'
DSRVLRD	104	X'08'
DSRVNSP	105	X'80'
DSRVNUP	104	X'01'
DSRVOPT1	104	(68)
DSRVOPT2	105	(69)
DSRVSEQ	104	X'80'
DSRVSKP	104	X'20'
DSRVUPD	105	X'40'

Constants in DSIDSRB

NAME	VALUE	MEANING
DSRBFNCD With Possible Values that Can Be Assigned to the Code (Halfword)		
DSRBFNRM	1	FIRST INVOCATION OF DSCP
DSRBFUNS	2	UNSOLICITED DATA PASSED TO DSCP
DSRBFSOL	3	SOLICITED DATA PASSED TO DSCP
DSRBFVSM	4	VSAM I/O OPERATION HAS COMPLETED
Constants For Setting and Testing DSRB Minor Return Codes (Fullword)		
(minor return code field settings are those defined by VSAM)		
DSRCGOOD	0	REQUESTED FUNCTION PERFORMED
DSRCNGRP	4	NEGATIVE RESPONSE WAS RECEIVED
DSRCSTOR	8	INSUFFICIENT NCCF STORAGE TO PROCESS REQUEST
DSRCUNSL	12	NO UNSOLICITED DSRB WAS AVAILABLE
DSRCNOEX	16	USER EXIT REJECTED THIS REQUEST
DSRCTRNC	20	DATA TRUNCATION OCCURRED DUE TO INSUFFICENT INPUT BUFFER LENGTH
DSRCEXTR	24	DATA TRUNCATION OCCURRED (USER EXIT SUBSTITUTION)
DSRCAREJ	28	ACCESS METHOD REJECTED REQUEST
DSRCLOSE	32	CNMI CLOSED DUE TO UNRECOVERABLE ERROR
Constants For Setting and Testing DSRBVRTP (Bit 8)		
DSRVGET	X'01'	GET REQUEST
DSRVPUT	X'02'	PUT REQUEST
DSRVPNT	X'03'	POINT REQUEST
DSRVERS	X'04'	ERASE REQUEST
DSRVNRQ	X'05'	ENDREQ REQUEST
Constants For Setting and Testing DSRB Major Fields (Fullword)		
DSRBVSUC	0	SUCCESSFUL COMPLETION
The Minor Return Code Settings For DSRCVLOG and DSRCVPHS are defined by VSAM		
DSRCVLOG	8	VSAM CHECK MACRO LOGICAL ERROR
DSRCVPHS	12	VSAM CHECK MACRO PHYSICAL ERROR
Constants For Setting and Testing DSRBTYPE (Bit 1)		
DSRBTSOL	B'0'	SOLICITED DSRB
DSRBTUNS	B'1'	UNSOLICITED DSRB
Constants For Setting and Testing DSRBACTV (Bit 1)		
DSRBAUSE	B'1'	DSRB IN USE BY DSCP
DSRBAVAL	B'0'	DSRB AVAILABLE FOR USE

IFR

DSIIFR: MAPS THE INTERNAL FUNCTION REQUEST PARAMETER LIST
BOUNDARY: FULLWORD
LENGTH: 2 BYTES + VARIABLE PARAMETERS
POINTED TO BY: TIB (TIOPSIFR) PSS IFR
NOTE: OVERLAYS HDRMSG FIELD IN DSITIB

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
0	(0) STRUCTURE	2	DSIIFR	INTERNAL FUNCTION REQUEST
0	(0) SIGNED	2	IFRCODE	SEE CODE VALUES
2	(2) CHARACTER	0	IFRPARMS	VARIABLE PARAMETERS
2	(2) STRUCTURE	19	IFRXDOM	XDOMAIN BUFFERS AND XTERM CLEANUP IFR REQUEST
2	(2) BITSTRING	1	IFRIND	INDICATORS
	1...		IFRXFREE	1 = FREEMAIN THIS BUFFER
	.1..		IFRXQTR	1 = TERMSESS REQUEST ON QUEUE
	..1.		IFRXQTP	1 = TERMSESS IN PROGRESS
3	(3) BITSTRING	3		ALIGNMENT
6	(6) SIGNED	2	IFRXMLEN	DOMAIN MESSAGE LENGTH
8	(8) A-ADDRESS	4	IFRXQCHN	QUEUE CHAIN PTR FOR SCB QUEUE
12	(C) CHARACTER	8	IFRXTNAT	PRIMARY DOMAIN APPLID (FROM A DSINAT ENTRY)
20	(14) CHARACTER	1	IFRXTEXT	TEXT
2	(2) STRUCTURE	10	IFRLGN	XTERM PARAMETERS
2	(2) SIGNED	2	IFRLGTYP	LOGON TYPE 1 = LOGON, 2 = SABOTEUR
4	(4) CHARACTER	8	IFRLGLUN	LUNAME
2	(2) STRUCTURE	26	IFRTWR	TCAM WRITE REQUEST
2	(2) CHARACTER	8	IFRTWDST	DESTINATION NAME
10	(A) SIGNED	2	IFRTWDLN	DATA LENGTH
12	(C) A-ADDRESS	4	IFRTWDAD	DATA ADDRESS NOTE: IF THIS FIELD IS 0, THE DATA IS ASSUMED TO BE CONTIGUOUS, I.E. AT IFRTWTXT
16	(10) A-ADDRESS	4	IFRTWRPL	RPL ADDRESS
20	(14) A-ADDRESS	4	IFRTWTVB	TVB ADDRESS
24	(18) A-ADDRESS	4		RESERVED
28	(1C) CHARACTER	0	IFRTWTXT	DATA, IF CONTIGUOUS WITH WRITE REQUEST

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
2	(2) STRUCTURE	10	IFRTRCSM	CSMI READ REQUEST
2	(2) A-ADDRESS	2		RESERVE
4	(4) A-ADDRESS	4	IFRTRRPL	RPL ADDRESS
8	(8) A-ADDRESS	4	IFRTRTVB	TVB ADDRESS
28	(1C) STRUCTURE	32	IFRSS	SCREEN SIZE REQUEST
28	(1C) CHARACTER	8	IFRSSCHR	SET TO '333333'X DSISS
36	(24) CHARACTER	8	IFRSSTRM	DEVICE
44	(2C) A-ADDRESS	4	IFRSSECB	ADDRESS OF ECB
48	(30) A-ADDRESS	4	IFRSSTVB	ADDRESS OF TVB
52	(34) A-ADDRESS	4	IFRSSPTR	POINTER TO SCREEN MATRX SIZE AREA IN REQUESTORS TIB
56	(38) CHARACTER	4	IFRSSDAT	SCREEN SIZE DATA
56	(38) SIGNED	2	IFRSSROW	ROW COUNT
58	(3A) SIGNED	2	IFRSSCOL	COL COUNT
28	(1C) STRUCTURE	13	IFRCW	CSMI WRITE DATA
28	(1C) CHARACTER	8	IFRCWCHR	SET TO '333333'X FWDRQ
36	(24) CHARACTER	1	IFRCWBLK	SET TO BLANK
37	(25) SIGNED	4	IFRCWCID	CORRELATION ID
41	(29) CHARACTER	0	IFRCWDAT	DATA
0	(0) STRUCTURE	4	IFRSSRC	MATRIX DSECT
0	(0) SIGNED	2	IFRSSTRW	ROW COUNT
2	(2) SIGNED	2	IFRSSTCL	COL COUNT
4	(4) CHARACTER	0	IFRSSEND	END OF DSECT
2	(2) STRUCTURE	38	IFRTIME	
2	(2) CHARACTER	8	IFRTITVB	TVBOPID OF TASK UNDER WHICH REQUEST WILL BE RUN
10	(A) CHARACTER	8	IFRTINAM	NAME OF ELEMENT
18	(12) CHARACTER	8	IFRTIPOP	OPID TO PURGE FOR
26	(1A) CHARACTER	8	IFRTITIM	TIME AS ENTERED
34	(22) SIGNED	2	IFRTIMIM	# MINUTES--EVERY COMMAND
36	(24) BITSTRING	1	IFRTIIND	FLAG BYTE
	1...		IFRTIAT	AT- TIME INDICATOR
	.1..		IFRTIEV	EVERY TIME INDICATOR
	..1.		IFRTIPR	PURGE- TIME INDICATOR
	...1		IFRTINXD	NEXT DAY INDICATOR
	 1111			ALIGNMENT
37	(25) BITSTRING	1		ALIGNMENT
38	(26) SIGNED	2	IFRTICR	FOR IFRCODCR
40	(28) CHARACTER	0	IFRTITXT	COMMAND TEXT AREA

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
2 (2)	STRUCTURE	12	IFRLP	LIST/PURGE MAPPING
2 (2)	UNSIGNED	1	IFRLPTYP	TYPE CODE (SEE CONSTANTS)
3 (3)	CHARACTER	8	IFRLPOP	AFFECTED OPERATOR ID
11 (B)	CHARACTER	3	IFRLPNUM	REQUEST NUM, PACKED, PURGE ONLY
2 (2)	STRUCTURE	10	IFRCR	CROSS TASK QUEUE COMMAND
2 (2)	CHARACTER	8	IFRCRVB	COMMAND VERB PADDED TO 8 CHARS
10 (A)	CHARACTER	2	IFRCRBLK	2 BLANK DELIMITER WORD BOUNDARY
12 (C)	CHARACTER	0	IFRCRPMS	COMMAND PARAMETERS

CROSS REFERENCE

DSIIFR	0 (0)	IFRTWR	2 (2)
IFRCODE	0 (0)	IFRTWRPL	16 (10)
IFRCR	2 (2)	IFRTWTVB	20 (14)
IFRCRBLK	10 (A)	IFRTWTXT	28 (1C)
IFRCRPMS	12 (C)	IFRXDOM	2 (2)
IFRCRVB	2 (2)	IFRXFREE	2 X'80'
IFRCW	28 (1C)	IFRXMLEN	6 (6)
IFRCWBLK	36 (24)	IFRXQCHN	8 (8)
IFRCWCHR	28 (1C)	IFRXQTP	2 X'20'
IFRCWCID	37 (25)	IFRXQTR	2 X'40'
IFRCWDAT	41 (29)	IFRXTEXT	20 (14)
IFRIND	2 (2)	IFRXTNAT	12 (C)
IFRLGLUN	4 (4)		
IFRLGN	2 (2)		
IFRLGTYP	2 (2)		
IFRLP	2 (2)		
IFRLPNUM	11 (B)		
IFRLPOP	3 (3)		
IFRLPTYP	2 (2)		
IFRPARMS	2 (2)		
IFRSS	28 (1C)		
IFRSSCHR	28 (1C)		
IFRSSCOL	58 (3A)		
IFRSSDAT	56 (38)		
IFRSSECB	44 (2C)		
IFRSSEND	4 (4)		
IFRSSPTR	52 (34)		
IFRSSRC	0 (0)		
IFRSSROW	56 (38)		
IFRSSTCL	2 (2)		
IFRSSTRM	36 (24)		
IFRSSTRW	0 (0)		
IFRSSTVB	48 (30)		
IFRTIAT	36 X'80'		
IFRTICR	38 (26)		
IFRTIEV	36 X'40'		
IFRTIIND	36 (24)		
IFRTIME	2 (2)		
IFRTIMIM	34 (22)		
IFRTINAM	10 (A)		
IFRTINXD	36 X'10'		
IFRTIPOP	18 (12)		
IFRTIPR	36 X'20'		
IFRTIITIM	26 (1A)		
IFRTITVB	2 (2)		
IFRTITXT	40 (28)		
IFRTRCSM	2 (2)		
IFRTRRPL	4 (4)		
IFRTRTVB	8 (8)		
IFRTWDAD	12 (C)		
IFRTWDLN	10 (A)		
IFRTWDST	2 (2)		

Constants in DSIIFR

LABEL	VALUE	MEANING
(Byte)		
IFRCODXT	1	XTERM CLEANUP REQUEST
IFRCODLG	2	LOGON REQUEST
IFRCODCR	3	CROSS-TASK QUEUE COMMAND
IFRCODTW	4	TCAM WRITE IFR
IFRCODSS	5	SCREEN SIZE REQUEST
IFRCODTR	6	CSMI READ ELEMENT
IFRCODCW	7	CSMI WRITE ELEMENT
IFRCODUS	8	USER IFR DRIVES EX013
IFRCODTP	9	TCAM PURGE I/O REQUEST
IFRCODTT	10	TCAM TERMINATE I/O REQUEST
IFRCODPN	11	SCREEN COMMAND IFR
IFRCODTI	12	TIMER REQUEST
IFRCODTC	14	TCAM RESETSR (CANCEL)
IFRCODLP	15	LIST/PURGE IMMEDIATE DST REQUEST
IFRCODLW	16	WRITE TO LOG INDICATOR
(Halfword)		
IFRSSRD	24	DEFAULT ROW COUNT
IFRSSCD	80	DEFAULT COLUMN COUNT
Type Codes in IFRLPTYP (Byte)		
IFRLPLST	1	IFRLPTYP=LIST
IFRLPPUR	2	IFRLPTYP=PURGE

LOGDS

DSILOGDS: MAPS THE NCCF DISK LOG
BOUNDARY: FULLWORD
LENGTH: 48 BYTES (X'30') + CALLERS TEXT RECORD
INCLUDED BLOCKS: NONE

OFFSETS		TYPE	LENGTH	NAME	DESCRIPTION
0	(0)	STRUCTURE	48	DSILOGDS	FORMAT OF LOG RECORD
0	(0)	CHARACTER	16	LOGKEY	KEY OF THE RECORD
0	(0)	SIGNED	4	LOGKEYDT	DATE: FORMAT = 00YYDDDC
4	(4)	SIGNED	4	LOGKEYTM	TIME: FORMAT = HHMMSS0C
4	(4)	CHARACTER	3		PLACE HOLDER
7	(7)	UNSIGNED	1	LOGUNIQ	0C UNLESS A DUPLICATE TIME DATE STAMP HAS BEEN OBTAINED
8	(8)	CHARACTER	8	LOGKEYEX	USER KEY FIELD
16	(10)	SIGNED	2	LOGDISP	DISPLACEMENT OF MESSAGE TEXT
18	(12)	CHARACTER	1	LOGIND	RECORD INDICATOR
19	(13)	CHARACTER	1	LOGMTYPE	MESSAGE TYPE
20	(14)	CHARACTER	8	LOGLUNAM	LUNAME FROM TVB
28	(1C)	SIGNED	4	LOGTIME	TIME FROM CALLER'S BFR HEADER (FORMAT: HHMMSS0C)
32	(20)	CHARACTER	8	LOGDOMID	DOMAIN ID
32	(20)	CHARACTER	7		PLACE HOLDER
39	(27)	CHARACTER	1	LOGAUTHP	% FOR MSG FOR AUTH OPERATOR
40	(28)	CHARACTER	8	LOGOPID	OPERATOR ID
48	(30)	CHARACTER	0	LOGTEXT	CALLERS TEXT RECORD

CROSS REFERENCE

DSILOGDS	0	(0)
LOGAUTHP	39	(27)
LOGDISP	16	(10)
LOGDOMID	32	(20)
LOGIND	18	(12)
LOGKEY	0	(0)
LOGKEYDT	0	(0)
LOGKEYEX	8	(8)
LOGKEYTM	4	(4)
LOGLUNAM	20	(14)
LOGMTYPE	19	(13)
LOGOPID	40	(28)
LOGTEXT	48	(30)
LOGTIME	28	(1C)
LOGUNIQ	7	(7)

MVT

DSIMVT: MAPS THE MAIN VECTOR TABLE
BOUNDARY: DOUBLEWORD
LENGTH: 2248 BYTES X'8C8'
POINTED TO BY: TVB (TVBMVT)
INCLUDED BLOCKS: CBH (MVTCBH)
SWB (MVTSWBM)

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
0 (0)	STRUCTURE	2248	DSIMVT	MAIN VECTOR TABLE
0 (0)	CHARACTER	4	MVTCBH	NCCF CTL BLK HDR
4 (4)	CHARACTER	4	MVTVER	NCCF VERSION INFORMATION
8 (8)	A-ADDRESS	4	MVTDPRAD	ADDR OF DSIDPR (VSE ONLY)
12 (C)	A-ADDRESS	4	MVTSNT	ADDR OF SPAN NAME TABLE
16 (10)	A-ADDRESS	4	MVTOIT	ADDR OPERATOR ID TABLE
20 (14)	A-ADDRESS	4	MVTART	ADDR OF AUTH ROUTING TABLE
24 (18)	A-ADDRESS	4	MVTDQT	ADDR DOMAIN QUALIFICATION TABLE
28 (1C)	A-ADDRESS	4	MVTDDT	ADDR DOMAIN DEFINITION TABLE
32 (20)	A-ADDRESS	4	MVTSCT	ADDR SYSTEM COMMAND TABLE
36 (24)	A-ADDRESS	4	MVTCDNID	ADDR OF CDRMNAME-TO-NCCFID TALBE
40 (28)	A-ADDRESS	4	MVTGRTTB	ADDR OF GLOBAL CNM ROUTING TABLE
44 (2C)	A-ADDRESS	4		RESERVED CRITICAL FIELD
48 (30)	A-ADDRESS	4		RESERVED CRITICAL FIELD
52 (34)	A-ADDRESS	4	MVTSVL	ADDR SERVICE-RTN VECTOR LIST
56 (38)	BITSTRING	8	MVTCWBQ	CWB CHAIN INFO
56 (38)	SIGNED	4	MVTCBOTH	COUNT CONTROL FIELDS
56 (38)	SIGNED	2	MVTCLIMT	CONTROL BLOCK COUNT LIMIT
58 (3A)	SIGNED	2	MVTCCOUN	CONTROL BLOCK CURRENT COUNT
60 (3C)	A-ADDRESS	4	MVTLCWB	ADDR OF 1ST CWB ON THE CHAIN
64 (40)	BITSTRING	8	MVTSWBQ	SWB CHAIN INFO
64 (40)	SIGNED	4	MVTSBOTH	COUNT CONTROL FIELDS
64 (40)	SIGNED	2	MVTSLIMT	CONTROL BLOCK COUNT LIMIT
66 (42)	SIGNED	2	MVTSCOUN	CONTROL BLOCK CURRENT COUNT
68 (44)	A-ADDRESS	4	MVTLSWB	ADDR OF 1ST SWB ON THE CHAIN
72 (48)	A-ADDRESS	4	MVTTVB	TVB CHAIN HEADER
76 (4C)	CHARACTER	8	MVTNOSPQ	QNAME FOR ENQ ON THE TVB CHAIN
76 (4C)	CHARACTER	8	MVTNCCFQ	QNAME FOR ENQ ON THE TVB CHAIN
84 (54)	CHARACTER	18	MVTTVBRN	RNAME FOR ENQ ON THE TVB CHAIN

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
102 (66)	SIGNED	2	MVTSCMAX	MAXIMUM SCOPECLASS VALUE
104 (68)	A-ADDRESS	4	MVTBPDCT	BPAM DSIDCT CHAIN HEADER
108 (6C)	A-ADDRESS	4		RESERVED FOR FUTURE USE
112 (70)	A-ADDRESS	4	MVTMCC	ADDR NCCF MESSAGE CSECT
116 (74)	A-ADDRESS	4	MVTEX01	USER OST INPUT EXIT RTN (DSIEX01)
120 (78)	A-ADDRESS	4	MVTEX02	USER OST OUTPUT EXIT RTN (DSIEX02)
124 (7C)	A-ADDRESS	4	MVTEX03	USER CMD INVOCATION EXIT RTN (DSIEX03)
128 (80)	A-ADDRESS	4	MVTEX04	USER LOG SERVICE EXIT RTN (DSIEX04)
132 (84)	A-ADDRESS	4	MVTEX05	USER POI OUTPUT EXIT RTN (DSIEX05)
136 (88)	A-ADDRESS	4	MVTEX06	USER POI INPUT EXIT RTN (DSIEX06)
140 (8C)	A-ADDRESS	4	MVTEX07	USER NNT OUTPUT EXIT RTN (DSIEX07)
144 (90)	A-ADDRESS	4	MVTEX08	USER NNT INPUT EXIT RTN (DSIEX08)
148 (94)	A-ADDRESS	4	MVTEX09	USER SYSTEM CONSOLE OUTPUT EXIT RTN (DSIEX09)
152 (98)	A-ADDRESS	4	MVTEX10	USER SYSTEM CONSOLE INPUT EXIT RTN (DSIEX10)
156 (9C)	A-ADDRESS	4	MVTEX11	USER PPT INPUT EXIT RTN (DSIEX11)
160 (A0)	A-ADDRESS	4	MVTEX12	OPERATOR LOGON EXIT RTN (DSIEX12)
164 (A4)	A-ADDRESS	4	MVTEX13	MESSAGE RECEIVER EXIT RTN (DSIEX13)
168 (A8)	A-ADDRESS	4	MVTEX14	OST TERMINATION EXIT RTN (DSIEX14)
172 (AC)	A-ADDRESS	4	MVTEX15	VSE ONLY TASK TERMINATION
176 (B0)	A-ADDRESS	4		RESERVED CRITICAL FIELD
180 (B4)	A-ADDRESS	4		RESERVED CRITICAL FIELD
184 (B8)	A-ADDRESS	4		RESERVED CRITICAL FIELD
188 (BC)	A-ADDRESS	4		RESERVED CRITICAL FIELD
192 (C0)	A-ADDRESS	4		RESERVED CRITICAL FIELD
196 (C4)	CHARACTER	1	MVTIND	INDICATOR FLAGS
1...			MVTINIT	1 = NCCF INIT IN PROGRESS
.1..			MVTTERM	1 = NCCF TERM IN PROGRESS
..1.			MVTCLOSE	1 = NCCF CLOSE ISSUED
...1			MVTTPEND	1 = TPEND HAS BEEN ENTERED
.... 1...			MVTSHTDN	1 = NCCF IS IN SHUTDOWN MODE
.... .1..			MVTWRMST	1 = NCCF IN WARM START MODE
.... ..1.			MVTWLOPN	1 = NCCF LOG IS ACTIVE

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
....1			MVTRCF	1 = RACF LOGON PROC
197 (C5)	CHARACTER	1	MVTSPCHR	00 = NO SUPPRESS CHAR OTHER = SUPPRESS CHAR
198 (C6)	SIGNED	2	MVTOCNT	MAX NUMBER OF ENTRIES IN DSIOIT
200 (C8)	SIGNED	2	MVTPRFCT	MAX NUMBER OF PROFILES DEFINED
202 (CA)	SIGNED	2	MVTDRTRY	MAX DEVICE RETRY COUNT
204 (CC)	SIGNED	4	MVTPRID	CURRENT DSM PRID VALUE
208 (D0)	A-ADDRESS	4	MVTMNTLE	MAIN TASK LOGON EXIT ADDRESS
212 (D4)	A-ADDRESS	4	MVTCMPRG	POINTER TO TCAM PURGE I/O PARAMETER LIST
216 (D8)	A-ADDRESS	4		RESERVED
220 (DC)	SIGNED	2	MVTMRC	MAX SUBTASK REINSTATE COUNT
222 (DE)	SIGNED	2	MVTTCNT	MAX NUMBER OF TVBS FOR THIS NCCF
224 (E0)	SIGNED	2	MVTSCNT	COUNT OF ENTRIES IN DSISNT (SPAN COUNT)
226 (E2)	SIGNED	2	MVTSNTLN	LENGTH OF EACH ENTRY IN DSISNT
228 (E4)	SIGNED	2	MVTRCNT	COUNT OF ENTRIES IN DSIART (RESOURCE COUNT)
230 (E6)	SIGNED	2	MVTARTLN	LENGTH OF EACH ENTRY IN DSIART
232 (E8)	SIGNED	2	MVTMLGON	MAX NO OF REENTRY OF LOGON PARMS FOR AN ATTEMPTED LOGON PROCESS
234 (EA)	SIGNED	2	MVTCDSES	MAX NO CROSS-DOMAIN SESSIONS ALLOWED FOR THIS NCCF
236 (EC)	CHARACTER	9	MVTCURAP	CURRENT DOMAIN VTAM APPLID (MNT)
236 (EC)	UNSIGNED	1	MVTCURAL	CURRENT DOMAIN APPLID LENGTH
237 (ED)	CHARACTER	8	MVTCURAN	VTAM APPLID NAME
245 (F5)	CHARACTER	9	MVTCURPW	CURRENT DOMAIN VTAM PASSWD (MNT)
245 (F5)	UNSIGNED	1	MVTCURPL	CURRENT DOMAIN PASSWD LENGTH
246 (F6)	CHARACTER	8	MVTCURPN	VTAM PASSWORD
254 (FE)	CHARACTER	2		ALIGNMENT
256 (100)	A-ADDRESS	4	MVTSWB	ADDR OF SWB USED BY DSIMNT
260 (104)	A-ADDRESS	4	MVTACB	ADDR OF ACB USED BY DSIMNT
264 (108)	A-ADDRESS	4	MVTGMSG	ADDR OF SPECIAL ERROR MESSAGE
268 (10C)	BITSTRING	8	MVTTOD	TOD CLOCK AT NCCF START
276 (114)	SIGNED	4		RESERVED
280 (118)	A-ADDRESS	4	MVTUFLD	NCCF USER FIELD
284 (11C)	A-ADDRESS	4		RESERVED
288 (120)	A-ADDRESS	4	MVTLAC	ADDR OF LOGON AUTH MODULE
292 (124)	A-ADDRESS	4	MVTGFMG1	WTOLIST MSG1 FOR GETMAIN FAILURE

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
296 (128)	SIGNED	4	MVTGFAIL	GETMAIN FAILURE COUNT
300 (12C)	A-ADDRESS	4	MVTGFMG2	WTOLIST MSG2 FOR GETMAIN FAILURE
304 (130)	A-ADDRESS	4	MVTPOOL	STORAGE POOL INDEX ADDRESS
308 (134)	SIGNED	4	MVTTASKC	TASK COUNT FOR ACF/TCAM ID
312 (138)	A-ADDRESS	4	MVTLOGGR	LOGGER BUFFER PTR
316 (13C)	SIGNED	2	MVTTVBSZ	TOTAL AREA GOTTEN FOR ALL TVBS
318 (13E)	CHARACTER	1	MVTMETH	ACCESS METHOD IN USE
319 (13F)	CHARACTER	1	MVTSNALV	SNA LEVEL IN USE
320 (140)	CHARACTER	8	MVTTPROC	TCAM AMH TPROCESS NAME
328 (148)	A-ADDRESS	4	MVTTLGNQ	TCAM LOGON QUEUE
332 (14C)	CHARACTER	2	MVTTSVC	NCCF USER SVC
332 (14C)	BITSTRING	1	MVTTSVCO	SVC OP CODE
333 (14D)	UNSIGNED	1	MVTTSVCN	SVC NUMBER HEX
334 (14E)	SIGNED	2		RESERVED
336 (150)	A-ADDRESS	4	MVTCTVB	ADDR OF TCT TVB
340 (154)	A-ADDRESS	4	MVTNDT	ADDR OF NCCF DOMAIN TABLE
344 (158)	SIGNED	4	MVTEXTRN	EXTERNAL INFO
344 (158)	BITSTRING	1	MVTDELAY	OPEN/CLOSE DELAY
345 (159)	UNSIGNED	1	MVTQREQ	EXTERNAL QUEUE REQUEST
348 (15C)	A-ADDRESS	4	MVTCCL	ADDR OF DSICCL

THE FOLLOWING FIELDS SUPPORT MNT REWRITE

352 (160)	A-ADDRESS	4	MVTSFXP	SUFFIX TABLE ADDR
356 (164)	CHARACTER	9	MVTAPID	APID NAME
356 (164)	CHARACTER	1	MVTAPLEN	APID LENGTH
357 (165)	CHARACTER	8	MVTSTNAM	NAME
365 (16D)	CHARACTER	3		RESERVED
368 (170)	A-ADDRESS	4	MVTSFXAD	SUFFIX ENTRY ADDR
372 (174)	CHARACTER	20	MVTCBL	API CB LENGTHS
372 (174)	SIGNED	4	MVTACBL	ACB LENGTH
376 (178)	SIGNED	4	MVTRPLL	RPL LENGTH
380 (17C)	SIGNED	4	MVTNIBL	NIB LENGTH
384 (180)	SIGNED	4	MVTEXLL	EX LIST LENGTH
388 (184)	SIGNED	4	MVTSCBL	SCB LENGTH
392 (188)	A-ADDRESS	4	MVTECBLS	ECB LIST ADDR
396 (18C)	SIGNED	4	MVTECBLN	ECB LIST LENGTH
400 (190)	A-ADDRESS	4	MVTSFXTB	APID SUFFIX TAB ADDR

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
404 (194)	SIGNED	4	MVTSFXLN	APID SUFFIX TAB LENGTH
408 (198)	CHARACTER	20	MVTECBG	GLOBAL ECBS
408 (198)	SIGNED	4	MVTECBT	TERMINATE ECB
412 (19C)	SIGNED	4	MVTECBW	WTOR ECB
416 (1A0)	SIGNED	4	MVTSFECB	STOP FORCE ECB
420 (1A4)	SIGNED	4	MVTECBD	DETATCH ECB
424 (1A8)	SIGNED	4	MVTECBA	ATTACH ECB
428 (1AC)	A-ADDRESS	4	MVTTVBM	MAIN TASK TVB ADDR
432 (1B0)	CHARACTER	64	MVTBLDLL	BLDL LIST AREA

ENTRIES IN THE FOLLOWING TABLES ARE ORDERED AND ACCESSED BY TASK TYPE VALUES AS DEFINED IN DSICBH. THEY ARE FILLED IN OR CALCULATED BY THE MAIN TASK.

496 (1F0)	CHARACTER	28	MVTRPLCT	COUNT OF RPLS BY TASK
496 (1F0)	SIGNED	4	MVTRPLP	PPT
500 (1F4)	SIGNED	4	MVTRPLN	NNT
504 (1F8)	SIGNED	4	MVTRPLS	OST
508 (1FC)	SIGNED	4	MVTRPLH	HCT
512 (200)	SIGNED	4	MVTRPLT	TCT
516 (204)	SIGNED	4	MVTRPLO	OPT
520 (208)	SIGNED	4	MVTRPLM	MNT
524 (20C)	CHARACTER	28	MVTNIBCT	COUNT OF NIB BY TASK
524 (20C)	SIGNED	4	MVTNIBP	PPT
528 (210)	SIGNED	4	MVTNIBN	NNT
532 (214)	SIGNED	4	MVTNIBS	OST
536 (218)	SIGNED	4	MVTNIBH	HCT
540 (21C)	SIGNED	4	MVTNIBT	TCT
544 (220)	SIGNED	4	MVTNIBO	OPT
548 (224)	SIGNED	4	MVTNIBM	MNT
552 (228)	CHARACTER	28	MVTSCBCT	S C B S
552 (228)	SIGNED	4	MVTSCBP	PPT
556 (22C)	SIGNED	4	MVTSCBN	NNT
560 (230)	SIGNED	4	MVTSCBS	OST
564 (234)	SIGNED	4	MVTSCBH	HCT
568 (238)	SIGNED	4	MVTSCBT	TCT
572 (23C)	SIGNED	4	MVTSCBO	OPT

OFFSETS		TYPE	LENGTH	NAME	DESCRIPTION
576	(240)	SIGNED	4	MVTSCBM	MNT
580	(244)	CHARACTER	28	MVTTOTLT	TOTAL CONTROL BLOCK STORAGE BY TASK
580	(244)	SIGNED	4	MVTTOTP	PPT
584	(248)	SIGNED	4	MVTTOTN	NNT
588	(24C)	SIGNED	4	MVTTOTS	OST
592	(250)	SIGNED	4	MVTTOTH	HCT
596	(254)	SIGNED	4	MVTTOTT	TCT
600	(258)	SIGNED	4	MVTTOTO	OPT
604	(25C)	SIGNED	4	MVTTOTM	MNT
608	(260)	CHARACTER	28	MVTTIBLT	COUNT OF TIB LENGTH BY TASK
608	(260)	SIGNED	4	MVTTIBP	PPT
612	(264)	SIGNED	4	MVTTIBN	NNT
616	(268)	SIGNED	4	MVTTIBS	OST
620	(26C)	SIGNED	4	MVTTIBH	HCT
624	(270)	SIGNED	4	MVTTIBT	TCT
628	(274)	SIGNED	4	MVTTIBO	OPT
632	(278)	SIGNED	4	MVTTIBM	MNT
636	(27C)	CHARACTER	1	MVTABLOK	ABEND LOCK FOR DSIMAB
637	(27D)	CHARACTER	139	MVTUXBUF	DSIEX10 BUFFER
637	(27D)	BITSTRING	4	MVTUXMBL	MLENG AND BLENG
641	(281)	BITSTRING	4	MVTUXTDS	TDISP
645	(285)	CHARACTER	16		PADDING WORDS
661	(295)	CHARACTER	115	MVTWTORA	WTOR REPLY AREA
776	(308)	CHARACTER	28	MVTUXPLS	DSIEX10 DSIUSE PLIST
776	(308)	BITSTRING	4	MVTUXCBL	CBH HDR AND LENGTH
780	(30C)	A-ADDRESS	4	MVTUXBFA	PTR TO MSG BUFFER
784	(310)	A-ADDRESS	4	MVTUXLUN	PTR TO SYSOP LUNAME
788	(314)	A-ADDRESS	4	MVTUXOID	PTR TO SYSOP OPID
792	(318)	A-ADDRESS	4	MVTUXSWB	PTR TO SWB
796	(31C)	A-ADDRESS	4	MVTUXTVB	PTR TO TVB
800	(320)	A-ADDRESS	4	MVTUXPDB	PTR TO PDB
804	(324)	CHARACTER	72	MVTMSVA1	MAIN TASK SAVE 1
876	(36C)	CHARACTER	72	MVTMSVA2	MAIN TASK SAVE 2
948	(3B4)	CHARACTER	72	MVTMSVA3	MAIN TASK SAVE 3
1020	(3FC)	CHARACTER	200	MVTMTWKA	MAIN TASK WORK AREA
1220	(4C4)	CHARACTER	136	MVTMSGPL	MSG PARM LIST AREA
1356	(54C)	CHARACTER	24	MVTMNTTP	MAIN TASK MNT TERM PARM LIST

OFFSETS		TYPE	LENGTH	NAME	DESCRIPTION
1356	(54C)	CHARACTER	9	MVTMNTPU	PU NAME
1365	(555)	CHARACTER	8	MVTMNTLU	LU NAME
1373	(55D)	CHARACTER	3	MVTMNTPD	PADDING
1376	(560)	A-ADDRESS	4	MVTMNTSW	MNT SWB ADDR
1380	(564)	SIGNED	4		RESERVED
1384	(568)	CHARACTER	12	MVTCPRGL	TCAM TVB PURGE LIST
1384	(568)	A-ADDRESS	4	MVTTCTVB	TVB TO PURGE
1388	(56C)	SIGNED	4	MVTCECBM	TCAM ECB NNT WAIT
1392	(570)	SIGNED	4	MVTCECBT	TCAM ECB TCT WAIT
1396	(574)	CHARACTER	600	MVTSWBM	WTOR REPLY ARES
1996	(7CC)	SIGNED	4	MV INSV	R15 SAVE FOR GENCB FAILURE
2000	(7D0)	SIGNED	4	MVTMAJSV	R0 SAVE FOR GENCB FAILURE
2004	(7D4)	CHARACTER	156	MVTMSGBF	MNT WCS MSG BFR

FOR TIMER-INITIATED COMMANDS

2160	(870)	CHARACTER	4	MVTBTIME	BASE TIME VALUE (PACKED)
2160	(870)	CHARACTER	1	MVTBTMHH	(HH) HOURS
2161	(871)	CHARACTER	1	MVTBTMMM	(MM) MINUTS
2162	(872)	CHARACTER	1	MVTBTMSS	(SS) SECONDS
2163	(873)	CHARACTER	1	MVTBTM0C	(0C) TENTHS/SIGN
2164	(874)	CHARACTER	8	MVTBSTCK	BASE TIME-STCK FORMAT
2172	(87C)	CHARACTER	8	MVTPOPT	POPTIME-STCK FORMAT

ADDRESS LIST OF MAIN TASK MODULES

2180	(884)	CHARACTER	4		RESERVED
2184	(888)	A-ADDRESS	4	MVTMIN02	ADDR OF MNT INIT STG2
2188	(88C)	A-ADDRESS	4	MVTMTE	ADDR OF MNT TERMINATION PROC
2192	(890)	A-ADDRESS	4	MVTMMP	ADDR OF MNT MSG PROC
2196	(894)	A-ADDRESS	4	MVTMAB	ADDR OF MNT ABEND PROC
2200	(898)	A-ADDRESS	4	MVTMCB	ADDR OF MNT BLD CTL BLKS PROC
2204	(89C)	A-ADDRESS	4	MVTMLG	ADDR OF MNT LOGON EXIT PROC
2208	(8A0)	A-ADDRESS	4	MVTMRP	ADDR OF MNT RPL EXIT PRO@D705709
2212	(8A4)	A-ADDRESS	4	MVTMLT	ADDR OF MNT LOST TERM EXIT PROC
2216	(8A8)	A-ADDRESS	4	MVTMTP	ADDR OF MNT TPEND EXIT PROC
2220	(8AC)	A-ADDRESS	4	MVTMEX	ADDR OF MNT END OF TASK EXIT PROC

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
2224 (8B0)	A-ADDRESS	4	MVTMNS	ADDR OF MNT NET SRVS EXIT PROC
2228 (8B4)	A-ADDRESS	4	MVTMST	ADDR OF MNT SUBTASK SERVICES PROC
2232 (8B8)	A-ADDRESS	16		RESERVED

CROSS REFERENCE

DSIMVT	0 (0)	MVTIND	196 (C4)	MVTSCBN	556(22C)
MVTABLOK	636(27C)	MVTINIT	196 X'80'	MVTSCBO	572(23C)
MVTACB	260(104)	MVTLAC	288(120)	MVTSCBP	552(228)
MVTACBL	372(174)	MVTLCWB	60 (3C)	MVTSCBS	560(230)
MVTAPID	356(164)	MVTLOGGR	312(138)	MVTSCBT	568(238)
MVTAPLEN	356(164)	MVTLSWB	68 (44)	MVTSCMAX	102 (66)
MVTART	20 (14)	MVTMAB	2196(894)	MVTSCNT	224 (E0)
MVTARTLN	230 (E6)	MVTMAJSV	2000(7D0)	MVTSCOUN	66 (42)
MVTBLDLL	432(1B0)	MVTMCB	2200(898)	MVTSCT	32 (20)
MVTBPDCT	104 (68)	MVTMCC	112 (70)	MVTSFECB	416(1A0)
MVTBSTCK	2164(874)	MVTMETH	318(13E)	MVTSFXAD	368(170)
MVTBTIME	2160(870)	MVTMEX	2220(8AC)	MVTSFXLN	404(194)
MVTBTMHH	2160(870)	MVTMINSV	1996(7CC)	MVTSFXP	352(160)
MVTBTMMM	2161(871)	MVTMIN02	2184(888)	MVTSFXTB	400(190)
MVTBTMSS	2162(872)	MVTMLG	2204(89C)	MVTSHTDN	196 X'08'
MVTBTM0C	2163(873)	MVTMLGON	232 (E8)	MVTSLIMT	64 (40)
MVTCBH	0 (0)	MVTMLT	2212(8A4)	MVTSNALV	319(13F)
MVTCBL	372(174)	MVTMMP	2192(890)	MVTSNT	12 (C)
MVTCBOTH	56 (38)	MVTMNS	2224(8B0)	MVTSNTLN	226 (E2)
MVTCCL	348(15C)	MVTMNTLE	208 (D0)	MVTSPCHR	197 (C5)
MVTCCOUN	58 (3A)	MVTMNTLU	1365(555)	MVTSTNAM	357(165)
MVTCDNID	36 (24)	MVTMNTPD	1373(55D)	MVTSVL	52 (34)
MVTCDSES	234 (EA)	MVTMNTPU	1356(54C)	MVTSWB	256(100)
MVTCECBM	1388(56C)	MVTMNTSW	1376(560)	MVTSWBM	1396(574)
MVTCECBT	1392(570)	MVTMNTTP	1356(54C)	MVTSWBQ	64 (40)
MVTCLIMT	56 (38)	MVTMRC	220 (DC)	MVTTASKC	308(134)
MVTCLOSE	196 X'20'	MVTMRP	2208(8A0)	MVTTCNT	222 (DE)
MVTCMPRG	212 (D4)	MVTMSGBF	2004(7D4)	MVTTCTVB	1384(568)
MVTCPRGL	1384(568)	MVTMSGPL	1220(4C4)	MVTTERM	196 X'40'
MVTCTVB	336(150)	MVTMST	2228(8B4)	MVTTIBH	620(26C)
MVTCURAL	236 (EC)	MVTMSVA1	804(324)	MVTTIBLT	608(260)
MVTCURAN	237 (ED)	MVTMSVA2	876(36C)	MVTTIBM	632(278)
MVTCURAP	236 (EC)	MVTMSVA3	948(3B4)	MVTTIBN	612(264)
MVTCURPL	245 (F5)	MVTMTE	2188(88C)	MVTTIBO	628(274)
MVTCURPN	246 (F6)	MVTMTP	2216(8A8)	MVTTIBP	608(260)
MVTCURPW	245 (F5)	MVTMTWKA	1020(3FC)	MVTTIBS	616(268)
MVTCWBQ	56 (38)	MVTNCCFQ	76 (4C)	MVTTIBT	624(270)
MVTDDT	28 (1C)	MVTNDT	340(154)	MVTTLGNQ	328(148)
MVTDELAY	344(158)	MVTNIBCT	524(20C)	MVTTOD	268(10C)
MVTDPRAD	8 (8)	MVTNIBH	536(218)	MVTTOTH	592(250)
MVTDQT	24 (18)	MVTNIBL	380(17C)	MVTTOTLT	580(244)
MVTDRTRY	202 (CA)	MVTNIBM	548(224)	MVTTOTM	604(25C)
MVTECBA	424(1A8)	MVTNIBN	528(210)	MVTTOTN	584(248)
MVTECBD	420(1A4)	MVTNIBO	544(220)	MVTTOTO	600(258)
MVTECBG	408(198)	MVTNIBP	524(20C)	MVTTOTP	580(244)
MVTECBLN	396(18C)	MVTNIBS	532(214)	MVTTOTS	588(24C)
MVTECBLS	392(188)	MVTNIBT	540(21C)	MVTTOTT	596(254)
MVTECBT	408(198)	MVTNOSPQ	76 (4C)	MVTTPEND	196 X'10'
MVTECBW	412(19C)	MVTOCNT	198 (C6)	MVTTPROC	320(140)
MVTEXLL	384(180)	MVTOIT	16 (10)	MVTTSVC	332(14C)
MVTEXTRN	344(158)	MVTPOOL	304(130)	MVTTSVCN	333(14D)
MVTEX01	116 (74)	MVTPOPT	2172(87C)	MVTTSVCO	332(14C)
MVTEX02	120 (78)	MVTPRFCT	200 (C8)	MVTTVB	72 (48)
MVTEX03	124 (7C)	MVTPRID	204 (CC)	MVTTVBM	428(1AC)
MVTEX04	128 (80)	MVTQREQ	345(159)	MVTTVBRN	84 (54)
MVTEX05	132 (84)	MVTRCF	196 X'01'	MVTTVBSZ	316(13C)
MVTEX06	136 (88)	MVTRCNT	228 (E4)	MVTUFLD	280(118)
MVTEX07	140 (8C)	MVTRPLCT	496(1F0)	MVTUXBFA	780(30C)
MVTEX08	144 (90)	MVTRPLH	508(1FC)	MVTUXBUF	637(27D)
MVTEX09	148 (94)	MVTRPLL	376(178)	MVTUXCBL	776(308)
MVTEX10	152 (98)	MVTRPLM	520(208)	MVTUXLUN	784(310)
MVTEX11	156 (9C)	MVTRPLN	500(1F4)	MVTUXMBL	637(27D)
MVTEX12	160 (A0)	MVTRPLO	516(204)	MVTUXOID	788(314)
MVTEX13	164 (A4)	MVTRPLP	496(1F0)	MVTUXPDB	800(320)
MVTEX14	168 (A8)	MVTRPLS	504(1F8)	MVTUXPLS	776(308)
MVTEX15	172 (AC)	MVTRPLT	512(200)	MVTUXSWB	792(318)
MVTGFAIL	296(128)	MVTSBOTH	64 (40)	MVTUXTDS	641(281)
MVTGFMG1	292(124)	MVTSCBCT	552(228)	MVTUXTVB	796(31C)
MVTGFMG2	300(12C)	MVTSCBH	564(234)	MVTVER	4 (4)
MVTGMSG	264(108)	MVTSCBL	388(184)	MVTWLOPN	196 X'02'
MVTGRTRN	176 (B0)	MVTSCBM	576(240)	MVTWRMST	196 X'04'
MVTGRTTB	40 (28)			MVTWTORA	661(295)

Constants in DSIMVT

NAME	VALUE	MEANING
Masks For Setting and Testing MVTIND Flags (Bit 1)		
MVTON	B'1'	FUNCTION IS ACTIVE
MVTOFF	B'0'	FUNCTION IS NOT ACTIVE
ACF/TCAM Trigger Characters--Read Only (Bit 64)		
MVTOPCHR	X'3333333333333333'	TCAM OPCTL CHARACTER STRING--READ ONLY
MVTSSCHR	X'333333C4E2C9E2E2'	SCREEN SIZE
MVTFWCHR	X'333333C6E6C4D9D8'	CSMI FORWARD
Masks For Setting and Testing MVTMETH (Char 1)		
MVTVTAM	'V'	ACF/VTAM METHOD
MVTTCAM	'T'	ACF/TCAM METHOD
Constants For Setting and Testing DSILCM Fields (Fullword)		
LCMCASE1	1	CONTROL BLOCK IS TVB WITH OPID
LCMCASE2	2	CONTROL BLOCK IS TVB WITH LU
LCMCASE3	3	CONTROL BLOCK IS NEXT TVB
LCMCASE4	4	CONTROL BLOCK IS SWB
LCMCASE5	5	CONTROL BLOCK IS CWB
LCMCASE6	6	AUTHORIZATION LOCATION REQUEST
LCMCASE7	7	FIND NEXT OPT.TASK
LCMCASE8	8	FIND TCT TVB
LCMCASE9	9	FIND PPT TVB
LCMFREE	1	FREE A CWB/SWB
LCMGET	0	LOCATE A CWB/SWB
LCMGOOD	0	CONTROL BLOCK FOUND
LCMINACT	4	TASK LOCATED IS INACTIVE
LCMBAD	8	NOT FOUND-- UNSUCCESSFUL

PDB

DSIPDB: MAPS THE PARSE DESCRIPTOR BLOCK USED TO ANALYZE ALL INPUT TO NCCF
BOUNDARY: DOUBLEWORD
LENGTH: 20 BYTES (X'14')
POINTED TO BY: CLB (CLBPSPDB) FROM &PAUSE CONTROL STATEMENT
(CLBGOPDB) FROM GO COMMAND
(CLBAPDB) ALTERNATE PDB
CWB (CWBPDB)
MVT (MVTUXPDB)
SWB (PSOPTAB)
(RDMPDB) PDB INPUT
(PAMPDB) INPUT PDB
(PAMENTRY) PDB ENTRY
TIB (TIBNCPDB) NORMAL COMMAND
(TIBMRPDB) MESSAGE RECEIVED AND RECEIVED COMMAND
(TIBICPDB) IMMEDIATE COMMAND
(TIBAGPDB) AGAIN PROCESSING
USE (USERPDB) PDB PASSED TO USER EXIT

INCLUDED BLOCKS: CBH (PDBCBH)

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
0	(0) STRUCTURE	16	DSIPDB	PARSE DESCRIPTOR BLOCK
0	(0) CHARACTER	16	PDBHDR	OVERALL PDB HEADER
0	(0) CHARACTER	4	PDBCBH	CONTROL BLOCK HEADER
4	(4) A-ADDRESS	4	PDBCMDA	ADDR OF COMMAND ROUTINE
8	(8) A-ADDRESS	4	PDBBUFA	ADDR OF INPUT BUFFER
12	(C) CHARACTER	1	PDBFLAGS	INDICATOR FLAGS
	1...		PDBIMMED	1=IMMEDIATE COMMAND 0=REGULAR COMMAND
13	(D) CHARACTER	1		FOR FUTURE USE/ALIGNMENT
14	(E) SIGNED	2	PDBNOENT	NUMBER OF ENTRIES IN THIS PDB
16	(10) CHARACTER	0	PDBENTRY	MULTIPLE ENTRIES
16	(10) CHARACTER	1	PDBTYPE	TYPE OF ENTRY (BY DELIMITER)
17	(11) UNSIGNED	1	PDBLENG	LENGTH OF THIS ENTRY
18	(12) SIGNED	2	PDBDISP	DISPLACEMENT TO BEGINNING OF CHARACTER STRING IN THE COMMAND

CROSS REFERENCE

DSIPDB	0	(0)
PDBBUFA	8	(8)
PDBCBH	0	(0)
PDBCMDA	4	(4)
PDBDISP	18	(12)
PDBENTRY	16	(10)
PDBFLAGS	12	(C)
PDBHDR	0	(0)
PDBIMMED	12	X'80'
PDBLENG	17	(11)
PDBNOENT	14	(E)
PDBTYPE	16	(10)

SCE

DSISCE: MAPS THE SYSTEM CONNMAND ELEMENT USED TO IDENTIFY
NCCF COMMAND PROCESSORS, USER-WRITTEN COMMAND
PROCESSORS, AND USER-WRITTEN COMMAND LISTS
BOUNDARY: DOUBLEWORD
LENGTH: 20 BYTES (X'14')
INCLUDED BLOCKS: CBH (DSICBH)

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
0 (0)	STRUCTURE	20	DSISCE	SYSTEM COMMAND ELEMENT
0 (0)	CHARACTER	8	SCEVERB	COMMAND VERB
8 (8)	CHARACTER	8	SCELNAME	LOAD MODULE NAME
16 (10)	A-ADDRESS	4	SCECADDR	ADDR OF THE COMMAND PROCESSOR

CROSS REFERENCE

DSISCE	0 (0)
SCECADDR	16 (10)
SCELNAME	8 (8)
SCEVERB	0 (0)

SWB

DSISWB: MAPS THE SERVICE ROUTINE WORK BLOCK USED BY NCCF
SERVICE ROUTINES
BOUNDARY: DOUBLEWORD
LENGTH: 600 BYTES (X'258')
POINTED TO BY: CLB (CLBSWB)
CWB (CWBSWB)
MVT (MVTLSWB) FIRST SWB ON CHAIN
(MVTSWB) SWB USED BY DSIMNT
(MVTMNTSW) MNT SWB ADDRESS
(MVTUXSWB)
NMB (NMBSWBA) NETWORK MANAGEMENT SERVICES SWB
TIB (TIBEXSWB) EXIT PROCESSING
(TIBNPSWB) NORMAL PROCESSING
(TIOPSSWB) PSS SWB
(TIMXNNSW)
(MMPPLSWB)
USE (USERSWB) IN USER EXIT
INCLUDED BLOCKS: CBH (SWBCBH)

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
0 (0)	STRUCTURE	600	DSISWB	SERVICE ROUTINE WORK BLOCK
0 (0)	CHARACTER	4	SWBCBH	CONTROL BLOCK HEADER
4 (4)	A-ADDRESS	72	SWBSAVEA	STANDARD SAVE AREA (18 ITEMS, EACH FIXED 31)
76 (4C)	A-ADDRESS	4	SWBNEXT	ADDR OF NEXT SWB ON CHAIN
80 (50)	A-ADDRESS	4	SWBTIB	ADDR OF CALLER'S TIB
84 (54)	CHARACTER	256	SWBADATD	AUTOMATIC WORK AREA
340 (154)	CHARACTER	256	SWBPLIST	PARAMETER LIST OVERLAY AREA
596 (254)	A-ADDRESS	4		RESERVED

DISK SERVICES INVOCATION PARAMETER LIST

340 (154)	STRUCTURE	12	DKSPARM	DISK SERVICES PARM LIST
340 (154)	A-ADDRESS	4	DKSDSB	ADDR OF DATA SERVICES BLOCK (DSIDSB)
344 (158)	A-ADDRESS	4	DKSIDPTR	ADDR OF A DCT ENTRY OR A DATA SET ID NAME OR NAME OF MEMBER
348 (15C)	BITSTRING	1	DKSOPT	OPTION INDICATORS
1...			DKSDCB	1=A DCT ADDR WAS SUPPLIED
.1..			DKSTTR	1=TTR SUPPLIED ON READ REQUEST
349 (15D)	BITSTRING	1	DKSREQ	REQUEST CODE
350 (15E)	CHARACTER	2	DKSRESV	FOR FUTURE USE/ALIGNMENT

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
MESSAGE BUILD INVOCATION PARAMETER LIST				
340	(154) STRUCTURE	92	MBSPARM	MESSAGE BUILD PARM LIST
340	(154) A-ADDRESS	4	MBSMSGA	ADDR OF SUPPLIED MSG SKELETON
344	(158) SIGNED	4	MBSMIDA	MSGID VALUE (IN BINARY)
348	(15C) A-ADDRESS	4	MBSBFRA	BUFFER ADDR FOR EDITED TEXT
352	(160) CHARACTER	4	MBSFLAGS	INDICATORS
352	(160) CHARACTER	1	MBSOPTS	OPTION INDICATORS
1...			MBSSKELA	0=MSGID SUPPLIED 1=SKELETON ADDR SUPPLIED
.1..			MBSSIZE	0=(SIZE=NO) SPECIFIED 1=(SIZE=ONLY) SPECFIED
..1.			MBSTBIND	0=USE NCCF MSG TABLE 1=CALLERS MSG TABLE
356	(164) CHARACTER	72	MBSTABLE	TEXT BUCKETS, 9 ENTRIES MAX
356	(164) CHARACTER	8	MBSENTRY	VARIABLE TEXT INFO
356	(164) A-ADDRESS	4	MBSTXTAD	ADDR OF VARIABLE TEXT
360	(168) UNSIGNED	1	MBSTXTLN	LENGTH OF VARIABLE TEXT
361	(169) UNSIGNED	1	MBSPADLN	TOTAL FIELD LENGTH IF PADDING IS REQUESTED
362	(16A) CHARACTER	1	MBSFILL	FILL CHARACTER
363	(16B) CHARACTER	1	MBSPNFLG	BUCKET FLAGS
1...			MBSLEFT	0=RIGHT FILL 1=LEFT FILL
428	(1AC) A-ADDRESS	4	MBSUMSGT	ADDR OF USERS MSG TABLE
COMMAND ANALASIS INVOCATION PARAMETER LIST				
340	(154) STRUCTURE	17	CAIPARM	CAI SERVICES PARM LIST
340	(154) A-ADDRESS	4	CAICMND	ADDR OF INPUT COMMAND
344	(158) A-ADDRESS	4	CAICPROC	ADDR OF CMD PROC FOR THE INPUT COMMAND OR 0 FOR INVALID CMD
348	(15C) CHARACTER	8	CAINAME	LOAD MODULE NAME TO BE FOUND
356	(164) BITSTRING	1	CAIIND	INDICATORS
1...			CAIPARSD	1=COMMAND HAS BEEN PARSED
PARSE INVOCATION PARAMETER LIST				
340	(154) STRUCTURE	13	PSOPARM	PARSE SERVICES PARM LIST
340	(154) A-ADDRESS	4	PSOCMND	ADDR OF INPUT COMMAND
344	(158) A-ADDRESS	4	PSOPTAB	ADDR OF LOCATION IN WHICH PARSE TABLE IS TO BE BUILT OR PDB REQUIRED LENGTH (RETURNED)
348	(15C) A-ADDRESS	4	PSODELCT	COUNT OF DELIMITERS TO BE USED
352	(160) CHARACTER	1	PSOIND	OPTION INDICATORS

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
	1...		PSOOPTN	1=CALCULATE PDB REQUIRED LENGTH FOR INPUT COMMAND (PSOCMND) 0=PARSE THE INPUT COMMAND
	.1..		PSOFIRST	0=CONTINUATION CARD 1=FIRST WORD ENDS IN BLANK
	..1.		PSOSUB	QUOTES TAKE PRECEDENCE
353	(161) CHARACTER	0	PSODELIM	ARRAY OF DELIMITERS TO BE USED

RECEIVE PROCESSING PARAMETER LIST

340	(154) STRUCTURE	81	RCVPARM	RECEIVE PROCESSING PARM LIST
340	(154) A-ADDRESS	72	RCVSAV	SAVE AREA (18 ITEMS, EACH FIXED 31)
412	(19C) A-ADDRESS	4	RCVRPL	RPL ADDR
416	(1A0) A-ADDRESS	4	RCVBUF	I/O BUFFER ADDR
420	(1A4) BITSTRING	1	RCVIND	OPTION INDICATORS
	1...		RCVASY	0=ASYNCHRONOUS OPERATION 1=SYNCHRONOUS OPERATION

PRESENTATION SERVICES PARAMETER LIST

340	(154) STRUCTURE	26	PSMPARM	PRESENTATION SERVICES PARM LIST
340	(154) A-ADDRESS	4	PSMTYPE	REQUEST TYPE INDICATOR
344	(158) A-ADDRESS	4	PSMAREA	ADDR OF CALLER'S DATA
348	(15C) A-ADDRESS	4	PSMRPL	ADDR OF CALLER'S RPL
352	(160) A-ADDRESS	4	PSMAPPL	ADDR OF 8-BYTE APPL NAME FIELD
356	(164) A-ADDRESS	4	PSMDOM	ADDR OF 8-BYTE DOMAIN ID
360	(168) A-ADDRESS	4	PSMRID	ADDR OF 3-BYTE REPLY ID
364	(16C) BITSTRING	1	PSMOPT	OPTION INDICATORS
	1...		PSMERASE	1=ERASE OPTION
	.1..		PSMSEG	1=SEGMENT OPTION
	..1.		PSMREST	1=KEYBOARD RESTORE OPTION
	...1		PSMREDY	1=SEND READY MSG OPTION
	.... 1...		PSMNREDY	1=ERASE READY MSG OPTION
	.... .1..		PSMFULL	1=FULL PROCESSING
	.... ..1.		PSMFRST	1=FIRST OR ONLY
	1		PSMLAST	1=LAST OR ONLY
365	(16D) BITSTRING	1	PSMOPT2	ADDITIONAL OPTIONS
	1...		PSMCMDLN	COMMANDLINE OPTIONS
	.1..		PSMWINDW	WINDOW REQUEST
	..11 1111			RESERVED

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
OPERATOR IDENTIFICATION SERVICES PARAMETER LIST				
340 (154)	STRUCTURE	12	OIMPARM	OIS PARM LIST
340 (154)	A-ADDRESS	4	OIMSNT	ADDR OF SNT POS (INPUT)
344 (158)	A-ADDRESS	4	OIMOPID	ADDR OF OPERATOR ID (INPUT)
348 (15C)	A-ADDRESS	4	OIMENT	ADDR OF POS NUMBER (OUTPUT)
ROUTING DETERMINATION SERVICES PARAMETER LIST				
340 (154)	STRUCTURE	28	RDMPARM	RDS PARM LIST
340 (154)	A-ADDRESS	4	RDMPDB	ADDR OF PDB (INPUT)
344 (158)	A-ADDRESS	4	RDMPOS	ADDR OF POSITION NUM (INPUT)
348 (15C)	A-ADDRESS	4	RDMENT	ADDR OF ADDR OR POS (OUTPUT)
352 (160)	A-ADDRESS	4	RDMLNAME	ADDR OF LU NAME (INPUT)
356 (164)	A-ADDRESS	4	RDMSTADD	ADDR OF START ADDRESS (INPUT)
360 (168)	CHARACTER	1	RDMIND	FLAG INDICATORS FOR RDM
1...			RDMACT	ACTIVATE ART ENTRY
.1..			RDMINACT	INACTIVATE ART ENTRY
..1.			RDMLOCAL	ONLY SEARCH THE LOCAL
...1 1111				RESERVED
361 (169)	CHARACTER	3		RESERVED
364 (16C)	A-ADDRESS	4	RDMDOM	IF XDOM RESOURCE, ADDR OF DOMAIN NAME
SPAN SEARCH SERVICES PARAMETER LIST				
340 (154)	STRUCTURE	20	SSMPARM	SSS PARM LIST
340 (154)	A-ADDRESS	4	SSMPOS	ADDR OF SPAN POS NUMBER (INPUT)
344 (158)	A-ADDRESS	4	SSMSNTAD	ADDR SNT ADDR (INPUT)
348 (15C)	A-ADDRESS	4	SSMSPNME	ADDR OF SPAN NAME (INPUT)
352 (160)	A-ADDRESS	4	SSMBIT	ADDR OF BIT POS (INPUT)
356 (164)	A-ADDRESS	4	SSMVAL	ADDR OF TVB
MESSAGE QUEING SERVICES PARAMETER LIST				
340 (154)	STRUCTURE	24	MQSPARM	MQS SERVICES PARM LIST
340 (154)	A-ADDRESS	4	MQSMADR	ADDR OF MESSAGE BUFFER
344 (158)	A-ADDRESS	4	MQSTADR	ADDR OF RECEIVER'S TVB
348 (15C)	A-ADDRESS	4	MQSOADR	ADDR OF RECEIVER'S OPERATOR ID
352 (160)	BITSTRING	1	MQSBFLG	INDICATORS
353 (161)	BITSTRING	1	MQSMFLG	TYPE INDICATORS
1...			MQSMLTO	MLWTO=ON
.1..			MQSMLC	CONTROL=ON

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
..1.			MQSMLL	LABEL=ON
...1			MQSMLD	DATA=ON
.... 1...			MQSMLDE	DATAEND=ON
.... .111				RESERVED
354 (162)	CHARACTER	2	MQSRESV	ALIGNMENT
356 (164)	A-ADDRESS	4	MQSSADR	ADDR OF SENDER ID
360 (168)	A-ADDRESS	4	MQSLADR	ADDR OF BUFFER LENGTH

PARAMETER ALIAS PARAMETER LIST

340 (154)	STRUCTURE	12	PAMPARM	PAM SERVICES PARM LIST
340 (154)	A-ADDRESS	4	PAMPDB	ADDR OF INPUT PDB
344 (158)	SIGNED	4	PAMENTRY	POSITION OF PDB ENTRY TO BE CKED
348 (15C)	A-ADDRESS	4	PAMOUT	ADDRESS OF AN 8-BYTE OUTPUT AREA TO CONTAIN THE REGULAR OPERAND, OR BLANKS

WRITE TO LOG PARAMETER LIST

340 (154)	STRUCTURE	8	WLSPARM	WLM SERVICES PARM LIST
340 (154)	A-ADDRESS	4	WLSHCT	ADDR OF HCT TVB
344 (158)	A-ADDRESS	4	WLSBFR	ADDR OF RECORD TO BE LOGGED

WRITE TO CONSOLE PARAMETER LIST

340 (154)	STRUCTURE	4	WCMPARM	WCM SERVICES PARM LIST
340 (154)	A-ADDRESS	4	WCMMADR	ADDR OF MESSAGE TEXT

CNMI SERVICES PARAMETER LIST

340 (154)	STRUCTURE	36	SWBCSPRM	CNMI SERVICES PARM LIST
340 (154)	A-ADDRESS	4	SWBCSDRB	ADDR OF DSRB
344 (158)	A-ADDRESS	4	SWBCSIPT	ADDR OF USER INPUT BFR
348 (15C)	A-ADDRESS	4	SWBCSILN	ADDR OF USER INPUT BUFFER LENGTH
352 (160)	A-ADDRESS	4	SWBCSRU	ADDR OF RU TO IMBED IN FOWARD RU
356 (164)	A-ADDRESS	4	SWBCSRLN	ADDR OF RU BUFFER LENGTH
360 (168)	A-ADDRESS	4	SWBCSDST	ADDR OF DEST NAME
364 (16C)	A-ADDRESS	4	SWBCSTAR	ADDR OF TARGET NAME
368 (170)	A-ADDRESS	4	SWBCSRSM	ADDR OF RESUME VERB (COMMAND PROCESSOR)
372 (174)	BITSTRING	1	SWBCSTYP	TYPE PARAMETER
1...			SWBCSCHN	CHAIN OPTION
.1..			SWBCSRUO	RU OPTION
373 (175)	BITSTRING	3		RESERVED

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
VSAM SERVICES PARAMETER LIST				
340	(154) STRUCTURE	29	SWBVPARM	VSAM SERVICES PARM LIST
340	(154) A-ADDRESS	4	SWBVDSRB	ADDR OF DSRB
344	(158) A-ADDRESS	4	SWBVKEY	ADDR OF VSAM KEY
348	(15C) SIGNED	4	SWBVKLN	LENGTH OF VSAM KEY
352	(160) A-ADDRESS	4	SWBVRESM	ADDR OF RESUME VERB
356	(164) A-ADDRESS	4	SWBVDAD	ADDR OF USER DATA AREA
360	(168) SIGNED	4	SWBVDLN	LENGTH OF USER DATA AREA
364	(16C) UNSIGNED	4	SWBVOPT	VSAM OPTION INDICATORS
364	(16C) BITSTRING	1	SWBVOPT1	ADDR OF RESUME VERB (COMMAND PROCESSOR)
	1...		SWBVSEQ	SEQUENTIIAL ACCESS
	.1..		SWBVDIR	DIRECT ACCESS
	..1.		SWBVSKP	SKIP SEQUENTIAL ACCESS
	...1		SWBVARD	USERS ARGUMENT
	.... 1...		SWBVLRD	LAST RECORD
	.... .1..		SWBVFWD	FORWARD PROCESSING
	.... ..1.		SWBVBWD	BACKWARD PROCESSING
	1		SWBVNUP	RECORD NOT UPDATED
365	(16D) BITSTRING	1	SWBVOPT2	
	1...		SWBVNSP	REMEMBER POSITION
	.1..		SWBVUPD	UPDATE RECORD
	..1.		SWBVKEQ	KEY = FOR ARGUMENT
	...1		SWBVKGE	KEY GREATER THAN OF EQUAL TO
	.... 1...		SWBVFKS	FULL KEY FOR ARGUMENT
	.... .1..		SWBVGEN	GENERIC KEY FOR ARGUMENT
366	(16E) BITSTRING	2		RESERVED
368	(170) CHARACTER	1	SWBVRTP	VSAM REQUEST TYPE INDICATOR (SEE CONSTANTS
NETWORK SERVICES INVOCATION PARAMETER LIST				
340	(154) STRUCTURE	228	SWBNM	NETWORK SERVICES PARM LIST
340	(154) CHARACTER	12	SWBNMIN	INPUT PARAMETER LIST
340	(154) A-ADDRESS	4	SWBNMBFR	BUFFER ADDR FOR FORWARD
344	(158) A-ADDRESS	4	SWBNMCID	RETURN ADDR FOR CORRELATION ID
348	(15C) A-ADDRESS	1	SWBNMTYP	REQUEST CODE
349	(15D) CHARACTER	3		RESERVED
352	(160) CHARACTER	16	SWBNMR	ADDITIONAL RETURN INFO
368	(170) CHARACTER	200	SWBNMWA	ADDITIONAL WORK AREA FOR NETWORK MANAGEMENT SERVICES

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
VSAM OPEN SERVICES PARAMETER LIST				
340 (154)	STRUCTURE	8	SWBVSOPL	VSAM OPEN SERVICES PARM LIST
340 (154)	A-ADDRESS	4	SWBVSACB	POINTER TO ACB
344 (158)	A-ADDRESS	4	SWBVSWRK	WORK AREA
SCOPE KEYWORD/VALUE SUPPORT PARAMETER LIST				
340 (154)	STRUCTURE	28	SWBSCPRM	SCOPE PARAMETER LIST FOR DSIKVM
340 (154)	A-ADDRESS	4	SWBSCSCT	SCTENTRY POINTER
344 (158)	CHARACTER	8	SWBSCCMD	COMMAND IF PTR NOT AVAILABLE
352 (160)	CHARACTER	8	SWBSCKEY	KEYWORD FOR AUTHORZATION
360 (168)	CHARACTER	8	SWBSCVAL	VALUE FOR AUTHORIZATION
CLIST DICTIONARY SERVICES PARAMETER LIST				
340 (154)	STRUCTURE	25	SWBCD	CLIST DICTIONARY PLIST
340 (154)	A-ADDRESS	4	SWBCDCLB	ADDR OF CLB
344 (158)	A-ADDRESS	4	SWBCDSYM	ADDR OF SYMBOL
348 (15C)	A-ADDRESS	4	SWBCDVAL	ADDR OF AREA FOR VALUE (DEF + CHG) OR ADDR OF WORD FOR PTR TO VALUE (SUB+ ANA)
352 (160)	A-ADDRESS	4	SWBCDST	ADDR OF BYTE FOR SYMBOL TYPE
356 (164)	A-ADDRESS	4	SWBCDVLN	ADDR OF WORD FOR VALUE LENGTH
360 (168)	A-ADDRESS	4	SWBCDSLN	ADDR OF WORD FOR SYMBOL LENGTH
364 (16C)	A-ADDRESS	1	SWBCDTYP	REQUEST TYPE CODE

CROSS REFERENCE

CAICMND	340(154)	PSMRPL	348(15C)	SWBRDFUN	340(154)
CAICPROC	344(158)	PSMSEG	364 X'40'	SWBRDSFN	344(158)
CAIIND	356(164)	PSMTYPE	340(154)	SWBSAVEA	4 (4)
CAINAME	348(15C)	PSMWINDW	365 X'40'	SWBSCCMD	344(158)
CAIPARM	340(154)	PSOCMND	340(154)	SWBSCKEY	352(160)
CAIPARSD	356 X'80'	PSODELCT	348(15C)	SWBSCPRM	340(154)
DKSDCB	348 X'80'	PSODELIM	353(161)	SWBSCSCT	340(154)
DKSDSB	340(154)	PSOFIRST	352 X'40'	SWBSCVAL	360(168)
DKSIDPTR	344(158)	PSOIND	352(160)	SWBTIB	80 (50)
DKSOPT	348(15C)	PSOOPTN	352 X'80'	SWBVARD	364 X'10'
DKSPARM	340(154)	PSOPARM	340(154)	SWBVBWD	364 X'02'
DKSREQ	349(15D)	PSOPTAB	344(158)	SWBVDAD	356(164)
DKSRESV	350(15E)	PSOSUB	352 X'20'	SWBVDIR	364 X'40'
DKSTTR	348 X'40'	RCVASY	420 X'80'	SWBVDLN	360(168)
DSISWB	0 (0)	RCVBUF	416(1A0)	SWBVDSRB	340(154)
MBSBFRA	348(15C)	RCVIND	420(1A4)	SWBVFKS	365 X'08'
MBSENTRY	356(164)	RCVPARM	340(154)	SWBVFWD	364 X'04'
MBSFILL	362(16A)	RCVRPL	412(19C)	SWBVGEN	365 X'04'
MBSFLAGS	352(160)	RCVSAV	340(154)	SWBVKEQ	365 X'20'
MBSLEFT	363 X'80'	RDMACT	360 X'80'	SWBVKEY	344(158)
MBSMIDA	344(158)	RDMDOM	364(16C)	SWBVKGE	365 X'10'
MBSMSGA	340(154)	RDMENT	348(15C)	SWBVKLN	348(15C)
MBSOPTS	352(160)	RDMINACT	360 X'40'	SWBVLRD	364 X'08'
MBSPADLN	361(169)	RDMIND	360(168)	SWBVNSP	365 X'80'
MBSPARM	340(154)	RDMLNAME	352(160)	SWBVNUP	364 X'01'
MBSPNFLG	363(16B)	RDMLOCAL	360 X'20'	SWBVOPT	364(16C)
MBSSIZE	352 X'40'	RDMPARM	340(154)	SWBVOPT1	364(16C)
MBSSKELA	352 X'80'	RDMPDB	340(154)	SWBVOPT2	365(16D)
MBSTABLE	356(164)	RDMPOS	344(158)	SWBVPARM	340(154)
MBSTBIND	352 X'20'	RDMSTADD	356(164)	SWBVRESM	352(160)
MBSTXTAD	356(164)	SSMBIT	352(160)	SWBVRTP	368(170)
MBSTXTLN	360(168)	SSMPARM	340(154)	SWBVSACB	340(154)
MBSUMSGT	428(1AC)	SSMPOS	340(154)	SWBVSEQ	364 X'80'
MQSBFLG	352(160)	SSMSNTAD	344(158)	SWBVSKP	364 X'20'
MQSLADR	360(168)	SSMSPNME	348(15C)	SWBVSOPL	340(154)
MQSMADR	340(154)	SSMVAL	356(164)	SWBVSWRK	344(158)
MQSMFLG	353(161)	SWBADATD	84 (54)	SWBVUPD	365 X'40'
MQSMLC	353 X'40'	SWBCBH	0 (0)	WCMMADR	340(154)
MQSMLD	353 X'10'	SWBCD	340(154)	WCMPARM	340(154)
MQSMLDE	353 X'08'	SWBCDCLB	340(154)	WLSBFR	344(158)
MQSMLL	353 X'20'	SWBCDSLN	360(168)	WLSHCT	340(154)
MQSMLTO	353 X'80'	SWBCDST	352(160)	WLSPARM	340(154)
MQSOADR	348(15C)	SWBCDSYM	344(158)		
MQSPARM	340(154)	SWBCDTYP	364(16C)		
MQSRESV	354(162)	SWBCDVAL	348(15C)		
MQSSADR	356(164)	SWBCDVLN	356(164)		
MQSTADR	344(158)	SWBCSCHN	372 X'80'		
OIMENT	348(15C)	SWBCSDRB	340(154)		
OIMOPID	344(158)	SWBCSDST	360(168)		
OIMPARM	340(154)	SWBCSILN	348(15C)		
OIMSNT	340(154)	SWBCSIPT	344(158)		
PAMENTRY	344(158)	SWBCSPRM	340(154)		
PAMOUT	348(15C)	SWBCSRLN	356(164)		
PAMPARM	340(154)	SWBCSRSM	368(170)		
PAMPDB	340(154)	SWBCSRU	352(160)		
PSMAPPL	352(160)	SWBCSRUO	372 X'40'		
PSMAREA	344(158)	SWBCSTAR	364(16C)		
PSMCMDLN	365 X'80'	SWBCSTYP	372(174)		
PSMDOM	356(164)	SWBNEXT	76 (4C)		
PSMERASE	364 X'80'	SWBNM	340(154)		
PSMFRST	364 X'02'	SWBNMBFR	340(154)		
PSMFULL	364 X'04'	SWBNMCID	344(158)		
PSMLAST	364 X'01'	SWBNMIN	340(154)		
PSMNREDY	364 X'08'	SWBNMR	352(160)		
PSMOPT	364(16C)	SWBNMTYP	348(15C)		
PSMOPT2	365(16D)	SWBNMWA	368(170)		
PSMPARM	340(154)	SWBPLIST	340(154)		
PSMREDY	364 X'10'	SWBRD	340(154)		
PSMREST	364 X'20'	SWBRDAPT	352(160)		
PSMRID	360(168)	SWBRDCPN	348(15C)		

Constants in DSISWB

NAME	VALUE	MEANING
Constants For Setting and Testing DKSREQ (Bit 8)		
DKSOPEN	X'01'	REQUEST FOR OPEN
DKSCLOS	X'02'	REQUEST FOR CLOSE
DKSCONN	X'03'	REQUEST FOR CONNECTION
DKSFIND	X'04'	REQUEST FOR FIND
DKSREAD	X'05'	REQUEST FOR READ OPERATION
DKSDISC	X'06'	REQUEST FOR DISCONNECTION
Constants For Setting and Testing DKS Return Codes (Fullword)		
DKSGOOD	0	OPERATION REQUEST SUCCESSFUL
DKSNTFND	4	DATA SET NAME NOT FOUND OR MEMBER NOT FOUND IN PDS
DKSEOD	4	END OF DATA
DKSIOERR	8	I/O ERROR
DKSFAIL	8	OPEN/CLOSE FAILURE
DKSRDNTP	12	DATA SET IS NOT OPENED OR READ IS NOT PERMITTED
DKSNOSTG	12	NO STORAGE AVAILABLE AVAILABLE
DKSNOBUF	16	NO DISK BUFFER ADDR
DKSINVLD	20	INVALID CTL BLK ID
DKSINVRC	24	INVALID REQUEST CODE
Constants For Setting and Testing MBS Return Codes (Fullword)		
MBSGOOD	0	MSG BUILD WAS SUCCESSFUL
MBSTXTNA	4	
MBSTRUNC	8	MSG TEXT HAS BEEN TRUNCATED
MBSNTFND	12	MSG TXT COULD NOT BE FOUND
MBSNOBUF	16	
MBSTRTNA	20	VAR TEXT NOT AVAILABLE AND TEXT IS TOO LONG
MBSTRNFD	24	MSGID NOT FOUND IN MSG DEIFINITION MODULE AND BUFFER IS TOO SMALL FOR FOR MSG DSI000
MBSINVAD	28	INVALID REQUEST
Command Analysis Return Code Values (Halfword)		
CAIGOOD	0	REGULAR COMMAND LOCATED
CAIBOTH	4	IMMED/REG COMMAND LOCATED
CAIMMED	8	IMMEDIATE COMMAND LOCATED
CAIBAD	12	NO COMMAND FOUND-- UNSUCCESSFUL
CAISCBAD	16	SCOPECLASS ERROR
CAIINVLD	20	COMMAND INCOMPATIBLE
Parse Return Code Values (Halfword)		
PSOGOOD	0	CMD PARSED OR SIZE FOUND
PSONULL	4	COMMAND NOT

```
                                        PARSED--NO DATA
              PSOSMALL      8           PARSE TABLE TOO
                                        SMALL FOR COMMAND
              PSOQUOTE      12          UNPAIRED
                                        APOSTROPHES
              PSONOPDB      100         NO PDB WAS PASSED

Constants For Setting and Testing PSM Option Code Byte 1 (Bit 8)

              PSMERAS       X'80'       SET ERASE OPTION ON
              PSMNERAS      X'00'       SET ERASE OPTION OFF
              PSMSEGMT      X'40'       SET SEGMENT
                                        OPTION ON
              PSMMSG        X'00'
              PSMRES        X'20'       SET KEYBOARD
                                        RESTORE ON
              PSMNRES       X'00'       SET KEYBOARD
                                        RESTORE OFF
              PSMRED        X'10'       SET READY MSG ON

              PSMNRED       X'08'       SET READY MSG OFF
              PSMNOOP       X'00'       DON'T CHANGE
                                        READY MESSAGE
              PSMFRSTF      X'06'       FULL--FIRST
              PSMMIDF       X'04'       FULL--MIDDLE
              PSMLASTF      X'05'       FULL--LAST
              PSMONLYF      X'07'       FULL--ONLY

Constants For Setting and Testing PSM Option Code Byte 2 (Bit 8)

              PSMCMDLF      X'80'       COMMAND LINE OPTION

Constants For Setting and Testing PSM Return Codes (Fullword)

              PSMGOOD       0           SUCCESSFUL
              PSMNOSND      4           DATA WAS NOT SENT
              PSMINVDL      8           DATA LENGTH INVALID
              PSMNOSTG      12          STORAGE UNAVAILABLE
              PSMFSERR      16          FILLED SCREEN ERROR
              PSMDUMMY      20          PSM FUNCTION NOT
                                        SUPPORTED
              PSMSEQER      24          ERROR IN SEQUENCE
                                        OF FORMATTED
                                        DISPLAY REQUEST
              PSMUEDEL      28          USER EXIT DELETED
                                        THIS MESSAGE
              PSMNOINP      32          SCREEN INPUT
                                        CANCELED,
                                        NO RESUME COMMAND
                                        WILL BE SCHEDULED
              PSMRETRY      36          ERROR DURING
                                        SCREEN I/O.  USER
                                        SHOULD RETRY REQUEST
              PSMIOERR      40          PERMANENT I/O
                                        ERROR DURING
                                        SCREEN I/O REQUEST
              PSMRESET      44          SCREEN INPUT RESET
                                        NORMAL; NO RESUME
                                        COMMAND WILL BE
                                        SCHEDULED
              PSMBUSY       48          REQUEST INVALID
                                        SINCE PREVIOUS
                                        ASYPANEL HAS NOT
                                        BEEN POSTED
                                        COMPLETE, OR RESET
                                        WITH TYPE=CANCEL
              PSMCANCL      52          INPUT REQUEST ENDED
                                        DUE TO TYPE=CANCEL
              PSMPOSTN      56          AT LEAST ONE NCCF
                                        ECB HAS BEEN POSTED

SYNAD Return Code Values (Fullword)

              SYNOK         0           REQUEST OK
```

SYNRETRY	4	RETRY REQUEST
SYNTMALF	8	TEMPORARY MALFUNCTION
SYNABORT	12	PERMANENT ERROR-- ABORT SUBTASK
SYNNVTAM	16	NOT A VTAM COMMAND SENDCMD ONLY
SYNOTACT	20	NOT ACTIVATED
SYNOAUTH	24	NO AUTHORIZATION FOR OPNDST ACQUIRE OR SIMLOGON
SYNRESET	28	RECEIVE RESET RESETSR

Operator Identification Services Return Codes (Fullword)

OIMGOOD	0	REQUEST SUCCESSFUL
OIMBAD	4	REQUEST UNSUCCESSFUL

Routing Services Return Code Values (fullword)

RDMASAME	0	AUTHORIZED FOR SAME DOMAIN
RDMNSAME	4	NOT AUTHORIZED FOR SAME DOMAIN
RDMCROSS	8	CROSS-DOMAIN SESSION ACTIVE
RDMNCROS	12	CROSS-DOMAIN SESSION NOT ACTIVE
RDMNOTBL	16	NO ART OR DQT TABLES EXIST
RDMFAILD	20	NO AUTHORIZATION

Span Search Services Return Code Values (Halfword)

SSMACT	0	SPAN FOUND--ACTIVE
SSMINACT	4	SPAN FOUND-- INACTIVE
SSMBAD	12	SPAN NOT FOUND

Message Queueing Return Code Values (Halfword)

MQSGOOD	0	MSG SEND OK
SWBMQSER	4	BUFFER SPEFICATION
MQSNFND	8	OPERATOR ID NOT FOUND
MQSNBFR	12	BUFFER OBTAIN FAILED
MQSTERM	16	NCCF IN TERMINATION

Message Queueing Indicator Values (Bit 8)

MQSBYES	B'00000000'	BUFFER NOT PROVIDED
MQSBNO	B'00000001'	USE BUFFER PASSED
MQSAUTH	B'00000010'	RECEIVER OF AUTHORIZED MSGS
MQSTCT	B'00000100'	QUEUE TO TCT
MQSLVAR	B'00001000'	QUEUE TO LU NAME
MQSEXT	B'00010000'	EXTERNAL INVOCATION
MQSPPT	B'00100000'	QUEUE TO PPT
MQSON	B'1'	TURN FUNCTION ON

Parameter Alias Return Code Values (Halfword)

PAMGOOD	0	REGULAR PARAMETER LOCATED
PAMSAME	4	NO PARAMETER FOUND SUCCESSFUL
PAMBAD	8	NO PARAMETER FOUND UNSUCCESSFUL

Write to Log Return Code Values (Halfword)

WLSGOOD	0	SUCCESSFUL
WLSNOSTG	4	NO STORAGE AVAIL

```
                WLSFAIL         8               OPEN FAILURE
                WLSHCLNA        12              HARDCOPY LOG FAILED
                                                QUEUE MANAGEMENT
                                                SERVICES FAILED
                WLSIOERR        16              PERMANENT I/O ERROR
                                                ON LOG FILE
                WLSEOFCL        20              OPEN FAILURE AFTER
                                                CLOSING LOG DATA
                                                SET AT END OF EXTENT

Write to Console Return Code Values (Halfword)

                WCMGOOD         0               SUCCESSFUL
                WLSNOSWB        8               NO SWB OBTAINED FOR
                                                CALL TO USER EXIT 09

Cits Constants For Testing and Setting Return Codes (Fullword)

                DCL CITSGOOD    0               CITS FUNCTION DONE
                DCL CITSFAIL    4               CITS FUNCTION
                                                FAILED

VSAM Services Request Indicators For SWBVRTP (Bit 8)

                SWBVGET         X'01'           GET REQUEST
                SWBVPUT         X'02'           PUT REQUEST
                SWBVPNT         X'03'           POINT REQUEST
                SWBVERS         X'04'           ERASE REQUEST
                DCL SWBVNRQ     X'05'           ENDREQ REQUEST

NMS Return Codes (Fullword)

                SWBNMSUC        0               NMS SUCCESSFUL
                SWBNMINV        4               INVALID CALL
                SWBNMREJ        8               ACCESS METHOD
                                                REJECTED REQUEST
                SWBNMNOS        12              INSUFFICIENT
                                                STORAGE

Constants For Type Codes (Bit 8)

                SWBNMOPN        X'01'           TYPE = OPEN
                SWBNMRDY        X'03'           TYPE = READY
                SWBNMFWD        X'04'           TYPE = FORWARD
                SWBNMCLS        X'02'           TYPE = CLOSE
                SWBNMSND        X'05'           TYPE = SEND

Constants For Setting and Testing VSAM Open Services (Fullword)

                SWBVPERR        8               INCORRECT PARAMETER
                                                SPECIFICATION

CNMI Services Routine Return Codes--Major (REG 15)

                SWBCGOOD        0               SUCCESSFUL
                SWBCFAIL        4               UNSUCCESSFUL
                SWBCNPUT        8               INPUT BUFFER TOO
                                                SMALL TO BUILD RU IN
                SWBCNMAC        12              INVALID MACRO
                                                SPECIFICATION
                SWBCNTSK        16              NOT RUNNING UNDER
                                                DATA SERVICES TASK
                SWBCNFNC        24              FUNCTION NOT
                                                SUPPORTED

CNMI Services Routine Return Codes--Minor (REG 0)

                SWBNOSWB        4               INVALID SWB
                SWBBADRB        8               INVALID DSRB
                SWBNUSRB        12              DSRB PASSED WAS
                                                IN USE
                SWBNSLRB        16              UNSOLICITED DSRB
                                                PASSED
```

SWBNOPID	20	INVALID OPERATOR ID IN DSRB
SWBNOVRB	24	UNDEFINED RESUME VERB
SWBNOSTR	28	INSUFFICIENT NCCF STORAGE TO PROCESS REQUEST
SWBCNACT	32	CNMI IS INACTIVE
SWBCAREJ	36	ACCESS METHOD REJECTED REQUEST
SWBCEXIT	40	USER EXIT REJECTED REQUEST
SWBCTRNC	44	DATA TRUNCATION OCCURRED DURING USER EXIT

Constants For DSIKVS Return Codes (Bit 8)

SWBSCGD	0	KEYWORD & VALUE OK
SWBSCKWD	4	KEYWRD NOT IN OPERATOR'S SCOPE
SWBSCNVL	8	VALUE NOT IN OPERATOR'S SCOPE
SWBSCPAR	9	NCCF INTERNAL USE ONLY
SWBSCINV	12	INVALID PARAMETER PASSED

Constants For Command List Dictionary Request Type Codes (Bit 8)

SWBCDSUB	X'01'	TYPE=SUBSTITUTE
SWBCDANA	X'02'	TYPE=ANALYZE
SWBCDDEF	X'03'	TYPE=DEFINE
SWBCDCHG	X'04'	TYPE=CHANGE
SWBCDTRM	X'05'	TYPE=TERMINATE

Command List Dictionary Symbol Type Codes (Bit 8)

(Must be the Same as in DSICDE)

SWBCDCS	X'01'	CONTROL SYMBOL
SWBCDCV	X'02'	CONTROL VARIABLE
SWBCDUV	X'03'	USER VARIABLE
SWBCDFCN	X'04'	BUILT-IN FUNCTION
SWBCDLBL	X'05'	LABEL

Return Codes From Dictionary Service Module (Fullword)

SWBCDSUC	0	CDS SUCCESSFUL
SWBCDINV	4	INVALID SYMBOL
SWBCDDUP	8	SYMBOL ALREADY DEFINED
SWBCDNOS	12	INSUFFICIENT STORAGE
SWBCDSNO	16	SHOULD NOT OCCUR
SWBCDLEX	20	VALUELN LIMIT EXCEEDED

Function Codes For Command List Control Symbols (Byte)

SWBCDCTL	1	&CONTROL FUNCTION
SWBCDEXT	2	&EXIT FUNCTION
SWBCDWRT	3	&WRITE FUNCTION
SWBCDBWT	4	&BEGWRITE FUNCTION
SWBCDGO	5	&GOTO FUNCTION
SWBCDIF	6	&IF FUNCTION
SWBCDPS	7	&PAUSE FUNCTION
SWBCDTHN	8	&THEN FUNCTION

Function Codes For Command List Built-in Function Symbols (Byte)

SWBCDCON	1	&CONCAT FUNCTION
SWBCDLTH	2	&LENGTH FUNCTION
SWBCDNID	3	&NCCFID FUNCTION
SWBCDNST	4	&NCCFSTAT FUNCTION

SWBCDSTR	5	&SUBSTR FUNCTION

Function Codes For Command List Control Variables (Byte)

SWBCDAPP	1	APPLID OF TASK
SWBCDHCY	2	HARDCOPY NAME
SWBCDLU	3	OPERATOR STATION NAME
SWBCDNCT	4	NUMBER DOMAINS
SWBCDOID	5	OPERATOR'S USERID
SWBCDPCT	6	NUMBER PARAMETERS ON CLIST INVOCATION
SWBCDPST	7	CHARACTER STRING ON CLIST INVOCATION
SWBCDRC	8	RETURN CODE FROM COMMAND PROCESSOR OR COMMAND LIST
SWBCDTSK	9	TASK TYPE: PPT, OST, OR NNT
SWBCDMMD	10	MSGMOD VALUE OR NULL
SWBCDTIM	11	5 CHARACTER TIME: HH:MM
SWBCDDAT	12	8 CHARACTER DATE: MM/DD/YY

Values for Setting and Testing Routing Definition Return Codes

SWBRDFOK	0	SUCCESS
SWBRDFND	4	NOT UNDER DST
SWBRDFDP	8	DUPLICATE DEFINIT
SWBRDFNS	12	NO STORAGE
SWBRDFIP	16	INVALID FUNC/SUBF
SWBRDFNA	20	LU TASK INACTIVE
SWBRDFIN	252	INVALID MACRO INVOCATION

Values for Setting and Testing LUS Return Code

SWBLUSIN	252	INVALID MACRO INVOCATION

TIB

DSITIB: MAPS THE TASK INFORMATION BLOCK FOR NCCF
BOUNDARY: DOUBLEWORD
LENGTH: 860 BYTES (X'35C') + VARIABLE LENGTH DATA
POINTED TO BY: TVB (TVBTIB)
CLB (CLBTIB)
CWB (CWBTIB) CALLERS TIB
DSRB (DSRBTIB) TIB OF DATA SERVICES TASK
SWB (SWBTIB) CALLERS TIB
TVB (TVBTIB)
INCLUDED BLOCKS: CBH (TIBCBH)
CWB (TIOCWB1,TIOCWB2,TIOCWB3,TIPCWB)
DSB (TIODSB,TIPDSB)
SWB (TIOSWBI,TIOSWBN,TIHSWBI,TIHSWBN,TIPSWBI,TIPSWBN)
BUFHDR (TIOPSHDR,TIOIMHDR,TIOI2HDR,TIOI3HDR,TIORCHDR,
TIOSHDR,TIO1HDR,TIO2HDR,TIPROCHDR,TIPSCHDR)

OFFSETS		TYPE	LENGTH	NAME	DESCRIPTION
0	(0)	STRUCTURE	860	DSITIB	TASK INFORMATION BLOCK
0	(0)	CHARACTER	4	TIBCBH	NCCF CONTROL BLOCK HEADER
4	(4)	A-ADDRESS	4	TIBTVB	ADDR OF THE TVB FOR THIS TIB
8	(8)	A-ADDRESS	4	TIBACB	ADDR OF ACF/VTAM ACB FOR THIS TASK
12	(C)	A-ADDRESS	4	TIBEXLST	ADDR OF ACF/VTAM EXLST
16	(10)	A-ADDRESS	4	TIBELT	ADDR OF THE ECBLIST IN THIS TIB
20	(14)	CHARACTER	9	TI3APID	ACF/VTAM APPLID FOR THIS TASK
20	(14)	UNSIGNED	1	TIBAPIDL	ACF/VTAM APPLID LENGTH
21	(15)	CHARACTER	8	TIBAPIDN	ACF/VTAM APPLID FOR THIS TASK
29	(1D)	CHARACTER	9	TIBAPWD	ACF/VTAM PASSWORD FOR THIS TASK
29	(1D)	UNSIGNED	1	TIBAPWDL	ACF/VTAM PASSWORD LENGTH
30	(1E)	CHARACTER	8	TIBAPWDN	ACF/VTAM PASSWORD FOR THIS TASK
38	(26)	CHARACTER	62	TIBAREA1	SUBTASK DEPENDENT AREA
38	(26)	SIGNED	2	TIBPOICT	CURRENT POI-HDR ID COUNT VALUE
38	(26)	UNSIGNED	2	TIBTSEQ	LAST SEQUENCE NUMBER--ACF/TCAM
40	(28)	A-ADDRESS	4	TIBPSSPT	ADDR DSIPSM WORK SPACE (SEE TIOPSS)
44	(2C)	A-ADDRESS	4	TIBNCCWB	NORMAL COMMAND CWB ADDR
48	(30)	A-ADDRESS	4	TIBICCWB	IMMED COMMAND CWB ADDR
52	(34)	A-ADDRESS	4	TIBMRCWB	RCVCMD CWB ADDR
56	(38)	A-ADDRESS	4	TIBEXSWB	SWB ADDR FOR EXIT PROCESSING
60	(3C)	A-ADDRESS	4	TIBNPSWB	SWB ADDR FOR NORMAL PROCESSING
64	(40)	A-ADDRESS	4	TIBNCPDB	NORMAL COMMAND PDB ADDR
68	(44)	A-ADDRESS	4	TIBMRPDB	MSGRCV & RCVCMD PDB ADDR
72	(48)	A-ADDRESS	4	TIBICPDB	IMMED COMMAND PDB ADDR

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
76 (4C)	A-ADDRESS	4	TIBAGPDB	AGAIN PROCESSING PDB ADDR
80 (50)	A-ADDRESS	4	TIBNCBFR	NORMAL CMD BUFFER ADDR
84 (54)	A-ADDRESS	4	TIBICBFR	IMMED CMD BUFFER ADDR
88 (58)	A-ADDRESS	4	TIBAGBFR	AGAIN PROCESSING CMD BUFFER ADDR
92 (5C)	A-ADDRESS	4	TIBSEND	ADDR OF PSS SEND ROUTINE
96 (60)	A-ADDRESS	4	TIBRECV	ADDR OF RECEIVE ROUTINE
100 (64)	A-ADDRESS	4	TIBUFLD	NCCF USER FIELD
104 (68)	A-ADDRESS	4	TIBTAFFY	POINTER TO STORAGE
108 (6C)	A-ADDRESS	4		RESERVED
112 (70)	CHARACTER	12	TIBAREA2	SUBTASK DEPENDENT AREA 2
112 (70)	A-ADDRESS	4	TIBCLBQ	DSICLB QUEUE HEADER
112 (70)	A-ADDRESS	4	TIBOSEXT	ADDR OF "OPTIONAL SUBTASK" EXTENTION
116 (74)	SIGNED	4	TIBCLECB	CLIST ECB
116 (74)	A-ADDRESS	4	TIBOSLST	ADDR OF OPTIONAL SUBTASK LIST CMD BUFFERS
120 (78)	SIGNED	4	TIBXECB	SECONDARY XDOMAIN ECB
124 (7C)	CHARACTER	72	TIBSAVES	STANDARD SAVE AREA
196 (C4)	CHARACTER	72	TIBSAVEE	EXIT SAVE AREA
268 (10C)	CHARACTER	256	TIBNDATD	NORMAL PROC OST AUTO WORK AREA
524 (20C)	CHARACTER	256	TIBEDATD	EXIT PROC OST AUTO WORK AREA
780 (30C)	CHARACTER	16	TIBINT	RESERVED
780 (30C)	A-ADDRESS	4	TIBINT1	RESERVED
784 (310)	A-ADDRESS	4	TIBINT2	RESERVED
788 (314)	CHARACTER	8	TIBMSGNM	RESERVED
796 (31C)	SIGNED	4	TIBRETCD	CMD PROC RETURN VALUE
800 (320)	A-ADDRESS	4	TIBSCTSK	SCOPEING ACTIVE IF NOT 0
804 (324)	SIGNED	4	TIBCIECB	COMMAND INPUT ECB
808 (328)	A-ADDRESS	4	TIBCLBWK	(PRIVATE) CLB WORK QUEUE
812 (32C)	A-ADDRESS	4	TIBLOGBF	POINTER TO LOGGER OUTPUT BUFFER
816 (330)	CHARACTER	8	TIBTINAM	NCCF-GENERATED TIMER ID
824 (338)	CHARACTER	1	TIBFLGS	FLAG BYTE
1...			TIBCCL	CLIST COMMAND
.iii iiii				RESERVED
825 (339)	CHARACTER	5	TIBMMD	MSGMOD IDENTIFIER
830 (33E)	CHARACTER	2		RESERVED--ALIGNMENT

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
832 (340)	A-ADDRESS	4	TIBLOGBE	POINTER TO LOGGER OUTPUT BUFFER WHEN NOT IN EXIT
836 (344)	A-ADDRESS	24		RESERVED--CRITICAL
860 (35C)	CHARACTER	0	TIBEXTEN	BEGINING OF UNIQUE EXTENSION

OST EXTENSION

860 (35C)	STRUCTURE	6972	TIBOST	OST EXTENSION
860 (35C)	CHARACTER	24	TIORPLCT	OST TASK RPLS
860 (35C)	A-ADDRESS	4	TIOORRPL	ADDR OF OPER RECEIVE ACF/VTAM RPL
864 (360)	A-ADDRESS	4	TIOOSRPL	ADDR OF OPER SEND ACF/VTAM RPL
868 (364)	A-ADDRESS	4	TIORCRPL	ADDR OF POI REC-CMD ACF/VTAM RPL
872 (368)	A-ADDRESS	4	TIOSCRPL	ADDR OF POI SEND-CMD ACF/VTAM RPL
876 (36C)	A-ADDRESS	4	TIORARPL	ADDR OF NNT RECEIVE ANY ACF/VTAM RPL
880 (370)	A-ADDRESS	4	TIORSRPL	ADDR OF NNT REQ SESSION ACF/VTAM RPL
884 (374)	CHARACTER	8	TIONIBCT	OST TASK NIBS
884 (374)	A-ADDRESS	4	TIOOSNIB	ADDR OF OPERATOR STATION NIB
888 (378)	A-ADDRESS	4	TIOCDNIB	ADDR OF CROSS-DOMAIN SESS NIB
892 (37C)	CHARACTER	32	TIOELT	ECB LIST USED FOR EXEC CONTROL (8 ELT ENTRIES, 4 CHARACTERS EACH)
892 (37C)	A-ADDRESS	4	TIOELTP	POINTER TO THE ECB
1...			TIOELTLA	LAST ECB FLAG
924 (39C)	SIGNED	4	TIORCECB	RECEIVE CMD ECB
928 (3A0)	SIGNED	4		RESERVED
932 (3A4)	SIGNED	4	TIORAECB	RECEIVE ANY ECB
936 (3A8)	SIGNED	4	TIOPAECB	PAUSE ECB
940 (3AC)	SIGNED	4	TIOLOECB	LOGON ECB
944 (3B0)	SIGNED	4	TIOQSECB	OUTPUT QUEUE ECB
948 (3B4)	CHARACTER	8	TIOPROFL	ISTATUS PROFILE NAME
956 (3BC)	A-ADDRESS	4	TIOSAUTH	ADDR OF SPAN AUTH TABLE
960 (3C0)	A-ADDRESS	4	TIONAUTH	ADDR OF NCCF-NCCF AUTH TABLE
964 (3C4)	SIGNED	2	TIOSCNT	COUNT OF SAT ENTRIES
966 (3C6)	SIGNED	2	TIONCNT	COUNT OF NAT ENTRIES
968 (3C8)	CHARACTER	146	TIOIBUF1	OST INPUT BUFFER 1
968 (3C8)	CHARACTER	24	TIOI1HDR	STANDARD BUFFER HEADER

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
992 (3E0)	CHARACTER	16		RESERVED--EXPANSION
1008 (3F0)	CHARACTER	20	TIOI1XTH	IMBEDDED XTH
1028 (404)	CHARACTER	6	TIOI1CTL	DEV CTL CHARS
1034 (40A)	CHARACTER	80	TIOI1BDY	BUFFER BODY
1114 (45A)	SIGNED	2	TIOOITSV	OIT INDEX FOR THIS OPERATOR
1116 (45C)	CHARACTER	146	TIOIBUF2	OST INPUT BUFFER 2
1116 (45C)	CHARACTER	24	TIOI2HDR	STANDARD BUFFER
1140 (474)	CHARACTER	16		RESERVED--EXPANSION
1156 (484)	CHARACTER	20	TIOI2XTH	IMBEDDED XTH
1176 (498)	CHARACTER	6	TIOI2CTL	DEVICE CONTROL CHARACTERS
1182 (49E)	CHARACTER	80	TIOI2BDY	BUFFER BODY
1262 (4EE)	BITSTRING	1	TIOFLGS	TIO INDICATOR SAVE AREA
1...			TIOSVRVR	SAVE AREA FOR TVBAUTH INDICATOR
.1..			TIOSVCTL	SAVE AREA FOR TVBNAUTH INDICATR
..11 1111				RESERVED
1263 (4EF)	CHARACTER	1		ALIGNMENT
1264 (4F0)	CHARACTER	146	TIOIBUF3	OST INPUT BUFFER 3
1264 (4F0)	CHARACTER	24	TIOI3HDR	STANDARD BUFFER HEADER
1288 (508)	CHARACTER	16		RESERVED--EXPANSION
1304 (518)	CHARACTER	20	TIOI3XTH	IMBEDDED XTH
1324 (52C)	CHARACTER	6	TIOI3CTL	DEVICE CONTROL CHARACTERS
1330 (532)	CHARACTER	80	TIOI3BDY	BUFFER BODY
1410 (582)	CHARACTER	2		ALIGNMENT
1412 (584)	CHARACTER	156	TIORCBUF	POI REC-CMD BUFFER
1412 (584)	CHARACTER	24	TIORCHDR	STANDARD BUFFER HEADER
1436 (59C)	CHARACTER	132	TIORCBDY	BUFFER BODY
1568 (620)	CHARACTER	188		RESERVED
1756 (6DC)	A-ADDRESS	4	TIOI1PTR	FIRST DYNAMIC BUFFER
1760 (6E0)	A-ADDRESS	4	TIOI2PTR	SECOND DYNAMIC BUFFER
1764 (6E4)	A-ADDRESS	4	TIOI3PTR	THIRD DYNAMIC BUFFER
1768 (6E8)	A-ADDRESS	4	TIORFPTR	REFRESH DYNAMIC BUFFER
1772 (6EC)	SIGNED	2	TIODBLEN	DYNAMIC BUFFER LENGTH
1774 (6EE)	SIGNED	2		RESERVED
1776 (6F0)	A-ADDRESS	4	TIORABUF	ADDR OF NNT RECEIVE ANY BUFFER
1780 (6F4)	A-ADDRESS	4	TIOACEE	POINTER TO RACF ACEE
1784 (6F8)	A-ADDRESS	4	TIOFLQ	FULL LINE QUEUE ANCHOR
1788 (6FC)	A-ADDRESS	4		RESERVED
1792 (700)	A-ADDRESS	4		RESERVED
1796 (704)	A-ADDRESS	4		RESERVED

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
1800 (708)	A-ADDRESS	4		RESERVED
1804 (70C)	CHARACTER	2736	TIOPSSSP	DSIPSM INFO BLOCK (TIOPSS)
4540 (11BC)	CHARACTER	364	TIOCWB1	A DSICWB
4904 (1328)	CHARACTER	364	TIOCWB2	ANOTHER DSICWB
5268 (1494)	CHARACTER	364	TIOCWB3	ONE MORE DSICWB
5632 (1600)	CHARACTER	600	TIOSWBI	DEFAULT (FIRST-IN-CHAIN) IMMED DSISWB
6232 (1858)	CHARACTER	600	TIOSWBN	DEFAULT (FIRST-IN-CHAIN) NORMAL DSISWB
6832 (1AB0)	CHARACTER	160	TIOPDB1	A DSIPDB
6992 (1B50)	CHARACTER	160	.IOPDB2	ANOTHER DSIPDB
7152 (1BF0)	CHARACTER	160	TIOPDB3	ONE MORE DSIPDB
7312 (1C90)	CHARACTER	160	TIOPDB4	AND STILL ANOTHER DSIPDB
7472 (1D30)	CHARACTER	360		RESERVED

DSIHCT TIB EXTENSION

860 (35C)	STRUCTURE	2330	TIBHCT	HCT EXTENTION
860 (35C)	CHARACTER	8	TIHRPLCT	HCT TASK RPLS
860 (35C)	A-ADDRESS	4	TIHSRPL	ADDR OF HCT SEND ACF/VTAM RPL
864 (360)	A-ADDRESS	4	TIHCLRPL	ADDR OF CLSDST ACF/VTAM RPL
868 (364)	CHARACTER	4	TIHNIBCT	HCT TASK NIBS
868 (364)	A-ADDRESS	4	TIHNIB	ADDR OF ACF/VTAM NIB
872 (368)	CHARACTER	12	TIHELT	ECB LIST USED FOR EXEC CONTROL (3 ELT ENTRIES OF 4 CHARS EACH)
872 (368)	A-ADDRESS	4	TIHELTP	POINTER TO THE ECB
1...			TIHELTLA	LAST ECB FLAG
884 (374)	CHARACTER	8	TIHHTI	HARDCPY TRANSFER INFORMATION
884 (374)	SIGNED	4	TIHHECB	HARDCPY ECB
888 (378)	A-ADDRESS	4	TIHHINQ	HARDCPY INPUT QUEUE
892 (37C)	CHARACTER	8	TIHDCHAR	DEVICE CHARACTERISTICS
900 (384)	CHARACTER	540	TIHOBUF1	HCT OUTPUT BUFFER 1
900 (384)	A-ADDRESS	4	TIHO1NXT	CHAIN PTR TO NEXT BUFFER
904 (388)	CHARACTER	24	TIHO1HDR	STANDARD BUFFER HEADER
928 (3A0)	CHARACTER	512	TIHO1BDY	BUFFER BODY
1440 (5A0)	CHARACTER	540	TIHOBUF2	HCT OUTPUT BUFFER 2
1440 (5A0)	A-ADDRESS	4	TIHO2NXT	CHAIN PTR TO NEXT BUFFER
1444 (5A4)	CHARACTER	24	TIHO2HDR	STANDARD BUFFER HEADER

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
1468 (5BC)	CHARACTER	512	TIHO2BDY	BUFFER BODY
1980 (7BC)	SIGNED	4	TIHRVECB	RECEIVE ECB
1984 (7C0)	CHARACTER	600	TIHSWBI	DEFAULT (FIC) IMMED DSISWB
2584 (A18)	CHARACTER	600	TIHSWBN	DEFAULT (FIC) NORMAL DSISWB
3184 (C70)	CHARACTER	1	TIHRPLRC	RPLRTNCD SAVE AREA
3185 (C71)	CHARACTER	1	TIHRPLFB	RPLFDB2 SAVE AREA
3186 (C72)	CHARACTER	4	TIHRPLSN	RPL SENSE SAVE AREA

DSIPPT TIB EXTENTION

860 (35C)	STRUCTURE	2444	TIBPPT	PPT EXTENTION
860 (35C)	CHARACTER	8	TIPRPLCT	PPT TASK RPLS
860 (35C)	A-ADDRESS	4	TIPRCRPL	ADDR OF POI REC-CMD ACF/VTAM RPL
864 (360)	A-ADDRESS	4	TIPSCRPL	ADDR OF POI SEND-CMD ACF/VTAM RPL
868 (364)	CHARACTER	0	TIPNIBCT	PPT TASK NIBS
868 (364)	CHARACTER	24	TIPELT	ECB LIST USED FOR EXEC CONTROL (6 ELT ENTRIES OF 4 CHARS EACH)
868 (364)	A-ADDRESS	4	TIPELTP	POINTER TO THE ECB
1...			TIPELTLA	LAST ECB FLAG
892 (37C)	SIGNED	4	TIPRCECB	RECEIVE CMD ECB
896 (380)	CHARACTER	156	TIPRCBUF	POI REC-CMD BUFFER
896 (380)	CHARACTER	24	TIPRCHDR	STANDARD BUFFER HEADER
920 (398)	CHARACTER	132	TIPRCBDY	BUFFER BODY
1052 (41C)	CHARACTER	116		RESERVED
1168 (490)	CHARACTER	364	TIPCWB	DSICWB FOR PPT
1532 (5FC)	CHARACTER	160	TIPPDB	DSIPDB FOR DSIPPT
1692 (69C)	CHARACTER	600	TIPSWBI	DEFAULT (FIC) IMMED DSISWB
2292 (8F4)	CHARACTER	600	TIPSWBN	DEFAULT (FIC) NORMAL DSISWB
2892 (B4C)	CHARACTER	32	TIPTILNK	TIMER LIFEBOAT LINKAGE
2924 (B6C)	A-ADDRESS	4	TIPNCPTR	NORMAL CMD DYNAMIC BFR
2928 (B70)	SIGNED	2		RESERVED
2930 (B72)	SIGNED	2	TIPNCLEN	NORMAL CMD BFR LENGTH
2932 (B74)	SIGNED	4	TIPCORNX	INDEX OF LAST TIPCOREL ENTRY: USE C S LOGIC
2936 (B78)	CHARACTER	200	TIPCOREL	SOLICITED MSG CORRELATION TABLE
2936 (B78)	CHARACTER	10	TIPCOREN	TIPCOREL ENTRY OVERLAY
2936 (B78)	BITSTRING	2	TIPCORSQ	SENDCMD SEQUENCE NUMBER
2938 (B7A)	CHARACTER	8	TIPCORIG	ORIGINATOR I.D.

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
3136 (C40)	A-ADDRESS	4	TIPTIMRQ	ADDR TIMER REQEST QUEUE
3140 (C44)	SIGNED	4	TIPTMECB	SERVICE TIMER ELEMENT ECB
3144 (C48)	SIGNED	4	TIPTIWK	OUTSTANDING TIMER UNIT VALUE
3148 (C4C)	CHARACTER	156	TIPNCBFR	PPT NORMAL CMD BUFFER
3148 (C4C)	CHARACTER	24	TIPNCHDR	STANDARD BUFFER HEADER
3172 (C64)	CHARACTER	132	TIPNCBDY	BUFFER BODY

DSITCT TIB EXTENSION

860 (35C)	STRUCTURE	4000	TIBTCT	TCT EXTENSION
860 (35C)	CHARACTER	4000		4000 BYTE EXTENSION
860 (35C)	CHARACTER	0	TICEND	END OF EXTENSION

DSIMNT TIB EXTENTION

860 (35C)	STRUCTURE	1984	TIBTIM	MNT TIB EXTENTION
860 (35C)	CHARACTER	16	TIMRPLCT	MNT TASK RPLS
860 (35C)	A-ADDRESS	4	TIMRPLX	EXITS RPL ADDR
864 (360)	A-ADDRESS	4	TIMRPLM	MAIN TASK RPL ADDR
868 (364)	A-ADDRESS	4	TIMRPLXR	CROSS-DOMAIN RECEIVE RPL
872 (368)	A-ADDRESS	4	TIMRPLXS	CROSS-DOMAIN SEND RPL
876 (36C)	CHARACTER	8	TIMNIBCT	MNT TASK NIBS
876 (36C)	A-ADDRESS	4	TIMNIBX	EXITS NIB ADDR
880 (370)	A-ADDRESS	4	TIMNIBM	MAIN TASK NIB ADDR
884 (374)	A-ADDRESS	4	TIMXACB	EXIT ACB ADDR
888 (378)	A-ADDRESS	72	TIMXSVA	EXITS SAVE AREA (18 ITEMS OF FIXED 31 EACH)
960 (3C0)	CHARACTER	24	TIMXNNPL	EXIT NNT TERM PARM LIST
960 (3C0)	CHARACTER	9	TIMXNNPU	PU NAME
969 (3C9)	CHARACTER	8	TIMXNNLU	LU NAME
977 (3D1)	CHARACTER	3	TIMXNNPD	PADDING
980 (3D4)	A-ADDRESS	4	TIMXNNSW	SWB ADDR
984 (3D8)	CHARACTER	151	TIMMSBUF	EXIT MESSAGE BUFFER
1135 (46F)	CHARACTER	5		RESERVED
1140 (474)	CHARACTER	8	TIMSVLUN	SAVE LU NAME
1148 (47C)	SIGNED	4	TIMERCOD	ENTRY REASON CODE
1152 (480)	CHARACTER	9	TIMDOMID	DOMAIN ID
1161 (489)	CHARACTER	3		RESERVED
1164 (48C)	CHARACTER	96	TIMLGSPS	LOGON EXIT SESSION PARMS
1260 (4EC)	CHARACTER	16	TIMDEVCH	DEVICE CHARACTERISTICS
1260 (4EC)	CHARACTER	1		SPACER

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
1261 (4ED)	CHARACTER	1	TIMDEVTP	DEVICE TYPE
1262 (4EE)	CHARACTER	1	TIMDEVMD	MODEL TYPE
1276 (4FC)	SIGNED	4	TIMGETSA	GETMAIN SAVE AREA
1280 (500)	SIGNED	4	TIMSVR14	EXIT REG14 SAVER
1284 (504)	CHARACTER	136	TIMMSGPL	EXIT DSIMMP MSG P-LIST
1420 (58C)	CHARACTER	200	TIMXWKA	EXITS WORK AREA
1620 (654)	CHARACTER	20	TIMXRBUF	CROSS-DOMAIN EXIT BUFFER
1640 (668)	SIGNED	4	TIMXMRTY	MACRO RETRY COUNT
1644 (66C)	CHARACTER	600	TIMSWBX	EXIT SWB
2244 (8C4)	CHARACTER	600	TIMSWBXL	LOGON EXIT SWB

0 (0)	STRUCTURE	4	TIBELTE	AN ELT ENTRY
0 (0)	A-ADDRESS	4	TIBEPTR	ADDR OF AN ECB
1...			TIBELAST	1=LAST ENTRY IN THE LIST

0 (0)	STRUCTURE	4	TIBBLKE	TIXBLK MAP
0 (0)	SIGNED	4	TIBECB	AN ECB
0 (0)	BITSTRING	1	TIBECBF	ECB FLAGS
1...			TIBECBWT	WAIT INDICATOR
.1..			TIBECBPO	POST INDICATOR
1 (1)	A-ADDRESS	3	TIBECBB	ECB BODY

STANDARD NCCF BUFFER HEADER

0 (0)	STRUCTURE	24	BUFHDR	BUFFER HEADER
0 (0)	SIGNED	2	HDRMLENG	MESSAGE LENGTH
2 (2)	SIGNED	2	HDRBLENG	LENGTH OF BUFFER IN USE
4 (4)	BITSTRING	1	HDRIND	HEADER INDICATORS
4 (4)	BITSTRING	1	HDRNMPOS	USED TO SAVE NM POST CODE FOR DELAYED DST POST
11..			HDRLNTYP	MULTILINE TYPE (SEE CONSTANTS)
5 (5)	CHARACTER	1	HDRMTYPE	MESSAGE TYPE
6 (6)	SIGNED	2	HDRTDISP	DISPLACEMENT TO 1ST TEXT LOCATION FROM BEGINNING OF THIS HEADER
8 (8)	SIGNED	4	HDRTSTMP	TIME STAMP FIELD
12 (C)	CHARACTER	8	HDRDOMID	DOMAIN ID
20 (14)	CHARACTER	4	HDRPOI	HDR USED FOR POI COMMANDS
20 (14)	A-ADDRESS	4	HDRWLHCT	HCT TVB ADDR FOR LOGGING SVCS
20 (14)	BITSTRING	1	HDRPID	POI HDR ID
21 (15)	BITSTRING	1	HDRPSTAT	POI STATUS FIELDS
22 (16)	SIGNED	2	HDRPMSG	UNIQUE MSG NUMBER FIELD
22 (16)	UNSIGNED	2	HDRCORID	DELIVER RU CORRELATION ID
24 (18)	CHARACTER	0	HDRTEXT	NON MESSAGE COMMAND TEXT

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
MESSAGE COMMAND EXTENSION				
24 (18)	STRUCTURE	12	HDRMCEXT	MSG CMD INFORMATION
24 (18)	A-ADDRESS	4	HDRNEXTM	NEXT MSG ON QUEUE
28 (1C)	CHARACTER	8	HDRSENDR	OPERATOR ID OF SENDER
36 (24)	CHARACTER	0	HDRMSG	MESSAGE-CMD TEXT
0 (0)	STRUCTURE	36	HDRMAR	MARGIN DATA OF PRINT LINE
0 (0)	CHARACTER	1	MARMTYPE	TYPE OF MESSAGE
1 (1)	CHARACTER	2		BLANKS
3 (3)	CHARACTER	8	MARMTIME	HH:MM:SS
3 (3)	CHARACTER	2	MARMHH	HOURS
5 (5)	CHARACTER	1	MARMC1	FIRST COLON
6 (6)	CHARACTER	2	MARMMM	MINUTES
8 (8)	CHARACTER	1	MARMC2	SECOND COLON
9 (9)	CHARACTER	2	MARMSS	SECONDS
11 (B)	CHARACTER	2		BLANKS
13 (D)	CHARACTER	8	MARMDOM	DOMAIN ID
21 (15)	CHARACTER	2		BLANKS
23 (17)	CHARACTER	8	MARMOPID	OPERATOR ID
31 (1F)	CHARACTER	5		MORE BLANKS
DSIOST TIB EXTENTION				
0 (0)	STRUCTURE	2736	TIOPSS	PRESENTATION SERVICES INFO (POINTED TO BY TIBPSSPT)
0 (0)	CHARACTER	1036	TIOOWBUF	PSS OUTPUT WORK BUFFER
0 (0)	A-ADDRESS	4	TIOOWBGN	ADDRESS OF BEGINNING OF BUFFER
4 (4)	A-ADDRESS	4	TIOOWEND	ADDR END OF BUFFER
8 (8)	SIGNED	4	TIOOWLEN	LENGTH OF BUFFER
12 (C)	CHARACTER	1024	TIOOWBDY	BUFFER BODY
1036 (40C)	CHARACTER	432	TIOMSAVE	RESERVED
1468 (5BC)	CHARACTER	432	TIOESAVE	SAVE AREA ARRAY (6 ITEMS OF 72 BYTES EACH)
1900 (76C)	A-ADDRESS	4	TIOOWNXT	ADDR OF NEXT BLOCK OF DATA TO BE SENT
1904 (770)	A-ADDRESS	4	TIOOWAVL	ADDR OF NEXT AVAILABLE LOCATION IN TIOOWBDY
1908 (774)	CHARACTER	536	TIOPSSND	PSM OUTPUT SEND BUFFER
1908 (774)	CHARACTER	24	TIOPSHDR	STANDARD BUFFER HEADER
1932 (78C)	CHARACTER	512	TIOPSBDY	BUFFER BODY
2444 (98C)	A-ADDRESS	4	TIOCFMAT	ADDR OF CURRENT DEVICE FORMAT
2448 (990)	CHARACTER	8	TIONFMAT	NAME OF NEW DEVICE FORMAT

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
2456 (998)	SIGNED	2	TIOSLCNT	CURRENT SCREEN LINE COUNT
2458 (99A)	UNSIGNED	1	TIOSLINP	INPUT LINE COUNT
2459 (99B)	UNSIGNED	1	TIOMARLN	SCREEN MARGIN LENGTH
2460 (99C)	CHARACTER	8	TIODCHAR	OST DEVICE CHARACTERISTICS
2460 (99C)	SIGNED	2	TIOMAXRU	MAX RU SIZE OUTBOUND
2468 (9A4)	CHARACTER	4	TIOSCREN	DEVICE SCREEN CHARACTERISTICS
2468 (9A4)	SIGNED	2	TIOROWCT	MAX ROWS ON SCREEN
2470 (9A6)	SIGNED	2	TIOCOLCT	MAX COLUMNS ON SCREEN
2472 (9A8)	SIGNED	4	TIOWRECB	WRAP ECB
2476 (9AC)	SIGNED	4	TIOSDECB	SEND ECB
2480 (9B0)	A-ADDRESS	4	TIORSPHD	ADDR OF RESPONSE MSG QUEUE
2484 (9B4)	A-ADDRESS	4	TIORSPTR	ADDR OF RESPONSE MSG POCESSING POINTER
2488 (9B8)	A-ADDRESS	4	TIOINADR	ADDR OF INPUT DATA AREA
2492 (9BC)	CHARACTER	1	TIOPSIND	PRESENTATION SERVICES INDICATORS
1...			TIOAWRAP	1=AUTOWRAP IS ACTIVE
.1..			TIOSCNUL	1=SCREEN UNLOCK (RESET OPERATION FIELD ON NEXT SEND)
..1.			TIOFSTLN	1=FIRST LINE OF A MSG BLOCK
...1			TIOSCNRF	1=SCREEN REFRESH (REFRESH CURRENT FORMAT)
.... 1...			TIONEWFM	1=REPLACE NEW FORMAT ON NEXT SEND
.... .1..			TIOIMSND	1=IMMEDIATE MSG WAITING TO BE SENT
.... ..1.			TIOFSENT	1=FORMAT SENT (SCREEN HAS BEEN REFRESHED)
....1			TIOSCNLK	1=SCREEN IS LOCKED
2493 (9BD)	CHARACTER	1	TIOPSIN2	MORE PRESENTATION SERVICES INDICATORS
1...			TIOEWA	1=ERASE/WRITE ALT REQUIRED
.1..			TIOPSOFM	1=OUTPUT PROCESSING IN FULL-LINE MODE
..1.			TIOPSMFL	1=HOLD BUFFER IN FULL-LINE MODE
...1			TIOCMDSD	1=OUTPUT TO INPUT LINE 0=NORMAL OUTPUT
.... 1...			TIOCMDGO	1=CMD REFRESH ACTIVE 0=REFRESH NOT ACTIVE
.... .1..			TIOFWRAP	1=FULL-LINE AUTOWRAP 0=NO FULL-LINE AUTOWRAP
2494 (9BE)	CHARACTER	1	TIOSNDLK	SEND LOCK (SEND IN PROG)
2495 (9BF)	UNSIGNED	1	TIOSTROW	SCREEN TIMER ROW ADDRESS
2496 (9C0)	UNSIGNED	1	TIOSTCOL	SCREEN TIMER COLUMN ADDRESS
2497 (9C1)	UNSIGNED	1	TIOIMROW	IMMED MSG AREA ROW ADDRESS
2498 (9C2)	UNSIGNED	1	TIOIMCOL	IMMED MSG AREA COLUMN ADDRESS
2499 (9C3)	UNSIGNED	1	TIORYROW	READY MSG ROW ADDRESS
2500 (9C4)	UNSIGNED	1	TIORYCOL	READY MSG COLUMN ADDRESS
2501 (9C5)	UNSIGNED	1	TIOSLROW	SCREEN LOCK IND ROW ADDRESS
2502 (9C6)	UNSIGNED	1	TIOSLCOL	SCREEN LOCK IND COL ADDRESS
2503 (9C7)	UNSIGNED	1	TIOCDROW	COMMAND AREA ROW ADDRESS
2504 (9C8)	UNSIGNED	1	TIOCDCOL	COMMAND AREA COLUMN ADDRESS
2505 (9C9)	UNSIGNED	1	TIOCLINP	CURRENT INPUT LINE COUNT
2506 (9CA)	CHARACTER	2		FOR FUTURE USE/ALIGNMENT

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
2508 (9CC)	A-ADDRESS	4	TIOFTTBL	PTR TO FMT TABLE FOR THIS SESSION
2512 (9D0)	CHARACTER	104	TIOIMBUF	IMMED MSG BUFFER
2512 (9D0)	CHARACTER	24	TIOIMHDR	STANDARD BUFFER HEADER
2536 (9E8)	CHARACTER	80	TIOIMBDY	TEXT AREA
2616 (A38)	SIGNED	4	TIOPSFLC	OUTPUT LINE COUNTER FOR FULL-LINE MODE
2620 (A3C)	A-ADDRESS	4	TIOPSIFR	PSS IFR POINTER
2624 (A40)	SIGNED	4	TIOPSIFL	PSS IFR LENGTH
2628 (A44)	A-ADDRESS	4	TIOPSSWB	PSS SWB
2632 (A48)	A-ADDRESS	4	TIOAPECB	ASYPANEL ECB PTR
2636 (A4C)	A-ADDRESS	4	TIOAPBUF	ASYPANEL INPUT BUFFER
2640 (A50)	SIGNED	2	TIOAPBLN	ASYPANEL BUFFER LENGTH
2642 (A52)	SIGNED	2	TIOAPTLN	TEMPORARY BUFFER LENGTH
2644 (A54)	A-ADDRESS	4	TIOAPPTR	TEMPORARY PANEL POINTER
2648 (A58)	A-ADDRESS	4	TIOAPRLN	RETURNED LENGTH POINTER
2652 (A5C)	CHARACTER	84		RESERVED

SCOPECLASS FLAGS EXTENTION

0 (0)	STRUCTURE	0	TIBSCO	TASK SCOPE FLAGS (POINTED TO BY TIBSCTSK)
0 (0)	CHARACTER	0		LENGTH IS VARIABLE

STRUCTURE TO MAP DSIMMP MTM MESSAGE PROCESSOR PARM LIST WHICH
WILL COVER MVTMSGPL AND TIMMSGPL PARMAMETER LIST AREAS

0 (0)	STRUCTURE	137	MMPPARML	DSIMMP PARM LIST
0 (0)	A-ADDRESS	72	MMPPLSAV	SAVE AREA (18 ITEMS OF FIXED 31 EACH)
72 (48)	A-ADDRESS	4	MMPPLSWB	SWB ADDRESS
76 (4C)	A-ADDRESS	4	MMPPLBUF	MSG BUFFER ADDRESS
80 (50)	A-ADDRESS	4	MMPPLPOP	OPERATOR ID ADDR FOR QUEUEING
84 (54)	CHARACTER	3	MMPPLPID	MESSAGE ID
87 (57)	CHARACTER	1	MMPPLRV1	RESERVED
88 (58)	A-ADDRESS	4	MMPPLVA1	MSG VARIABLE 1 ADDRESS
92 (5C)	A-ADDRESS	4	MMPPLVA2	MSG VARIABLE 2 ADDRESS
96 (60)	A-ADDRESS	4	MMPPLVA3	MSG VARIABLE 3 ADDRESS
100 (64)	A-ADDRESS	4	MMPPLVA4	MSG VARIABLE 4 ADDRESS
104 (68)	A-ADDRESS	4	MMPPLVA5	MSG VARIABLE 5 ADDRESS
108 (6C)	A-ADDRESS	4	MMPPLVA6	MSG VARIABLE 6 ADDRESS

OFFSETS		TYPE	LENGTH	NAME	DESCRIPTION
112	(70)	A-ADDRESS	4	MMPPLVA7	MSG VARIABLE 7 ADDRESS
116	(74)	A-ADDRESS	4	MMPPLVA8	MSG VARIABLE 8 ADDRESS
120	(78)	A-ADDRESS	4	MMPPLVA9	MSG VARIABLE 9 ADDRESS
124	(7C)	UNSIGNED	1	MMPPLVL1	MSG VARIABLE 1 LENGTH
125	(7D)	UNSIGNED	1	MMPPLVL2	MSG VARIABLE 2 LENGTH
126	(7E)	UNSIGNED	1	MMPPLVL3	MSG VARIABLE 3 LENGTH
127	(7F)	UNSIGNED	1	MMPPLVL4	MSG VARIABLE 4 LENGTH
128	(80)	UNSIGNED	1	MMPPLVL5	MSG VARIABLE 5 LENGTH
129	(81)	UNSIGNED	1	MMPPLVL6	MSG VARIABLE 6 LENGTH
130	(82)	UNSIGNED	1	MMPPLVL7	MSG VARIABLE 7 LENGTH
131	(83)	UNSIGNED	1	MMPPLVL8	MSG VARIABLE 8 LENGTH
132	(84)	UNSIGNED	1	MMPPLVL9	MSG VARIABLE 9 LENGTH
133	(85)	CHARACTER	1	MMPPLFLG	MSG REQUEST TYPE FLAG
	1...			MMPPLMBS	1=BUILD MSG
	.1..			MMPPLWCS	1=WRITE MSG TO SYS OP
	..1.			MMPPLQOP	1=QUEUE TO OPER ID
	...1			MMPPLQAR	1=QUEUE TO AUTHORIZED RECEIVER
	 1111			MMPPLRV2	RESERVED
134	(86)	CHARACTER	3	MMPPLRV3	RESERVED

CROSS REFERENCE

BUFHDR 0 (0)
DSITIB 0 (0)
HDRBLENG 2 (2)
HDRCORID 22 (16)
HDRDOMID 12 (C)
HDRIND 4 (4)
HDRLNTYP 4 X'C0'
HDRMAR 0 (0)
HDRMCEXT 24 (18)
HDRMLENG 0 (0)
HDRMSG 36 (24)
HDRMTYPE 5 (5)
HDRNEXTM 24 (18)
HDRNMPOS 4 (4)
HDRPID 20 (14)
HDRPMSG 22 (16)
HDRPOI 20 (14)
HDRPSTAT 21 (15)
HDRSENDR 28 (1C)
HDRTDISP 6 (6)
HDRTEXT 24 (18)
HDRTSTMP 8 (8)
HDRWLHCT 20 (14)
MARMC1 5 (5)
MARMC2 8 (8)
MARMDOM 13 (D)
MARMHH 3 (3)
MARMMM 6 (6)
MARMOPID 23 (17)
MARMSS 9 (9)
MARMTIME 3 (3)
MARMTYPE 0 (0)
MMPPARML 0 (0)
MMPPLBUF 76 (4C)
MMPPLFLG 133 (85)
MMPPLMBS 133 X'80'
MMPPLPID 84 (54)
MMPPLPOP 80 (50)
MMPPLQAR 133 X'10'
MMPPLQOP 133 X'20'
MMPPLRV1 87 (57)
MMPPLRV2 133 X'0F'
MMPPLRV3 134 (86)
MMPPLSAV 0 (0)
MMPPLSWB 72 (48)
MMPPLVA1 88 (58)
MMPPLVA2 92 (5C)
MMPPLVA3 96 (60)
MMPPLVA4 100 (64)
MMPPLVA5 104 (68)
MMPPLVA6 108 (6C)
MMPPLVA7 112 (70)
MMPPLVA8 116 (74)
MMPPLVA9 120 (78)
MMPPLVL1 124 (7C)
MMPPLVL2 125 (7D)
MMPPLVL3 126 (7E)
MMPPLVL4 127 (7F)
MMPPLVL5 128 (80)
MMPPLVL6 129 (81)
MMPPLVL7 130 (82)
MMPPLVL8 131 (83)
MMPPLVL9 132 (84)
MMPPLWCS 133 X'40'
TIBACB 8 (8)
TIBAGBFR 88 (58)
TIBAGPDB 76 (4C)
TIBAPID 20 (14)
TIBAPIDL 20 (14)
TIBAPIDN 21 (15)
TIBAPWD 29 (1D)
TIBAPWDL 29 (1D)
TIBAPWDN 30 (1E)
TIBAREA1 38 (26)
TIBAREA2 112 (70)
TIBBLKE 0 (0)
TIBCBH 0 (0)
TIBCCL 824 X'80'
TIBCICB 804(324)
TIBCLBQ 112 (70)
TIBCLBWK 808(328)
TIBCLECB 116 (74)
TIBECB 0 (0)
TIBECBB 1 (1)
TIBECBF 0 (0)
TIBECBPO 0 X'40'
TIBECBWT 0 X'80'
TIBEDATD 524(20C)
TIBELAST 0 X'80'
TIBELT 16 (10)
TIBELTE 0 (0)
TIBEPTR 0 (0)
TIBEXLST 12 (C)
TIBEXSWB 56 (38)
TIBEXTEN 860(35C)
TIBFLGS 824(338)
TIBHCT 860(35C)
TIBICBFR 84 (54)
TIBICCWB 48 (30)
TIBICPDB 72 (48)
TIBINT 780(30C)
TIBINT1 780(30C)
TIBINT2 784(310)
TIBLOGBE 832(340)
TIBLOGBF 812(32C)
TIBMMD 825(339)
TIBMRCWB 52 (34)
TIBMRPDB 68 (44)
TIBMSGNM 788(314)
TIBNCBFR 80 (50)
TIBNCCWB 44 (2C)
TIBNCPDB 64 (40)
TIBNDATD 268(10C)
TIBNPSWB 60 (3C)
TIBOSEXT 112 (70)
TIBOSLST 116 (74)
TIBOST 860(35C)
TIBPOICT 38 (26)
TIBPPT 860(35C)
TIBPSSPT 40 (28)
TIBRECV 96 (60)
TIBRETCD 796(31C)
TIBSAVEE 196 (C4)
TIBSAVES 124 (7C)
TIBSCO 0 (0)
TIBSCTSK 800(320)
TIBSEND 92 (5C)
TIBTAFFY 104 (68)
TIBTCT 860(35C)
TIBTIM 860(35C)
TIBTINAM 816(330)
TIBTSEQ 38 (26)
TIBTVB 4 (4)
TIBUFLD 100 (64)
TIBXECB 120 (78)
TICEND 860(35C)
TIHCLRPL 864(360)
TIHDCHAR 892(37C)
TIHELT 872(368)
TIHELTLA 872 X'80'
TIHELTP 872(368)
TIHHECB 884(374)
TIHHINQ 888(378)
TIHHTI 884(374)
TIHNIB 868(364)
TIHNIBCT 868(364)
TIHOBUF1 900(384)
TIHOBUF2 1440(5A0)
TIHO1BDY 928(3A0)
TIHO1HDR 904(388)
TIHO1NXT 900(384)
TIHO2BDY 1468(5BC)
TIHO2HDR 1444(5A4)
TIHO2NXT 1440(5A0)
TIHRPLCT 860(35C)
TIHRPLFB 3185(C71)
TIHRPLRC 3184(C70)
TIHRPLSN 3186(C72)
TIHRVECB 1980(7BC)
TIHSRPL 860(35C)
TIHSWBI 1984(7C0)
TIHSWBN 2584(A18)
TIMDEVCH 1260(4EC)
TIMDEVMD 1262(4EE)
TIMDEVTP 1261(4ED)
TIMDOMID 1152(480)
TIMERCOD 1148(47C)
TIMGETSA 1276(4FC)
TIMLGSPS 1164(48C)
TIMMSBUF 984(3D8)
TIMMSGPL 1284(504)
TIMNIBCT 876(36C)
TIMNIBM 880(370)
TIMNIBX 876(36C)
TIMRPLCT 860(35C)
TIMRPLM 864(360)
TIMRPLX 860(35C)
TIMRPLXR 868(364)
TIMRPLXS 872(368)
TIMSVLUN 1140(474)
TIMSVR14 1280(500)
TIMSWBX 1644(66C)
TIMSWBXL 2244(8C4)
TIMXACB 884(374)
TIMXMRTY 1640(668)
TIMXNNLU 969(3C9)
TIMXNNPD 977(3D1)
TIMXNNPL 960(3C0)
TIMXNNPU 960(3C0)
TIMXNNSW 980(3D4)
TIMXRBUF 1620(654)
TIMXSVA 888(378)
TIMXWKA 1420(58C)
TIOACEE 1780(6F4)
TIOAPBLN 2640(A50)
TIOAPBUF 2636(A4C)
TIOAPECB 2632(A48)
TIOAPPTR 2644(A54)
TIOAPRLN 2648(A58)
TIOAPTLN 2642(A52)
TIOAWRAP 2492 X'80'
TIOCDCOL 2504(9C8)
TIOCDNIB 888(378)
TIOCDROW 2503(9C7)
TIOCFMAT 2444(98C)
TIOCLINP 2505(9C9)
TIOCMDGO 2493 X'08'
TIOCMDSD 2493 X'10'
TIOCOLCT 2470(9A6)
TIOCWBI 454(11BC)

CROSS REFERENCE

TIOCWB2	490(1328)	TIOPSIFR	2620(A3C)	TIPSWBN	2292(8F4)
TIOCWB3	526(1494)	TIOPSIND	2492(9BC)	TIPTILNK	2892(B4C)
TIODBLEN	1772(6EC)	TIOPSIN2	2493(9BD)	TIPTIMRQ	3136(C40)
TIODCHAR	2460(99C)	TIOPSMFL	2493 X'20'	TIPTIWK	3144(C48)
TIOELT	892(37C)	TIOPSOFM	2493 X'40'	TIPTMECB	3140(C44)
TIOELTLA	892 X'80'	TIOPSS	0 (0)		
TIOELTP	892(37C)	TIOPSSND	1908(774)		
TIOESAVE	1468(5BC)	TIOPSSSP	1804(70C)		
TIOEWA	2493 X'80'	TIOPSSWB	2628(A44)		
TIOFLGS	1262(4EE)	TIOQSECB	944(3B0)		
TIOFLQ	1784(6F8)	TIORABUF	1776(6F0)		
TIOFSENT	2492 X'02'	TIORAECB	932(3A4)		
TIOFSTLN	2492 X'20'	TIORARPL	876(36C)		
TIOFTTBL	2508(9CC)	TIORCBDY	1436(59C)		
TIOFWRAP	2493 X'04'	TIORCBUF	1412(584)		
TIOIBUF1	968(3C8)	TIORCECB	924(39C)		
TIOIBUF2	1116(45C)	TIORCHDR	1412(584)		
TIOIBUF3	1264(4F0)	TIORCRPL	868(364)		
TIOIMBDY	2536(9E8)	TIORFPTR	1768(6E8)		
TIOIMBUF	2512(9D0)	TIOROWCT	2468(9A4)		
TIOIMCOL	2498(9C2)	TIORPLCT	860(35C)		
TIOIMHDR	2512(9D0)	TIORSPHD	2480(9B0)		
TIOIMROW	2497(9C1)	TIORSPTR	2484(9B4)		
TIOIMSND	2492 X'04'	TIORSRPL	880(370)		
TIOINADR	2488(9B8)	TIORYCOL	2500(9C4)		
TIOI1BDY	1034(40A)	TIORYROW	2499(9C3)		
TIOI1CTL	1028(404)	TIOSAUTH	956(3BC)		
TIOI1HDR	968(3C8)	TIOSCNLK	2492 X'01'		
TIOI1PTR	1756(6DC)	TIOSCNRF	2492 X'10'		
TIOI1XTH	1008(3F0)	TIOSCNT	964(3C4)		
TIOI2BDY	1182(49E)	TIOSCNUL	2492 X'40'		
TIOI2CTL	1176(498)	TIOSCREN	2468(9A4)		
TIOI2HDR	1116(45C)	TIOSCRPL	872(368)		
TIOI2PTR	1760(6E0)	TIOSDECB	2476(9AC)		
TIOI2XTH	1156(484)	TIOSLCNT	2456(998)		
TIOI3BDY	1330(532)	TIOSLCOL	2502(9C6)		
TIOI3CTL	1324(52C)	TIOSLINP	2458(99A)		
TIOI3HDR	1264(4F0)	TIOSLROW	2501(9C5)		
TIOI3PTR	1764(6E4)	TIOSNDLK	2494(9BE)		
TIOI3XTH	1304(518)	TIOSTCOL	2496(9C0)		
TIOLOECB	940(3AC)	TIOSTROW	2495(9BF)		
TIOMARLN	2459(99B)	TIOSVCTL	1262 X'40'		
TIOMAXRU	2460(99C)	TIOSVRVR	1262 X'80'		
TIOMSAVE	1036(40C)	TIOSWBI	563(1600)		
TIONAUTH	960(3C0)	TIOSWBN	623(1858)		
TIONCNT	966(3C6)	TIOWRECB	2472(9A8)		
TIONEWFM	2492 X'08'	TIPCOREL	2936(B78)		
TIONFMAT	2448(990)	TIPCOREN	2936(B78)		
TIONIBCT	884(374)	TIPCORIG	2938(B7A)		
TIOOITSV	1114(45A)	TIPCORNX	2932(B74)		
TIOORRPL	860(35C)	TIPCORSQ	2936(B78)		
TIOOSNIB	884(374)	TIPCWB	1168(490)		
TIOOSRPL	864(360)	TIPELT	868(364)		
TIOOWAVL	1904(770)	TIPELTLA	868 X'80'		
TIOOWBDY	12 (C)	TIPELTP	868(364)		
TIOOWBGN	0 (0)	TIPNCBDY	3172(C64)		
TIOOWBUF	0 (0)	TIPNCBFR	3148(C4C)		
TIOOWEND	4 (4)	TIPNCHDR	3148(C4C)		
TIOOWLEN	8 (8)	TIPNCLEN	2930(B72)		
TIOOWNXT	1900(76C)	TIPNCPTR	2924(B6C)		
TIOPAECB	936(3A8)	TIPNIBCT	868(364)		
TIOPDB1	683(1AB0)	TIPPDB	1532(5FC)		
TIOPDB2	699(1B50)	TIPRCBDY	920(398)		
TIOPDB3	715(1BF0)	TIPRCBUF	896(380)		
TIOPDB4	731(1C90)	TIPRCECB	892(37C)		
TIOPROFL	948(3B4)	TIPRCHDR	896(380)		
TIOPSBDY	1932(78C)	TIPRCRPL	860(35C)		
TIOPSFLC	2616(A38)	TIPRPLCT	860(35C)		
TIOPSHDR	1908(774)	TIPSCRPL	864(360)		
TIOPSIFL	2624(A40)	TIPSWBI	1692(69C)		

Constants in DSITIB

NAME	VALUE	MEANING
Constants for HDRMTYPE Defined Values (Char 1)		
HDRTYPEA	'T'	SOLICITED MSG FROM ACF/TCAM
HDRTYPEB	'?'	SUPPRESSION CHAR
HDRTYPEC	'C'	CMD/MSG FROM CLIST
HDRTYPED	'!'	IMMEDIATE CMD MSG
HDRTYPEE	'E'	EXTERNAL NON-NCCF MESSAGE
HDRTYPEF	'F'	VSAM RECORD
HDRTYPEG	'G'	CSMI RECORD
HDRTYPEI	'I'	INTERNAL FUNCTION REQUEST
HDRTYPEJ	'''	NCCF-GENERATED FULL-LINE MESSAGE
HDRTYPEK	'"'	IBM-WRITTEN NON-NCCF GENERATED FULL-LINE MESSAGE
HDRTYPEL	'='	USER-WRITTEN FULL-LINE MESSAGE
HDRTYPEM	'M'	MESSAGE FROM MESSAGE COMMAND
HDRTYPEN	'-'	NCCF-GENERATED MSG
HDRTYPEP	'P'	MSG FROM COMMAND OR CLIST UNDER PPT
HDRTYPEQ	'Q'	UNSOLICITED MESSAGE FROM ACF/VTAM
HDRTYPER	'R'	RESPONSE TO ACF/VTAM
HDRTYPES	'S'	MSG TEXT SUBSTITUTED BY USER EXIT
HDRTYPET	'*'	COMMAND INPUT FROM TERMINAL
HDRTYPEU	'U'	RESERVED FOR USER IN USER EXIT OR IN COMMAND PROCESSOR
HDRTYPEV	' '	SOLICITED MESSAGE FROM ACF/VTAM
HDRTYPEW	'+'	NON-NCCF IBM-WRITTEN COMMAND PROCESSOR GENERATED MESSAGE
HDRTYPEX	'X'	CROSS DOMAIN (NNT-TO-OST) CMD
HDRTYPEY	'>'	REPLY REQUIRED
HDRTYPEZ	'Z'	DST-GENERATED MSG
Values for Testing HDRLNTYP, Indicating Multiline Type (Bit 2)		
(HDRMTYPE=HDRTYPEJ, HDRTYPEK, HDRTYPEL Only)		
HDRLNCTL	B'11'	CONTROL LINE
HDRLNLBL	B'10'	LABEL LINE
HDRLNDAT	B'00'	DATA LINE
HDRLNEND	B'01'	DATA/END LINE
HDRLNCTL	B'11'	CONTROL LINE
Masks for Setting and Testing DSITIB Flags (Bit 1)		
TIBON	B'1'	FUNCTION IS ACTIVE
TIBOFF	B'0'	FUNCTION NOT ACTIVE
Constants for Setting and Testing DSITIB Fields (Fullword)		
TIBZERO	0	FOR 0 POINTER VALUE SETTINGS
TIOSCBCT	4	NUMBER OF OST SCBS
TIHSCBCT	2	NUMBER OF HCT SCBS
TIPSCBCT	1	NUMBER OF PPT SCBS
TIMSCBCT	0	NUMBER OF MNT SCBS

TVB

DSITVB: MAPS THE TASK VECTOR BLOCK FOR NCCF
BOUNDARY: DOUBLEWORD
LENGTH: 144 BYTES (X'90')
POINTED TO BY: IFR (IFRTWTVB,IFRTRTVB,IFRSSTVB)
TIB (TIBTVB)
(HDRWLHCT) HCT TCB FOR LOGGING SERVICES
MVT (MVTTVB) CHAIN HEADER
(MVTCTVB) TCT TVB
(MVTTCTVB) TVB TO PURGE
(MVTUXTVB)
SSB (SSBTVBD) LUNAME DONNER
(SSBTVBO) OBJECT TVB FOR START/MOVE
(SSBTVBI) INIT TVB FOR START/MOVE
SWB (SSMVAL)
(MQSTADR) RECEIVER'S TVB
(WLSHCT) HCT TVB
USE (USERTVB) SESSION TVB
INCLUDED BLOCKS: CBH (TVBCBH)

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
0	(0) STRUCTURE	144	DSITVB	TASK VECTOR BLOCK
0	(0) CHARACTER	4	TVBCBH	NCCF CONTROL BLOCK HEADER
4	(4) A-ADDRESS	4	TVBNEXT	ADDR OF NEXT TVB IN THE CHAIN
8	(8) A-ADDRESS	4	TVBTIB	ADDR OF THE TIB FOR THIS TASK
12	(C) A-ADDRESS	4	TVBTCB	ADDR OF SYSTEM TCB FOR THE TASK
16	(10) A-ADDRESS	4	TVBMVT	ADDR OF NCCF MAIN VECTOR TABLE
20	(14) SIGNED	4	TVBECB	ECB USED BY DSIMNT FOR THE TASK
24	(18) SIGNED	4	TVBTECB	TERMINATION ECB--INDICATES EOT PROCESSING WHEN POSTED
28	(1C) A-ADDRESS	4	TVBEXMSG	PTR TO AN EXCEPTION MSG TO BE HANDLED BY DSIMNT
32	(20) SIGNED	4	TVBMECB	MESSAGE ECB--INDICATES MSG IN PROCESSING WHEN POSTED
36	(24) A-ADDRESS	4	TVBMPRIQ	PRIVATE MESSAGE QUEUE
40	(28) A-ADDRESS	4	TVBMPUBQ	PUBLIC MESSAGE QUEUE
44	(2C) A-ADDRESS	4	TVBHCTVB	ADDR OF HCT TVB FOR THIS TASK
48	(30) CHARACTER	1	TVBIND1	INDICATOR FLAGS
	1...		TVBREIN	1=TASK REINSTATEMENT REQUEST
	.1..		TVBREDP	1=TASK REDISPATCH REQUEST
	..1.		TVBTERM	1=TASK TERMINATION IN PROG
	...1		TVBDETCH	1=TASK IS TO BE DETACHED
	.... 1...		TVBATTCH	1=TASK IS TO BE ATTACHED
	.... .1..		TVBCLSD	1=CLSDST PASS REQUESTED
	.... ..1.		TVBLABT	1=LOGON ABORT
	1		TVBSTART	1=START CMD ISSUED FOR TASK
49	(31) CHARACTER	1	TVBIND2	INDICATOR FLAGS
	1...		TVBSTOP	1=STOP CMD ISSUED FOR TASK
	.1..		TVBBYAP	1=BYPASS AUTH PROCESSING
	..1.		TVBCNRM	1=CLOSE NORMAL ISSUED BY TASK
	...1		TVBCIMD	1=CLOSE IMMEDAITE ISSUED FOR TASK

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
	.... 1...		TVBVCLOS	1=ACF/VTAM CLOSE ACB IS REQUIRED
	.... .1..		TVBMOVE	1=MOVE CMD ISSUED FOR TASK
	.... ..1.		TVBCDMP	1=CLOSE DUMP ISSUED FOR TASK
	1			RESERVED
50	(32) CHARACTER	1	TVBIND3	INDICATOR FLAGS
	1...		TVBACTV	1=TASK IS ACTIVE
	.1..		TVBLGON	1=LOGON IN PROCESS
	..1.		TVBLGOFF	1=LOGOFF IN PROGRESS
	...1		TVBAUTH	1=OPID IS AUTHORIZED
	.... 1...		TVBRESET	1=ATTN IND (RESET) FIELDED
	.... .1..		TVBNAUTH	1=NO AUTH CHKNG NEC
	.... ..1.		TVBRCVAI	1=RCV ANY HAS BEEN ISSUED
	1		TVBINXIT	1=PROCESSING IN AN EXIT
51	(33) CHARACTER	1	TVBIND4	INDICATOR FLAGS
	1...		TVBPAUSE	1=PAUSE HAS BEEN ISSUED
	.1..		TVBRCVRY	1=RECOVERY IN PROGRESS
	..1.		TVBNWDVC	1=NEW DEVICE ASSIGNED
	...1		TVBERIMM	1=ERASE IMMEDIATE MSG AREA AFTER NEXT INPUT
	.... 1...		TVBLGN	1=MAINTASK LOGON EXIT ENTERED
	.... .1..		TVBETXR	1=MAINTASK ETXR ENTERED
	.... ..1.		TVBSIMRQ	1=SIMLOGON REQUIRED FOR START OR MOVE COMMAND
	1		TVBSTOPF	1=TVB STOP FORCE INDICATOR
52	(34) SIGNED	2	TVBERCT	ERROR RETRY COUNT FOR THIS TASK
54	(36) SIGNED	2	TVBTCODE	TERMINATION CODE
54	(36) CHARACTER	1	TVBMTCOD	MODULE INDICATOR
55	(37) UNSIGNED	1	TVBNTCOD	INCIDENT INDICATOR
	1...		TVBPTCOD	TERMECB INDICATOR
56	(38) SIGNED	4	TVBHCUSE	COUNT OF SESSIONS TO (THIS) HCT
60	(3C) CHARACTER	8	TVBLUNAM	ACF/VTAM LU NAME
68	(44) CHARACTER	8	TVBOPID	OPERATOR ID USING THIS OST
76	(4C) A-ADDRESS	4	TVBUFLD	NCCF USER FIELD
80	(50) BITSTRING	1	TVBINUSE	IN USE BY START/STOP/MOVE CMD PROCESSOR WHEN SET
81	(51) UNSIGNED	3		RESERVED
84	(54) A-ADDRESS	4	TVBEXITQ	EXIT QUEUE STORAGE ANCHOR
88	(58) SIGNED	4	TVBTASKQ	TASK QUEUE STORAGE ANCHOR
92	(5C) A-ADDRESS	4	TVBSSB	ADDRESS OF SIMLOGON SERVICES BLOCK
96	(60) CHARACTER	8	TVBMODNM	OPTIONAL TASK LOAD MODULE NAME
104	(68) CHARACTER	8	TVBMEMNM	OPTIONAL TASK INIT PARAMITERS MEMBER NAME
112	(70) CHARACTER	4		
112	(70) CHARACTER	1	TVBATPRI	OPTIONAL TASK ATTACH PRIORITY
116	(74) BITSTRING	4	TVBSTAT	ADDITIONAL TASK FLAGS
116	(74) BITSTRING	4	TVBZSTAT	ADDITIONAL TASK FLAGS
116	(74) BITSTRING	1	TVBZIND1	DSM STATUS FLAGS
	1...		TVBZPUP	PRIMARY VSAM DATA SET UPDATE

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
	.1..		TVBZPIT	PRIMARY VSAM DATA SET IN TERMINATION
	..1.		TVBZSUP	SECONDARY VSAM DATA SET UPDATE
	...1		TVBZSIT	SECONDARY VSAM DATA SET IN TERMINATION
	.... 1...		TVBZCACT	CSMI IS ACTIVE
	.... .1..		TVBXACM	NNT ACCESS METHOD: 0=ACF/VTAM, 1=ACF/TCAM
	.... ..1.		TVBNOSP	0=NCCF
	1		TVBPNMOD	0=CURRENT SCREEN NOT MODIFIED, 1=SCREEN WAS MODIFIED
117	(75) BITSTRING	1	TVBZIND2	RESERVED
	1...		TVBPANEL	1=SCREEN REQUEST IN PROCESS, 0=NORMAL PSM PROCESS
	.1..		TVBASYIN	1=ASYPANEL INPUT REQUEST, 0=NORMAL PANEL PROCESS
	..1.		TVBUNPOS	1=UNDEFINED POS, 0=DEFINED POS
118	(76) BITSTRING	1	TVBZIND3	RESERVED
119	(77) BITSTRING	1	TVBZIND4	RESERVED
120	(78) A-ADDRESS	4	TVBTASKC	ACF/TCAM TASK COUNT FOR ID
124	(7C) CHARACTER	8	TVBAPID	LOGICAL APPL ID ASSIGNED IN ACF/TCAM ENVIRONMENT
132	(84) A-ADDRESS	4	TVBTOPQ	ACF/TCAM OPCTL QUEUE ANCHOR
132	(84) A-ADDRESS	4	TVBRDECB	ACF/TCAM CMSI READ ECB
136	(88) A-ADDRESS	4	TVBTXRQ	ACF/TCAM XDOM RCV ANY QUEUE ANCHOR
140	(8C) A-ADDRESS	4	TVBTEXQ	ACF/TCAM EXIT QUEUE ANCHOR
140	(8C) A-ADDRESS	4	TVBTRDYQ	TCT READY FOR READ QUEUE

CROSS REFERENCE

DSITVB	0	(0)	TVBTIB	8	(8)
TVBACTV	50	X'80'	TVBTOPQ	132	(84)
TVBAPID	124	(7C)	TVBTRDYQ	140	(8C)
TVBASYIN	117	X'40'	TVBTXRQ	136	(88)
TVBATPRI	112	(70)	TVBUFLD	76	(4C)
TVBATTCH	48	X'08'	TVBUNPOS	117	X'20'
TVBAUTH	50	X'10'	TVBVCLOS	49	X'08'
TVBBYAP	49	X'40'	TVBXACM	116	X'04'
TVBCBH	0	(0)	TVBZCACT	116	X'08'
TVBCDMP	49	X'02'	TVBZIND1	116	(74)
TVBCIMD	49	X'10'	TVBZIND2	117	(75)
TVBCLSD	48	X'04'	TVBZIND3	118	(76)
TVBCNRM	49	X'20'	TVBZIND4	119	(77)
TVBDETCH	48	X'10'	TVBZPIT	116	X'40'
TVBECB	20	(14)	TVBZPUP	116	X'80'
TVBERCT	52	(34)	TVBZSIT	116	X'10'
TVBERIMM	51	X'10'	TVBZSTAT	116	(74)
TVBETXR	51	X'04'	TVBZSUP	116	X'20'
TVBEXITQ	84	(54)			
TVBEXMSG	28	(1C)			
TVBHCTVB	44	(2C)			
TVBHCUSE	56	(38)			
TVBIND1	48	(30)			
TVBIND2	49	(31)			
TVBIND3	50	(32)			
TVBIND4	51	(33)			
TVBINUSE	80	(50)			
TVBINXIT	50	X'01'			
TVBLABT	48	X'02'			
TVBLGN	51	X'08'			
TVBLGOFF	50	X'20'			
TVBLGON	50	X'40'			
TVBLUNAM	60	(3C)			
TVBMECB	32	(20)			
TVBMEMNM	104	(68)			
TVBMODNM	96	(60)			
TVBMOVE	49	X'04'			
TVBMPRIQ	36	(24)			
TVBMPUBQ	40	(28)			
TVBMTCOD	54	(36)			
TVBMVT	16	(10)			
TVBNAUTH	50	X'04'			
TVBNEXT	4	(4)			
TVBNOSP	116	X'02'			
TVBNTCOD	55	(37)			
TVBNWDVC	51	X'20'			
TVBOPID	68	(44)			
TVBPANEL	117	X'80'			
TVBPAUSE	51	X'80'			
TVBPNMOD	116	X'01'			
TVBPTCOD	55	X'80'			
TVBRCVAI	50	X'02'			
TVBRCVRY	51	X'40'			
TVBRDECB	132	(84)			
TVBREDP	48	X'40'			
TVBREIN	48	X'80'			
TVBRESET	50	X'08'			
TVBSIMRQ	51	X'02'			
TVBSSB	92	(5C)			
TVBSTART	48	X'01'			
TVBSTAT	116	(74)			
TVBSTOP	49	X'80'			
TVBSTOPF	51	X'01'			
TVBTASKC	120	(78)			
TVBTASKQ	88	(58)			
TVBTCB	12	(C)			
TVBTCODE	54	(36)			
TVBTECB	24	(18)			
TVBTERM	48	X'20'			
TVBTEXQ	140	(8C)			

Constants in DSITVB

NAME	VALUE	MEANING
Masks For Setting and Testing TVBIND Flags (Bit 1)		
TVBON	B'1'	FUNCTION IS ACTIVE
TVBOFF	B'0'	FUNCTION IS NOT ACTIVE
Constants For Setting and Testing TVB Fields		
(Fullword)		
TVBZERO	0	FOR 0 POINTER VALUE SETTINGS
(Char 1)		
TVBTCODA	'A'	TVBMTCOD SETTING FOR DSILAR
TVBTCODD	'D'	TVBMTCOD SETTING FOR DSIDFA
TVBTCODF	'F'	TVBMTCOD SETTING FOR DSIENP
TVBTCODG	'G'	TVBMTCOD SETTING FOR DSILGN
TVBTCODH	'H'	TVBMTCOD SETTING FOR DSIHCT
TVBTCODL	'L'	TVBMTCOD SETTING FOR DSILTM
TVBTCODM	'M'	TVBMTCOD SETTING FOR DSILAM
TVBTCODN	'N'	TVBMTCOD SETTING FOR DSILAN
TVBTCODO	'O'	TVBMTCOD SETTING FOR DSIOST
TVBTCODP	'P'	TVBMTCOD SETTING FOR DSIPSM
TVBTCODQ	'Q'	TVBMTCOD SETTING FOR DSIPPT
TVBTCODR	'R'	TVBMTCOD SETTING FOR DSIRCV
TVBTCODS	'S'	TVBMTCOD SETTING FOR DSISTP
TVBTCODT	'T'	TVBMTCOD SETTING FOR DSITPE
TVBTCODV	'V'	TVBMTCOD SETTING FOR DSIDST
TVBTCODX	'X'	TVBMTCOD SETTING FOR DSINSE
TVBTCODZ	'Z'	TVBMTCOD SETTING FOR DSIDPR

USE

DSIUSE: MAPS THE USER EXIT PARAMETER LIST USED TO INTERFACE WITH ALL USER EXITS
BOUNDARY: DOUBLEWORD
LENGTH: 28 BYTES (X'18') + EXTENSION
INCLUDED BLOCKS: CBH (USERCBH)

OFFSETS		TYPE	LENGTH	NAME	DESCRIPTION
0	(0)	STRUCTURE	28	DSIUSE	USER EXIT PARAMETER LIST
0	(0)	CHARACTER	4	USERCBH	CONTROL BLOCK HEADER
0	(0)	CHARACTER	1		CONTROL BLOCK ID VALUE
1	(1)	UNSIGNED	1	USERCODE	SPECIFIC EXIT ROUTINE INDICATOR
4	(4)	A-ADDRESS	4	USERMSG	ADDR OF MESSAGE BUFFER
8	(8)	A-ADDRESS	4	USERLU	ADDR OF SESSION LUNAME
12	(C)	A-ADDRESS	4	USEROPID	ADDR OF SESSION OPERATOR ID
16	(10)	A-ADDRESS	4	USERSWB	ADDR OF A SWB FOR USE IN EXIT PROCESSING
20	(14)	A-ADDRESS	4	USERTVB	ADDR OF SESSION TVB
24	(18)	A-ADDRESS	4	USERPDB	ADDR OF THE PDB ASSOCIATED WITH THE MESSAGE BUFFER PASSED TO THE EXIT
28	(1C)	CHARACTER	0	USEREXT	EXTENSION FOR EXIT DSIEX12
28	(1C)	STRUCTURE	28	USERLGON	LOGON EXIT EXTENSION
28	(1C)	CHARACTER	8	USERPSWD	PASSWORD
36	(24)	CHARACTER	8	USERHCPY	HARDCOPY DEVICE NAME
44	(2C)	CHARACTER	8	USERPROF	PROFILE NAME
52	(34)	A-ADDRESS	4	USEDSRB	PROFILE NAME

CROSS REFERENCE

DSIUSE	0	(0)
USEDSRB	52	(34)
USERCBH	0	(0)
USERCODE	1	(1)
USEREXT	28	(1C)
USERHCPY	36	(24)
USERLGON	28	(1C)
USERLU	8	(8)
USERMSG	4	(4)
USEROPID	12	(C)
USERPDB	24	(18)
USERPROF	44	(2C)
USERPSWD	28	(1C)
USERSWB	16	(10)
USERTVB	20	(14)

Constants in DSIUSE

NAME	VALUE	MEANING
Symbolic Return Code Values--Returned From User Exits (Byte)		
USERASIS	0	USE BUFFER AS IS
USERDROP	4	DO NOT PROCESS THIS BUFFER--RC 4 FROM EXIT DSIEX12 WILL CANCEL LOGON
USERSWAP	8	MOVE BUFFER POINTED TO BY REG ZERO TO THE BUFFER POINTED TO BY USERMSG AND PROCESS IT IN PLACE OF THE ORIGINAL MSG
USERLOG	12	LOG TO DISK ONLY
USERLOGR	16	REPLACE MESSAGE AND LOG TO DISK
USERHCL	20	LOG TO HCL ONLY
USERHCLR	24	REPLACE MESSAGE AND LOG TO HARD-COPY
Constants For Setting and Testing USERCODE Field (Bit 8)		
USERDINT	X'E9'	DSM INITIALIZATION EXIT
USERVINT	X'EA'	VSAM INITIALIZATION EXIT
USERVINP	X'EB'	VSAM INPUT EXIT
USERVOUT	X'EC'	VSAM OUTPUT EXIT
USERCINP	X'ED'	CNMI INPUT EXIT
USERCOUT	X'EE'	CNMI OUTPUT EXIT
USERTRE	X'EF'	TCT INPUT EXIT

Appendix D. Sample User-Written Command Processor

This appendix is an example of a user-written command processor, DSIUSP, which displays an NCCF control block and its length, or displays data in storage. **Note:** This command processor is not executable as it is shown in this appendix.

```
         TITLE '     DSIUSP USER COMMAND PROCESSOR *** SHOW *** '
*********************************************************************
*********************************************************************
* N A M E    :-                                                     *
*      DSIUSP (NCCF USER COMMAND PROCESSOR FOR VERB 'SHOW').        *
*                                                                   *
* O B J E C T:-                                                     *
*      1. DISPLAY A NCCF CONTROL BLOCK FOR EXPLICIT/IMPLICIT LENGTH.*
*      2. DISPLAY DATA IN THE STORAGE.                              *
*                                                                   *
* DESCRIPTION:-                                                     *
*      THIS COMMAND PROCESSOR IS INVOKED WHEN 'SHOW' VERB IS ENTERED*
*      IN ONE OF THE FOLLOWING FORMS:                               *
*                                                                   *
*          (1)   SHOW CB=<NAME>                                     *
*          (2)   SHOW CB=<NAME>,LEN=<VALUE2>|*                      *
*          (3)   SHOW CB=<NAME>,LEN=<VALUE2>|*,TERM=<TERM ID>       *
*          (4)   SHOW CB=<NAME>,LEN=<VALUE2>|*,OPER=<OPER ID>       *
*          (5)   SHOW CB=<NAME>,LEN=<VALUE2>|*,TERM=<ID-1>,OPER=<ID-2> *
*          (6)   SHOW ADDR=<VALUE1>                                 *
*          (7)   SHOW ADDR=<VALUE1>,LEN=<VALUE2>                    *
*                                                                   *
*      KEYWORDS ARE NOT POSITIONAL AND AS SUCH MAY BE SPECIFIED IN ANY *
*      ORDER. KEYWORD 'CB' IS REQUIRED FOR COMMANDS 1 THRU 5, AND KEY- *
*      WORD 'ADDR' IS REQUIRED FOR COMMANDS 6 AND 7.  OTHER KEYWORDS*
*      ARE OPTIONAL.                                                *
*                                                                   *
*      WHERE NAME IS ABBREVIATED NAME OF A NCCF CONTROL BLOCK, VALUE1 *
*            IS ADDRESS IN HEX WHERE DISPLAY IS TO START, VALUE2 IS  *
*            LENGTH FOR WHICH DISPLAY IS DESIRED. LENGTH VALUE MAY BE *
*            ENTERED IN DECIMAL, OR IN HEX (PREFIXING THE VALUE BY X). *
*            WHEN C'*' IS ENTERED FOR THE LENGTH THEN IT IS ASSUMED TO *
*            BE THE IMPLIED LENGTH OF NCCF CONTROL BLOCK (THE VALUE IN *
*            CB-HEADER). IF LENGTH IS OMITTED THEN IT IS DEFAULTED TO  *
*            FOUR (4) BYTES. IF BOTH TERM AND OPER ARE CODED THEN OPER *
*            IS IGNORED.                                            *
*                                                                   *
*      IF 'HELP' IS ENTERED FOR <NAME>, THEN A BRIEF HELP-MENU IS   *
*      PRESENTED ON THE TERMINAL.                                   *
*                                                                   *
*      DISPLAY OF DATA CAN BE STOPPED ANY TIME BY SETTING THE TVBRESET *
*      INDICATOR VIA RESET COMMAND OF NCCF.                         *
*                                                                   *
*      ANY SYNTAX ERROR IN THE INPUT WILL CAUSE AN ERROR MESSAGE TO *
*      BE DISPLAYED ON THE TERMINAL. RETURN CODE IS SET TO VALUE 8. *
*      CONTROL IS RETURNED TO THE USER FOR NECESSARY CORRECTIONS.   *
```

```
*      AFTER STANDARD ENTRY LINKAGE, ADDRESSABILITY IS ESTABLISHED     *
*      FOR MVT, PDB, TIB, TVB AND THE WORKAREA IN CWB.  THE VALUE FOR  *
*      NUMBER OF ENTRIES IN THE PDB PASSED IS TESTED. IF THE VALUE IS  *
*      LESS THAN 3, THEN AN ERROR MESSAGE IS PRESENTED AND CONTROL IS  *
*      RETURNED TO THE USER WITH A RETURN CODE OF 8. IF THE VALUE IS   *
*      AN EVEN NUM, THEN ALSO AN ERR-MESSAGE IS PRESENTED AND CONTROL  *
*      IS RETURNED TO THE USER WITH A RETURN CODE OF 8.                *
*                                                                      *
*      CMD FORMAT 1 - 5 : THE NAME ENTERED IS COMPARED AGAINST A LIST  *
*         OF NAMES (SEE LIST LISTOFCB). IF A MATCH IS FOUND, THEN THE  *
*         ADDRESS OF THAT CONTROL BLOCK IS FOUND FROM EITHER MVT OR    *
*         FROM TIB. THE LENGTH OF THE CONTROL BLOCK IS IN THE FIRST    *
*         WORD (2ND HALF WORD / THE 4TH BYTE IN THE WORD). THE CONTROL *
*         BLOCK IS THEN DISPLAYED FOR THE EXPLICIT/IMPLICIT LENGTH.    *
*                                                                      *
*      CMD FORMAT 6 - 7 : STORAGE IS DISPLAYED FROM THE ADDRESS VALUE  *
*         ENTERED TO THE DESIRED LENGTH. INPUT IS CHECKED FOR NON-HEX  *
*         DIGITS IN HEX VALUE AND NON-DECIMAL DIGITS IN DECIMAL VALUE. *
*         IF SOME ERROR IS FOUND THEN AN ERROR MESSAGE IS DISPLAYED,   *
*         AND CONTROL IS RETURNED TO THE USER WITH RET CODE SET TO 8.  *
*         IF NO ERROR IS FOUND IN THE INPUT AND  EVERY THING IS GOOD,  *
*         DATA IS PRESENTED FOR DISPLAY ON THE TERMINAL 16-BYTES AT A  *
*         TIME. A CHECK IS MADE AT THIS POINT TO SEE IF THE TVBRESET   *
*         BIT IS ON (RESET KEY ENTERED).  IF SO, THEN THE DISPLAY IS   *
*         TERMINATED AND CONTROL IS RETURNED TO THE USER WITH RETURN   *
*         CODE SET TO 0.                                               *
*                                                                      *
* ENTRY POINT:-                                                        *
*      DSIUSP.                                                         *
*                                                                      *
* MODULE CHARACTERISTICS:-                                             *
*      PROCESSOR - ASSEMBLER                                           *
*      SIZE      - APPROXIMATELY 4K DECIMAL BYTES                      *
*      ATTRIBUTES- REENTRANT                                           *
*      MODE      - PROBLEM PROGRAM                                     *
*      PROTECTION- USER KEY                                            *
*                                                                      *
* I N P U T:-                                                          *
*      REGISTER 1  - ADDRESS OF PARAMETER LIST (CONTAINED IN CWB)      *
*      REGISTER 13 - ADDRESS OF HIS SAVE AREA                          *
*      REGISTER 14 - RETURN ADDRESS                                    *
*      REGISTER 15 - ENTRY POINT ADDRESS OF THIS PROGRAM               *
*      OTHER REGS. - UNRELATED INFORMATION                             *
*                                                                      *
* O U T P U T:-                                                        *
*      REGISTER 15 - RETURN CODE VALUE                                 *
*      OTHER REGS. - UNCHANGED FROM INPUT                              *
*                                                                      *
* NORMAL EXIT:-                                                        *
*      NORMAL RETURN TO THE CALLER.                                    *
*                                                                      *
* ERROR EXIT:-                                                         *
*      NONE.
```

```
* CONTROL BLOCKS:-                                                    *
*     FOLLOWING NCCF CONTROL BLOCKS ARE USED:                         *
*        DSICBH,  DSICWB,  DSIMVT,  DSIPDB,  DSISVL,  DSISWB,         *
*        DSITIB,  DSITVB.                                             *
*                                                                     *
* NCCF/SYSTEM MACROS:-                                                *
*        DSIPSS, DSILCS, AND DSIDATIM.                                *
*                                                                     *
* MESSAGES ISSUED:-                                                   *
*     VARIOUS ERROR MESSAGE ARE PRESENTED USING NCCF 'DSIPSS' MACRO   *
*     TO IDENTIFY SYNTAX ERROR, ERROR IN COMMAND FORMAT ETC.          *
*                                                                     *
* INTERNAL TABLES:-                                                   *
*     A TABLE OF ABBREVIATED NAMES FOR VARIOUS NCCF CONTROL BLOCKS.   *
*     VARIOUS TRANSLATE TABLES TO TRANSLATE INPUT/OUTPUT DATA.        *
*                                                                     *
***********************************************************************
***********************************************************************


***********************************************************************
*   ADDRESSABILITY IS ESTABLISHED VIA REG 12. SAVE AREA AND THE WORK  *
*   AREA IN THE CWB PASSED IS ADDRESSED VIA REG 13. THE WORK AREA IS  *
*   CLEARED BY PROPOGATING NULLS (00) THROUGH IT. REG 11 IS USED AS   *
*   BASE REG FOR TIB, REG 10 FOR TVB, REG 9 FOR MVT.                  *
*   TEMPORARILY USED BASE REGS: REG 8 FOR PDB, REG 2 FOR PDBENTRY OR  *
*   BUFHDR IN TIB EXTENSION.                                          *
***********************************************************************


DSIUSP   CSECT
         STM   R14,R12,12(R13)     SAVE HIS REGISTERS.
         BALR  R12,0               ESTABLISH ADDRESSABILITY
         USING *,R12               *** FOR THE PROGRAM.
         USING DSICWB,R1           ADDRESSABILITY FOR CWB PASSED.
         LA    R9,CWBSAVEA         POINT TO SAVE AREA IN CWB.
         ST    R9,8(R13)           SAVE IN HIS SAVE AREA (BACK PTR).
         ST    R13,4(R9)           SAVE HIS SAVE AREA ADDRESS.
         LR    R13,R9              POINT TO SAVE AREA AND USE IT AS
         USING CWBSAVEA,R13        *** BASE REG FOR WORK AREA TOO.
         MVI   WORKAREA,X'00'      MOVE X'00' THROUGH THE
         MVC   WORKAREA+1(255),WORKAREA   ***WORK AREA IN CWB.
         ST    R1,PARMADDR         SAVE PARM-LIST ADDRESS.
         DROP  R1                  RELEASE REG 1.
         L     R11,CWBTIB          ESTABLISH ADDRESSABILITY FOR
         USING DSITIB,R11          ** TASK INFORMATION BLOCK (TIB).
         L     R10,TIBTVB          ESTABLISH ADDRESSABILITY FOR
         USING DSITVB,R10          *** TASK VECTOR BLOCK (TVB).
         ST    R10,ADDRTVB         SAVE TVB ADDRESS TEMPORARILY.
         L     R9,TVBMVT           ESTABLISH ADDRRESSABILITY FOR
         USING DSIMVT,R9           *** MAIN VECTOR TABLE (MVT).
         L     R8,CWBPDB           ESTABLISH ADDRESSABILITY FOR
         USING DSIPDB,R8           ***PARSE DESCRIPTOR BLOCK (PDB).
```

```
*********************************************************************
*   NUMBER OF ENTRIES IN PDB IS CHECKED. IT SHOULD BE ATLEAST 3 AND  *
*   IT MUST BE AN ODD NUMBER. IF THIS IS NOT THE CASE THEN ERRMSG 1  *
*   IS PRESENTED. OTHERWISE OUTPUT AREA IS BLANKED OUT AND SEARCH IS *
*   MADE FOR ALL OF THE FORMAT KEYWORDS.                             *
*         REG 7  - POINTS TO THE OUTPUT AREA                         *
*         REG 6  - POINTS TO A GROUP OF FORMAT KEYWORDS              *
*         REG 4  - POINTS TO KEYWORD/KEYWORD VALUE IN BUFFER         *
*         REG 2  - POINTS TO A PDB ENTRY IN THE PDB PASSED           *
*         REG 1  - VALUE TO CONTROL SEARCH LOOP                      *
*         REG 0  - LENGTH OF A PDB ENTRY                             *
*********************************************************************


         CLI   PDBNOENT+1,X'03'    3 > NUMBER OF ENTRIES IN PDB ?
         BL    ERROR1              YES - TOO FEW ENTRIES.
         TM    PDBNOENT+1,X'01'    NUM OF PDB ENTRIES SHOULD BE ODD ?
         BZ    ERROR1              NO - SOMETHING IS MISSING.
         MVI   OUTAREA,C' '        PROPOGATE C' ' THROUGH THE
         MVC   OUTAREA+1(31),OUTAREA *** 32 BYTES OF OUTAREA.
         SLR   R3,R3               ZERO OUT REGISTER 3.
         LA    R6,FMTOKYWD         ADDRESS OF FMT 0 KEYWORDS.
         LA    R0,PDBENTND-PDBENTRY ** LENGTH OF A PDB ENTRY.
FNDKYWD1 EQU   *
         LA    R7,OUTAREA          ADDRESS OF OUTPUT AREA.
FNDKYWD2 EQU   *
         CLI   0(R6),C' '          LAST KEYWORD SEARCHED ?
         BE    DONEFMT             YES - DONE WITH THE SEARCH.
         SLR   R1,R1               ZERO REGISTER 1 AND INSERT
         IC    R1,PDBNOENT+1       *** NUMBER OF ENTRIES IN PDB.
         SRL   R1,1                DIVIDE BY 2 FOR LOOP CONTROL.
         LA    R2,PDBTABLE         ADDRESS OF 1ST PDB ENTRY.
         USING PDBENTRY,R2         ADDRESSABILITY FOR PDBENTRY.
```

```
*********************************************************************
*   IF ANY UNRECOGNISEABLE KEYWORD IS FOUND, ERRMSG 9 IS PRESENTED  *
*   AND CONTROL IS RETURNED TO THE USER. OTHERWISE PROCESSING CONTI- *
*   NUES WITH FINDING EACH KEYWORD IN A PRE-DEFINED ORDER.          *
*         REG 7  - POINTS TO OUTPUT AREA FOR KEYWORD/KEYWORD-VALUE  *
*         REG 6  - POINTS TO A GROUP OF FORMAT KEYWORDS             *
*         REG 5  - POINTS TO KEYWORDS TABLE                         *
*         REG 4  - POINTS TO KEYWORD/KEYWORD VALUE IN BUFFER        *
*         REG 3  - LENGTH OF KEYWORD IN THE BUFFER                  *
*         REG 2  - POINTS TO A PDB ENTRY IN THE PDB PASSED          *
*         REG 1  - VALUE TO CONTROL SEARCH LOOP                     *
*         REG 0  - LENGTH OF A PDB ENTRY                            *
*********************************************************************

FNDKYWD3 EQU   *
         AR    R2,R0              ADDRESS OF NEXT PDB ENTRY.
         L     R4,CWBBUF          ADDRESS OF THE BUFFER PASSED,
         AH    R4,PDBDISP         *** ADD DISPLACEMENT IN BUFFER.
         IC    R3,PDBLENG         GET LENGTH OF KEYWORD IN BUFFER
         LTR   R3,R3              LENGTH OF KEYWORD = 0 ?
         BZ    ERROR4             YES - IT IS WRONG, PUT ERR-MSG.
         CH    R3,FOUR            LENGTH OF KEYWORD > 4 ?
         BH    ERROR4             YES - IT IS WRONG, PUT ERR-MSG.
         BCTR  R3,0               DECREASE IT BY 1 FOR EXECUTE.
         EX    R3,GETKYWD         MOVE KEYWORD IN OUTPUT AREA.
         LA    R5,KYWRDTAB        ADDRESS OF KEY WORDS TABLE
CHKKYWD  EQU   *
         CLI   0(R5),C' '         END OF TABLE W/O FINDING KEYWORD ?
         BE    ERROR9             YES - GO PUT ERROR MESSAGE.
         CLC   0(4,R7),0(R5)      IS KEY WORD IN KEYWORDS TABLE ?
         BE    GOODKYWD           YES - CONTINUE PROCESSING.
         LA    R5,4(R5)           ADDR OF NEXT KEYWORD IN TABEL.
         B     CHKKYWD            GO CHECK AGAINST THIS KEY WORD.
GETKYWD  MVC   0(1,R7),0(R4)      MOVE KEYWORD OUT OF THE BUFFER.
GOODKYWD EQU   *
         MVC   0(4,R7),4(R7)      BLANK OUT THE OUTPUT AREA
         AR    R2,R0              POINT TO NEXT PDB ENTRY.
         CLC   0(2,R4),0(R6)      COMPARE WITH THE FMT KEYWORD ?
         BE    GETVALUE           SAME - GET KEYWORD VALUE.
         BCT   R1,FNDKYWD3        LOOP TILL PDB END REACHED.
         B     NEXTKYWD           SEARCH FOR NEXT KEYWORD.
```

```
*****************************************************************
*   KEYWORD VALUE IS MOVED TO THE OUTPUT AREA, IF THE LENGTH OF KEY- *
*   WORD VALUE IS BETWEEN 0-8 CHARACTERS. OTHERWISE ERRMSG 4 IS PRE- *
*   SENTED. FORMAT SWITCH IS SET TO VALUE 00 OR 01 DEPENDING ON THE  *
*   TYPE OF COMMAND ENTERED.                                        *
*        REG 7  - POINTS TO THE OUTPUT AREA                         *
*        REG 6  - POINTS TO APPROPRIATE SET OF FORMAT WORDS         *
*        REG 4  - POINTS TO THE KEYWORD VALUE IN THE BUFFER         *
*        REG 3  - LENGTH OF THE KEYWORD VALUE IN BUFFER             *
*****************************************************************


GETVALUE EQU   *
         L     R4,CWBBUF          ADDRESS OF THE BUFFER PASSED,
         AH    R4,PDBDISP         *** ADD DISPLACEMENT IN BUFFER.
         IC    R3,PDBLENG         LENGTH OF OPERAND IN BUFFER.
         LTR   R3,R3              LENGTH OF OPERAND = 0 ?
         BZ    ERROR4             YES - IT IS WRONG, PUT ERR-MSG.
         CH    R3,EIGHT           LENGTH OF OPERAND > 8 ?
         BH    ERROR4             YES - IT IS WRONG, PUT ERR-MSG.
         BCTR  R3,0               DECREASE IT BY 1 FOR EXECUTE.
         EX    R3,OUTOFBUF        MOVE A OPERAND IN OUTPUT AREA.
NEXTKYWD EQU   *
         LA    R6,2(R6)           POINT TO NEXT KEYWORD.
         LA    R7,8(R7)           AREA FOR NEXT KEYWORD VALUE.
         B     FNDKYWD2           GO FIND THIS NEW KEYWORD.
DONEFMT  EQU   *
         CLI   FMTSW,X'0.'        IS THIS SECOND PASS THRU HERE ?
         BE    ITISFMT1           YES - IT IS FORMAT 1 COMMAND.
         CLI   OUTAREA,C' '       CB-NAME SPECIFIED ?
         BNE   ITISFMT0           YES - IT IS FORMAT 0 COMMAND.
         MVI   FMTSW,X'01'        SET PASS THRU (FMT) SWITCH.
         LA    R6,FMT1KYWD        ADDRESS OF FORMAT1 KEYWORDS.
         B     FNDKYWD1           GO FIND KEYWORDS.
OUTOFBUF MVC   0(1,R7),0(R4)      MOVE VALUE OUT OF BUFFER.
```

```
*****************************************************************
*   VALUES FOR VARIOUS FORMAT 0 KEYWORDS IS CHECKED. IF NO VALUE IS  *
*   ENTERED THEN APPROPRIATE DEFAULT VALUE IS SUBSTITUTED.           *
*   FOR LENGTH    - DEFAULT VALUE IS FOUR BYTES                      *
*       TERM ID   - DEVICE FROM WHICH COMMAND IS ENTERED             *
*       OPER ID   - OPERATOR WHO ENTERED THE COMMAND                 *
*   IF TERM-ID IS ENTERED, THEN TVB CHAIN IS SEARCHED TO LOCATE TVB  *
*   FOR THAT TERM-ID. WHEN A TVB IS FOUND, BASE REGS FOR TVB AND TIB *
*   ARE UPDATED TO POINT TO CORRECT TVB AND TIB. IF NO TVB IS FOUND  *
*   FOR THE TERM-ID, THEN ERRMSG 5 IS PRESENTED.                     *
*        REG 2  - POINTS TO FIRST TVB IN TVB CHAIN                   *
*        REG 4  - POINTS TO OUTPUT AREA FOR DSILCS                   *
*        REG 15 - RETURN CODE FROM DSILCS (ZERO INDICATES SUCCESS)   *
*****************************************************************


ITISFMT0 EQU   *
         MVI   FMTSW,X'00'          RESET FMT SWITCH TO X'00'.
         CLI   OUTAREA+8,C' '       VALUE FOR LENGTH SPECIFIED ?
         BNE   CHECKLU              YES - CHECK IF LUNAME SPECIFIED.
         MVI   OUTAREA+8,C'4'       NO - INIT WITH DEFAULT LENGTH.
CHECKLU  EQU   *
         L     R2,MVTTVB            ADDRESS OF FIRST TVB OFF MVT.
         LA    R4,ADDRTVB           ADDR FOR OUTPUT FROM DSILCS.
         CLI   OUTAREA+16,C' '      VALUE FOR LUNAME SPECIFIED ?
         BE    CHECKOP              NO; CHECK IF OP-ID SPECIFIED.
         LA    R3,OUTAREA+16        ADDR OF 8-BYTE LUNAME AREA.
         DSILCS TVB=(R2),LU=(R3),CBADDR=(R4) ** LOCATE TVB FOR LUNAME.
         LTR   R15,R15              LOCATE OK (RET CODE = 0) ?
         BZ    FMT0ISOK             YES - FORMAT 0 IS OK.
         B     ERROR5               NO - GO PUT ERROR MESSAGE.
```

```
*****************************************************************
*   IF OPER-ID IS ENTERED THEN TVB CHAIN IS SEARCHED FOR OPER-ID.    *
*   THE ADDR RETURNED BY DSILCS IS USED TO UPDATE THE TVB AND TIB    *
*   BASE REGS (REG 10 AND 11). A ZERO TIB ADDRESS INDICATES THAT     *
*   THE TERMINAL/OPERATOR DOES NOT HAVE AN ACTIVE TIB. ERROR MESS-   *
*   AGE 8 IS PRESENTED, OTHERWISE CTL BLK NAME IS COMPARED AGAINST   *
*   A LIST OF CB-NAME. IF NO MATCH IS FOUND, ERRMSG 2 IS PRESENTED.  *
*        REG 2  - FIRST TVB IN TVB CHAIN                           *
*        REG 4  - ADDRESS OF OUTPUT AREA FOR DSILCS                *
*        REG 15 - RET CODE FROM DSILCS (ZERO INDICATES SUCCESS)    *
*****************************************************************

CHECKOP  EQU   *
         CLI   OUTAREA+24,C' '     VALUE FOR OP-ID SPECIFIED ?
         BE    FMTOISOK            NO - FORMAT 0 IS OK AS IT IS.
         LA    R3,OUTAREA+24       ADDR OF 8-BYTE OP-ID AREA.
         DSILCS TVB=(R2),OPID=(R3),CBADDR=(R4) ** LOCATE TVB FOR OPID.
         LTR   R15,R15             LOCATE OK (RET CODE = 0) ?
         BNZ   ERROR5              NO - GO PUT ERROR MESSAGE.
FMTOISOK EQU   *
         L     R10,ADDRTVB         GET ADDRESS OF TVB.
         L     R11,TVBTIB          ADDRESS OF TIB FROM TVB.
         LTR   R11,R11             DOES TIB EXIST FOR OPER/TERM ?
         BZ    ERROR8              NO - GO PUT ERROR MESSAGE.
         L     R3,OUTAREA          YES - GET CB-NAME VALUE IN REG.
         LA    R2,CBNAMEOK         BR TO ADDRESS IF CB-NAME IS GOOD.
         LA    R6,LISTOFCB         ADDRESS OF CB-NAME TABLE.
TESTNAME EQU   *
         LM    R4,R5,0(R6)         GET NAME AND BRANCH ADDRESS.
         LTR   R4,R4               END OF TABLE WITHOUT A MATCH ?
         BZ    ERROR2              YES - GO PUT ERROR MESSAGE.
         CR    R3,R4               NO. ARE TWO NAME SAME ?
         BER   R5                  YES - GET ADDRESS AND LENGTH OF CB.
         LA    R6,8(R6)            NO - CHECK NEXT ENTRY IN TABLE.
         B     TESTNAME            LOOP BACK TO TEST AGAIN.
```

```
*****************************************************************
*   DEPENDING ON THE CONTROL BLOCK NAME ENTERED, THE CONTROL BLOCK   *
*   ADDRESS IS FOUND EITHER FROM THE MVT OR THE TIB.                 *
*         REG 9  - POINTS TO THE MAIN VECTOR TABLE (MVT)             *
*         REG 11 - POINTS TO THE TASK INFORMATION BLOCK (TIB)        *
*         REG 4  - POINTS TO THE DESIRED CONTROL BLOCK               *
*****************************************************************

ITISMVT  EQU   *
         LR    R4,R9              IT IS MVT. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISSNT  EQU   *
         L     R4,MVTSNT          IT IS SNT. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISOIT  EQU   *
         L     R4,MVTOIT          IT IS OIT. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISART  EQU   *
         L     R4,MVTART          IT IS ART. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISDQT  EQU   *
         L     R4,MVTDQT          IT IS DQT. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISDDT  EQU   *
         L     R4,MVTDDT          IT IS DDT. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISSCT  EQU   *
         L     R4,MVTSCT          IT IS SCT. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISSVL  EQU   *
         L     R4,MVTSVL          IT IS SVL. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISTIB  EQU   *
         LR    R4,R11             IT IS TIB. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISTVB  EQU   *
         L     R4,TIBTVB          IT IS TVB. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
```

```
*****************************************************************
*   DEPENDING ON THE CONTROL BLOCK NAME ENTERED, THE CONTROL BLOCK   *
*   ADDRESS IS FOUND EITHER FROM THE MVT OR THE TIB.                 *
*         REG 9  - POINTS TO THE MAIN VECTOR TABLE (MVT)             *
*         REG 11 - POINTS TO THE TASK INFORMATION BLOCK (TIB)        *
*         REG 4  - POINTS TO THE DESIRED CONTROL BLOCK               *
*****************************************************************


ITISACB  EQU   *
         L     R4,TIBACB          IT IS ACB. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISNCWB EQU   *
         L     R4,TIBNCCWB        IT IS NCCWB. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISICWB EQU   *
         L     R4,TIBICCWB        IT IS ICCWB. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISMCWB EQU   *
         L     R4,TIBMRCWB        IT IS MRCWB. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISESWB EQU   *
         L     R4,TIBEXSWB        IT IS EXSWB. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISNSWB EQU   *
         L     R4,TIBNPSWB        IT IS NPSWB. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISNPDB EQU   *
         L     R4,TIBNCPDB        IT IS NCPDB. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISMPDB EQU   *
         L     R4,TIBMRPDB        IT IS MRPDB. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISIPDB EQU   *
         L     R4,TIBICPDB        IT IS ICPDB. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
ITISAPDB EQU   *
         L     R4,TIBAGPDB        IT IS AGPDB. LOAD ITS ADDRESS.
         BR    R2                 NOW GO GET ITS LENGTH.
```

```
*****************************************************************
*   DEPENDING ON THE CONTROL BLOCK NAME ENTERED, THE CONTROL BLOCK   *
*   ADDRESS IS FOUND EITHER FROM THE MVT OR THE TIB.                 *
*         REG 9  - POINTS TO THE MAIN VECTOR TABLE (MVT)             *
*         REG 11 - POINTS TO THE TASK INFORMATION BLOCK (TIB)        *
*         REG 4  - POINTS TO THE DESIRED CONTROL BLOCK               *
*****************************************************************

ITISRPL1 EQU   *
         BAL   R14,OSTTIBX          GO CHECK THE TYPE OF TIB.
         L     R4,TIOORRPL          YES - IT IS ORRPL. LOAD ITS ADDRESS.
         B     MOVETYPE             NOW GO GET ITS LENGTH.
ITISRPL2 EQU   *
         BAL   R14,OSTTIBX          GO CHECK THE TYPE OF TIB.
         L     R4,TIOOSRPL          IT IS OSRPL. LOAD ITS ADDRESS.
         B     MOVETYPE             NOW GO GET ITS LENGTH.
ITISRPL3 EQU   *
         BAL   R14,OSTTIBX          GO CHECK THE TYPE OF TIB.
         L     R4,TIORCRPL          IT IS RCRPL. LOAD ITS ADDRESS.
         B     MOVETYPE             NOW GO GET ITS LENGTH.
ITISRPL4 EQU   *
         BAL   R14,OSTTIBX          GO CHECK THE TYPE OF TIB.
         L     R4,TIOSCRPL          IT IS SCRPL. LOAD ITS ADDRESS.
         B     MOVETYPE             NOW GO GET ITS LENGTH.
ITISRPL5 EQU   *
         BAL   R14,OSTTIBX          GO CHECK THE TYPE OF TIB.
         L     R4,TIORARPL          IT IS RARPL. LOAD ITS ADDRESS.
         B     MOVETYPE             NOW GO GET ITS LENGTH.
ITISRPL6 EQU   *
         BAL   R14,OSTTIBX          GO CHECK THE TYPE OF TIB.
         L     R4,TIORSRPL          IT IS RSRPL. LOAD ITS ADDRESS.
         B     MOVETYPE             NOW GO GET ITS LENGTH.
ITISRPL7 EQU   *
         BAL   R14,HCTTIBX          GO CHECK THE TYPE OF TIB.
         L     R4,TIHSRPL           IT IS HSRPL. LOAD ITS ADDRESS.
         B     MOVETYPE             NOW GO GET ITS LENGTH.
ITISRPL8 EQU   *
         BAL   R14,HCTTIBX          GO CHECK THE TYPE OF TIB.
         L     R4,TIHCLRPL          IT IS HCLRPL. GET ITS ADDRESS.
         B     MOVETYPE             NOW GO GET ITS LENGTH.
ITISRPL9 EQU   *
         BAL   R14,PPTTIBX          GO CHECK THE TYPE OF TIB.
         L     R4,TIPRCRPL          IT IS PRCRPL. GET ITS ADDRESS.
         B     MOVETYPE             NOW GO GET ITS LENGTH.
ITISRPLA EQU   *
         BAL   R14,PPTTIBX          GO CHECK THE TYPE OF TIB.
         L     R4,TIPSCRPL          IT IS PSCRPL. GET ITS ADDRESS.
         B     MOVETYPE             NOW GO GET ITS LENGTH.
```

```
*****************************************************************
*   DEPENDING ON THE CONTROL BLOCK NAME ENTERED, THE CONTROL BLOCK   *
*   ADDRESS IS FOUND EITHER FROM THE MVT OR THE TIB.                 *
*         REG 9  - POINTS TO THE MAIN VECTOR TABLE (MVT)             *
*         REG 11 - POINTS TO THE TASK INFORMATION BLOCK (TIB)        *
*         REG 4  - POINTS TO THE DESIRED CONTROL BLOCK               *
*****************************************************************


ITISONIB EQU   *
         BAL   R14,OSTTIBX          GO CHECK THE TYPE OF TIB.
         L     R4,TIOOSNIB          IT IS OSNIB. LOAD ITS ADDRESS.
         B     MOVETYPE             NOW GO GET ITS LENGTH.
ITISCNIB EQU   *
         BAL   R14,OSTTIBX          GO CHECK THE TYPE OF TIB.
         L     R4,TIOCDNIB          IT IS CDNIB. LOAD ITS ADDRESS.
         B     MOVETYPE             NOW GO GET ITS LENGTH.
ITISHNIB EQU   *
         BAL   R14,HCTTIBX          GO CHECK THE TYPE OF TIB.
         L     R4,TIHNIB            IT IS HNIB. LOAD ITS ADDRESS.
         B     MOVETYPE             NOW GO GET ITS LENGTH.
ITISSAT  EQU   *
         BAL   R14,OSTTIBX          GO CHECK THE TYPE OF TIB.
         L     R4,TIOSAUTH          IT IS SAT. LOAD ITS ADDRESS.
         BR    R2                   NOW GO GET ITS LENGTH.
ITISNAT  EQU   *
         BAL   R14,OSTTIBX          GO CHECK THE TYPE OF TIB.
         L     R4,TIONAUTH          IT IS NAT. LOAD ITS ADDRESS.
         BR    R2                   NOW GO GET ITS LENGTH.

*****************************************************************
*   THE ADDRESS AND THE LENGTH OF THE DESIRED CONTROL BLOCK IS SAVED *
*   IF THE ADDRESS IS ZERO, ERRMSG 7 IS PRESENTED. OTHERWISE THE CTL *
*   BLOCK IS DISPALYED FOR THE APPROPRIATE LENGTH VALUE.             *
*         REG 4  - ADDRESS OF THE CONTROL BLOCK                      *
*         REG 5  - LENGTH OF THE CONTROL BLOCK                       *
*****************************************************************


MOVETYPE EQU   *
         MVI   CBTYPE,X'FF'         INDICATE CB IS RPL OR NIB.
CBNAMEOK EQU   *
         CLI   CBTYPE,X'FF'         CONTROL BLOCK A NIB OR RPL ?
         BE    LOADBYTE             YES - CB LENGTH FIELD IS 1 BYTE.
         LH    R5,2(R4)             NO - IT IS HALFWORD, GET IT.
         B     STOREIT              SAVE ADDRESS AND LENGTH.
LOADBYTE EQU   *
         SLR   R5,R5                ZERO REGISTER.
         IC    R5,3(R4)             GET LENGTH VALUE FOR RPL/NIB.
STOREIT  EQU   *
         STM   R4,R5,PARMSPSS       SAVE ADDRESS AND LENGTH OF CB.
         LTR   R4,R4                IS CTL BLK ADDRESS ZERO ?
         BZ    ERROR7               YES - GO PUT ERROR MESSAGE.
         B     DISPLYCB             GO DISPLAY CONTROL BLOCK.
```

```
*****************************************************************
*   THE HELP DIRECTIONS TO USE THIS COMMAND PROCESSOR ARE PRESENTED  *
*   ON THE TERMINAL. DIRECTIONS INCLUDE VARIOUS COMMAND FORMATS ETC. *
*         REG 2  - POINTS TO THE BUFFER CONTAINING HELP DIRECTIONS    *
*         REG 5  - ADDRESS OF THE HELP INFORMATION DIRECTIONS         *
*         REG 6  - VALUE TO CONTROL LOOP FOR DISPLAY OF DIRECTIONS    *
*****************************************************************

ITISHELP EQU   *                      IT IS HELP. DISPLAY DIRECTIONS.
         LA    R2,BUFFER              ADDRESS OF OUTPUT BUFFER.
         USING BUFHDR,R2              ADDRESSABILITY FOR BUF-HEADER.
         LA    R1,50                  LENGTH OF MSG-TEXT IN BUFFER.
         STH   R1,HDRMLENG            SAVE MSG-LENGTH IN BFR HEADER.
         L     R5,ADDRHELP            ADDRESS OF HELP DIRECTIONS.
         MVC   MSGAREA(50),0(R5)      MOVE A LINE OF HELP DIRECTION.
         BAL   R14,PUTBUFER           GO DISPLAY THE LINE.
         LA    R6,19                  VALUE TO CONTROL 'PUTAGAIN' LOOP.
PUTAGAIN EQU   *
         LA    R1,50                  LENGTH OF MESSAGE TEXT.
         STH   R1,HDRMLENG            SAVE IT IN 'MLENG' FIELD.
         LA    R5,50(R5)              ADDRESS OF NEXT HELP INSTRUCTION.
         MVC   MSGAREA(50),0(R5)      MOVE IT IN MSG AREA OF BUFFER.
         L     R1,CWBSWB              ADDRESS OF SWB FOR DSIPSS.
         DSIPSS SWB=(R1),TYPE=OUTPUT,BFR=(R2),OPTIONS=SEG
         BCT   R6,PUTAGAIN            CONTINUE UNTIL ALL INFORMATION IS
         B     RETURN                 ***DISPLAYED, AND THEN RETURN.
```

```
*****************************************************************
*   FOR FORMAT 1 COMMAND, THE VALUE ENTERED FOR ADDRESS AND LENGTH   *
*   IS TRANSLATED USING APPROPRIATE TRANSLATE TABLE. IF LENGTH IS    *
*   ENTERED IN DECIMAL THEN IT IS CONVERTED INTO HEX.                *
*****************************************************************


ITISFMT1 EQU   *
         CLI   OUTAREA,C' '         VALUE FOR ADDRESS SPECIFIED ?
         BE    ERROR1               NO - PUT ERROR MESSAGE.
         CLI   OUTAREA+8,C' '       VALUE FOR LENGTH SPECIFIED ?
         BNE   FMT1ISOK             YES - FORMAT 1 IS OK.
         MVI   OUTAREA+8,C'4'       NO - SET LENGTH TO DEFAULT VALUE.
FMT1ISOK EQU   *
         TR    OUTAREA(6),TRTABLE1 **X-LATE ADDRESS FOR SYNTAX CHECK.
         MVI   OUTAREA+6,C' '       MOVE C' ' AT THE END OF FIELD.
         MVI   PACKAREA,X'00'       CLEAR THE PACK AREA BY
         MVC   PACKAREA+1(7),PACKAREA   ** PROPOGATING X'00'.
         SLR   R6,R6                REG TO COUNT NUMBER OF DIGITS.
         LA    R7,OUTAREA           ADDRESS OF THE OUTPUT AREA.
         LA    R4,CBADDR            AREA TO SAVE START OF DISPLAY ADDR.
         BAL   R14,SYNTAXCK         GO CHECK THE SYNTAX.
DISPLYCB EQU   *                    FOUND CBNAME. PREPARE TO DISPLAY.
         CLI   FMTSW,X'00'          MAKE SURE IS IT FORMAT 0 ?
         BNE   NOTFMT0              NO - IT IS FORMAT 1.
         CLI   OUTAREA+8,C'*'       YES; LENGTH OF CB IS IMPLIED.
         BE    DISPDATA             YES - GO DISPLAY CONTROL BLOCK.
NOTFMT0  EQU   *                    NO - IT IS EXPLICIT.
         CLI   OUTAREA+8,C'X'       IS LENGTH SPECIFIED IN HEX ?
         BE    ITISHEX              YES - TRANSLATE IT ACCORDINGLY.
         TR    OUTAREA+8(6),TRTABLE2 ** NO; IT IS DECIMAL. X-LATE IT.
         MVC   PACKAREA,OUTAREA+8   TEMPORARY MOVE DATA TO AN AREA.
         MVI   OUTAREA+8,C'D'       PREFIX WITH 'D' TO INDICATE
         MVC   OUTAREA+9(6),PACKAREA ** DECIMAL LENGTH VALUE.
         B     TRDONE               AFTER X-LATE CHECK SYNTAX.
ITISHEX  EQU   *
         TR    OUTAREA+9(6),TRTABLE1 **TRANSLATE HEX LENGTH ALSO.
TRDONE   EQU   *
         MVI   OUTAREA+15,C' '      MOVE C' ' AT THE END OF OUTPUT AREA.
         MVI   PACKAREA,X'00'       CLEAR OUT THE PACK AREA.
         MVC   PACKAREA+1(7),PACKAREA   ***BY MOVING X'00'.
         SLR   R6,R6                REG TO COUNT NUMBER OF DIGITS.
         LA    R7,OUTAREA+9         ADDRESS OF OUTPUT AREA.
         LA    R4,CBLENTH           ADDRESS TO SAVE LENGTH VALUE.
         BAL   R14,SYNTAXCK         GO CHECK SYNTAX.
         CLI   OUTAREA+8,C'X'       LENGTH IS SPECIFIED IN HEX ?
         BE    DISPDATA             YES - START DISPLAY OF DATA.
         CVB   R6,PACKAREA          CONVERT LENGTH IN BINARY (HEX).
         ST    R6,CBLENTH           SAVE IT.
```

```
*****************************************************************
*   DATA IS DISPLAYED FROM THE ADDRESS ENTERED OR DETERMINED FOR THE *
*   DESIRED OR DEFAULT LENGTH VALUE.                            *
*         REG 3  - ADDRESS OF THE CURRENT DISPLAY BYTE          *
*         REG 2  - LENGTH WHICH REMAINED TO BE DISPLAYED        *
*****************************************************************

DISPDATA EQU   *                    DISPLAY THE DATA REQUESTED.
         MVI   FMTSW,X'00'          RESET FORMAT INDICATOR SWITCH.
         L     R2,CBLENTH           LENGTH OF DATA TO BE DISPLAYED.
         L     R3,CBADDR            ADDRESS WHERE DISPLAY IS TO START.
         ST    R3,DISPADDR          SAVE THE VALUE.
         MVI   SWITCH2,C'N'         INDICATES BFR-HDR NOT INITIALIZED.
LOOP010  EQU   *
         MVI   SWITCH1,C'N'         INDICATES NO DATA IN BUFFER.
         LTR   R2,R2                LENGTH IS = 0 ?
         BNP   MAYBDONE             YES - WE MAY BE DONE.
         MVI   MSGAREA,C' '         NO - PROPOGATE C' ' THROUGH
         MVC   MSGAREA+1(67),MSGAREA ***THE MESSAGE AREA.
         LA    R4,MSGAREA+8         AREA FOR TRANSLATED DATA.
         LA    R7,MSGAREA+50        AREA FOR CHARACTER FORM OF DATA.
         MVI   MSGAREA+49,C'+'      LEFT MARGIN INDICATOR.
         MVI   MSGAREA+66,C'+'      RIGHT MARGIN INDICATOR.
         SLR   R1,R1                ZERO OUT A REGISTER.
         LA    R6,4                 OUTER LOOP COUNT (LOOP020).
LOOP020  EQU   *
         LA    R5,4                 INNER LOOP COUNT (LOOP030).
         MVI   SWITCH1,C'Y'         INDICATES BUFFER HAS SOME DATA.
LOOP030  EQU   *
         SLR   R0,R0                ZERO ANOTHER REGISTER.
         IC    R1,0(R3)             GET A BYTE TO TRANSLATE.
         STC   R1,0(R7)             SAVE FOR CHARACTER FORM DISPLAY.
         D     R0,SIXTEEN           SPLIT THE TWO NIBBLES OF BYTE.
         STC   R1,0(R4)             SAVE LEFT NIBBLE.
         STC   R0,1(R4)             SAVE RIGHT NIBBLE.
         TR    0(2,R4),TRTABLE3     TRANSLATE THE BYTES JUST SAVED.
         LA    R3,1(R3)             ADDRESS OF NEXT DATA BYTE.
         LA    R7,1(R7)             ADDRESS OF NEXT CHAR FORM AREA.
         LA    R4,2(R4)             ADDRESS OF NEXT MSG-AREA BYTE.
         BCTR  R2,0                 DECREASE CB-LENGTH BY 1.
         LTR   R2,R2                CHECK IF CB-LENGTH = 0 ?
         BNP   MAYBDONE             YES - WE MAY BE DONE.
         BCT   R5,LOOP030           CONTINUE WITH INNER LOOP.
         LA    R4,2(R4)             LEAVE 2 BLANKS BEFORE NEXT DATA.
         BCT   R6,LOOP020           CONTINUE WITH OUTER LOOP.
```

```
*********************************************************************
*   DATA IS DISPLAYED FROM THE ADDRESS ENTERED OR DETERMINED FOR THE *
*   DESIRED OR DEFAULT LENGTH VALUE.                                 *
*         REG 3  - ADDRESS OF THE CURRENT DISPLAY BYTE               *
*         REG 2  - LENGTH WHICH REMAINED TO BE DISPLAYED             *
*********************************************************************
MAYBDONE EQU   *
         CLI   SWITCH1,C'N'        IS BUFFER EMPTY (WITH OUT DATA) ?
         BE    RETURN              YES - WE ARE DONE AND CAN RETURN.
         LA    R4,MSGAREA          NO - PREPARE TO X-LATE ADDRESS.
         LA    R5,DISPADDR+1       POINT TO WHERE ADDRESS IS SAVED.
         LA    R6,3                VALUE TO CONTROL NEXT LOOP.
LOOP040  EQU   *
         SLR   R0,R0               ZERO A REGISTER.
         IC    R1,0(R5)            GET A BYTE OF ADDRESS FOR X-LATION.
         D     R0,SIXTEEN          SPLIT TWO NIBBLES OF THE BYTE.
         STC   R1,0(R4)            SAVE THE LEFT HALF.
         STC   R0,1(R4)            SAVE THE OTHER HALF.
         TR    0(2,R4),TRTABLE3    TRANSLATE THE TWO SAVED BYTES.
         LA    R4,2(R4)            POINT TO NEXT BYTE  IN MSG-AREA.
         LA    R5,1(R5)            POINT TO NEXT BYTE OF ADDRESS.
         BCT   R6,LOOP040          CONTINUE WITH THIS LOOP.
         LA    R7,MSGAREA+50       ADDRESS OF CHARACTER FORM DATA.
         TR    0(16,R7),TRTABLE4   TRANSLATE TO REMOVE 3270 CC.
         ST    R3,DISPADDR         ADDR OF NEXT BYTE TO BE DISPLAYED.
         LR    R4,R2               TEMPORARILY SAVE REG 2.
         LA    R2,BUFFER           ADDRESS OF OUTPUT BUFFER.
         USING BUFHDR,R2           ADDRESSABILITY FOR BUF-HEADER.
         CLI   SWITCH2,C'Y'        BUFFER HEADER INITIALIZED ?
         BE    HDRDONE             YES.
         MVI   SWITCH2,C'Y'        INDICATE HEADER IS INITIALIZED.
         LA    R1,68               VALUE OF MESSAGE AREA LENGTH.
         STH   R1,HDRMLENG         SAVE IT IN 'MLENG' FIELD.
         BAL   R14,PUTBUFER        GO DISPLAY BUFFER.
         B     TSTRESET            CHECK IF RESET IS ON IN TVB.

*********************************************************************
*   DATA IS DISPLAYED FROM THE ADDRESS ENTERED OR DETERMINED FOR THE *
*   DESIRED OR DEFAULT LENGTH VALUE. IF TVBRESET BIT IS SET ON THEN  *
*   THE DISPLAY OF DATA IS STOPPED AND CONTROL IS RETURNED.          *
*         REG 3  - ADDRESS OF THE CURRENT DISPLAY BYTE               *
*         REG 2  - TEMPORARILY ALSO USED AS POINTER TO BUFFER        *
*         REG 2  - LENGTH WHICH REMAINED TO BE DISPLAYED             *
*********************************************************************

HDRDONE  EQU   *
         LA    R1,68               LENGTH OF MSG-AREA IN BUFFER.
         STH   R1,HDRMLENG         SAVE IT IN 'MLENG' FIELD.
         L     R1,CWBSWB           ADDRESS OF SWB PASSED.
         DSIPSS SWB=(R1),TYPE=OUTPUT,BFR=(R2),OPTIONS=SEG
TSTRESET EQU   *
         TM    TVBIND3,TVBRESET    RESET INDICTOR ON IN TVB ?
         BO    RETURN              YES - RETURN IMMEDIATELY.
         LR    R2,R4               RESTORE REG 2 BACK.
         B     LOOP010             GO DISPLAY MORE DATA.
```

```
*****************************************************************
*   ERRMSG 1 - INVALID COMMAND FORMAT IS USED                   *
*   ERRMSG 2 - CONTROL BLOCK NAME NOT IN THE LIST               *
*   ERRMSG 3 - NON-DECIMAL/NON-HEX DIGIT IS ENCOUNTERED.        *
*   ERRMSG 4 - KEYWORD VALUE LENGTH IS OUTSIDE ITS RANGE (0 - 8)*
*   ERRMSG 5 - NO TVB EXIST FOR TERM-ID/OPER-ID                 *
*   ERRMSG 6 - DESIRED CTL BLK CAN NOT BE FOUND IN THIS TIB     *
*   ERRMSG 7 - CONTROL BLOCK STARTS AT ADDRESS ZERO (00000000)  *
*   ERRMSG 8 - OPERATOR/TERMINAL DOES NOT HAVE AN ACTIVE TIB    *
*   ERRMSG 9 - RECOGNISED KEYWORDS ARE CB, LEN, TERM, OPER, ADDR*
*****************************************************************

ERROR1   EQU   *
         LA    R1,ERRMSG01          ADDRESS OF ERROR MESSAGE 1.
         B     PUTERROR             DISPLAY ERROR MSG, AND RETURN.
ERROR2   EQU   *
         LA    R1,ERRMSG02          ADDRESS OF ERROR MESSAGE 2.
         B     PUTERROR             DISPLAY ERROR MSG, AND RETURN.
ERROR3   EQU   *
         LA    R1,ERRMSG03          ADDRESS OF ERROR MESSAGE 3.
         B     PUTERROR             DISPLAY ERROR MSG, AND RETURN.
ERROR4   EQU   *
         LA    R1,ERRMSG04          ADDRESS OF ERROR MESSAGE 4.
         B     PUTERROR             DISPLAY ERROR MSG, AND RETURN.
ERROR5   EQU   *
         LA    R1,ERRMSG05          ADDRESS OF ERROR MESSAGE 5.
         B     PUTERROR             DISPLAY ERROR MSG, AND RETURN.
ERROR6   EQU   *
         LA    R1,ERRMSG06          ADDRESS OF ERROR MESSAGE 6.
         B     PUTERROR             DISPLAY ERROR MSG, AND RETURN.
ERROR7   EQU   *
         LA    R1,ERRMSG07          ADDRESS OF ERROR MESSAGE 7.
         B     PUTERROR             DISPLAY ERROR MSG, AND RETURN.
ERROR8   EQU   *
         LA    R1,ERRMSG08          ADDRESS OF ERROR MESSAGE 8.
         B     PUTERROR             DISPLAY ERROR MSG, AND RETURN.
ERROR9   EQU   *
         LA    R1,ERRMSG09          ADDRESS OF ERROR MESSAGE 9.
PUTERROR EQU   *
         MVC   MSGAREA(50),0(R1)    MOVE ERROR MESSAGE TEXT.
         LA    R2,BUFFER            ADDRESS OF OUTPUT BUFFER.
         USING BUFHDR,R2            ADDRESSABILITY FOR BUF-HEADER.
         LA    R1,50                LENGTH OF MESSAGE TEXT.
         STH   R1,HDRMLENG          SAVE IN 'MLENG' FIELD OF BUFHDR.
         BAL   R14,PUTBUFER         GO DISPLAY ERROR MESSAGE.
         MVI   RETCODE,X'08'        SET RETURN CODE VALUE.
```

```
*********************************************************************
*   RESTORE THE CALLERS REGISTER FROM HIS SAVE AREA AND RETURN TO    *
*   THE CALLER WITH APPROPRIATE RETURN CODE VALUE IN REG 15.         *
*         REG 14 - RETURN ADDRESS                                    *
*         REG 15 - RETURN CODE VALUE, EITHER 0 OR 8.                 *
*********************************************************************

RETURN   EQU   *
         SLR   R15,R15              ZERO RETURN CODE REGISTER.
         IC    R15,RETCODE          GET RETURN CODE VALUE.
         L     R13,4(R13)           ADDRESS OF HIS SAVE AREA.
         LM    R0,R12,20(R13)       RESTORE REGISTERS 0 - 12.
         L     R14,12(R13)          LOAD RETURN ADDRESS.
         BR    R14                  RETURN FOR GOOD.




*********************************************************************
*   TIB TYPE IS CHECKED FOR THE CONTROL BLOCK NAME ENTERED. IF IT IS *
*   THE RIGHT TIB, PROCESSING CONTINUES ELSE ERRMSG 6 IS PRESENTED.  *
*         REG 11 - BASE REG FOR TIB                                  *
*********************************************************************

OSTTIBX  EQU   *
         USING DSICBH,R11           ADDRESSABILITY FOR DSICBH.
         CLI   CBHTYPE,X'02'        CHECK IF OST-TIB EXTENSION ?
         BER   R14                  YES - RETURN AND CONTINUE.
         B     ERROR6               NO - GO PUT ERROR MESSAGE.
HCTTIBX  EQU   *
         CLI   CBHTYPE,X'03'        CHECK IF HCT-TIB EXTENSION ?
         BER   R14                  YES - RETURN AND CONTINUE.
         B     ERROR6               NO - GO PUT ERROR MESSAGE.
PPTTIBX  EQU   *
         CLI   CBHTYPE,X'00'        CHECK IF PPT-TIB EXTENSION ?
         BER   R14                  YES - RETURN AND CONTINUE.
         B     ERROR6               NO - GO PUT ERROR MESSAGE.
```

```
*********************************************************************
*   INPUT SYNTAX IS CHECKED FOR DECIMAL/HEX DIGITS AFTER TRANSLATION *
*   ERRMSG 3 IS PRESENTED IF A NON-HEX/NON-DECIMAL DIGIT IS FOUND.   *
*         REG 7  - POINTS TO AREA CONTAINING TRANSLATED INPUT         *
*********************************************************************

SYNTAXCK EQU   *
         ST    R14,SAVE14C          SAVE RETURN ADDRESS.
ONCEMORE EQU   *
         CLI   0(R7),X'40'          IS IT C' ' (END OF DATA) ?
         BE    EXECPACK             YES - GO PACK THE DATA.
         CLI   0(R7),X'00'          NO - IS IT X'00' (INVALID CHAR) ?
         BE    ERROR3               YES - THERE IS SYNTAX ERROR.
         LA    R6,1(R6)             NO - ADD 1 TO COUNT OF VALID CHAR.
         LA    R7,1(R7)             ADDRESS OF NEXT BYTE.
         B     ONCEMORE             GO CHECK THE BYTE.
EXECPACK EQU   *
         LTR   R6,R6                COUNT OF VALID CHAR = 0 ?
         BZ    ERROR4               YES - NULL OPERAND NOT ALLOWED.
         SR    R7,R6                NO. POINT TO START OF OUTPUT AREA.
         BCTR  R6,0                 REDUCE COUNT OF VALID CHAR BY ONE.
         EX    R6,GOPACKIT          GO PACK THE DATA.
         L     R6,PACKAREA+4        GET PACKED DATA IN REG.
         SRL   R6,4                 GET RID OF LOW ORDER 4 BITS.
         ST    R6,0(R4)             SAVE THE VALUE.
         BR    R14                  RETURN BACK.
GOPACKIT EQU   *
         PACK  PACKAREA+4(4),0(0,R7)  ***PACK THE DATA.

*********************************************************************
*   DISPLAY THE DATA IN THE BUFFER AFTER INITIALIZING VARIOUS FIELDS *
*   IN THE BUFHDR, SUCH AS DOMID, TSTMP, BLENG, TDISP ETC.          *
*         REG 2  - POINTS TO THE BUFFER                             *
*         REG 1  - POINTS TO THE SWB FOR CALL TO DSIPSS             *
*********************************************************************

PUTBUFER EQU   *
         ST    R14,SAVE14A          SAVE RETURN ADDRESS.
         CLI   HDRTDISP+1,X'00'     'TDISP', 'DOMID' ETC DONE ?
         BNE   HALFDONE             YES - NEED ONLY 'BLENG' AND 'TSTMP'.
         LA    R0,BUFHDRND-BUFHDR   DISPLACEMENT OF MSG FROM BFR HDR.
         STH   R0,HDRTDISP          SAVE IT IN 'TDISP' FIELD.
         MVC   HDRDOMID(8),MVTCURAN **MOVE DOMAIN-ID INFORMATION.
         MVI   HDRMTYPE,C'U'        INDICATE USER TYPE MESSAGE.
         LA    R0,120               LENGTH OF TOTAL BUFFER.
         STH   R0,HDRBLENG          SAVE IT IN 'BLENG' FIELD.
HALFDONE EQU   *
         BAL   R14,GETTIME          ISSUE SVC 34 FOR TIME (?HHMMSS+).
         ST    R1,HDRTSTMP          SAVE IT IN 'TSTMP' FIELD.
         L     R1,CWBSWB            GET ADDRESS OF SWB.
         DSIPSS SWB=(R1),TYPE=OUTPUT,BFR=(R2)
         L     R14,SAVE14A          GET THE RETURN ADDRESS.
         BR    R14                  RETURN BACK.
```

```
*********************************************************************
*    ISSUE DSIDATIM TO GET TIME THE TIME OF DAY                     *
*     OUTPUT: REG 1 - TIME IN HHMMSSOC FORM                         *
*********************************************************************


GETTIME  EQU   *
         ST    R14,SAVE14B          SAVE RETURN ADDRESS.
         DSIDATIM AREA=PACKAREA,FORMAT=BINARY  GET THE TIME OF DAY
         L     R1,PACKAREA+4        RETURN ONLY THE TIME
         L     R14,SAVE14B          LOAD RETURN ADDRESS.
         BR    R14                  AND RETURN.


LISTOFCB DS    0F                   LIST OF CTL BLK NAME AND BR ADDRESS
         DC    CL4'ACB '            IF NAME IS 'ACB ', THEN
         DC    A(ITISACB)           ***BRANCH TO LABEL 'ITISACB'.
         DC    CL4'ART '            IF NAME IS 'ART ', THEN
         DC    A(ITISART)           ***BRANCH TO LABEL 'ITISART'.
         DC    CL4'AGPD'            IF NAME IS 'AGPDB', THEN
         DC    A(ITISAPDB)          ***BRANCH TO LABEL 'ITISAPDB'.
         DC    CL4'CDNI'            IF NAME IS 'CDNIB', THEN
         DC    A(ITISCNIB)          ***BRANCH TO LABEL 'ITISCNIB'.
         DC    CL4'DDT '            IF NAME IS 'DDT ', THEN
         DC    A(ITISDDT)           ***BRANCH TO LABEL 'ITISDDT'.
         DC    CL4'DQT '            IF NAME IS 'DQT ', THEN
         DC    A(ITISDQT)           ***BRANCH TO LABEL 'ITISDQT'.
         DC    CL4'EXSW'            IF NAME IS 'EXSWB', THEN
         DC    A(ITISESWB)          ***BRANCH TO LABEL 'ITIESWB'.
         DC    CL4'ICCW'            IF NAME IS 'ICCWB', THEN
         DC    A(ITISICWB)          ***BRANCH TO LABEL 'ITISICWB'.
         DC    CL4'ICPD'            IF NAME IS 'ICPDB', THEN
         DC    A(ITISIPDB)          ***BRANCH TO LABEL 'ITISIPDB'.
         DC    CL4'MRCW'            IF NAME IS 'MRCWB', THEN
         DC    A(ITISMCWB)          ***BRANCH TO LABEL 'ITISMCWB'.
         DC    CL4'MRPD'            IF NAME IS 'MRPDB', THEN
         DC    A(ITISMPDB)          ***BRANCH TO LABEL 'ITISMPDB'.
         DC    CL4'MVT '            IF NAME IS 'MVT ', THEN
         DC    A(ITISMVT)           ***BRANCH TO LABEL 'ITISMVT'.
         DC    CL4'NAT '            IF NAME IS 'NAT ', THEN
         DC    A(ITISNAT)           ***BRANCH TO LABEL 'ITISNAT'.
         DC    CL4'NCCW'            IF NAME IS 'NCCWB', THEN
         DC    A(ITISNCWB)          ***BRANCH TO LABEL 'ITISNCWB'.
         DC    CL4'NCPD'            IF NAME IS 'NCPDB', THEN
         DC    A(ITISNPDB)          ***BRANCH TO LABEL 'ITISNPDB'.
         DC    CL4'NPSW'            IF NAME IS 'NPSWB', THEN
         DC    A(ITISNSWB)          ***BRANCH TO LABEL 'ITISNSWB'.
         DC    CL4'OIT '            IF NAME IS 'OIT ', THEN
         DC    A(ITISOIT)           ***BRANCH TO LABEL 'ITISOIT'.
         DC    CL4'ORRP'            IF NAME IS 'ORRPL', THEN
         DC    A(ITISRPL1)          ***BRANCH TO LABEL 'ITISRPL1'.
```

```
DC    CL4'OSNI'          IF NAME IS 'OSNIB', THEN
DC    A(ITISONIB)        ***BRANCH TO LABEL 'ITISONIB'.
DC    CL4'OSRP'          IF NAME IS 'OSRPL', THEN
DC    A(ITISRPL2)        ***BRANCH TO LABEL 'ITISRPL2'.
DC    CL4'RARP'          IF NAME IS 'RARPL', THEN
DC    A(ITISRPL5)        ***BRANCH TO LABEL 'ITISRPL5'.
DC    CL4'RCRP'          IF NAME IS 'RCRPL', THEN
DC    A(ITISRPL3)        ***BRANCH TO LABEL 'ITISRPL3'.
DC    CL4'RSRP'          IF NAME IS 'RSRPL', THEN
DC    A(ITISRPL6)        ***BRANCH TO LABEL 'ITISRPL7'.
DC    CL4'SAT '          IF NAME IS 'SAT ', THEN
DC    A(ITISSAT)         ***BRANCH TO LABEL 'ITISSAT'.
DC    CL4'SCRP'          IF NAME IS 'SCRPL', THEN
DC    A(ITISRPL4)        ***BRANCH TO LABEL 'ITISRPL4'.
DC    CL4'SCT '          IF NAME IS 'SCT ', THEN
DC    A(ITISSCT)         ***BRANCH TO LABEL 'ITISSCT'.
DC    CL4'SNT '          IF NAME IS 'SNT ', THEN
DC    A(ITISSNT)         ***BRANCH TO LABEL 'ITISSNT'.
DC    CL4'SVL '          IF NAME IS 'SVL ', THEN
DC    A(ITISSVL)         ***BRANCH TO LABEL 'ITISSVL'.
DC    CL4'TIB '          IF NAME IS 'TIB ', THEN
DC    A(ITISTIB)         ***BRANCH TO LABEL 'ITISTIB'.
DC    CL4'TVB '          IF NAME IS 'TVB ', THEN
DC    A(ITISTVB)         ***BRANCH TO LABEL 'ITISTVB'.
DC    CL4'HSRP'          IF NAME IS 'HSRPL', THEN
DC    A(ITISRPL7)        ***BRANCH TO LABEL 'ITISRPL7'.
DC    CL4'HCLR'          IF NAME IS 'HCLRPL', THEN
DC    A(ITISRPL8)        ***BRANCH TO LABEL 'ITISRPL8'.
DC    CL4'PRCR'          IF NAME IS 'PRCRPL', THEN
DC    A(ITISRPL9)        ***BRANCH TO LABEL 'ITISRPL9'.
DC    CL4'PSCR'          IF NAME IS 'PSCRPL', THEN
DC    A(ITISRPLA)        ***BRANCH TO LABEL 'ITISRPLA'.
DC    CL4'HNIB'          IF NAME IS 'HNIB', THEN
DC    A(ITISHNIB)        ***BRANCH TO LABEL 'ITISHNIB'.
DC    CL4'HELP'          IF NAME IS 'HELP', THEN
DC    A(ITISHELP)        ***BRANCH TO LABEL 'ITISHELP'.
DC    XL4'00000000'      END OF TABLE.
```

```
ADDRHELP DC    A(HELPINFO)          ADDRESS OF HELP DIRECTIONS
THREE00  DC    F'300'               FULLWORD VALUE 300.
SIXTY    DC    F'60'                FULLWORD VALUE 60.
SIXTEEN  DC    F'16'                FULLWORD VALUE 16.
EIGHT    DC    H'8'                 HALFWORD VALUE 8.
FOUR     DC    H'4'                 HALFWORD VALUE 4.
FMTOKYWD DC    C'CBLETEOP           FMT 0 KEYWORDS (CB, LEN, TERM, OP)
FMT1KYWD DC    C'ADLE  '            FORMAT 1 KEYWORDS (ADDR, LEN)
KYWRDTAB DC    C'CB  '              LIST OF RECOGNISED KEYWORD 'CB'
         DC    C'LEN '              **** RECOGNISED KEYWORD 'LEN'
         DC    C'ADDR'              **** RECOGNISED KEYWORD 'ADDR'
         DC    C'TERM'              **** RECOGNISED KEYWORD 'TERM'
         DC    C'OPER'              **** RECOGNISED KEYWORD 'OPER'
         DC    C'  '                END OF THE LIST



TRTABLE1 DC    64X'00',C' ',64X'00',X'FAFBFCFDFEFF',58X'00',X'FAFB'
         DC    X'FCFDFEFF',41X'00',X'F0F1F2F3F4F5F6F7F8F9',6X'00'


TRTABLE2 DC    64X'00',C' ',175X'00',X'F0F1F2F3F4F5F6F7F8F9',6X'00'


TRTABLE3 DC    CL16'0123456789ABCDEF'


TRTABLE4 DC    64X'4B',C' ',9X'4B',X'4A4B4C4D4E4F50',9X'4B',X'5A5B5C'
         DC    X'5D5E5F6061',9X'4B',X'6B6C6D6E6F',10X'4B',X'7A7B7C7D'
         DC    X'7E7F',65X'4B',C'ABCDEFGHI',7X'4B',C'JKLMNOPQR',8X'4B'
         DC    C'STUVWXYZ',6X'4B',C'0123456789',6X'4B'


ERRMSG01 DC    CL50'INVALID COMMAND FORMAT IS USED. TRY *SHOW CB=HELP*'
ERRMSG02 DC    CL50'CTL BLOCK NAME NOT IN THE LIST. TRY *SHOW CB=HELP*'
ERRMSG03 DC    CL50'NON-DECIMAL/NON-HEX DIGIT IS FOUND. PLEASE CORRECT'
ERRMSG04 DC    CL50'KEYWORD/OPERAND WITH =0 OR >8 CHARS IS NOT ALLOWED'
ERRMSG05 DC    CL50'NO TVB EXISTS FOR THE SPECIFIED LU-NAME OR OPER-ID'
ERRMSG06 DC    CL50'THE DESIRED NCCF CONTROL BLOCK IS NOT IN THIS TIB '
ERRMSG07 DC    CL50'CB STARTING ADDRESS IS 0. MAY BE IT DOES NOT EXIST'
ERRMSG08 DC    CL50'THE OPERATOR/TERMINAL DOES NOT HAVE AN ACTIVE TIB '
ERRMSG09 DC    CL50'RECOGNISED KEYWORDS ARE: CB, LEN, TERM, OPER, ADDR'
```

```
HELPINFO DC    CL50'FOR THE DISPLAY OF ANY NCCF CONTROL BLOCK, ENTER--
         DC    CL50' SHOW CB=XXXX,LEN=<VALUE>,TERM=<ID-1>,OPER=<ID-2>
         DC    CL50' WHERE LEN, TERM AND OPER ARE OPTIONAL PARAMETERS.
         DC    CL50' LENGTH CAN BE CODED IN HEX (PREFIXING BY X)/DECI
         DC    CL50' OR AS AN *, IN WHICH CASE IT IS THE IMPLICIT LEN
         DC    CL50' OF THE NCCF CONTROL BLOCK. IF BOTH TERM AND OPER
         DC    CL50' ARE CODED THEN THE OPER-ID IS IGNORED. FOR THE CB
         DC    CL50' CODE ONE OF THE FOLLOWING NAMES (ALL TASKS):
         DC    CL50'ACB  ART    AGPDB  ICPDB  MRPDB  NCPDB  DDT    DQT
         DC    CL50'OIT  EXSWB  NPSWB  ICCWB  MRCWB  NCCWB  TVB    SAT
         DC    CL50'MVT  NAT    SCT    SNT    SVL    TIB    HELP
         DC    CL50' FOR OST-TASKS ADD ----   ORRPL  OSRPL  RARPL
         DC    CL50'             RCRPL  RSRPL  SCRPL  CDNIB  OSNIB
         DC    CL50' FOR HCT-TASKS ADD ----   HNIB   HSRPL  HCLRPL
         DC    CL50' FOR PPT-TASKS ADD ----   PRCRPL     PSCRPL
         DC    CL50'FOR DISPLAY OF DATA IN USERS ADDRESS SPACE ENTER--
         DC    CL50'      SHOW ADDR=<VALUE1>,LEN=<VALUE2>
         DC    CL50'WHERE ADDRESS IS THE STARTING ADDRESS IN HEX, AND
         DC    CL50'LENGTH CAN BE CODED IN HEX (PREFIXING BY X) OR IN
         DC    CL50'DECIMAL. TO STOP DISPLAY HIT RESET KEY.
```

```
         DSICWB                        COMMAND WORK BLOCK DSECT.


DSICWB   DSECT                         CONTINUE DEFINING CWB.
         ORG   CWBADATD                REDEFINE DATA AREA IN CWB.
WORKAREA DS    0CL256                  WORK AREA FOR COMMAND PROCESSOR
OUTAREA  DS    0CL32                   OUTPUT AREA FOR OPERANDS.
         DS    CL8                     AREA FOR 1ST KEYWD CB/ADDR VALUE.
         DS    CL8                     AREA FOR 2ND KEYWD (LENGTH) VALUE.
         DS    CL8                     AREA FOR 3RD KEYWD (LUNAME) VALUE.
         DS    CL8                     AREA FOR 4TH KEYWD (OPID) VALUE.
PACKAREA DS    D                       DOUBLE WORD AREA TO PACK/UNPACK.
PARMADDR DS    A                       SAVE AREA FOR PARM-LIST ADDRESS.
ADDRTVB  DS    A                       SAVE AREA FOR LOCATE TVB ADDRESS.
SAVE14A  DS    F                       SAVE AREA FOR LINKAGE REGISTER.
SAVE14B  DS    F                       SAVE AREA FOR LINKAGE REGISTER.
SAVE14C  DS    F                       SAVE AREA FOR LINKAGE REGISTER.
DISPADDR DS    F                       ADDRESS OF CURRENT DISPALY BYTE.
FMTSW    DS    0C                      FORMAT INDICATOR BYTE.
RETCODE  DS    C                       RETURN CODE VALUE.
CBTYPE   DS    C                       INDICATOR IF CB-TYPE IS RPL/NIB.
SWITCH1  DS    C                       SWITCH FOR EMPTY/FULL BUFFER
SWITCH2  DS    C                       SWITCH FOR BUF-HEADER INITIALIZATION
PARMSPSS DS    0CL8                    PARMS USED TO CALL DSIPSS
CBADDR   DS    A                       ADDRESS OF CONTROL BLOCK
CBLENTH  DS    F                       LENGTH OF CONTROL BLOCK
         DS    F                       ADDITIONAL FULL WORD AREA.
BUFFER   DS    0F                      COMPLETE BUFFER AREA
         DS    XL(BUFHDRND-BUFHDR)     BUFFER HEADER AREA.
MSGAREA  DS    0CL96                   MESSAGE AREA IN THIS BUFFER
LABEL1   EQU   *                       LABEL TO COMPUTE LENGTH.
         DS    XL(256-(LABEL1-WORKAREA)) REMAINDER OF THE WORKAREA.
```

Appendix E. Sample Data Services Command Processors

This appendix contains examples of data services command processors (DSCPs). DSITDSRD uses the DSIZVSMS macro instruction and reads keyed records from a VSAM data set. DSITDSOL uses the DSIZCSMS macro instruction to send a request for data to the CNM interface.

Note: These command processors are not executable as shown in this appendix.

DSITDSRD Command Processor

```
*****************************************************************************
*                                                                           *
*  NAME = DSITDSRD                                                          *
*                                                                           *
*  FUNCTION = THIS COMMAND PROCESSOR WILL READ KEYED RECORDS FROM           *
*             A VSAM DATA SET AS PER THE CNMREAD COMMAND PASSED.            *
*                                                                           *
*  ENTRY POINT = DSITDSRD                                                   *
*                                                                           *
*  INPUT =                                                                  *
*      REGISTERS:                                                           *
*           R1  = DSICWB ADDRESS                                            *
*           R13 = ADDRESS OF STANDARD SAVEAREA                              *
*           R14 = RETURN ADDRESS                                            *
*           R15 = ENTRY POINT OF ROUTINE                                    *
*                                                                           *
*      DSICWB (POINTED TO BY R1 - PSCP AND DSCP):                           *
*           CWBSAVEA = AN 18 WORD SAVE AREA FOR USE IN THIS CMD PROC.       *
*           CWBBUF = ADDRESS OF A MESSAGE BUFFER. THE BUFFER                *
*                    CONTAINS A STANDARD NCCF BUFFER HEADER.                *
*           CWBPDB = ADDRESS OF A PDB FOR USE IN THIS CMD PROC.             *
*                    NOTE - THE PDB WILL CONTAIN VALID INFORMATION          *
*                    ONLY WHEN CONTROL IS RECEIVED INITIALLY FROM           *
*                    THE TASK (OST OR DST) DRIVING THIS CMD PROC.           *
*           CWBSWB = ADDRESS OF A SWB FOR USE IN THIS CMD PROC.             *
*           CWBTIB = ADDRESS OF THE DSIOST OR DSIZDST TIB.                  *
*           CWBADATD = A 256 BYTE WORK AREA FOR USE IN THIS CMD PROC.       *
*           CWBDSRB = FOR A DSCP ADDRESS OF A DSRB CONTAINING               *
*                     INFORMATION RELATED TO PROCESSING OF THE              *
*                     COMMAND. FOR A PSCP THIS FIELD IS MEANINGLESS.        *
*                                                                           *
*      DSIDSRB (POINTED TO BY CWBDSRB - DSCP ONLY):                         *
*                                                                           *
*                                                                           *
*           DSRBUSER = THE ADDRESS OF THE READ GLOBAL WORK AREA(RGWA)       *
*                      WILL BE PLACED HERE AFTER IT IS OBTAINED.THE         *
*                      HIGH ORDER BYTE WILL BE USED AS A FLAG TO            *
*                      DETERMINE IF THE RGWA SHOULD BE FREE'D UP            *
*                      UPON COMPLETION.                                     *
*                                                                           *
```

```
*  OUTPUT =                                                          *
*      REGISTERS:                                                    *
*           R0 - R14 = RESTORED TO INPUT STATUS                      *
*           R15 = RETURN CODE                                        *
*                                                                    *
*      RETURN CODE (R15 ON EXIT):                                    *
*                                                                    *
*         FOR A DSCP:                                                *
*            0  = MAINTAIN THE DSRB PASSED TO THIS CMD PROC          *
*                 RESUME PROCESSING HAS TO BE SCHEDULED.             *
*            8  = RELEASE THE DSRB PASSED TO THIS CMD PROC -         *
*                 THE FUNCTION IS COMPLETE.                          *
*                                                                    *
*  DESCRIPTION = DSITDSRD IS A DATA SERVICES COMMAND PROCESSOR(DSCP). *
*                ITS FUNCTION IS TO READ THE NUMBER OF RECORDS       *
*                SPECIFIED IN THE CNMREAD COMMAND AREA STARTING      *
*                WITH THE INITIAL KEY SPECIFIED.OUTPUT RECORDS ARE   *
*                BUILT FOR THE FIRST AND LAST RECORDS READ PLUS      *
*                EACH READ ERROR.UPON COMPLETION OF READING THE      *
*                NUMBER OF RECORDS SPECIFIED A FINAL SUMMARY RECORD IS *
*                BUILT.EACH RECORD(MESSAGE) BUILT IS THEN QUEUED BACK *
*                TO THE INVOKING PSCP VIA THE NCCF MACRO DSIMQS.     *
*                A SCHEDULING ERROR WILL RESULT IN A MESSAGE BEING   *
*                SENT DIRECTLY TO THE USER AND THE TERMINATION OF    *
*                COMMAND PROCESSING.                                 *
*                                                                    *
*  EXIT = RETURN TO ADDRESS IN R14, WITH RETURN CODE SET IN R15.     *
*                                                                    *
*  EXTERNAL REFERENCES = NONE                                        *
*                                                                    *
*  CONTROL BLOCKS =                                                  *
*            DSICWB - COMMAND-PROCESSOR WORK BLOCK                   *
*            DSIDSRB - DATA SERVICES REQUEST BLOCK (DSCP)            *
*            DSIPDB - PARSE DATA BLOCK                               *
*            DSISWB - SERVICE-ROUTINE WORK BLOCK                     *
*            DSITIB - TASK INFORMATION BLOCK                         *
*            DSITID - DSIZDST TASK INFORMATION BLOCK EXTENSION (DSCP) *
*            DSIMVT - NCCF MAIN VECTOR TABLE                         *
*                                                                    *
*  MACROS = DSICBS,DSIGET,DSIFRE,DSIMQS,DSIABN,DSIZVSMS,DSIMBS,DSIPSS *
*                                                                    *
*  ABENDS:                                                           *
*        651 = FAILURE OF MQS TO PUT OUT MSG 261                     *
*        652 = FAILURE TO FREE THE RGWA                              *
*        653 = FAILURE TO QUEUE READ ERROR MSG                       *
*        654 = FAILURE TO QUEUE LAST RECORD MSG                      *
*        655 = FAILURE TO QUEUE FIRST RECORD                         *
*        656 = FAILURE TO QUEUE SUMMARY RECORD                       *
*        657 = FAILURE OF MQS TO PUT OUT MSG 256                     *
*                                                                    *
**********************************************************************
```

```
         MACRO
&LABEL   HEXEBC  &TO=,&FROM=
&LABEL   UNPK    &TO.(7),&FROM.(4)
         PACK    &TO+7(1),&TO+6(1)
         OI      &TO+6,X'F0'
         OI      &TO+7,X'F0'
         TR      &TO.(8),HEXTAB-240
         MEND
*
         MACRO
&LABEL   UPONE &ADDR=,&DIGITS=4
&LABEL   LA     15,&ADDR
         LA     14,&DIGITS
         LA     15,0(14,15)
         BCTR   15,0
UP&SYSNDX TR     0(1,15),UPTAB-240
         CLI    0(15),C'0'
         BNE    UPND&SYSNDX
         BCTR   15,0
         BCT    14,UP&SYSNDX
UPND&SYSNDX EQU  *
         MEND
DSITDSRD CSECT
         USING *,15
         B      INSAVE
         DC     C'DSITDSRD
         DC     C'&SYSDATE'
INSAVE   EQU    *
         DROP   15
*************************************************************************
*  MODULE ADDRESSABILITY IS ESTABLISHED, ADDRESSABILITY TO THE CWB IS *
* ESTABLISHED, AND THE CWB SAVEAREA IS USED IN PERFORMING STANDARD    *
* ENTRY LINKAGE.                                                      *
*************************************************************************
         STM    14,12,12(13)        SAVE CALLERS REGS
         LR     12,15               ESTABLISH MODULE
         USING DSITDSRD,12           ADDRESSABILITY
         USING DSICWB,1              CMD-PROC WORK BLOCK ADDRESSABILITY
         LA     2,CWBSAVEA          PICK UP ADDRESS OF MY SAVE AREA
         ST     13,4(2)             SET BACK POINTER
         XC     8(4,2),8(2)         CLEAR FORWARD POINTER
         ST     2,8(13)             SET CALLERS FWD POINTER
         LR     13,2                R13 NOW POINTS TO MY SAVE AREA
         L      6,CWBBUF            SET REG 6 TO ADDRESS OF I/P BUFFER
         L      10,CWBTIB                START SET UP OF MVT ADDRESS
         USING DSITIB,10
         L      11,TIBTVB
         ST     11,0(,13)      PUT MY TVB ADDR IN 1ST WORD OF SAVEAREA
         USING DSITVB,11
         DROP   10
         L      8,TVBMVT
         DROP   11
         USING DSIMVT,8                  FINIALLY GOT THE MVT
         LR     11,1                     ALSO NEED THE CWB IN 11
         DROP   1
```

```
         USING DSICWB,11
         L     2,CWBDSRB                 GET THE ADD OF THE DSRB
         USING DSIDSRB,2
         LA    4,RGWALEN                 SET RGWA LENGTH IN REG 4
*****************************************************************
*         BUILD AN OUTPUT BUFFER IN THE CWBADATD AREA            *
*       WITH A TEXT AREA LENGTH EQUAL TO THE RDCNMCMD DSECT WHICH *
*       IS THE READ RESPONSE AREA QUEUED BACK TO THE PSCP.       *
*****************************************************************
         LA    7,CWBADATD                GET THE • OF THE ADATD AREA
         USING BUFHDR,7                  SET REG 7 AS BASE FOR BUFHDR
         LA    10,HDRTEXT                GET ADDR OF TEXT PORTION
         USING RDCNMCMD,10               SET REG 10 AS BASE FOR
*     RESPONSE AREA FOR QUEUEING BACK TO THE PSCP
         MVC   CNMVERB(RDCNMEND-CNMVERB),BLANKS  BLANK RESP AREA
         MVC   CNMIFR(2),IFRCDE          SET IFR CODE IN RESP AREA
         MVC   CNMVERB(8),READRSP        MOVE COMMAND NAME
         DROP  10                        DROP RESP AREA BASE
         SR    10,7                      TEXT - START = HEADER LENGTH
         STH   10,HDRTDISP               STORE DISPL. TO TEXT IN HDR
         LA    10,256                   GEN ACTUAL BUFFER SIZE
         STH   10,HDRBLENG               STORE TOTAL BUFLEN IN HDR
         LA    10,RDCNMLEN               GET MSG LENGTH
         STH   10,HDRMLENG               SET HEADER MSG LENGTH
         MVI   HDRMTYPE,HDRTYPEI         SET MSGTYPE
         CLI   DSRBFNCD,DSRBFNRM         IS THIS 1ST TIME ENTERED?
         BNE   NEXTRD                    IF NOT GO TO READ NEXT RECORD
*****************************************************************
*                                                               *
*             CODE FOR 1ST TIME ENTERED FOLLOWS.                *
*                                                               *
*    1) ISSUE DSIGET FOR A READ GLOBAL WORK AREA(RGWA).         *
*    2) STORE ITS ADDRESS IN THE DSRB.                          *
*    3) INITIALIZE THE RGWA AND THE VSAM BUFFER HEADER.         *
*    4) SCHEDULE A READ FOR THE FIRST KEY REQUESTED.            *
*                                                               *
*****************************************************************
         LA    3,DSRBUSER                SET ADD OF WHERE STOR ADD GOES
         LA    5,CWBADATD+240            GET ADD OF WORK AREA FOR LIST
         DSIGET LV=(4),A=(3),BNDRY=DBLWD,LISTA=(5),REENT=YES
         LTR   15,15                     TEST RETURN CODE
         BZ    INITRGWA                  IF GOOD GO SET UP RGWA
*****************************************************************
*         COULD NOT GET STORAGE FOR RGWA THEREFORE ISSUE        *
*       INSUFFICIENT STORAGE MESSAGE,SET 'FREE DSRB' RETURN CODE *
*       AND RETURN TO DST.                                      *
*****************************************************************
         MVI   HDRMTYPE,HDRTYPEU         SET MSGTYPE FOR OUTPUT
         L     10,MVTUFLD               GEN ADDR WHERE MSG CSECT
         LA    10,8(,10)                 ADDR IS STORED
         L     10,0(,10)                GET MSG CSECT ADDR
         DSIMBS MID=261,SWB=CWBSWB,BFR=(7),MSGTBL=(10)
         C     15,RTRNCD12   WAS ERR MSG BUILT O.K.
         BNH   MQSMSG1       YES, GO SEND MSG
         BAL   14,SENDERR    NO, GO BUILD INTERNAL MSG & SEND
```

```
         LTR   14,15          WAS INTERNAL MSG SENT O.K.
         LA    15,8           INDICATE FREE DSRB
         BZ    SRDEXIT        GO EXIT
         DC    F'0'           MQS ERROR, HALT EXECUTION
MQSMSG1  EQU   *
         DSIMQS SWB=CWBSWB,BFR=(7),TASKID=DSRBOID
         LTR   15,15                  TEST MQS RETURN CODE
         BNZ   ABEND651
         LA    15,8                   INDICATE FREE THE DSRB
         B     SRDEXIT                GO TO RETURN LOGIC
ABEND651 DC    F'0'     MQS OUTPUT FAILED, HALT EXECUTION
*****************************************************************
*         GET OF RGWA SUCCESSFUL.................               *
*       SET UP VSAM BUFFER,INITIALIZE RGWA AND SCHEDULE READ FOR  *
*       1ST KEY REQUESTED.                                        *
*****************************************************************
INITRGWA EQU   *
         L     3,DSRBUSER             GET ADDRESS OF THE RGWA
         USING RGWA,3                 SET REG 3 AS BASE FOR RGWA
*****************************************************************
*                                                               *
*       INITIALIZE THE VSAM BUFFER HEADER & STORE THE BUFFER ADDRESS *
*       IN THE DSRB.                                            *
*                                                               *
*****************************************************************
         MVC   RGWAID(RGWALEN),BLANKS   BLANK OUT THE RGWA
         LA    9,RBUFHDR              GET ADDR OF VSAM BUFFER HDR
         ST    9,DSRBVDAD             STORE IT IN THE DSRB
         DROP  7                      DROP TO REBASE BUFHDR
         USING BUFHDR,9               SET REG 9 AS BASE FOR BUFHDR
         LA    10,HDRTEXT             GET ADDR OF VSAM BUFFER TEXT
         SR    10,9                   TEXT - START = HEADER LENGTH
         STH   10,HDRTDISP            STORE DISPL. TO TEXT IN HDR
         LA    10,L'READBUF(10)       CALC. HDR + TEXT
         STH   10,HDRBLENG            STORE TOTAL BUFLEN IN HDR
*****************************************************************
*                                                               *
*       INITIALIZE THE READ GLOBAL WORK AREA (RGWA) FROM THE CNMRDCMD *
*       INPUT BUFFER.                                           *
*                                                               *
*****************************************************************
         DROP  9                      DROP TO REBASE BUFHDR
         USING BUFHDR,6               SET REG 6 AS BASE FOR BUFHDR
         LH    10,HDRTDISP            GET ADDR OF I/P COMMAND TEXT
         AR    10,6                   START+DISPLACEMENT = TEXT
         USING CNMRDCMD,10            SET REG 10 AS BASE FOR BUFHDR
         MVC   RGWAID(4),RGWACHRS     MOVE IN ID
         MVC   IKEY(4),RDIKEY         MOVE IN INITIAL KEY TO READ
         MVC   CURRKEY(4),RDIKEY      SET CURRENT READ KEY
         MVC   TOTREC(3),TREC         MOVE IN TOT RECORDS TO READ
         MVC   CURRTOT(3),ZERO        SET NO. RECORDS READ TO ZERO
         MVC   SUCCESS(3),ZERO        SET SUCCESSFULLY READ TO 0
         MVC   ENOFBUF(8),ENDBUF      SET END OF BUFFER CONSTANT
```

```
*********************************************************************
*                                                                   *
*       SCHEDULE A READ FOR THE RECORD WHICH HAS A KEY EQUAL TO THE *
*       KEY IN THE RGWA FIELD CURRKEY.                              *
*                                                                   *
*********************************************************************
SCHEDRD  EQU   *
         MVC   READBUF(L'READBUF),ASTERIKS MOVE ASTERISKS IN BUFFER
         LA    9,CURRKEY                SET REG TO • OF CURRENT KEY
         DSIZVSMS SWB=CWBSWB,DSRB=CWBDSRB,FUNC=GET,KEY=(9),KEYLEN=FOUR,*
               OPTION=(DIR,KEQ,FKS),DATAREA=RBUFHDR
         LTR   15,15                    TEST RETURN CODE
         BZ    SRDEXIT                  IF GOOD GO TO EXIT
*********************************************************************
*                                                                   *
*             SCHEDULED READ FAILED ISSUE MESSAGE AND GET OUT.      *
*                                                                   *
*********************************************************************
         MVI   TYPRD,C'S'               SET ERROR TYPE TO SCHEDULING
         ST    15,FDBK1                 STORE MAJOR RETURN CODE
         ST    0,FDBK2                  STORE MINOR RETURN CODE

         HEXEBC  TO=MAJOR,FROM=FDBK1
         HEXEBC  TO=MINOR,FROM=FDBK2
         DROP  6
         USING BUFHDR,7       BASE BUFFER HEADER AT CWBADATD
         MVI   HDRMTYPE,HDRTYPEU      SET MSGTYPE FOR OUTPUT
         L     10,MVTUFLD         GEN ADDR WHERE MSG CSECT
         LA    10,8(,10)           ADDR IS STORED
         L     10,0(,10)          GET MSG CSECT ADDR
         DSIMBS MID=256,SWB=CWBSWB,BFR=(7),MSGTBL=(10),
               P1=(*MAJOR,8),                                       *
               P2=(*MINOR,8),                                       *
               P3=(*CURRKEY,4),                                     *
               P4=(*TYPRD,1)
         C     15,RTRNCD12   WAS ERR MSG BUILT O.K.
         BNH   MQSMSG2       YES, GO SEND MSG
         BAL   14,SENDERR NO,GO BUILD INTERNAL ERRMSG & SEND
         LTR   15,15         WAS INTERNAL MSG SENT O.K.
         BZ    CLEANUP       YES, GO CLEAN UP AND EXIT
         DC    F'0'          MQS ERROR, HALT EXECUTION
MQSMSG2  EQU   *
         DSIMQS SWB=CWBSWB,BFR=(7),TASKID=DSRBOID
         LTR   15,15                    TEST MQS RETURN CODE
         BNZ   ABEND657
         B     CLEANUP                  GO TO CLEANUP LOGIC
ABEND657 DC    F'0'      MQS OUTPUT FAILED, HALT EXECUTION
```

```
*****************************************************************
*                                                               *
*           CODE FOR READ OF NEXT RECORD FOLLOWS.               *
*                                                               *
*****************************************************************
NEXTRD   EQU   *
         L     3,DSRBUSER              GET ADDRESS OF THE RGWA
         USING RGWA,3                  SET REG 3 AS BASE FOR RGWA
         UPONE ADDR=CURRTOT,DIGITS=3   INCREMENT CURRENT TOTAL
         USING BUFHDR,7                SET REG 7 AS BASE FOR BUFHDR
         LH    10,HDRTDISP             GET ADDR OF O/P COMMAND TEXT
         AR    10,7                    START+DISPLACEMENT = TEXT
         USING RDCNMCMD,10             SET REG 10 AS BASE FOR
*     RESPONSE AREA FOR QUEUEING BACK TO THE PSCP
         L     15,DSRBRCMA             GET PREVIOUS READ MAJ CODE
         LTR   15,15                   WAS THE READ SUCCESSFUL?
         BNZ   READERR                 IF NOT GO TO READ ERROR RTN.
         L     15,DSRBRCMI             GET PREVIOUS READ MIN CODE
         LTR   15,15                   WAS THE READ SUCCESSFUL?
         BZ    LASTREC                 IF O.K. GO TEST FOR LAST RECD
READERR  EQU   *
*****************************************************************
*                                                               *
*            READ ERROR PROCESSING FOLLOWS.                     *
*                                                               *
*****************************************************************
         MVI   CNMCODE,C'E'            MOVE IN CODE FOR ERROR RECORD
         MVC   DBKEY(4),CURRKEY        MOVE IN KEY OF RECORD
         HEXEBC  TO=MAJRC,FROM=DSRBRCMA
         HEXEBC  TO=MINRC,FROM=DSRBRCMI
         MVI   TYPERR,C'R'             SET ERROR TYPE TO READ
*****************************************************************
*                                                               *
*          ISSUE MQS MACRO TO INVOKE READVS PROCESSOR(DSITDSRD)  *
*                                                               *
*****************************************************************
MQSREAD  EQU   *
         DSIMQS SWB=CWBSWB,BFR=(7),TASKID=DSRBOID
         LTR   15,15                   WAS MESSAGE QUEUED O.K. ?
         BZ    SUMMTEST                IF ZERO IT WAS
         DC    F'0'      QUEUE FAILED, HALT EXECUTION
LASTREC  EQU   *
         UPONE ADDR=SUCCESS,DIGITS=3   INCREMENT SUCCESSFUL READS CTR
SUMMTEST EQU   *
         CLC   CURRTOT(3),TOTREC       COMPARE CURRENT TOTAL TO TOTAL
*                                      TO BE READ
         BNL   SUMMREC    IF NOT < GO SEND LAST & SUM RECS
FIRSTREC EQU   *
         CLC   SUCCESS,ONE             COMPARE # SUCCESSES TO ONE
         BE    SENDFRST                GO SEND 1ST SUCCESSFUL RECORD
         UPONE ADDR=CURRKEY,DIGITS=4   INCREMENT SUCCESSFUL READS CTR
         B     SCHEDRD                 GO SCHEDULE A READ
```

```
*********************************************************************
*                                                                   *
*            FIRST RECORD PROCESSING FOLLOWS.                       *
*                                                                   *
*********************************************************************
SENDFRST EQU   *
         MVI   CNMCODE,C'F'              MOVE IN CODE FOR FIRST RECORD
         MVC   DBKEY(4),CURRKEY          MOVE IN KEY OF RECORD
         MVC   DBREC(48),READBUF         MOVE IN RECORD READ
*********************************************************************
*                                                                   *
*          ISSUE MQS MACRO TO INVOKE READVS PROCESSOR(DSITDSRD)     *
*                                                                   *
*********************************************************************
         DSIMQS SWB=CWBSWB,BFR=(7),TASKID=DSRBOID
         LTR   15,15                     WAS MESSAGE QUEUED O.K. ?
         BNZ   ABEND655                  IF NONZERO ABEND
         UPONE ADDR=CURRKEY,DIGITS=4     INCREMENT SUCCESSFUL READS CTR
         B     SCHEDRD                   GO TO SCHEDULE READ
ABEND655 DC    F'0'      QUEUE FAILED, HALT EXECUTION
*********************************************************************
*                                                                   *
*         LAST RECORD AND SUMMARY RECORD PROCESSING FOLLOWS.        *
*                                                                   *
*********************************************************************
SUMMREC  EQU   *
         L     15,DSRBRCMA               DETERMINE IF THE LAST
         LTR   15,15                     READ WAS SUCCESSFUL,
         BNZ   SENDSMRC                  IF NOT DO NOT ATTEMPT TO SEND
         L     15,DSRBRCMI               BUT FINISH BY SENDING SUMMARY
         LTR   15,15                     RECORD.
         BNZ   SENDSMRC
         MVI   CNMCODE,C'L'               MOVE IN CODE FOR LAST RECORD
         MVC   DBKEY(4),CURRKEY           MOVE IN KEY OF RECORD
         MVC   DBREC(L'DBREC),READBUF     MOVE IN RECORD READ
*********************************************************************
*                                                                   *
*          ISSUE MQS MACRO TO INVOKE READVS PROCESSOR(DSITDSRD)     *
*                                                                   *
*********************************************************************
         DSIMQS SWB=CWBSWB,BFR=(7),TASKID=DSRBOID
         LTR   15,15                      WAS MESSAGE QUEUED O.K. ?
         BZ    SENDSMRC                   IF ZERO IT WAS, GO SEND SUMMARY
         DC    F'0'      QUEUE FAILED, HALT EXECUTION
SENDSMRC EQU   *
         MVI   CNMCODE,C'S'               MOVE IN CODE FOR SUMMARY REC
         MVC   IKEYRSP(4),IKEY            MOVE IN INITIAL KEY
         MVC   TRECRSP(3),TOTREC          MOVE IN NBR OF ATTEMPTED READS
         MVC   RDSUCESS(3),SUCCESS        MOVE IN NBR OF SUCCESSFUL READS
         MVC   DBREC(L'DBREC),BLANKS      BLANK OUT LAST RECORD IN BUF
```

```
*****************************************************************
*                                                               *
*        ISSUE MQS MACRO TO INVOKE READVS PROCESSOR(DSITDSRD)   *
*                                                               *
*****************************************************************
         DSIMQS SWB=CWBSWB,BFR=(7),TASKID=DSRBOID

         LTR   15,15                    WAS MSG QUEUED O.K. ?
         BZ    CLEANUP                  IF ZERO IT WAS, GO EXIT
         DC    F'0'      QUEUE FAILED, HALT EXECUTION
*****************************************************************
*                                                               *
*            FREE THE RGWA CODE FOLLOWS.                        *
*                                                               *
*****************************************************************
CLEANUP  EQU   *
         LA    4,RGWALEN                GET LENGTH OF RGWA
         DSIFRE  R,LV=(4),A=(3)
         LTR   15,15                    TEST FREE RETURN CODE
         BNZ   ABEND652
         LA    15,8                     INDICATE FREE THE DSRB
         B     SRDEXIT                  EXIT MODULE
ABEND652 DC    F'0' FREE OF WORK AREA FAILED,HALT EXECUTION
*****************************************************************
*  STANDARD EXIT LINKAGE IS PERFORMED. REGISTER 15 WILL CONTAIN A     *
* RETURN CODE.                                                  *
*****************************************************************
SRDEXIT  L     13,4(13)            RESET TO CALLERS SAVE AREA
         L     14,12(13)           RESTORE CALLERS
         LM    0,12,20(13)          REGISTERS - EXCEPT 15
         BR    14                   AND RETURN
*
*****************************************************************
*  THE FOLLOWING IS AN INTERNAL SUBROUTINE TO CONSTRUCT AND           *
*  QUEUE AN ERROR MSG TO THE USER INDICATING THAT DSIMBM              *
*  FAILED TO BUILD THE REQUESTED MESSAGE                              *
*****************************************************************
*
SENDERR  EQU   *
         LA    5,HDRTEXT      GET END OF BUFFER ADDR
         LA    6,MBMERRLN     GET MSG LENGTH
         STH   6,HDRMLENG     SET MSG LENGTH IN BUFFER HEADER
         LA    6,256                 GEN TOTAL BUFFER LENGTH
         STH   6,HDRBLENG     SET BUFFER LENGTH IN BUF HEADER
         USING MBMERRDS,5     BASE ERROR MSG DSECT
         LA    6,MBMERRLN+HDRTEXT GEN ADDR OF TEMP AREA
         MVC   MBMERRDS(MBMERRLN),MBMERR MOVE ERR MSG TO BUF
         CVD   15,0(,6)  CONVERT RETURN CODE TO PACKED DECIML
         UNPK  MBMRTRN(8),4(4,6)  UNPACK TEMP CODE INTO MSG
         OI    MBMRTRN+7,X'F0'  CHANGE SIGN 'C' TO 'F'
         L     1,CWBSWB       GET ADDR OF SWB
         L     15,MBSMIDA-DSISWB(,1)  GET MSG NBR
         CVD   15,0(,6)  CONVERT MSG NBR TO PACKED DECIMAL
         UNPK  MBMMSGNM(8),4(4,6) UNPACK MSG NBR INTO ERRMSG
         OI    MBMMSGNM+7,X'F0'  CHANGE SIGN 'C' TO 'F'
         DSIMQS  SWB=CWBSWB,BFR=(7),TASKID=DSRBOID
         BR    14             RETURN TO MAINLINE
*****************************************************************
```

```
*
*        CONSTANTS FOLLOW

UPTAB    DC    CL16'1234567890000000'
HEXTAB   DC    CL16'0123456789ABCDEF'
FOUR     DC    F'4'                  KEY LENGTH CONSTANT
LAST     DC    C'L'
FIRST    DC    C'F'
SUMM     DC    C'S'
ERROR    DC    C'E'
READRSP  DC    CL8'READRESP'
ENDBUF   DC    CL8'
RGWACHRS DC    CL4'RGWA'
BLANKS   DC    256C' '
ONE      DC    CL3'001'
IFRCDE   DC    XL2'0003'
ZERO     DC    CL3'000'
RTRNCD12 DC    F'12'          DSIMBM ERROR CHECK
MBMERR   DS    0C       INTERNAL ERR MSG FOR DSIMBM FAILURE
         DC    C'***ERROR IN DSIMBM, RETURN CODE='
         DC    CL8'00000000'
         DC    C', MESSAGE NBR='
         DC    CL8'00000000'
         DC    C' DSITDSRD TERMINATING COMMAND PROCESSING'
MBMERRND EQU   *
MBMERRLN EQU   MBMERRND-MBMERR      LENGTH OF ERROR MSG

RGWA     DSECT                     CNMREAD WORK AREA
RGWAID   DS    CL4                 RGWA ID FIELD
         DS    C                   BLANK
IKEY     DS    CL4                 INITIAL KEY TO READ
         DS    C                   BLANK
TOTREC   DS    CL3                 NUMBER OF RECORDS TO READ
         DS    C                   BLANK
CURRKEY  DS    CL4                 CURRENT KEY BEING READ
         DS    C                   BLANK
CURRTOT  DS    CL3                 NUMBER OF RECORDS READ SO FAR
         DS    C                   BLANK
SUCCESS  DS    CL3                 NO. OF RECORDS READ SUCCESSFULLY
         DS    CL2                 BLANK
FDBK1    DS    CL4                 RET CODE FEEDBACK AREA
FDBK2    DS    CL4                 RET CODE FEEDBACK AREA
         DS    C                   BLANK
RBUFHDR  DS    CL(BUFHDRND-BUFHDR) NCCF BUFFER HEADER
READBUF  DS    CL48                VSAM READ BUFFER
ENOFBUF  DS    CL8                 '        ' END OF BUFFER IND.
MAJOR    DS    CL8                 MAJOR RETURN CODE
MINOR    DS    CL8                 MINOR RETURN CODE
         DS    C                   BLANK
TYPRD    DS    CL1                 TYPE OF READ ERROR
RGWAEND  EQU   *
RGWALEN  EQU   RGWAEND-RGWA
```

```
CNMRDCMD DSECT                        CNMREAD COMMAND DSECT
RDVERB   DS    CL8                    'CNMREAD' COMMAND ID
         DS    CL4                    BLANKS
RDIKEY   DS    CL4                    KEY OF 1ST RECORD TO BE READ
         DS    C                      BLANK
TREC     DS    CL3                    NUMBER OF RECORDS TO BE READ
         DS    CL2                    BLANKS


RDCNMCMD DSECT                        READ RESPONSE DSECT
CNMIFR   DS    CL2                    IFR CODE AREA
CNMVERB  DS    CL8                    'READRESP' COMMAND
         DS    CL2                    BLANKS
CNMCODE  DS    C                      TYPE CODE FOR MESSAGE O/P
*
*                 E = ERROR MESSAGE
*                 F = FIRST RECORD READ
*                 L = LAST RECORD READ
*                 S = SUMMARY RECORD
*
         DS    C                      BLANK
DBKEY    DS    CL4                    KEY OF RECORD READ
         DS    CL4                    BLANKS
MAJRC    DS    CL8                    MAJOR RETURN CODE ON READ
MINRC    DS    CL8                    MINOR RETURN CODE ON READ
         DS    C                      BLANK
IKEYRSP  DS    CL4                    INITIAL KEY READ
         DS    C                      BLANK
TRECRSP  DS    CL3                    TOTAL RECORDS TO BE READ
         DS    C                      BLANK
RDSUCESS DS    CL3                    TOTAL READ SUCCESSFULLY
         DS    C                      BLANK
TYPERR   DS    CL1                    TYPE OF READ ERROR
*
*                 S = SCHEDULING ERROR
*                 R = READ ERROR
*
         DS    CL4                    BLANKS
DBREC    DS    CL48                   RECORD READ FRO DATA BASE
RDCNMEND EQU   *
RDCNMLEN EQU   RDCNMEND-RDCNMCMD


MBMERRDS DSECT  DSECT FOR INTERNAL ERRMSG FOR DSIMBM FAILURE
         DS    CL32
MBMRTRN  DS    CL8           RETURN CODE FROM DSIMBM
         DS    CL14
MBMMSGNM DS    CL8           MSG NBR THAT CAUSED FAILURE


DSITDSRD CSECT
ASTERIKS DC    (L'READBUF)C'*'
         END DSITDSRD
```

```
*****************************************************************
*                                                               *
*  NAME = DSITDSOL                                              *
*                                                               *
*  FUNCTION = THIS COMMAND PROCESSOR WILL SEND REQUESTS TO CNMI  *
*  AND HANDLE THE RESPONSES.                                    *
*                                                               *
*  ENTRY POINT = DSITDSOL                                       *
*                                                               *
*  INPUT =                                                      *
*      REGISTERS:                                               *
*           R1  = DSICWB ADDRESS                                *
*           R13 = ADDRESS OF STANDARD SAVEAREA                  *
*           R14 = RETURN ADDRESS                                *
*           R15 = ENTRY POINT OF ROUTINE                        *
*                                                               *
*      DSICWB (POINTED TO BY R1 - PSCP AND DSCP):               *
*           CWBSAVEA = AN 18 WORD SAVE AREA FOR USE IN THIS CMD PROC. *
*           CWBBUF = ADDRESS OF A MESSAGE BUFFER. THE BUFFER    *
*                    CONTAINS A STANDARD NCCF BUFFER HEADER.    *
*           CWBPDB = ADDRESS OF A PDB FOR USE IN THIS CMD PROC. *
*                    NOTE - THE PDB WILL CONTAIN VALID INFORMATION *
*                    ONLY WHEN CONTROL IS RECEIVED INITIALLY FROM *
*                    THE TASK (OST OR DST) DRIVING THIS CMD PROC. *
*           CWBSWB = ADDRESS OF A SWB FOR USE IN THIS CMD PROC. *
*           CWBTIB = ADDRESS OF THE DSIOST OR DSIZDST TIB.      *
*           CWBADATD = A 256 BYTE WORK AREA FOR USE IN THIS CMD PROC. *
*           CWBDSRB = FOR A DSCP ADDRESS OF A DSRB CONTAINING   *
*                     INFORMATION RELATED TO PROCESSING OF THE  *
*                     COMMAND. FOR A PSCP THIS FIELD IS MEANINGLESS. *
*                                                               *
*      DSIDSRB (POINTED TO BY CWBDSRB - DSCP ONLY):             *
*           DSRBFNCD = REASON FOR INVOCATION                    *
*           DSRBRCMA = MAJOR RETURN CODE ON COMPLETION OF REQUEST *
*           DSRBRCMI = MINOR RETURN CODE ON COMPLETION OF REQUEST *
*           DSRBOID  = OPID OF OST THAT INVOKED US              *
*           DSRBUSER = WORK FIELD                               *
*           DSRBINPT = ADDRESS OF CNMI BUFFER                   *
*                                                               *
*  OUTPUT =                                                     *
*      REGISTERS:                                               *
*           R0 - R14 = RESTORED TO INPUT STATUS                 *
*           R15 = RETURN CODE                                   *
*                                                               *
*      RETURN CODE (R15 ON EXIT):                               *
*         FOR A DSCP:                                           *
*            8  = RELEASE THE DSRB PASSED TO THIS CMD PROC - FCT *
*                 REQUESTED IS COMPLETE.                        *
*            0  = MAINTAIN THE DSRB PASSED TO THIS CMD PROC -   *
*                 RESUME PROCESSING HAS BEEN SCHEDULED.         *
*                                                               *
```

```
*  DESCRIPTION =                                                        *
*     THIS COMMAND PROCESSOR IS INVOKED BY DSITPSOL WHICH HAS QUEUED   *
*     A COMMAND WITH THE FOLLOWING POSSIBLE SYNTAXES:                  *
*                                                                      *
*          1) - NCFCNMI <PUNAME>                                       *
*          2) - NCFCNMI <PUNAME> <OPTION>                              *
*                       WHERE OPTION := REJECT | REPLACE               *
*                                                                      *
*     DSITPSOL HAS VERIFIED THE PUNAME FOR LENGTH AND THE OPTION,      *
*     IF PRESENT.  DSITDSOL PUTS THE PUNAME IN A WORK AREA, SELECTS    *
*     A REAL OR DUMMY RU, AND INVOKES DSIZCSMS TO MAKE THE CNMI        *
*     REQUEST.  IF THE REQUEST COMPLETES SUCCESSFULLY OR WITH A        *
*     NEGATIVE RESPONSE, AN APPROPRIATE COMMAND IS FORMATTED AND       *
*     QUEUED TO DSITPSOL.                                              *
*                                                                      *
*     THE REJECT OPTION CAUSES THE USER FORWARD EXIT, DSICPUTE,        *
*     TO REJECT THE REQUEST.  THE REPLACE OPTION CAUSES DSICPUTE       *
*     TO SUBSTITUTE A GOOD RU FOR THE DUMMY RU AND TO REQUEST BUFFER   *
*     SUBSTITUTION.                                                    *
*                                                                      *
*                                                                      *
*                                                                      *
*                                                                      *
*                                                                      *
*  EXIT = RETURN TO ADDRESS IN R14, WITH RETURN CODE SET IN R15.       *
*                                                                      *
*  EXTERNAL REFERENCES = NONE                                          *
*                                                                      *
*  CONTROL BLOCKS =                                                    *
*           DSICWB - COMMAND-PROCESSOR WORK BLOCK                      *
*           DSIDSRB - DATA SERVICES REQUEST BLOCK (DSCP)               *
*           DSIPDB - PARSE DATA BLOCK                                  *
*           DSISWB - SERVICE-ROUTINE WORK BLOCK                        *
*           DSITIB - TASK INFORMATION BLOCK                            *
*           DSITVB - TASK VECTOR BLOCK                                 *
*           DSIMVT - MAIN VECTOR TABLE                                 *
*           DSISVL - SERVICE ROUTINE VECTOR LIST                       *
*           DSIIFR - INTERNAL FUNCTION REQUEST                         *
*                                                                      *
*  MACROS = DSICBS,DSIGET,DSIFRE,DSIMQS,DSIZCSMS                       *
*                                                                      *
************************************************************************
```

```
DSITDSOL CSECT
*****************************************************************
*  MODULE ADDRESSABILITY IS ESTABLISHED, ADDRESSABILITY TO THE CWB IS *
* ESTABLISHED, AND THE CWB SAVEAREA IS USED IN PERFORMING STANDARD    *
* ENTRY LINKAGE.                                                      *
*****************************************************************
*
         USING *,R15               ESTABLISH ADDRESSABILITY FOR BRANCH
         B     SAVEREGS            BRANCH AROUND NAME AND DATE
         DC    C'DSITDSOL
         DC    C'&SYSDATE'
SAVEREGS EQU   *
         DROP  R15                 DROP ADDRESSABILITY FOR BRANCH
         STM   R14,R12,12(R13)     SAVE REGS IN CALLERS AREA
         LR    R12,R15             PUT MY ENTRY POINT INTO REG 12
         USING DSITDSOL,R12        ESTABLISH MODULE ADDRESSABILITY
         LR    R3,R1               PUT CWB ADDR INTO R3
         USING DSICWB,R3           BASE THE CWB
         LA    R2,CWBSAVEA         GET MY SAVEAREA ADDR
         ST    R13,4(,R2)          SET MY BACKWARD POINTER
         ST    R2,8(,R13)          SET CALLERS FORWARD POINTER
         LR    R13,R2              PUT MY SAVEAREA INTO REG 13
         XC    8(4,R13),8(R13)     ZERO MY FORWARD POINTER
         L     R4,CWBTIB           GET MY TIB ADDR
         L     R6,TIBTVB-DSITIB(,R4)  GET MY TVB ADDR
         ST    R6,0(,R13)          PUT MY TVB ADDR IN 1ST WORD OF SAVEA
         L     R4,TVBMVT-DSITVB(,R6)  GET MY MVT ADDR
         USING DSIMVT,R4           BASE THE MVT
         L     R5,CWBDSRB          GET MY DSRB ADDR
         USING DSIDSRB,R5          BASE THE DSRB
*****************************************************************
*                                                               *
*  INITIALIZE THE OUTPUT BUFFER HEADER FOR MESSAGES TO THE OST  *
*                                                               *
*****************************************************************
*
         LA    R7,CWBADATD         GET OUTPUT BUFFER ADDR
         USING BUFHDR,R7           BASE THE BUF HEADER ON THE OUTPUT BUF
         LA    R8,BUFHDRND-BUFHDR  GEN LENGTH OF BUFFER HEADER
         STH   R8,HDRTDISP         STORE OFFSET TO MSG IN HEADER
         LA    R8,256              GEN LENGTH OF BUFFER
         STH   R8,HDRBLENG         STORE BUFFER LENGTH IN HEADER
         MVI   HDRMTYPE,HDRTYPEU   INIT MESSAGE TYPE TO USER
         MVI   HDRIND,X'00'        ZERO INDICATORS IN HEADER
         MVC   HDRDOMID(8),MVTCURAN  STORE DOMAIN ID IN HEADER
         XC    HDRPOI(L'HDRPOI),HDRPOI  ZERO POI INFO IN HEADER
         XC    HDRTSTMP(4),HDRTSTMP   PUT A PACKED ZERO
         MVI   HDRTSTMP+3,X'0C'          INTO THE TIME STAMP
*****************************************************************
*                                                               *
*  IF THIS IS THE INITIAL INVOCATION, GETMAIN AN AREA FOR THE CNMI  *
*  BUFFER AND VERIFY THE OPERANDS.  IF EVERYTHING IS OK,        *
*  ISSUE DSIZCSMS TO SEND CNMI REQUEST.                         *
*                                                               *
```

```
*********************************************************************
*
         CLI   DSRBFNCD,DSRBFNRM  IS THIS THE INITIAL INVOCATION
         BNE   CNMIRESP           NO, GO LOOK AT THE CNMI RESPONSE
         DSIGET  LV=256,          GET THE CNMI BUF AND QUEUE TO TVB    >
               A=DSRBUSER,                                             >
               LISTA=BUFHDRND,                                         >
               Q=YES,                                                  >
               TASKA=(R6),                                             >
               REENT=YES,                                              >
               EXIT=NO
         LTR   R15,R15            TEST THE RETURN CODE
         BNZ   GETMERR            IF NOT ZERO, GETMAIN ERROR
         L     R6,CWBPDB          GET ADDR OF PDB
         L     R8,CWBBUF          GET ADDR OF INPUT COMMAND BUF
         USING DSIPDB,R6          BASE THE PDB
         LA    R9,PDBTABLE        GET STARTING ADDR OF PDB ENTRIES
         LA    R9,PDBENTND-PDBENTRY(,R9)  GET ADDR OF 2ND ENTRY
         USING PDBENTRY,R9        BASE THE PDBENTRY
         MVI   BUFHDRND,X'40'     INITIALIZE TEMP AREA FOR
         MVC   BUFHDRND+1(7),BUFHDRND   PUNAME TO BLANKS
         AH    R8,PDBDISP         GEN ADDR OF PUNAME IN INPUT BUF
         XR    R10,R10            ZERO REG 10
         IC    R10,PDBLENG        GET LENGTH OF PUNAME
         BCTR  R10,R0             DECREMENT MOVE LENGTH
         EX    R10,MOVE           MOVE PUNAME TO TEMP AREA
         LA    R10,3              PUT 3 IN REG 10 FOR ENTRIES TEST
         CH    R10,PDBNOENT       TEST FOR 3 ENTRIES IN COMMAND
         BE    SETOPTN            IF 3, GO SEE WHAT OPTION SPECIFIED
         LA    R10,REQMS          GET ADDR OF THE REAL RU
         B     SENDRU             GO SEND THE RU
SETOPTN  EQU   *
*        ***SELECT THE DUMMY RU ASSOCIATED WITH THE SPECIFIED OPTION
         LA    R9,PDBENTND-PDBENTRY(,R9)  GEN ADDR OF 3RD PDB ENTRY
         L     R8,CWBBUF          GET STARTING ADDR OF INPUT BUF
         AH    R8,PDBDISP         GEN ADDR OF OPTION IN INPUT BUF
         CLC   0(3,R8),REJ        COMPARE FOR REJECT
         BNE   REPLRU             IF NOT EQUAL, OPTION MUST BE REPLACE
         LA    R10,REJ            GET ADDR OF DUMMY REJECT RU
         B     SENDRU             GO SEND THE RU
REPLRU   EQU   *
*        ***SINCE OPTION WAS VERIFIED BY DSITPSOL IT MUST BE REPLACE
         LA    R10,REPL           GET ADDR OF DUMMY REPLACE RU
SENDRU   EQU   *
*        ***ISSUE DSIZCSMS TO SEND FORWARD RU, CHECK RESULT
         LA    R8,BUFHDRND        GET ADDRESS OF PUNAME
         DSIZCSMS  SWB=CWBSWB,                                        >
               DSRB=(R5),                                             >
               INPUT=DSRBUSER,                                        >
               LENGTH=LNGTH256,                                       >
               RU=(R10),                                              >
               RULENG=LNGTH8,                                         >
               DEST=(R8)
         LTR   R0,R0              TEST MAJOR RETURN CODE
```

```
         BNZ   CHKFWDEX           IF NOT ZERO, EXIT MAY HAVE REJECTED
*        ***RU SCHEDULED OK, INIT BUFHDR AND MOVE OK MSG TO OUTPUT BUF
         LA    R11,L'SCHELDOK     GET LENGTH OF MSG
         STH   R11,HDRMLENG       STORE MSG LENGTH IN BUF HEADER
         MVC   BUFHDRND(L'SCHELDOK),SCHELDOK  MOVE MSG TO OUTPUT BUF
         UNPK  BUFHDRND+37(3),DSRBPRID(2) UNPACK PRID INTO OUT BUF
         MVC   BUFHDRND+40(1),DSRBPRID+1  MOVE LAST BYTE TO OUT BUF
         OI    BUFHDRND+39,X'F0'  MAKE ZONE CORRECT
         OI    BUFHDRND+40,X'F0'  MAKE ZONE CORRECT
         TR    BUFHDRND+37(4),TRANSTBL-240  MAKE ALL CHARS PRINTABLE
         LA    R11,0              INDICATE DSRB SHOULD NOT BE FREED
         B     EXIT               GO SEND MSG AND EXIT
CHKFWDEX EQU   *
*        ***DETERMINE IF THE USER FORWARD EXIT REJECTED THE REQUEST
         LA    R11,SWBCEXIT       GET TEST VALUE FOR USER EXIT REJECT
         CR    R0,R11             TEST MINOR RETURN CODE FOR REJECT
         BNE   CNMMERR            IF NOT EQUAL, WE HAVE A CNMM ERROR
*        ***INIT BUF HEADER FOR MSG INDICATING USER EXIT REJECT
         LA    R11,L'FWEXITRJ     GET MSG LENGTH
         STH   R11,HDRMLENG       STORE MSG LENGTH IN OUTPUT BUF HDR
         MVC   BUFHDRND(L'FWEXITRJ),FWEXITRJ  MOVE EXIT REJ MSG TO BUF
         B     FREEBUF            GO FREE BUF, SEND MSG, AND EXIT
CNMMERR  EQU   *
*        ***DSIZCSMM COULD NOT EXECUTE THE REQUEST, SEND ERROR MSG
         LA    R11,L'CNMMREJ      GET LENGTH OF MSG
         STH   R11,HDRMLENG       STORE MSG LENGTH IN OUTPUT BUF HDR
         MVC   BUFHDRND(L'CNMMREJ),CNMMREJ  MOVE MSG TO OUTPUT BUF
*
         STCM  R15,X'F',BUFHDRND+43  PUT MAJOR RETURN CODE IN BUF
*
         UNPK  BUFHDRND+43(7),BUFHDRND+43(4)  UNPACK THE RETURN CODE
*        STCM  R15,X'1',BUFHDRND+50  STICK LAST BYTE IN BUF
         OI    BUFHDRND+49,X'F0'  MAKE ZONE CORRECT
         OI    BUFHDRND+50,X'F0'  MAKE ZONE CORRECT
         TR    BUFHDRND+43(8),TRANSTBL-240  MAKE ALL CHARS PRINTABLE
*
         STCM  R0,X'F',BUFHDRND+61  PUT MINOR RETURN CODE IN BUF
*
         UNPK  BUFHDRND+61(7),BUFHDRND+61(4)  UNPACK THE RETURN CODE
*        STCM  R0,X'1',BUFHDRND+68  STICK LAST BYTE IN BUF
         OI    BUFHDRND+67,X'F0'  MAKE THE ZONE CORRECT
         OI    BUFHDRND+68,X'F0'  MAKE THE ZONE CORRECT
         TR    BUFHDRND+61(8),TRANSTBL-240  MAKE ALL CHARS PRINTABLE
         B     FREEBUF            GO FREE BUFFER, SEND MSG, AND EXIT
GETMERR  EQU   *
*        ***GETMAIN FAILED FOR CNMI BUFFER, INIT BUF HDR AND MOVE MSG
         LA    R11,L'GMERRMSG     GET LENGTH OF GETMAIN ERROR MSG
         STH   R11,HDRMLENG       STORE MSG LENGTH IN OUTPUT BUF HDR
         MVC   BUFHDRND(L'GMERRMSG),GMERRMSG  MOVE MSG TO OUTPUT BUF
         LA    R11,8              INDICATE THAT DSRB SHOULD BE FREED
         B     EXIT               GO SEND MSG AND EXIT
```

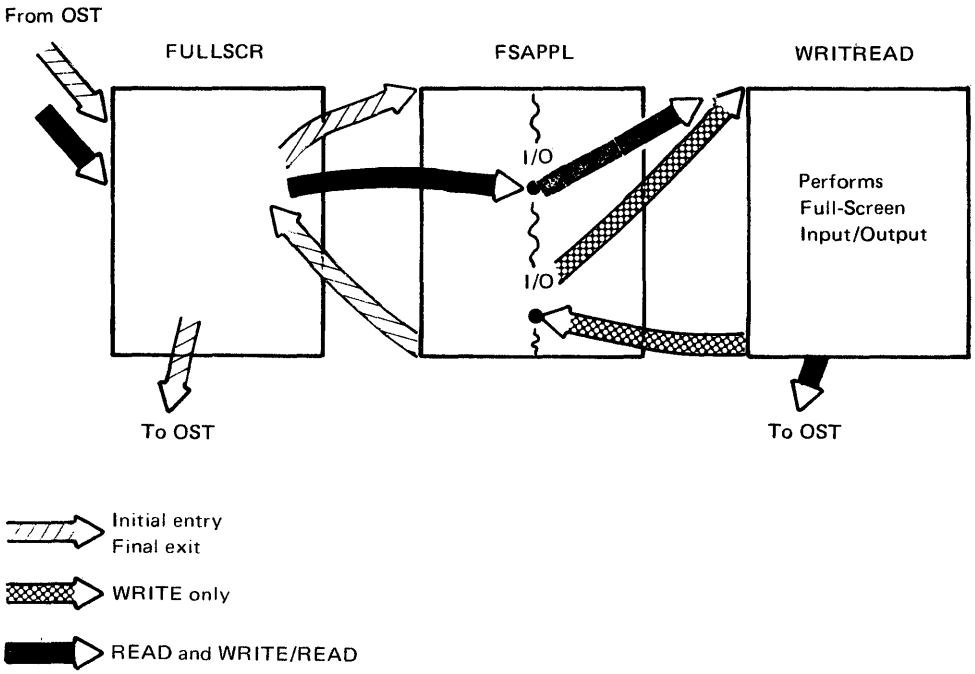
```
*****************************************************************
*                                                               *
*  IF THE REQUESTED FUNCTION WAS SUCCESSFULLY COMPLETED, BUILD THE    *
*  APPROPRIATE COMMAND TO REINVOKE DSITPSOL.                    *
*  IF UNSUCCESSFUL COMPLETION, FORMAT THE ERROR MSG.            *
*                                                               *
*****************************************************************
*
CNMIRESP EQU   *
         L     R8,DSRBINPT         GET ADDR OF CNMI BUFFER
         CLI   DSRBFNCD,DSRBFSOL   TEST FOR SOLICITED FUNCTION CODE
         BNE   FNCTERR             IF NOT EQUAL, GO SEND ERROR MSG
         CLC   DSRBRCMI(4),RSPGOOD  TEST FOR GOOD MINOR RTN CODE
         BNE   CHKNEGRP            IF NOT GOOD, GO CHECK FOR NEG RESP
         MVC   BUFHDRND+2(9),CNMIDELV  MOVE DELIVER CMND TO OUTPUT BUF
         B     BLDCMND             GO COMPLETE COMMAND BUILD
CHKNEGRP EQU   *
         CLC   DSRBRCMI(4),RSPNGR  TEST FOR NEGATIVE RESPONSE
         BNE   FNCTFAIL            IF NOT EQUAL, GO FORMAT ERROR MSG
         MVC   BUFHDRND+2(9),CNMINEGR  MOVE IN NEG RESPONSE CMND
BLDCMND  EQU   *
*        ***COMPLETE BUILD OF OUTPUT CMND WITH IFR AND DELIVERED INFO
         MVI   HDRMTYPE,HDRTYPEI  SET MSG TYPE TO COMMAND REQUEST
         MVC   BUFHDRND(2),INTRNLRQ  SET IFR CODE
         LH    R9,HDRMLENG-BUFHDR(,R8)  GET INPUT RESPONSE LENGTH
         AH    R8,HDRTDISP-BUFHDR(,R8)  POINT 8 TO START OF RESPONSE
         BCTR  R9,R0               DECREMENT MOVE LENGTH
         EX    R9,MOVERU           MOVE RESPONSE TO OUTPUT BUFFER
         LA    R9,12(,R9)          GEN OUTPUT MSG LENGTH
         STH   R9,HDRMLENG         PUT MSG LENGTH IN OUTPUT BUF HDR
         B     FREEBUF             GO FREE CNMI BUF AND EXIT PROCESSING
FNCTERR  EQU   *
*        ***FUNCTION CODE DID NOT INDICATE A SOLICITED RESPONSE
         LA    R11,L'BADFNCT       GET LENGTH OF ERROR MSG
         STH   R11,HDRMLENG        PUT MSG LENGTH IN OUTPUT BUF HDR
         MVC   BUFHDRND(L'BADFNCT),BADFNCT  MOVE MSG TO OUTPUT BUF
         MVC   BUFHDRND+31(1),DSRBFNCD  MOVE FUNCTION CODE TO BUFFER
         UNPK  BUFHDRND+30(1),BUFHDRND+31(1)  SWAP NIBBLES
         NI    BUFHDRND+30,X'0F'   ZERO THE ZONE
         NI    BUFHDRND+31,X'0F'   ZERO THE ZONE
         TR    BUFHDRND+30(2),TRANSTBL  MAKE CHARS PRINTABLE
         B     FREEBUF             GO FREE CNMI BUFFER AND EXIT
FNCTFAIL EQU   *
*        ***REQUEST DID NOT COMPLETE SUCCESSFULLY, BUILD ERROR MSG***
         LA    R11,L'UNSUCCES      GET LENGTH OF ERROR MSG
         STH   R11,HDRMLENG        PUT MSG LENGTH IN OUTPUT BUF HDR
         MVC   BUFHDRND(L'UNSUCCES),UNSUCCES  MOVE MSG TO OUTPUT BUF
*
         MVC   BUFHDRND+42(4),DSRBRCMA  MOVE MAJOR RETURN CODE TO BUF
*
         UNPK  BUFHDRND+42(7),BUFHDRND+42(4)  UNPACK THE RETURN CODE
         MVC   BUFHDRND+49(1),DSRBRCMA+3  MOVE LAST BYTE
         OI    BUFHDRND+48,X'F0'   MAKE THE ZONE CORRECT
         OI    BUFHDRND+49,X'F0'   MAKE THE ZONE CORRECT
         TR    BUFHDRND+42(8),TRANSTBL-240  MAKE ALL CHARS PRINTABLE
```

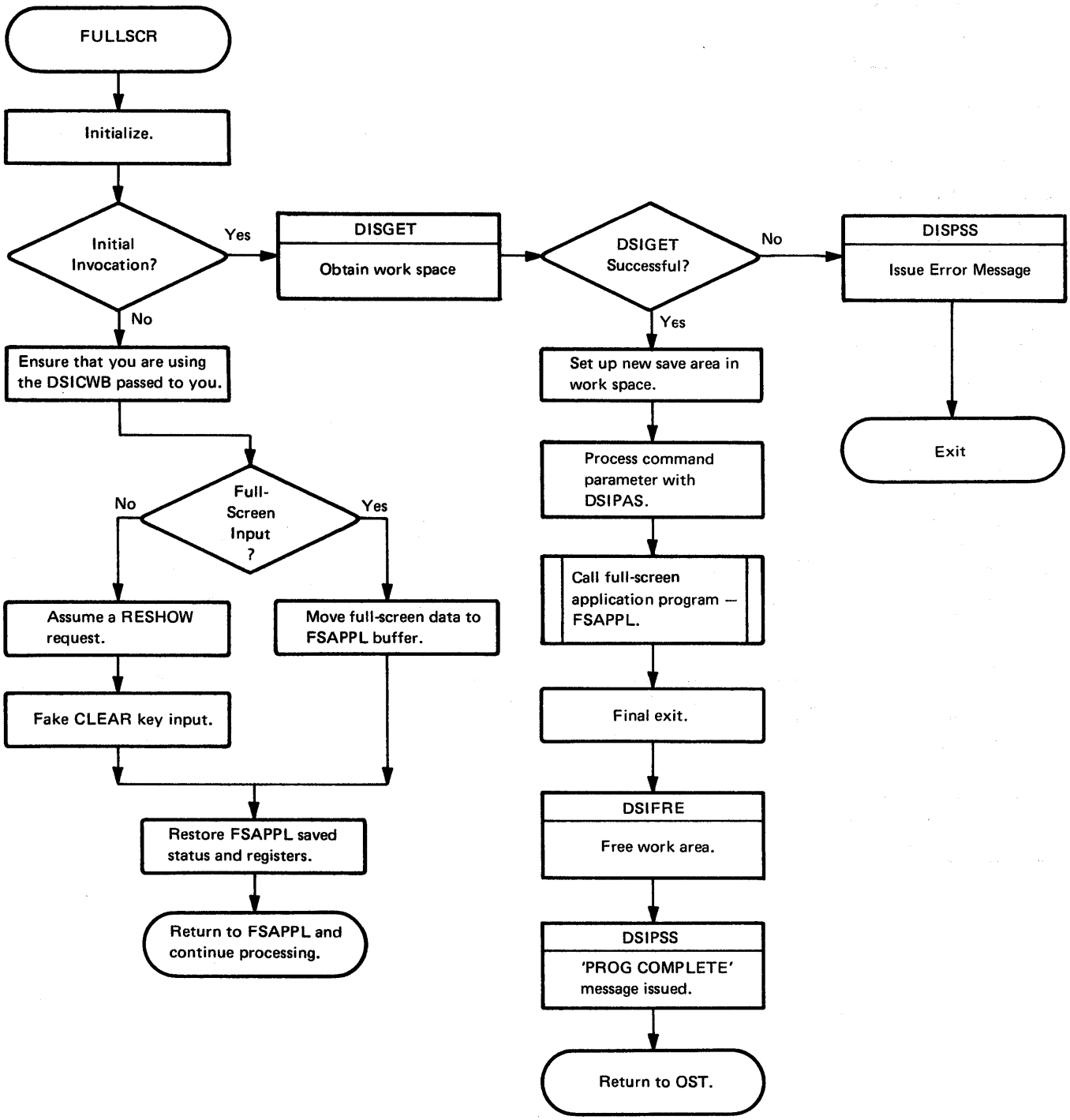
```
*
         MVC   BUFHDRND+60(7),DSRBRCMI  MOVE MINOR RETURN CODE TO BUF
         UNPK  BUFHDRND+60(7),BUFHDRND+60(4)  UNPACK THE RETURN CODE
         MVC   BUFHDRND+67(1),DSRBRCMI+3  MOVE LAST BYTE
         OI    BUFHDRND+66,X'F0'  MAKE THE ZONE CORRECT
         OI    BUFHDRND+67,X'F0'  MAKE THE ZONE CORRECT
         TR    BUFHDRND+60(8),TRANSTBL-240  MAKE ALL CHARS PRINTABLE
*****************************************************************
*                                                               *
*  FREE THE CNMI BUFFER AND CONTINUE WITH EXIT PROCESSING       *
*                                                               *
*****************************************************************
*
FREEBUF  EQU   *
         LA    R11,CWBADATD+241   GET ADDR OF WORK AREA
         L     R2,CWBTIB          GET MY TIB ADDR
         L     R2,TIBTVB-DSITIB(,R2)  GET MY TVB ADDR
         DSIFRE  LV=256,                                        >
               A=DSRBINPT,                                      >
               LISTA=(R11),                                     >
               Q=YES,                                           >
               TASKA=(R2),                                      >
               EXIT=NO
         LA    R11,8              INDICATE THAT DSRB SHOULD BE FREED
         LTR   R15,R15            TEST FOR GOOD FREE
         BZ    EXIT               IF OK, GO SEND OUTPUT BUF AND EXIT
         DC    F'0'               BAD FREE, HALT EXECUTION
*****************************************************************
*                                                               *
*  QUEUE THE OUTPUT BUFFER TO THE OST THAT INVOKED US, RESTORE  *
*  THE REGS AND RETURN TO THE DST                               *
*                                                               *
*****************************************************************
*
EXIT     EQU   *
         DSIMQS  SWB=CWBSWB,                                    >
               BFR=(R7),                                        >
               TASKID=DSRBOID
         LTR   R15,R15            TEST FOR GOOD MQS
         BZ    RESTOR             IF OK, GO RESTORE REGS AND EXIT
         DC    F'0'               MQS FAILED, HALT EXECUTION
RESTOR   EQU   *
*        ***PERFORM STANDARD EXIT LINKAGE PROCESSING***
         LR    R15,R11            SET THE RETURN CODE
         L     R13,4(,R13)        GET CALLERS SAVEAREA ADDR
         L     R14,12(,R13)       RESTORE REG 14
         LM    R0,R12,20(R13)     RESTORE REGS 0-12
         BR    R14                RETURN TO DST
*****************************************************************
*                                                               *
*                    DECLARES                                   *
*                                                               *
*****************************************************************
```

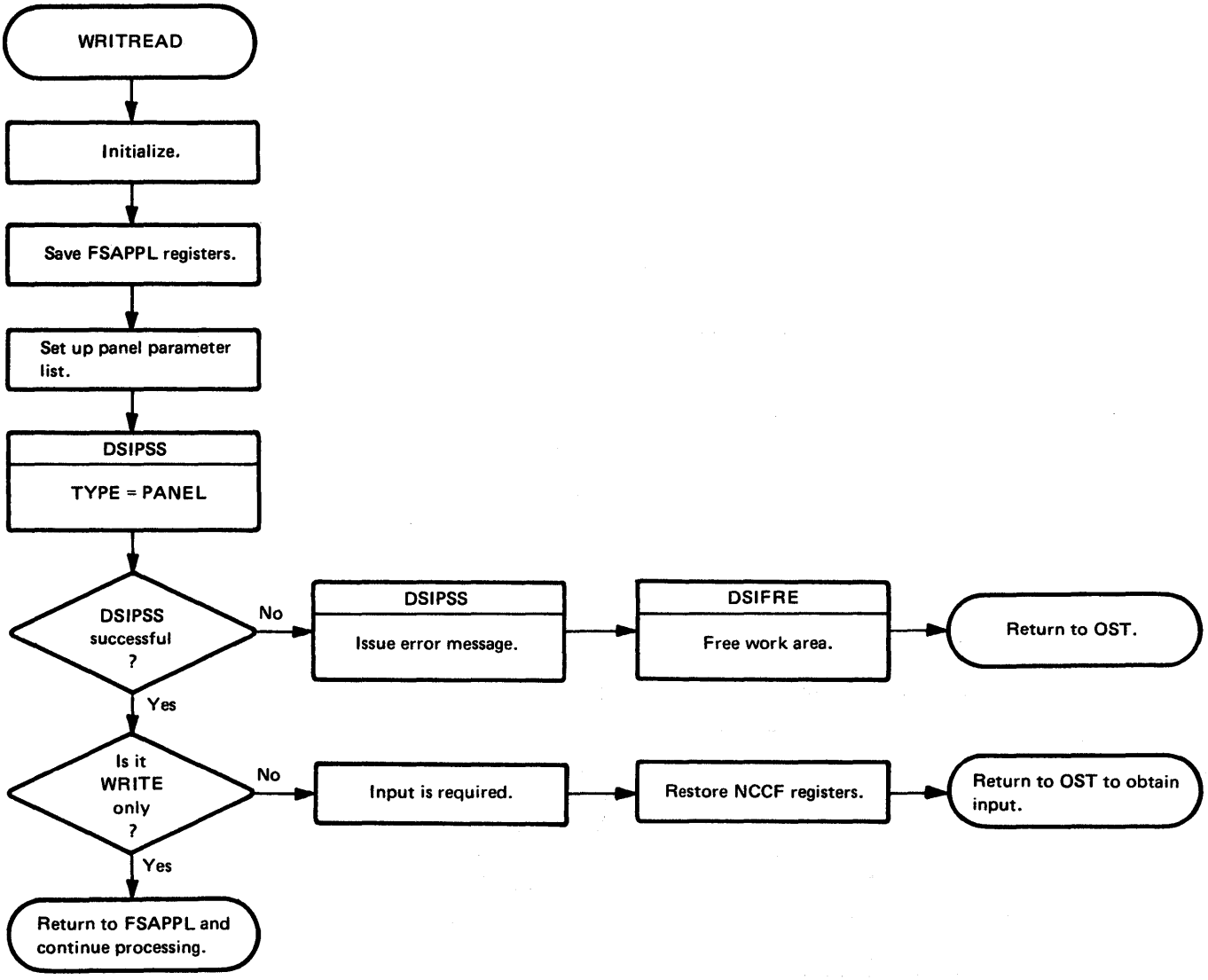
```
*
RSPGOOD  DC    A(DSRCGOOD)
RSPNGR   DC    A(DSRCNGRP)
LNGTH256 DC    F'256'
LNGTH8   DC    F'8'
MOVE     MVC   BUFHDRND(0),0(R8)  OBJECT OF EXECUTE TO MOVE PUNAME
MOVERU   MVC   BUFHDRND+11(0),0(R8)  OBJECT OF EXECUTE TO MOVE CNMI RSP
REJ      DC    XL8'D9C5D10000000000'  DUMMY REJECT RU
REPL     DC    XL8'D9C5D70000000000'  DUMMY REPLACE RU
REQMS    DC    XL8'4103040000000000'  THE REAL RU
SCHELDOK DC    C'CNMI REQUEST HAS BEEN SCHEDULED-PRID=0000'
FWEXITRJ DC    C'USER FORWARD EXIT HAS REJECTED THIS REQUEST'
CNMMREJ  DC    C'DSIZCSMM INDICATED ERROR CONDITION MAJOR=X''        ''-
               MINOR=X''        '''
GMERRMSG DC    C'GETMAIN FOR CNMI BUFFER FAILED'
TRANSTBL DC    C'0123456789ABCDEF'
UNSUCCES DC    C'FUNCTION COMPLETED UNSUCCESSFULLY MAJOR=X''       '' M-
               INOR=X''       '''
BADFNCT  DC    C'INVALID FUNCTION CODE--CODE=X''  '''
INTRNLRQ DC    Y(IFRCODCR)        IFR CODE FOR CROSS TASK CMND QUEUEING
CNMIDELV DC    C'CNMIDELV '       VERB FOR DELIVER RESPONSE
CNMINEGR DC    C'CNMINEGR '       VERB FOR NEGATIVE RESPONSE
```

Appendix F. Sample Full-Screen Command Processor

This appendix is an example of a user-written full-screen command processor.
Note: This command processor is not executable as it is shown in this appendix.




FULLSCR
Initialize.
Initial Invocation?
Yes
No
DISGET
Obtain work space
DSIGET Successful?
No
Yes
DISPSS
Issue Error Message
Exit
Ensure that you are using the DSICWB passed to you.
Full-Screen Input ?
No
Yes
Assume a RESHOW request.
Move full-screen data to FSAPPL buffer.
Fake CLEAR key input.
Restore FSAPPL saved status and registers.
Return to FSAPPL and continue processing.
Set up new save area in work space.
Process command parameter with DSIPAS.
Call full-screen application program — FSAPPL.
Final exit.
DSIFRE
Free work area.
DSIPSS
'PROG COMPLETE' message issued.
Return to OST.


WRITREAD
Initialize.
Save FSAPPL registers.
Set up panel parameter list.
DSIPSS
TYPE = PANEL
DSIPSS successful ?
No
DSIPSS
Issue error message.
DSIFRE
Free work area.
Return to OST.
Yes
Is it WRITE only ?
No
Input is required.
Restore NCCF registers.
Return to OST to obtain input.
Yes
Return to FSAPPL and continue processing.

```
         TITLE 'FULLSCR - NCCF 3270 PANEL DRIVER'
* /*****************************************************************
*                                                                   *
*MODULE NAME: FULLSCR                                               *
*                                                                   *
*DESCRIPTIVE NAME: NCCF PANEL DRIVER                                *
*                                                                   *
*FUNCTION: THIS MODULE DRIVES A NCCF FULL SCREEN COMMAND PROCESSOR.  *
*          NCCF.                                                    *
*                                                                   *
*ATTRIBUTES: REENTRANT                                              *
*                                                                   *
*COMMAND SYNTAX: VERB <XYZ>                                         *
*  'XYZ' IS OPTIONAL. IF NOT SPECIFIED, IT IS NOT PASSED TO FSAPPL. *
*  'XYZ' MAY HAVE PARAMETER SYNONYMS DEFINED WITH THE 'PARMSYN' NCCF*
*     DEFINITION STATEMENT. DSIPAS IS USED TO OBTAIN THE REGULAR    *
*     VALUE.                                                        *
*                                                                   *
*ENTRY POINT: FULLSCR                                               *
*  PURPOSE: TO DEMONSTRATE FULL SCREEN PANEL MODE                   *
*  LINKAGE: CALL                                                    *
*  INPUT: STANDARD NCCF COMMAND PROCESSOR LINKAGE                   *
*    REGISTERS: R1=CWB ADDRESS CONTAINING:                          *
*                  TIB ADDRESS                                      *
*                  SWB ADDRESS                                      *
*                  PDB ADDRESS                                      *
*                  COMMAND BUFFER ADDRESS                           *
*               R13=SAVE AREA ADDRESS                               *
*               R14=RETURN ADDRESS                                  *
*               R15=ADDR OF FULLSCR                                 *
*                                                                   *
*EXIT NORMAL:                                                       *
*  LINKAGE: RETURN TO CALLER                                        *
*  OUTPUT:                                                          *
*    REGISTERS:                                                     *
*      UNCHANGED REGISTERS: ALL REGISTERS EXCEPT R15                *
*      OUTPUT REGISTERS: R15 CONTAINS A RETURN CODE FOR CALLER.     *
*                        R15 = 0 - NO ERRORS                        *
*EXIT ERROR: NONE.                                                  *
*                                                                   *
*EXTERNAL REFERENCES:                                               *
*PROCEDURES INVOKED: FSAPPL                                         *
*  PURPOSE: TO PERFORM PROCESSING AND BUILD 3270 DATA STREAMS.      *
*  LINKAGE: CALL                                                    *
*  INPUT: STANDARD NCCF COMMAND PROCESSOR LINKAGE                   *
*    REGISTERS: R0=CWB ADDRESS                                      *
*               R1=WORK AREA ADDRESS. MAPPED BY DSECT FSSAVE.       *
*                  IF PARAMETER 'XYZ' WAS SPECIFIED, PARMBIT IS SET.*
*               R13=SAVE AREA ADDRESS                               *
*               R14=RETURN ADDRESS                                  *
*               R15=ADDR OF FSAPPL                                  *
*    NOTE: FSAPPL WILL CALL WRITREAD TO PERFORM THE TERMINAL I/O    *
*          OF FULL SCREEN 3270 DATA STREAMS BUILT BY FSAPPL.        *
*          ADDRESSABILITY TO WRITREAD IS BY 'DC V(WRITREAD)' IN     *
*          IN FSAPPL.                                               *
*                                                                   *
*NCCF CONTROL BLOCKS: DSICWB  DSIMVT  DSIPDB  DSISVL  DSISWB       *
*                     DSITIB  DSITVB                                *
*NCCF MACROS: DSICBS  DSIGET  DSIFRE  DSIPAS  DSIPSS               *
*                                                                   *
*******************************************************************/
```

```
FULLSCR  CSECT ,                       /*********************************/
         DS    0H                      /*                               */
         USING *,R15                   /*            ENTRY              */
         B     PROLOG                  /*                               */
         DC    C'FULLSCR &SYSDATE.'    /*                               */
         DROP  R15                     /*                               */
PROLOG   STM   R14,R12,12(R13)         /*           LINKAGE             */
         BALR  R12,0                   /*                               */
PSTART   DS    0H                      /*                               */
         USING PSTART,R12              /*********************************/



         LR    R2,R1                   /* MOVE THE CWB BASE             */
         USING DSICWB,R2               /* BASE THE COMMAND WORK BLOCK   */
         LA    R1,CWBSAVEA             /* POINT TO FULLSCR SAVEAREA     */
         ST    R13,4(,R1)              /* SAVE CALLER-S SAVEAREA ADDRESS*/
         ST    R1,8(,R13)              /* SAVE FULLSCR SAVEAREA ADDRESS */
         LR    R13,R1                  /*                               */
         L     R4,CWBTIB               /* OBTAIN TIB ADDRESS            */
         USING DSITIB,R4               /* BASE TASK INFO BLOCK          */
         L     R10,TIBTVB              /* GET TVB ADDRESS               */
         ST    R10,0(,R13)             /* DEBUGGING AID: SAVEAREA -> TVB*/
         L     R5,TVBMVT-DSITVB(,R10) /* GET MVT BASE                   */
         USING DSIMVT,R5               /* BASE NCCF MAIN VECTOR TABLE   */
         LA    R11,CWBADATD            /* POINT TO AUTO DATA AREA       */
         USING DATD,R11                /* DECLARE BASE REG              */

*    /*****************************************************************/
*    /*                                                               */
*    /* COPY THE BUFFER HEADER PASSED. USE IT AS A PROTOTYPE.         */
*    /*                                                               */
*    /*****************************************************************/
*
         USING BUFHDR,R10              /* BASE BUFHDR TEMPORARILY       */
         L     R10,CWBBUF              /* INITIALIZE WORKING BUFHDR...  */
         MVC   BUFFER(24),BUFHDR       /*   FROM THE ONE PASSED         */
         LA    R10,BUFFER              /* CHANGE THE LENGTH             */
         MVC   HDRBLENG,BUFFERLN       /* TO THE REAL LENGTH            */
         MVC   HDRTDISP,LNBUFHDR       /* SET THE TEXT DISP TO ...      */
*                                      /*                  END OF BUFHDR*/
         MVI   HDRMTYPE,HDRTYPEU       /* SET TYPE TO USER GEN-ED MSG   */
         DROP  R10                     /* DROP BUFHDR COVER             */



*****************************************************************
*                                                               *
*   TEST TO SEE IF FSAPPL IS ALREADY RUNNING. IF SO, THE WORK AREA   *
*   ADDRESS WILL BE IN THE TIB USER FIELD (TIBUFLD).            *
*                                                               *
*****************************************************************
         L     R6,TIBUFLD              /*                               */
         LTR   R6,R6                   /* TEST FOR ZERO                 */
         BNZ   RUNNING                 /*                               */
```

```
*****************************************************************
*                                                               *
*                     SET UP ROUTINE                            *
*                                                               *
*****************************************************************
*
*       /*************************************************************/
*       /*                                                           */
*       /* GET WORK SPACE. STORE ADDRESS IN TIBUFLD.                 */
*       /*                                                           */
*       /*************************************************************/
*
         L     R10,GETSIZE          /* PUT LENGTH IN REGISTER        */
         L     R9,TIBTVB            /* POINT TO THE TVB              */
         DSIGET LV=(R10),           /* LENGTH IS IN REGISTER         */*
               A=TIBUFLD,           /* RETURN ADDRESS IN TIBUFLD     */*
               REENT=YES,           /* CALL IS REENTANT              */*
               LISTA=GETWORK,       /* DSIGET WORKAREA = GETWORK     */*
               Q=YES,               /* NCCF WILL KEEP TRACK OF AREA  */*
               TASKA=(R9)           /* TVB ADDRESS IS IN REG9        */
*
         LTR   R15,R15              /* WAS DSIGET SUCCESSFUL?        */
         BZ    GETOK                /* BRANCH IF YES                 */
         LA    R10,BUFFER
         USING BUFHDR,R10           /* BASE THE BUFHDR ON BUFFER     */
         MVC   HDRTEXT(42),MSG3     /* MOVE IN THE MESSAGE TEXT      */
         MVC   HDRMLENG,MSG3L       /* SET MESSAGE LENGTH IN BUFHDR  */
         DSIPSS TYPE=OUTPUT,        /* SEND MESSAGE TO THE TERMINAL  */*
               BFR=(R10),           /*   MESSAGE ADDRESS IS IN R10   */*
               SWB=CWBSWB           /*   USE THE SWB PASSED          */
*
         DROP  R10                  /* DROP BUFHDR COVER             */
*
         B     CMDXIT               /* RETURN TO NCCF                */
*
*       /*************************************************************/
*       /*                                                           */
*       /* MOVE FULLSCR SAVEAREA TO THE GETMAINED AREA               */
*       /*                                                           */
*       /*************************************************************/
*
GETOK    L     R6,TIBUFLD           /* COPY THE BACK POINTER         */
         USING FSSAVE,R6            /*                               */
         L     R3,4(,R13)           /*                               */
         ST    R3,SAVEREGS+4        /*                               */
         ST    R6,8(,R13)           /* MOVE FWD PTR TO NEW SAVEAREA  */
         LR    R13,R6               /* POINT TO NEW SAVEAREA         */
         ST    R2,SAVEREGS+28       /*                               */


*       /*************************************************************/
*       /*                                                           */
*       /* DO INPUT PARAMETER PROCESSING                             */
*       /*                                                           */
*       /*************************************************************/
*
         NI    PARMBYTE,X'FF'-PARMBIT /* SET PARAMETER BIT OFF       */
```

```
*
*         /*************************************************************/
*         /*                                                           */
*         /* IF THERE IS ONE OPERAND, AND IF IT IS 3 CHARACTERS AND    */
*         /* IF IT IS THE CHARACTERS 'XYZ' THEN SET THE PARMBIT.       */
*         /*                                                           */
*         /*************************************************************/
*
         L     R3,CWBPDB           /* POINT TO PARSE DESCRIP BLOCK  */
         USING DSIPDB,R3           /* SET UP TEMPORARY BASE REGISTER*/
         LH    R9,PDBNOENT         /* IF NUMBER OF PDB ENTRIES IS...*/
         C     R9,TWO              /*   LESS THAN 2, THE COMMAND... */
         BL    NOTXYZ              /*   HAS NO OPERANDS.            */
         MVI   CMDPARM,C' '        /* INITIALIZE WITH A BLANK       */
*                                  /*                               */
         DSIPAS PDB=(CWBPDB,'2'),  /* LOOK FOR FIRST OPERAND ALIAS  */*
               OUT=CMDPARM,        /*   PLACE OUTPUT IN CMDPARM     */*
               SWB=CWBSWB          /*   USE THE SWB PASSED          */
*                                  /*                               */
         CLC   CMDPARM,XYZ         /* IS IT 'XYZ'?                  */
         BNE   NOTXYZ              /* BRANCH IF NOT                 */
         OI    PARMBYTE,PARMBIT    /* SET FSAPPL PARAMETER BIT      */
         DROP  R3                  /* DROP PDB COVER                */

NOTXYZ   DS    0H


*        /*************************************************************/
*        /*                                                           */
*        /* PERFORM INITIAL CALL TO FSAPPL                            */
*        /*                                                           */
*        /*************************************************************/
*
         LR    R0,R2               /* MOVE CWB ADDRESS TO REG ZERO  */
         LR    R1,R6               /* POINT TO THE WORKAREA         */
         L     R15,FSADDR          /* LOAD THE ENTRY POINT ADDRESS  */
         BALR  R14,R15             /* CALL FSAPPL                   */
*
*        /*************************************************************/
*        /*                                                           */
*        /* FINAL CLEAN-UP -- WHEN RETURNING HERE, FSAPPL IS FINISHED */
*        /*                                                           */
*        /* ISSUE DSIFRE TO RELEASE THE WORK AREA POINTED TO BY       */
*        /* TIBUFLD AND R6.                                           */
*        /*                                                           */
*        /*************************************************************/
*
         LA    R13,CWBSAVEA        /* DONT POINT AT GETMAIN SAVEAREA*/
*                                  /* DSIFRE FREE WORKAREA HUNG ON  */
*                                  /*   TIBUFLD                     */
         L     R10,GETSIZE         /* PUT LENGTH IN REGISTER        */
         L     R9,TIBTVB           /* POINT TO THE TVB              */
         DSIFRE LV=(R10),          /* LENGTH IS IN REGISTER         */*
               A=TIBUFLD,          /* RETURN ADDRESS IN TIBUFLD     */*
               LISTA=GETWORK,      /* DSIGET WORKAREA = GETWORK     */*
               Q=YES,              /* NCCF WILL KEEP TRACK OF AREA  */*
               TASKA=(R9)          /* TVB ADDRESS IS IN R9          */
*
         XC    TIBUFLD,TIBUFLD     /* CLEAR USER FIELD              */
```

```
*
*       /*****************************************************************/
*       /*                                                               */
*       /* PUT OUT COMMAND COMPLETE MESSAGE                              */
*       /*                                                               */
*       /*****************************************************************/
*
         LA    R10,BUFFER
         USING BUFHDR,R10           /* BASE THE BUFHDR ON BUFFER     */
         MVC   HDRTEXT(28),MSG2     /* MOVE IN THE MESSAGE TEXT      */
         MVC   HDRMLENG,MSG2L       /* SET MESSAGE LENGTH IN BUFHDR  */
         DSIPSS TYPE=OUTPUT,        /* SEND MESSAGE TO THE TERMINAL  */*/
               BFR=(R10),           /*   MESSAGE ADDRESS IS IN R10   */*/
               SWB=CWBSWB           /*   USE THE SWB PASSED          */

         DROP  R10                  /* DROP BUFDHR COVER             */

         B     CMDXIT
************************* END SETUP ROUTINE ****************************



*/*****************************************************************/
*/*                                                               */
*/*  CMDXIT:  THE COMMAND COMMON EXIT POINT. RETURN TO NCCF.      */
*/*                                                               */
*/*****************************************************************/
*
CMDXIT   DS    0H                  /*                               */
         L     R13,CWBSAVEA+4      /* RESTORE R13 TO ORIG R13 VALUE */
         LM    R14,R12,12(R13)     /*                               */
         SLR   R15,R15             /* ALWAYS GIVE A GOOD RETURN CODE*/
         BR    R14                 /*                               */
```

```
*****************************************************************/
*                                                               */
*       ALREADY RUNNING FSAPPL. REINVOKED FOR FULL SCREEN INPUT.*/
*                                                               */
*****************************************************************/
*
RUNNING  DS    0H
*       /*************************************************************/
*       /*                                                           */
*       /* MOVE FULLSCR SAVEAREA TO THE GETMAINED AREA               */
*       /*                                                           */
*       /*************************************************************/
*
         L     R3,4(,R13)          /*                               */
         ST    R3,SAVEREGS+4       /*                               */
         ST    R6,8(,R13)          /* MOVE FWD PTR TO NEW SAVEAREA  */
         LR    R13,R6              /* POINT TO NEW SAVEAREA         */
         ST    R2,SAVEREGS+28      /* STORE THE NEW CWB ADDRESS IN  */
*                                  /* OLD SAVEAREA FOR REG 2        */


*         /*************************************************************/
*         /*                                                           */
*         /* IF THERE IS NO DATA (NOT EVEN THE 3270 AID), MUST BE A    */
*         /* RESHOW (REFRESH THE SCREEN) REQUEST.                      */
*         /*                                                           */
*         /*************************************************************/
*
         L     R3,CWBPDB           /* GET THE PDB ADDRESS           */
         USING DSIPDB,R3           /* BASE PARSE DESCRIPTOR BLOCK   */
         CLC   PDBNOENT,ONE        /* ONLY 1 PARSE ENTRY (VERB ONLY)*/
         BNE   INPUT               /* BRANCH IF MORE (CANT BE LESS) */
         DROP  R3                  /* DROP PDB COVER                */


*         /*************************************************************/
*         /*                                                           */
*         /* RESHOW ROUTINE                                            */
*         /*                                                           */
*         /*************************************************************/
*                                  /*                               */
         L     R3,SAVEINA          /* FAKE INPUT = CLEAR KEY        */
         MVC   0(3,R3),CLEARKEY    /* MOVE CLEAR, 40, 40 TO INPUT   */
         LA    R10,CLEARLEN        /* GET 'CLEARKEY' LENGTH         */
         STH   R10,INPUTLEN        /* PASS LENGTH BACK TO FSAPPL    */
*                                  /*                               */
         LR    R0,R2               /* HAVE TO RETURN THE CWB ADDRESS*/
         L     R13,WRSAVA+4        /* RESTORE FSAPPL S.A. POINTER   */
         SLR   R15,R15             /* SEND ZERO RETURN CODE         */
         L     R14,12(,R13)        /* RESTORE REG 14                */
         LM    R1,R12,24(R13)      /* RESTORE REGS 1 - 12           */
         BR    R14                 /* RETURN TO FSAPPL              */
*         /************** END RESHOW ROUTINE ************************/
```

```
*          /*****************************************************************/
*          /*                                                               */
*          /* NORMAL PANEL INPUT ROUTINE                                    */
*          /*                                                               */
*          /*****************************************************************/
*
INPUT     DS    0H
*
*            /*****************************************************************/
*            /*                                                               */
*            /* MOVE INPUT TO FSAPPL. DO NOT COPY THE VERB OR BLANK.          */
*            /* ONLY COPY THE 3270 INPUT DATA.                                */
*            /*                                                               */
*            /*****************************************************************/
*
          L     R7,CWBBUF             /* POINT TO THE INPUT BUFFER          */
          USING BUFHDR,R7             /* ESTABLISH BASE REG                 */
          LH    R9,HDRMLENG           /* GET THE INPUT DATA LENGTH          */
          SL    R9,NINE               /* DECREMENT BY 9 (8 FOR VERB,        */
*                                     /*   1 FOR BLANK)                     */
          STH   R9,INPUTLEN           /* PASS INPUT LENGTH TO FSAPPL        */
*                                     /*                                    */
          LH    R8,HDRTDISP           /* GET INPUT TEXT DISPLACEMENT        */
          ALR   R8,R7                 /* ADD TO THE BUFFER ADDRESS          */
*                                     /*                                    */
          L     R14,SAVEINA           /* LOAD THE INPUT AREA ADDRESS        */
          LR    R15,R9                /* COPY THE LENGTH OF THE DATA        */
          MVCL  R14,R8                /* MOVE THE DATA READ                 */
          DROP  R7                    /* DROP BUFHDR COVER                  */
*
          LR    R0,R2                 /* HAVE TO RETURN THE CWB ADDRESS*/
          L     R13,WRSAVA+4          /* RESTORE FSAPPL SAVEAREA PTR        */
          SLR   R15,R15               /* SEND ZERO RETURN CODE              */
          L     R14,12(,R13)          /* RESTORE REG 14                     */
          LM    R1,R12,24(R13)        /* RESTORE REGS 1 - 12                */
          BR    R14                   /* RETURN TO FSAPPL                   */
********************** END INPUT ROUTINE ********************************/
```

```
*/*****************************************************************/
*/*                                                               */
*/* WRITREAD - THE TERMINAL WRITE/READ INTERFACE                  */
*/*                                                               */
*/* FUNCTION: TO PERFORM FULL SCREEN I/O BY ISSUING THE NCCF MACRO */
*/* DSIPSS TYPE=PANEL.                                            */
*/*                                                               */
*/* INPUT:   R0 = CWB ADDRESS                                     */
*/*          R1 = PARAMETER LIST ADDRESS. THE PARMLIST FORMAT IS: */
*/*               +---------------+---------------+--------+-------+ */
*/*               |    PARMOUT    |    PARMIN     |PARMOUTL|PARMINL| */
*/*               +---------------+---------------+--------+-------+ */
*/*                0               4               8        10    11 */
*/*               PARMOUT  = OUTPUT BUFFER ADDRESS                */
*/*               PARMIN   = INPUT BUFFER ADDRESS                 */
*/*               PARMOUTL = OUTPUT BUFFER LENGTH                 */
*/*               PARMINL  = INPUT BUFFER LENGTH                  */
*/*          R13 = SAVE AREA ADDRESS                              */
*/*          R14 = RETURN ADDRESS                                 */
*/*          R15 = ENTRY POINT (WRITREAD) ADDRESS                 */
*/*                                                               */
*/* NOTES: (1) IF PARMOUT=0, NO OUTPUT IS DONE.                   */
*/*        (2) IF PARMIN=0, NO INPUT IS DONE.                     */
*/*        (3) IF DATA IS TO BE READ FROM THE TERMINAL, RETURN IS */
*/*            NOT MADE TO FSAPPL UNTIL THE DATA HAS BEEN READ.   */
*/*            THIS REQUIRES A RETURN TO NCCF AND THE RESUME COMMAND */
*/*            TO BE EXECUTED.                                    */
*/*        (4) ANY DATA IN THE OUTPUT BUFFER MUST BE A COMPLETE   */
*/*            3270 DATA STREAM INCLUDING THE COMMAND CODE, WCC   */
*/*            AND DATA.                                          */
*/*                                                               */
*/*****************************************************************/
```

```
WRITREAD DS    0H                  /*                            */
         ENTRY WRITREAD            /* TERMINAL I/O ENTRY POINT   */
*                                  /*                            */
         USING *,R15               /*                            */
         STM   R14,R12,12(R13)     /* SAVE REGS                  */
         L     R12,AENTRY          /* POINT TO MAIN CSECT ENTRY  */
         USING PSTART,R12          /*                            */
         DROP  R15                 /*                            */
         CNOP  0,4                 /* ALIGN TO A FULL WORD       */
         B     *+8                 /*                            */
AENTRY   DC    A(PSTART)           /* ADDR OF MAIN ENTRY         */
*                                  /*                            */
         LR    R2,R0               /* RE-ESTABLISH CWB COVERAGE  */
         LA    R11,CWBADATD        /* POINT TO THE AUTO AREA     */
         L     R6,TIBUFLD          /* POINT TO WRITREAD SAVEAREA */
         LA    R15,WRSAVA          /*                            */
         ST    R13,4(,R15)         /* CHAIN SAVE AREAS           */
         ST    R15,8(,R13)         /* CHAIN THE OTHER WAY        */
         LR    R13,R15             /* POINT TO SAVEAREA WITH R13 */
```

```
*    /*****************************************************************/
*    /*                                                               */
*    /* SET UP THE DSIPSS TYPE=PANEL PARAMETER LIST.                  */
*    /*                                                               */
*    /*****************************************************************/
*
*      /*************************************************************/
*      /*                                                           */
*      /* SET UP RESUME VERB. USE THE VERB FULLSCR WAS INVOKED WITH.*/
*      /*                                                           */
*      /*************************************************************/
*
         USING PARMLIST,R1          /* DECLARE BASE REGISTER         */
         MVC   PSSCMD,BLANKS        /* START BY BLANKING COMMAND VERB*/
         L     R10,CWBPDB           /* GET VERB FROM FIRST PDB ENTRY */
         USING DSIPDB,R10           /* BASE THE PDB                  */
         LA    R10,PDBTABLE         /* POINT TO THE FIRST ENTRY      */
         USING PDBENTRY,R10         /* BASE THE ENTRY TEMPORARILY    */
         SLR   R3,R3                /* CLEAR FOR THE 'IC'            */
         IC    R3,PDBLENG           /* GET THE LENGTH OF THE VERB    */
         BCTR  R3,0                 /* DECREMENT FOR EXECUTE         */
         LH    R9,PDBDISP           /* GET DISPLACEMENT TO VERB,...  */
         AL    R9,CWBBUF            /*  ADD INPUT BUFFER ADDRESS,... */
*                                   /*  TO POINT TO THE VERB.        */
         EX    R3,EXMOVE2           /* MOVE JUST THE VERB            */
         DROP  R10                  /* DROP PDB COVER                */
*
*      /*************************************************************/
*      /*                                                           */
*      /* SET UP REMAINDER OF THE PARAMETER LIST.                   */
*      /*                                                           */
*      /*************************************************************/
*
         MVC   PSSOUT,PARMOUT       /* MOVE THE OUTPUT BUFFER ADDR   */
         MVC   PSSOUTL,PARMOUTL     /* MOVE THE OUTPUT DATA LENGTH   */
         MVC   PSSINL,PARMINL       /* MOVE THE EXPECTED INPUT...    */
*                                   /*                       LENGTH  */
*
         L     R3,PARMIN            /* SAVE THE INPUT BUFFER ADDRESS */
         ST    R3,SAVEINA           /*   IN THE WORK AREA            */
         DROP  R1                   /* DROP PARMLIST COVER           */
*
*      /*************************************************************/
*      /*                                                           */
*      /* ISSUE DSIPSS TYPE = PANEL.                                */
*      /*                                                           */
*      /*************************************************************/
*
         LA    R10,PSSPARM          /* POINT TO PANEL PARAM LIST     */
         DSIPSS TYPE=PANEL,         /* REQUEST FULL SCREEN I/O       */*
               PANEL=(R10),         /* SPECIFY PARMLIST ADDRESS      */*
               SWB=CWBSWB           /* SPECIFY SWB ADDRESS           */
*
         LTR   R15,R15              /* TEST THE DSIPSS RETURN CODE   */
         BZ    PANELOK              /* IF ZERO (GOOD), BRANCH        */
```

```
*    /*****************************************************************/
*    /*                                                               */
*    /* DSIPSS HAS RETURNED A NONZERO RETURN CODE. NO ERROR ANALYSIS  */
*    /* IS DONE. A MESSAGE WILL BE ISSUED IN STANDARD NCCF MODE AND   */
*    /* THEN FINAL EXIT IS TAKEN.                                     */
*    /*                                                               */
*    /*****************************************************************/
*
         LA    R10,BUFFER           /* SET UP BUFFER COVER           */
         USING BUFHDR,R10           /* BASE THE BUFHDR               */
         MVC   HDRTEXT(41),MSG1     /* MOVE IN THE ERROR MSG TEXT    */
         MVC   HDRMLENG,MSG1L       /* SET MSG LENGTH                */
         ST    R15,ONEWORD          /* PUT RETURN CODE INTO STORAGE  */
         UNPK  UNPACK1,FIVEBYTE     /* UNPACK THE RETURN CODE        */
         TR    UNPACK1(8),HEXTAB    /* CONVERT TO PRINTABLE HEX      */
         MVC   HDRTEXT+22(8),UNPACK1 /* MOVE IN INSERT               */
*
         DSIPSS TYPE=OUTPUT,        /* REQUEST SINGLE LINE OUTPUT    */*/
               BFR=(R10),           /* TEXT IS IN BUFFER             */*/
               SWB=CWBSWB           /* USE THIS SWB                  */
*
         DROP  R10                  /* DROP BUFHDR COVER             */
*
*        /*****************************************************************/
*        /* DSIFRE FREE WORKAREA HUNG ON TIBUFLD                          */
*        /*****************************************************************/
*
         LA    R13,CWBSAVEA         /* MOVE SAVEAREA BACK TO THE CWB */
*
         L     R10,GETSIZE          /* PUT LENGTH IN REGISTER        */
         L     R9,TIBTVB            /* POINT TO THE TVB              */
         DSIFRE LV=(R10),           /* LENGTH IS IN REGISTER         */*/
               A=TIBUFLD,           /* RETURN ADDRESS IN TIBUFLD     */*/
               LISTA=GETWORK,       /* DSIGET WORKAREA = GETWORK     */*/
               Q=YES,               /* NCCF WILL KEEP TRACK OF AREA  */*/
               TASKA=(R9)           /* TVB ADDRESS IS IN R9          */
*
         SLR   R3,R3                /* CLEAR USER FIELD SO THAT...   */
         ST    R3,TIBUFLD           /* NEXT TIME WILL BE FIRST TIME  */
         B     CMDXIT               /* EXIT THE COMMAND PROCESSOR    */


*    /*****************************************************************/
*    /*                                                               */
*    /* DSIPSS TYPE=PANEL WAS SUCCESSFUL. PROCEED.                    */
*    /*                                                               */
*    /*****************************************************************/
*
PANELOK  DS    0H                   /*                               */
         LH    R10,PSSINL           /* WAS A READ REQUESTED? TEST... */
         LTR   R10,R10              /* THE INPUT AREA LENGTH         */
         BNZ   CMDXIT               /* EXIT TO NCCF IF READ WAS...   */
*                                   /* REQUESTED. COMMAND WILL BE... */
*                                   /* REDRIVEN.                     */
*                                   /*                               */
*                                   /* REQUEST WAS WRITE ONLY        */
         STH   R10,INPUTLEN         /* ZERO NUMBER OF BYTES READ.    */
*                                   /*                               */
         L     R13,4(,R13)          /* RESTORE CALLER-S SAVEAREA     */
         LM    R14,R12,12(R13)      /* RESTORE REGS                  */
         BR    R14                  /* RETURN TO CALLER (WRITE ONLY) */
*
*  /* ******************  END WRITREAD ***************************** */
```

```
*   /*****************************************************************/
*   /*                                                               */
*   /*                        CONSTANTS                              */
*   /*                                                               */
*   /*****************************************************************/
*
DATA      DS    0H                    /*                            */
ONE       DC    H'1'                  /* CONSTANT '1'               */
LNBUFHDR  DC    H'24'                 /* LENGTH OF BUFHDR           */
BUFFERLN  DC    AL2(BUFFERND-BUFFER)  /* LENGTH OF 'BUFFER'         */
EXMOVE1   MVC   0(0,R3),9(R7)         /* EXECUTED MOVE              */
EXMOVE2   MVC   PSSCMD(0),0(R9)       /* EXECUTED MOVE              */
          DS    0F                    /*                            */
TWO       DC    F'2'                  /* CONSTANT '2'               */
NINE      DC    F'9'                  /* CONSTANT '9'               */
GETSIZE   DC    F'10000'              /* GETMAINED WORK SPACE SIZE  */
*
          NOPR  ((ENDDATD-DATD)/257*16) /* GUARANTEE THAT CWB WORK... */
*                                       /* SPACE DOES NOT GROW OVER 256 */
*                                       /* IF TOO BIG, WILL GET ASM ERROR*/
          ORG   *-2                     /* NOPR NOT REALLY NEEDED.    */
*
FSADDR    DC    V(FSAPPL)               /* ADDRESS OF FULL SCREEN APPL */
XYZ       DC    CL8'XYZ'                /* CONSTANT FOR OPERAND CHECK  */
CLEARKEY  DC    X'6D4040'               /* CLEAR AID, ROW 0, COLUMN 0  */
CLEARLEN  EQU   *-CLEARKEY              /* LENGTH OF 'CLEARKEY'        */
BLANKS    DC    CL8'        '           /* BLANKS                      */
MSG1      DC    CL41'FULLSCR: RETURN CODE XXXXXXXX FROM DSIPSS'
MSG1L     DC    AL2(*-MSG1)             /* LENGTH OF MSG1              */
MSG2      DC    CL28'FULLSCR: PROCESSING COMPLETE'
MSG2L     DC    AL2(*-MSG2)             /* LENGTH OF MSG2              */
MSG3      DC    CL42'FULLSCR: DSIGET FAILED. COMMAND TERMINATED'
MSG3L     DC    AL2(*-MSG3)             /* LENGTH OF MSG3              */
HEXTAB    EQU   *-240                   /* PRINTABLE HEX CONVERSION TABLE*/
          DC    C'0123456789ABCDEF'     /*                             */
```

```
*   /*****************************************************************/
*   /*                                                               */
*   /*                          DSECTS                               */
*   /*                                                               */
*   /*****************************************************************/
*
*                                    /*********************************/
DATD     DSECT                       /* CWB WORK SPACE DSECT          */
GETWORK  DS    CL16                  /* DSIGET/DSIFRE PARMLIST        */
*                                    /*                               */
BUFFER   DS    CL156                 /* WORK BUFFER                   */
BUFFERND EQU   *                     /* END OF 'BUFFER'               */
*                                    /*                               */
         DS    0F                    /* FULL WORD ALIGN               */
PSSPARM  DS    0CL16                 /* DSIPSS TYPE=PANEL PARMLIST    */
PSSCMD   DS    CL8                   /*   VERB                        */
PSSOUT   DS    AL4                   /*   OUTPUT BUFFER ADDRESS       */
PSSOUTL  DS    FL2                   /*   OUTPUT BUFFER LENGTH        */
PSSINL   DS    FL2                   /*   INPUT DATA LENGTH           */
*                                    /*                               */
*                                    /* MISC. SCRATCH VARIABLES       */
         DS    0F                    /* FULL WORD ALIGN               */
FIVEBYTE DS    0CL5                  /* 5 BYTES USED FOR UNPACK/TR    */
ONEWORD  DS    CL4                   /* FIRST 4 BYTES OF FIVEBYTE     */
UNPACK1  DS    CL9                   /* 9 BYTES USED FOR UNPACK/TR    */
CMDPARM  DS    CL8                   /* COMMAND PARAMETER FROM DSIPAS */
ENDDATD  EQU   *                     /* END OF CWB WORK AREA DSECT    */
*                                    /*********************************/


*                                    /*********************************/
PARMLIST DSECT                       /* MAP FOR WRITREAD PARMLIST     */
PARMOUT  DS    A                     /*   OUTPUT AREA ADDRESS         */
PARMIN   DS    A                     /*   INPUT AREA ADDRESS          */
PARMOUTL DS    H                     /*   OUTPUT AREA LENGTH          */
PARMINL  DS    H                     /*   INPUT AREA LENGTH           */
*                                    /*********************************/


*                                    /*********************************/
FSSAVE   DSECT                       /* FSAPPL WORK AREA DSECT        */
SAVEREGS DS    18F                   /* SAVE AREA                     */
WRSAVA   DS    18F                   /* SAVE AREA FOR CALLING WRITREAD*/
SAVEINA  DS    A                     /* INPUT AREA ADDR SAVE AREA     */
INPUTLEN DS    H                     /* NUMBER OF BYTES READ          */
PARMBYTE DS    X                     /* PARM FLAG BYTE                */
PARMBIT  EQU   X'01'                 /* FSAPPL PARAMETER BIT          */
FSWORK   EQU   *                     /* REMAINDER OF SPACE FOR USE    */
*                                    /*   BY FSAPPL.                  */
*                                    /*********************************/
```

Appendix G. Glossary of Terms and Abbreviations

This glossary defines terms and abbreviations that are important in Network Communications Control Facility (NCCF) publications. It does not include terms previously established for IBM operating systems and for products used with NCCF. Additional terms can be found by referring to the index, to prerequisite and corequisite books, and to the *IBM Data Processing Glossary,* GC20-1699.

ACF/TCAM. Advanced Communications Function for the Telecommunications Access Method.

ACF/VTAM. Advanced Communications Function for the Virtual Telecommunications Access Method.

ACF/VTAME. Advanced Communications Function for the Virtual Telecommunications Access Method Entry.

alphameric. Pertaining to a character set that contains letters, digits, and usually other characters, such as punctuation marks.

AMH Application message handler.

application message handler (AMH). In ACF/TCAM, a routine that routes messages between application programs or between an application program and a device message handler. NCCF in ACF/TCAM is supplied with an AMH. See also *device message handler.*

application program. (1) A program written for or by a user that applies to a particular application. (2) In data communication, a program used to connect and communicate with terminals in a network, enabling users to perform application-oriented activities.

authorization message. An NCCF message that is directed to an authorized operator. An example is a message about the use of NCCF, such as a successful logon, repeated unsuccessful logons, logon rejected because of invalid password, a DSM error message, and logoff.

authorized operator. In NCCF, an operator who has been authorized to receive undeliverable messages, authorization messages, and lost terminal messages. Authorization is specified on the AUTH statement during NCCF definition.

both regular and immediate command. A NCCF command that may be executed as either a regular or an immediate command, depending on where it is encountered. If the command is received from an operator terminal, it is executed as an immediate command. If it is received in another way (for example, in a command list), it is executed as a regular command.

CNM. Communications network management.

command. A request from a terminal for the performance of an operation or the execution of a particular program. A command may be entered from a terminal by an operator, or generated from a command list, or implied in a received message, or issued by a command processor.

command list. A sequential list of commands and/or control statements that is assigned a name. When the name is invoked (as a command) the commands in the list are executed.

command processor. A problem program executed to perform an operation specified by a command.

communication controller. A type of communication control unit whose operations are controlled by a program stored and executed in the unit. Examples are the IBM 3704 and 3705 Communication Controllers.

communication network management (CNM). The process of designing, installing, operating, and managing the distribution of information and control among end users of communication systems.

communication network management application. A combination of the components and elements that comprise the problem determination, operational facilities, and performance functions of CNM. An example is NCCF with added CNM processors.

communication network management interface. The interface provided to application programs by the access method for handling data and commands associated with communication network management. CNM data and commands are handled across this interface.

communication network management processor. A command processor that manages one of the functions of a communication network management application. A CNM processor is executed under control of NCCF and requires NCCF as a prerequisite program.

conditional command list. An NCCF command list consisting of control statements and variables that control the sequence of execution of the command list.

control statement. In NCCF, a statement in a command list that controls the processing sequence of the command list or allows the command list to send messages to the operator and receive input from the operator.

cross-domain communication. In a multiple-domain network, communication between domains.

cursor. A movable spot of light on the screen of a display device, usually indicating where the next character will be entered.

data services command processor (DSCP). An NCCF component that structures the request for recording and retrieving data in the application program's data base, and also structures the request to solicit data from a network device.

data services manager (DSM). A function in NCCF that provides VSAM services for data storage and retrieval and provides the interface between DSCPs and the CNM interface.

data services request block (DSRB). The NCCF control block that allows communication between the data services task and a data services command processor.

data services task (DST). The NCCF subtask that provides support to gather, record, and manage data in a VSAM file that contains communication network management information.

device message handler. In ACF/TCAM, a user-defined routine that routes messages between a device and an application message handler, or between devices. See also *message handler.* Contrast with *application message handler.*

domain. In a data communication system, the portion of the total network that is controlled by the SSCP in one telecommunication access method.

DSCP. Data services command processor.

DSM. Data services manager.

DSRB. Data services request block.

DST. Data services task.

exit routine. Any of several types of special-purpose routines that handle processing for certain conditions in a program. NCCF provides for user-written exit routines. NCCF has its own data communication access method exit routines an system exit routines.

full-line mode. A form of screen presentation in NCCF where the message area of the terminal screen consists of 80-byte messages. Full-line mode is used by NPDA. Contrast with *standard NCCF mode.*

full-screen mode. A form of screen presentation in NCCF where the contents of an entire terminal screen can be displayed at once. Full-screen mode is often used for fill-in-the-blanks prompting.

hard-copy log. In NCCF, a file written on a hard-copy device (such as a printer) that contains a record of all messages passing through NCCF that are associated with a specific operator or operators.

hard-copy task (HCT). The NCCF subtask that controls the passage of data between NCCF and the hard-copy log device.

HCT. Hard-copy task.

immediate command. In NCCF, a command (such as GO, CANCEL, or RESET) that can be executed while a regular command is being processed.

log. A collection of messages or message segments placed on a secondary storage device for accounting or data collection.

logger. In NCCF, a subtask that records errors from EP mode and local mode devices to the EP data base and transmits errors from NCP mode devices supported by ACF/VTAM and ACF/TCAM to the NCP data base.

message. In telecommunications, a combination of characters and symbols transmitted from one point to another.

message handler (MH). Under ACF/TCAM, a sequence of user-specified macro instructions that examine and process control information in message headers, and perform the functions necessary to prepare message segments for forwarding to their destinations. One message handler is required for each line group having unique message-handling requirements.

MH. Message handler.

MSNF. Multisystem Networking Facility.

Multisystem Networking Facility (MSNF). An optional feature of ACF/VTAM and ACF/TCAM that permits these access methods, together with ACF/NCP/VS, to control a multiple-domain network.

NCCF. Network Communications Control Facility.

NCP. Network Control Program

Network Communications Control Facility (NCCF). A program product consisting of a base for command processors that can monitor, control, and improve the operation of a data communication network.

Network Control Program (NCP). A program, generated by the user from a library of IBM-supported modules, that controls the operation of the communication controller.

network operator. In SNA, a person or program responsible for controlling the operation of all or part of a network.

Network Problem Determination Application (NPDA). A program product that assists the user in identifying communication network problems from a central control point using interactive display techniques. NCCF is required fo NPDA.

NPDA. Network Problem Determination Application.

network resource. Any named entry known to the access method. Network resources include network control programs (NCPs), local and remote terminals, lines, application programs, cross-domain resource tables, and cross-domain resource managers.

operand. Information entered with a command name to define the data on which a command processor operates and to control the execution of the command processor.

operator. See network operator.

operator control. The ACF/TCAM facility that allows users to enter ACF/TCAM operator control commands to examine or alter the status of the communication network. Operator control commands may be entered from an authorized station on a nonswitched link, from the system console, or from an application program.

operator station. A control point in NCCF that links a terminal, an operator, and the control environment assigned to the operator (such as profile and span of control). The logical unit from which an operator logged on.

operator station task (OST). The NCCF subtask that establishes and maintains the online session with the network operator. There is one operator station task for each network operator who logs on to NCCF.

optional subtask. A user-defined subtask specified on the TASK definition statement.

OST. Operator station task.

overlapped span of control. A condition that exists when the network resource name appears in a span or spans associated with more than one active network operator. Under such a condition, either operator may control the resource. The status of the device depends on the cumulative effect of commands entered and the sequence in which the commands are received by the access method.

password. (1) A unique string of characters that a program, computer operator, or user must supply to meet security requirements before gaining access to data. (2) In systems with time sharing, a 1- to 8-character symbol that the user may be required to supply at the time he logs on the system. The password is confidential, as opposed to the user identification.

POI. ACF/VTAM's program operator interface.

PPT. Primary POI task.

presentation services command processor (PSCP). An NCCF component that processes requests from a user terminal and formats displays to be presented at the user terminal.

primary POI task (PPT). The NCCF subtask that processes all unsolicited messages received from the ACF/VTAM program operator interface (POI) and either delivers them to the controlling operator or command processor. The primary POI task also processes the initial command specified to execute when NCCF is initialized, and timer request commands scheduled to execute under the PPT.

profile. In NCCF, a record that describes the control available to a particular network operator. The profile includes the operator's span of control, the name of the terminal to be used as a hard-copy device, whether the operator is authorized (see *authorized operator*), and (optionally) the name of a command or command list that is executed immediately after logon is successfully completed.

program operator. An ACF/VTAM application program that is authorized to issue ACF/VTAM operator commands and receive ACF/VTAM operator messages.

PSCP. Presentation services command processor.

regular command. Any access method or NCCF command that is not an immediate command and is processed by a regular command processor. Only one regular command may be executed at one time; regular commands issued while other regular commands are being processed are stacked. Contrast with *immediate command.*

resource. See *network resource.*

response. (1) An answer to an inquiry. (2) The unit of information that is exchanged between the access method or an application program and an SNA terminal to describe how a request arrived.

routing qualifier. An explicit parameter added to commands in NCCF to accommodate cross-domain execution. NCCF removes the routing qualifier before the command is passed to the appropriate access method.

scope of commands. An NCCF facility that allows restriction of NCCF commands and operands to a subset of all NCCF operators in the network.

SNA. Systems Network Architecture.

span. In NCCF, a user-defined group of network resources within a single domain. Each major or minor node is defined as belonging to one or more spans. See also *span of control*

span of control. The total network resources over which a particular network operator has control. All the network resources listed in spans associated through profile definition with a particular network operator are within that operator's span of control.

standard NCCF mode. A form of screen presentation in NCCF where the message area of the terminal screen consists of 69 bytes for each message and an 11 byte previx. Contrast with *full-line mode.*

station. (1) One of the input or output points of a system that uses communication facilities; for example, the telephone set in the telephone system or the point where the business machine interfaces with the channel on a leased private line. (2) One or more computers, terminals or devices at a particular location.

suppression character. In NCCF, a user-defined character that is coded at the beginning of a command list statement or a command to prevent the statement or command from appearing on the operator's terminal screen, the hard-copy log, and the NCCF log.

TCAM control task (TCT). The NCCF subtask that controls communication between NCCF and ACF/TCAM.

TCT. ACF/TCAM control task.

terminal. A device, often equipped with a keyboard and some kind of display, capable of sending and receiving information over a communication link.

timer initiation. An NCCF facility that allows the operator to schedule a command or command list to be executed based on a timer, either at a specific time or repetitively at specified time intervals.

timer request. A command or command list scheduled to execute either at a specific time or repetitively at specified time intervals.

unsolicited message routing. (1) A method of routing replies to CNM application programs by using a routing table instead of a process request identifier. (2) A method of routing access method messages to an NCCF operator by using the PPT.

variable. In NCCF, a character string beginning with & that is coded in a command list and is assigned a value during execution of the command list.

VSAM. Virtual Storage Access Method.

VSE/OCCF. VSE/Operator Communication Control Facility.

VSE/Operator Communication Control Facility (VSE/OCCF). A program product designed to run with the VSE operating system and NCCF. VSE/OCCF minimizes required operator interaction with the VSE system console by intercepting messages from VSE and application programs and responding automatically with pre-coded actions.

Index

&BEGWRITE 2-12
&CONCAT 2-16
&CONTROL 2-12
&EXIT 2-13
&GOTO 2-13
&IF 2-14
&LENGTH 2-17
&NCCFID 2-18
&NCCFSTAT 2-18
&PAUSE 2-15
&SUBSTR 2-17
&THEN 2-14
&WRITE 2-16

A operand
  DSIFRE macro 3-23
  DSIGET macro 3-24
alias, command operand 3-36
alphanumeric G-1
application message handler (AMH) G-1
application program G-1
APPLID operand, DSIPSS macro 3-40
AREA operand, DSIDATIM macro 3-20
ARTPOS operand, DSIRDS macro 3-47
assignment statements
  in command lists 2-11
asynchronous full-screen command processors 4-21
ASYPANEL operand, DSIPSS macro 3-42
authorization and resource table (*see* DSIART)
authorization message G-1
authorized operator
  definition of G-1
  locating (*see* DSILCS macro)
AUTHRCV operand
  DSILCS macro 3-28
  DSIMQS macro 3-35

BFR operand
  DSICES macro 3-19
  DSIMBS macro 3-31
  DSIMQS macro 3-35
  DSIPRS macro 3-38
  DSIPSS macro 3-43
  DSIWCS macro 3-50
  DSIWLS macro 3-50
BNDRY operand, DSIGET macro 3-25
"both" command processor 4-4
buffer, command
  obtaining 4-11
buffer header (BUFHDR)
  example of use 3-9
  fields in 3-7
  format of 3-8
  in command processors 4-11
  listed in DSITIB C-42
BUFHDR (*see* buffer header)

CANCEL operand, DSIPSS macro 3-42
CBADDR operand, DSILCS macro 3-28
cbname operand, DSICBS macro 3-18
CMD operand, DSIKVS macro 3-26
coding guidelines
  command list 2-6
  command processor 4-1
  exit routine 5-13
  NCCF, generally 1-1
  subtask 6-3
command analysis (*see* DSICES macro)
command list
  assignment statements 2-11
  coding guidelines 2-6
  commands 2-10
  comments 2-10
  control statements 2-11
  control variables 2-8
  defining to NCCF 2-2
  examples of 2-21–2-27
  filing 2-2
  invoking
    from an access method message 2-4
    from an operator terminal 2-3
    from another command list 2-3
    from a user-written command processor 2-3
  labels 2-9
  naming 2-2
  null statements 2-10
  parameters 2-7
  PPT restrictions on 2-5
  source compatibility with NCCF Release 1 1-1
  suppression character in 2-6
  user variables 2-9
  variables
    control variables 2-8
    definition of G-4
    in general 2-6
    parameters 2-7
    user variables 2-9
command processors, "both" 4-4
command processors, data services
  definition of G-2
  example of processing logic 4-14
  in general 4-5
  initial revocation of 4-13
  restrictions for 4-5
  sample CNM data E-12
  sample VSAM data E-1
  use in program design 4-6
command processors, generally
  address of command 4-12
  both regular and immediate commands 4-4
  calling 4-12
  coding guidelines for 4-1
  control block considerations for 4-6, 4-7
  definition of G-1
  example of
    data services E-1, E-12
    regular D-1
  executed under DSIPPT 4-4
  executed under DST (*see* command processors, data services)
  full-line 4-19
  full-screen 4-29
  immediate commands 4-3
  invoking 4-1, 4-11
  looking up address of 4-12
  operating environment 4-2
  PPT restrictions on 4-4
  register usage for 4-2
  regular commands 4-2
  return codes for 4-15–4-19
  source compatibility with NCCF Release 1 1-1

command processors, full-line 4-19
command processors, full-screen
  asynchronous 4-21
  coding guidelines 4-1, 4-20
  DSIPSS return codes in 4-25
  escaping from 4-25
  in general 4-20, 4-24
  logging input and output 4-25
  RESHOW key 4-25
  reshow option 4-25
  sample F-1
  screen formatting in 4-24
  suspending 4-25
  synchronous 4-23
command processors, immediate
  how called 4-3, 4-4
  in general 4-3
  return codes for 4-16
command processors, presentation services
  definition of G-3
  relation to data services 4-6
command processors, regular
  how called 4-2, 4-3
  in general 4-2
  return codes for 4-15
command work block (*see* DSICWB)
commands
  building a buffer for 4-11 (*see also* buffer header)
  definition of G-1
  forwarding to another domain 4-13
  immediate G-2
  in command lists 2-8
  parsing 3-38, 4-11
  passing to access method 4-15
  passing to another subtask in same domain 4-13
  regular G-3
  returning to another domain 4-15
  summary of A-2
comments in command lists 2-10
communication network management (CNM)
  completion of I/O request 4-17
  completion of receipt of unsolicited data 4-18
  definition of G-1
communication network management application G-1
communication network management interface
  definition of G-1
  requesting data from (*see* DSIZCSMS macro)
communication network management
  processor G-1
compcode operand, DSIPOS macro 3-37
conditional command list 2-1
CONN operand, DSIDKS macro 3-21
console, system operator
  writing to (*see* DSIWCS macro)
control block header (*see* DSICBH)
control blocks, NCCF
  descriptions of C-1
  including (*see* DSICBS macro)
  in command processors 4-6, 4-7
  in exit routines 5-14
  in subtasks 6-9
  listing of C-1, C-63
  locating (*see* DSILCS macro)
  overview of 3-5
  printing (*see* DSICBS macro)
  used to invoke service routines 3-4
cross-domain communication G-1
CWB control block (*see* DSICWB)
CWB operand, DSILCS macro 3-28
data services, program design example of 4-6
data services command processor (DSCP) (*see* command processor, data services)
data services macro instruction (*see* DSIZCSMS macro and DSIZVSMS macro)
data services manager G-2
data services request block (*see* DSIDSRB)
data services task (DST) G-2
DATAREA operand, DSIZVSMS macro 3-54
date, obtaining (*see* DSIDATIM macro)
DCB operand, DSILOD macro 3-30
DEFER operand, DSICBS macro 3-18
defining command lists to NCCF 2-2
DEST operand, DSIZCSMS macro 3-52
device message handler (DMH) G-2
DISC Operand, DSIDKS macro 3-22
disk services (*see* DSIDKS macro)
DPR Operand, DSIWAT macro 3-49
DSB control block (*see* DSIDSB)
DSBWORD operand, DSIDKS macro 3-21
DSCP (*see* command processor, data services)
DSIART (authorization and resource table)
  relationship to DSIOIT and DSISNT 3-17
  search of (*see* DSIRDS macro)
DSICBH (control block header)
  fields in 3-9
  listing of C-2
DSICBS macro
  explanation of 3-6
  in general 3-18
  overview of 3-2
DSICES macro
  in general 3-19, 4-12
  overview of 3-2
  return codes for 3-19
DSICWB (command work block)
  fields in 4-6
  freeing (*see* DSILCS macro)
  listing of C-4
  obtaining 4-11 (*see also* DSILCS macro)
DSIDATIM macro
  in general 3-20
  overview of 3-2
DSIDEL macro
  in general 3-20
  overview of 3-2
  return codes for 3-21
DSIDKS macro
  explanation of 3-12
  in general 3-21
  overview of 3-2
  return codes for 3-22
DSIDSB (data service block) C-6
DSIDSRB (data services request block)
  definition of G-2
  fields in 4-9
  listing of C-7
  return codes for CNM I/O request 4-17
  return codes for unsolicited CNM data 4-18
  return codes for VSAM services 4-16
DSIEX01 exit routine 5-4
DSIEX02 exit routine 5-5
DSIEX03 exit routine 5-5
DSIEX04 exit routine 5-6
DSIEX05 exit routine 5-6
DSIEX06 exit routine 5-6
DSIEX07 exit routine 5-7
DSIEX08 exit routine 5-7
DSIEX09 exit routine 5-7

DSIEX10 exit routine 5-8
DSIEX11 exit routine 5-8
DSIEX12 exit routine 5-8
DSIEX13 exit routine 5-9
DSIEX14 exit routine 5-9
DSIEX15 exit routine 5-9
DSIFRE macro
  explanation of 3-11
  in general 3-23
  overview of 3-2
  return codes for 3-24
DSIGET macro
  explanation of 3-11
  in general 3-24
  overview of 3-2
  return codes for 3-26
DSIIFR (internal function request)
  explanation of 3-9
  listing of C-11
DSIKVS macro
  in general 3-26
  overview of 3-2
  return codes for 3-27
DSILCS macro
  explanation of 3-12
  in general 3-27
  overview of 3-2
  return codes for 3-29
DSILOD macro
  in general 3-29
  overview of 3-2
  return codes for 3-30
DSILOGDS
  format of B-2
  in listing C-15
DSIMBS macro
  in general 3-30
  overview of 3-3
  return codes for 3-32
DSIMDS macro
  end message format 3-34
  in general 3-33
  message text format 3-33
  overview of 3-3
  start message format 3-33
DSIMQS macro
  explanation of 3-13
  in general 3-34
  overview of 3-3
  return codes for 3-35
DSIMVT (main vector table)
  establishing addressability in 3-1
  field sin 6-9
  listing of C-16
  requirement of addressability to 3-1
DSIOIS
  examples of 3-17
  in general 3-36
  overview of 3-3
  return codes for 3-36
DSIOIT (operator identification table), relationship to DSIART and DSISNT 3-17
DSIPAS macro
  in general 3-36
  overview of 3-3
  return codes for 3-37
DSIPDB (parse descriptor block)
  creating (*see* DSIPRS)
  fields in 3-10
  listing of C-26
  obtaining 4-11
  using 4-11
DSIPOS macro
  in general 3-37
  overview of 3-3
DSIPPT, restriction on command processors 4-4
DSIPRS macro
  in general 3-38
  overview of 3-3
  return codes for 3-39
DSIPRT (print utility) B-1
DSIPSS macro
  ECB post codes for 3-46
  examples of 3-15
  explanation of 3-14
  in general 3-39
  output from full-screen command processor 4-25
  overview of 3-3
  return codes for 3-45, 3-46
  use with a full-screen command processor 4-25
DSIRDS macro
  explanation of 3-16
  examples of 3-16, 3-17
  in general 3-46
  overview of 3-3
  return codes for 3-47
DSISCE (system command entry)
  fields in 4-9
  in general 3-19, 4-12
  listing of C-27
DSISCT (system command table) 4-12
DSISNT (span name table)
  contents of 3-48
  relationship to DSIART and DSIOIT 3-17
  search of (*see* DSISSS macro)
DSISSS macro
  example of 3-17
  in general 3-47
  overview of 3-3
  return codes for 3-48
DSISWB (service work block)
  explanation of 3-6
  freeing (*see* DSILCS macro)
  listing of C-28
  obtaining 4-11 (*see also* DSILCS macro) 5-17
DSITDSOL (sample CNM data command processor) E-12
DSITDSRD (sample VSAM services command processor) E-1
DSITIB (task information block) C-42
  fields in 6-13
DSITRE 5-11
DSITVB (task vector block)
  explanation of 3-7
  fields in 6-11
  listin of C-57
DSIUSE (user exit parameter list)
  contents of 5-15
  listing of C-62
DSIUSP (sample command processor) D-1
DSIWAT macro
  example of 3-49
  in general 3-49
  overview of 3-3
DSIWCS macro
  explanation of 3-13
  in general 3-50
  overview of 3-3
DSIWLS macro
  explanation of 3-13
  in general 3-50

overview of 3-3
return codes for 3-50
DSIZCSMS macro
in general 3-51
overview of 3-4
return codes for 3-52, 4-18
DSIZVSMS macro
completion of request by 4-16
in general 3-53
overview of 3-4
return codes for 3-54, 4-17
DSRB control block (*see* DSIDSRB)
DSRB operand
DSIZCMS macro 3-51
DSIZVSMS macro 3-54

E operand, DSIFRE macro 3-23
ecbaddress operand, DSIPOS macro 3-37
ECB operand, DSIWAT macro 3-49
ECBLIST operand
DSIPSS macro 3-42
DSIWAT macro 3-49
entry linkage
in command processor 4-1
in exit routine 5-1
in subtask 6-3
EJECT operand, DSICBS macro 3-18
EP operand
DSIDEL macro 3-20
DSILOD macro 3-29
EPLOC operand
DSIDEL macro 3-21
DSILOD macro 3-29
Escape key 4-25
EXIT operand
DSIFRE macro 3-24
DSIGET macro 3-25
exit routines
coding guidelines 5-13
control block considerations for 5-14
data services (XIT–) 5-1, 5-10, 5-11
definition of G-2
DSIEX01–DSIEX15 5-4–5-9
environment of 5-3
examples of
DSIEX01 example 5-20
exit routine prototype 5-18
in general 5-1
input parameters 5-14
installation 5-13
interfaces for 5-2
output parameters 5-17
overview of 5-1–5-4
parameter list for (DSIUSE) 5-15
registers
on input 5-14
on output 5-17
return codes set 5-18

FIND operand, DSIDKS macro 3-22
FIRST operand, DSIPRS macro 3-39
FORMAT operand, DSIDATIM macro 3-20
freeing a control block (*see* DSILCS)
freeing storage (*see* DSIFRE)
full-line mode 3-14
full-line title-line output 4-19
full-screen command processors (*see* command processors, full-screen)
full-screen mode 3-14
FUNC operand, DSIZVSMS macro 3-54

getting a control block (*see* DSILCS)
getting storage (*see* DSIGET)

hard-copy task (HCT) G-2
header
buffer (*see* buffer header)
control block (*see* DSICBH)

IFR control block (*see* DSIIFR)
IMMED operand, DSIPSS macro 3-41
immediate command processor 2-44, 4-3
INPUT operand, DSIZCSMS macro 3-51
including a control block (*see* DSICBS)
internal function request (*see* DSIIFR)

KEY operand, DSIZVSMS macro 3-54
KEYLEN operand, DSIZVSMS macro 3-54
KEYWORD operand, DSIKVS macro 3-26

LENGTH operand, DSIZCSMS macro 3-52
LIST command 6-5
LISTA operand
DSIFRE macro 3-23
DSIGET macro 3-25
DSILOD macro 3-29
loading (*see* DSILOD macro)
locating a control block (*see* DSILCS)
log, hard-copy
definition of G-2
in general B-1
sample printout of B-5
log, NCCF
definition of G-2
in general B-1
record format B-2
sample printout of B-3, B-4
sending messages to (*see* DSIWLS macro)
logger G-2
LU operand, DSILCS macro 3-28
LUNAME operand, DSIRDS macro 3-47
LV operand
DSIFRE macro 3-23
DSIGET macro 3-25

macro instructions
overview of 3-2, 3-3, 3-4
in communication from an OST 3-13
syntax in 1-1
message
building (*see* DSIMBS macro)
command list started by 2-4
defining module for (*see* DSIMDS macro)
definition of G-2
in subtask 6-7, 6-8
queuing 3-15
sending 3-13
table for 3-30
message handler (MH) G-2
MID operand, DSIMBS macro 3-31
MODNAME operand, DSICES macro 3-19
MSGA operand, DSIMBS macro 3-31
MSGSIZE operand, DSIMBS macro 3-31
MSGTBL operand, DSIMBS macro 3-32
MVT control block (*see* DSIMVT)

NAME operand, DSIDKS macro 3-22
network resource G-2
NEXT operand, DSILCS macro 3-29

OITPOS operand
  DSIOIS macro 3-36
  DSISSS macro 3-48
operand G-2
operator, network G-2
operator control (ACF/TCAM) G-3
operator identification, searching for (*see* DSIOIS macro)
operator identification table (*see* DSIOIT)
operator station G-3
operator station task (OST)
  definition of G-3
  macros for communication with 3-13
OPID operand
  DSILCS macro 3-28
  DSIOIS macro 3-36
OPTION operand, DSIZVSMS macro 3-54
optional subtask G-3
OPTIONS operand, DSIPSS macro
  FIRST 3-43
  LAST 3-43
  MIDDLE 3-43
  MSG 3-43
  ONLY 3-43
  SEG 3-43
OUT operand, DSIPAS macro 3-37
OUTPUT operand, DSIPSS macro 3-40
overlapped span of control G-3

PANEL operand, DSIPSS macro 3-41, 3-44–3-46
parse descriptor block (*see* DSIPDB)
parsing 3-38, 4-11
password G-3
PDB operand
  DSICES macro 3-19
  DSIPAS macro 3-37
  DSIPRS macro 3-39
PDBSIZE operand, DSIPRS macro 3-38
positional fields, message 3-30
PPT (*see* primary POI task)
PPT restrictions on command processors 4-4
presentation services 3-13, 3-14 (*see also* DSIPSS macro)
primary POIT task (DSIPPT) G-3
PRINT operand, DSICBS macro 3-18
profile G-3
program operator G-3
PSSWAIT operand, DSIPSS macro 3-42
publications
  corequisite, TCAM i
  corequisite, VTAM i
  prerequisite i
P1 . . . P9 operand, DSIMBS macro 3-31

Q operand
  DSIFRE macro 3-24
  DSIGET macro 3-25

R operand, DSIFRE macro 3-23
READ operand, DSIDKS macro 3-22
REENT operand, DSIGET macro 3-25
regular command processor 4-2
request/response unit (*see* RU)
RESHOW key, full-screen 4-25
reshow option, full-screen 4-25
response G-3
resource, locating (*see* DSIRDS macro)
resource location 3-16
routing qualifier G-3
RU (request/response unit)
  definition of G-3
  use in DSIZCSMS macro 3-52
RU operand, DSIZCSMS macro 3-52
RULENG operand, DSIZCSMS macro 3-52

SCE control block (*see* DSISCE)
scope checking (*see* DSIKVS)
scope of commands G-3
screen formatting (*see* DSIPSS macro)
SCRSIZE operand, DSIPSS macro 3-41
SCTADDR operand
  DSICES macro 3-19
  DSIKVS macro 3-26
service facilities
  control block considerations for 3-4
  in general 3-1
  macro instructions invoked by 3-2–3-4, 3-18–3-55
  obtaining MVT addressability for 3-1
service work block (SWB) (*see* DSISWB)
SIZE operand, DSIPSS macro 3-43
SNTADDR operand, DSISSS macro 3-48
service compatibility 1-1
SP operand
  DSIFRE macro 3-23
  DSIGET macro 3-25
span G-3
span name table (*see* DSISNT)
span of control
  in general G-3
  overlapped G-3
standard NCCF mode 3-14
station G-3
status of resource, indicating (*see* DSIRDS macro)
STATUS operand, DSIRDS macro 3-47
storage
  freeing 3-11 (*see also* DSIFRE macro)
  getting 3-11 (*see also* DSIGET macro)
subtask
  attachment of 6-3
  coding guidelines 6-3
  command processing 6-9
  control block considerations for 6-9
  defining to NCCF 6-1
  displaying status of 6-5
  entry linkage 6-3
  example of 6-14
  exit linkage 6-3
  freeing DSIMQS buffers 6-8
  indicating when ready 6-4
  in general 6-1
  initialization 6-2, 6-3
  LIST command 6-5
  managing queued storage 6-6, 6-8
  message handling 6-7, 6-8
  optional facilities 6-5
  organization 6-1, 6-2
  reading initialization deck 6-6
  requirements 6-3
  termination 6-2, 6-5
suppression character in command lists 2-4
SWB control block (*see* DSISWB)
SWB operand
  DSICES macro 3-19
  DSIDKS macro 3-21
  DSIKVS macro 3-26
  DSILCS macro 3-28
  DSIMBS macro 3-30
  DSIMQS macro 3-35

DSIOIS macro 3-36
DSIPAS macro 3-37
DSIPRS macro 3-38
DSIPSS macro 3-40
DSIRDS macro 3-47
DSISSS macro 3-48
DSIWCS macro 3-50
DSIWLS macro 3-50
DSIZCSMS macro 3-51
DSIZVSMS macro 3-53
synchronous full-screen command processor 4-13
system command entry (*see* DSISCE)

TARGET operand, DSIZCSMS macro 3-52
task vector block (TVB)
listing of C-57
locating (*see* DSILCS macro)
TASKA operand
DSIFRE macro 3-24
DSIGET macro 3-25
TASKID operand, DSIMQS macro 3-35
TCAM control task (DSITCT) G-3
terminal G-3
TESTWAIT operand, DSIPSS macro 3-42
TIB control block (*see* DSITIB)
time, obtaining (*see* DSIDATIM macro)
timer request G-4
timer initiation G-4
title-line processing, full-screen 4-20
title-line processing, full-line 4-19
TRE exit routine 5-11
TVB control block (*see* DSITVB)
TVB operand, DSICLS macro 3-28
TVBRESET bit 4-26
TVBPNMOD bit 4-26
TYPE operand
DSIDKS macro
CONN 3-21
DISC 3-22
FIND 3-22
READ 3-22
DSIMDS macro 3-33, 3-34
DSIPSS macro
ASYPANEL 3-42
CANCEL 3-24
IMMED 4-41
OUTPUT 3-40
PANEL 3-41
PSSWAIT 3-42
SCRSIZE 3-41
TESTWAIT 3-42
WINDOW 3-41
XSEND 3-41
DSIZCSMS macro 3-52

unsolicited message routing G-4
user-defined modules, loading (*see* DSILOD macro)
user exit (*see* exit routines)

VALUE operand, DSIKVS macro 3-26
variables, command list
control variables 2-8
definition of G-4
in general 2-6
user variables 2-9
VSAM disk log B-1
VSAM services, NCCF (*see* DSIZVSMS macro)

wait, subtask (*see* DSIWAT macro)
WINDOW operand, DSIPSS macro 3-41

XITCI exit routine 5-10
XITCO exit routine 5-10
XITDI exit routine 5-10
XITVI exit routine 5-11
XITVO exit routine 5-11
XITVN exit routine 5-11
XSEND operand, DSIPSS macro 3-41

3270 data stream
example 4-23
full-screen formatting 4-24

Network Communications Control Facility Customization (File No. S370/4300-30) Printed in U.S.A. SC27-0433-5

SC27-0433-5