60480300

CONTROL DATA CORPORATION

MESSAGE CONTROL SYSTEM VERSION 1 REFERENCE MANUAL

CDC® OPERATING SYSTEMS:
NOS 1

60480300

CONTROL DATA CORPORATION

MESSAGE CONTROL SYSTEM VERSION 1 REFERENCE MANUAL

CDC® OPERATING SYSTEMS:
NOS 1

REVISION RECORD	
REVISION	**DESCRIPTION**
A	Original release.
(11/30/79)	
Publication No. 60480300	

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086

or use Comment Sheet in the back of this manual

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Revision
Cover	–
Inside front cover	A
Title Page/ii	–
iii/iv	A
v/vi	A
vii/viii	A
ix/x	A
xi	A
1-1 thru 1-5	A
2-1 thru 2-5	A
3-1 thru 3-4	A
4-1 thru 4-28	A
5-1 thru 5-11	A
6-1 thru 6-11	A
7-1 thru 7-4	A
8-1 thru 8-8	A
9-1 thru 9-3	A
A-1 thru A-5	A
B-1 thru B-17	A
C-1 thru C-4	A
D-1 thru D-10	A
E-1	A
Index-1 thru -5	A
Inside back cover	A
Back Cover	–

Page	Revision

Page	Revision

PREFACE

This manual describes the CONTROL DATA® Message Control System (MCS) Version 1.0. This manual is written for the programmer familiar with the COBOL language, the Network Operating System (NOS), and Network Host Products (NHP).

The Message Control System provides a method of queuing, routing, and journaling messages passed between COBOL programs and the communication network.

As described in this publication, the Message Control System Version 1.0 operates under control of the NOS 1.4 operating system for the CDC® CYBER 170 Series; CYBER 70 Models 71, 72, 73, and 74; and 6000 Series Computer Systems. MCS uses the Network Access Method (NAM) Version 1.2 for communication with terminals and interfaces with COBOL 5.3.

Related information can be found in the publications listed below. The NOS manual abstracts is an instant-sized manual containing a brief description of the contents and intended audience of all NOS and NOS product set manuals. The abstracts manual can be useful in determining which manuals are of greatest interest to a particular reader.

Publication	Publication Number
COBOL Version 5 Reference Manual	60497100
Network Products Interactive Facility Version 1 Reference Manual	60455250
Network Products Network Access Method Version 1 Reference Manual	60499500
Network Products Network Access Method Version 1 Network Definition Language Reference Manual	60480000
NOS Version 1 Manual Abstracts	84000420
NOS Version 1 Operator's Guide	60435600
NOS Version 1 Reference Manual (Volume 1 of 2)	60435400
NOS Version 1 System Maintenance Reference Manual	60455380

CDC manuals can be ordered from Control Data Corporation, Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103.

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

NOTATIONS USED IN THIS MANUAL

UPPERCASE	Uppercase words are Application Definition Language reserved words. They must be spelled correctly, including any hyphens; they cannot be used in a source program except as indicated. When used in examples of terminal dialog, uppercase indicates information generated by MCS or by the network.
UNDERLINED	Underlined uppercase words are required when the format in which they appear is used.
lowercase	Lowercase words are generic terms which represent the words or symbols supplied by the programmer. When generic terms are repeated in a format, a number is appended to the term for identification in the subsequent discussion. When used in examples of terminal dialog, lowercase indicates information entered by the terminal user.
[ ]	Brackets indicate an optional portion of a format. All of the format within the brackets can be omitted or included at programmer option. If items are stacked vertically within the brackets, only one of the stacked items can be used.
{ }	Braces indicate the portion of a format that is required, but a selection of one of the vertically stacked items within the braces must be made.
. . .	The ellipses is a repitition indicator. The portion of the format enclosed in the immediately preceding braces or brackets can be repeated at programmer option.

GENERAL DESCRIPTION 1

The Message Control System (MCS) provides the COBOL programmer with a telecommunications message handling capability. MCS is a network application program that utilizes the Network Access Method (NAM) to transfer data between COBOL programs and terminals and between COBOL programs.

FEATURES

A central feature of MCS is the Application Definition Language that allows an application developer to tailor an MCS application to specific needs. Other features are as follows:

- A set of commands provides the terminal user with execution-time control over certain aspects of the MCS application with which the user is communicating.
- A set of application operator commands provides a designated application operator with additional control over MCS operations.
- Application status information can be copied to a monitor file to be used for analyzing the activities of an application. MCS provides facilities for specifying how often or under what circumstances the copying is to be performed.
- MCS applications can execute independently of the network for purposes of testing message routing and Application Definition Language program logic.
- Essential application status information can be copied to a dump file for recovery purposes. MCS provides facilities for naming the dump file and for specifying how often or under what circumstances the copying is to be performed.
- A COBOL Communication Facility allows COBOL programs to interface with MCS.

CONFIGURATION

The hardware and software configuration needed to support MCS is similar to that of any network application program using the network software to service interactive terminals.

NETWORK SOFTWARE INTERFACES

MCS is a network application program that interfaces with the Network Access Method (NAM). Through NAM, MCS interfaces with the Communications Supervisor module and the Network Validation Facility of the network software. The Communications Supervisor supervises the establishment and maintenance of a communication path between a terminal and MCS. Login dialog and access security are provided by the Network Validation Facility (NVF). The relationships of MCS with NAM, NVF, the Communications Supervisor, and COBOL programs are shown in figure 1-1.

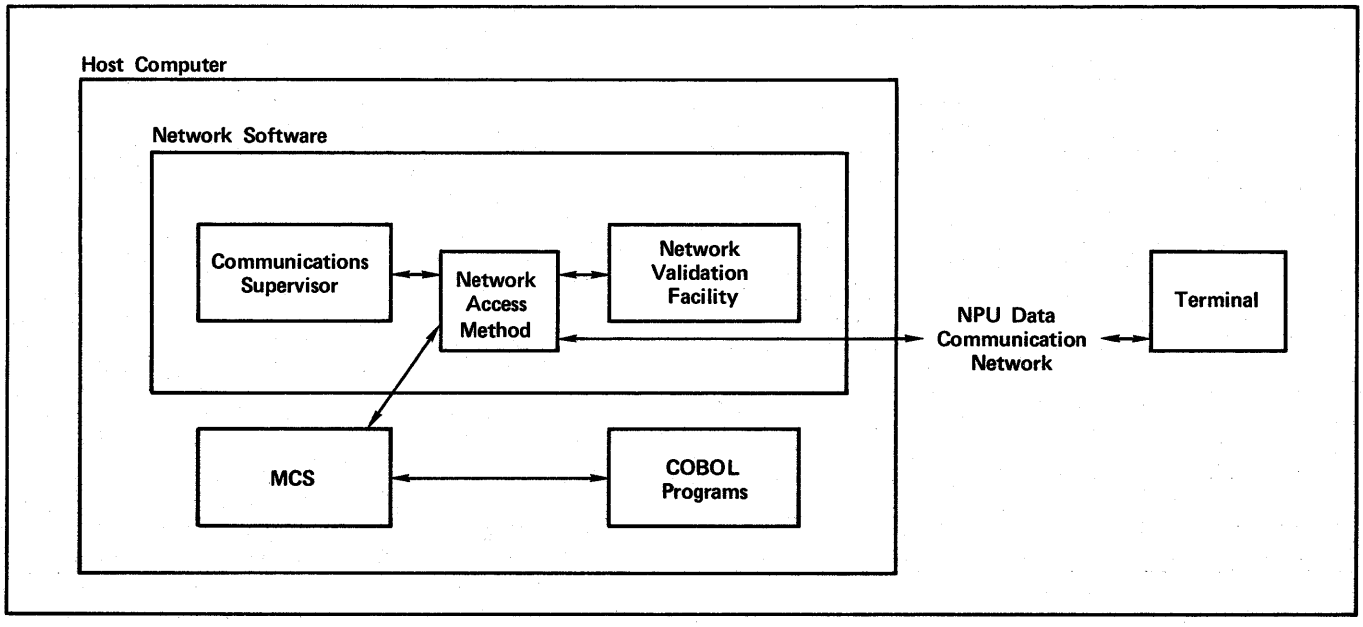


Figure 1-1. MCS Software Interfaces

MCS uses the system control point feature of the NOS operating system. MCS, when running, implies the use of three control points, one for each of the following modules:

- NAM
- MCS
- COBOL program

For installations where control point resources are limitied, MCS should be shut down when applications are not in use; the control points are then available to other programs.

HARDWARE REQUIREMENTS

The MCS software requires one CDC host computer (configured for network software operation), one 255x series Network Processing Unit and communications line adapter, and any terminal supported by the Interactive Virtual Terminal (IVT) interface. The IVT interface supports 14 classes of terminals. Most terminal classes correspond to an actual terminal, as shown in table 1-1.

TABLE 1-1. INTERACTIVE VIRTUAL TERMINAL CLASSES

Line Protocol	Terminal Class	Archetype Terminal
Asynchronous	1	Teletype Corporation Model 30 Series
	2	CDC 713
	4	IBM 2741
	5	Teletype Corporation Model 40-2
	6	Hazeltine 2000
	7	CDC 751
	8	Tektronix 4000 Series Asynchronous
HASP Synchronous	9	HASP Protocol Multileaving Workstation
Mode 4 Synchronous	10	CDC 200 User Terminal
	11	CDC 214-1X
	12	CDC 711-10
	13	CDC 714-10/20
	14	CDC 731-12 or 732-12
	15	CDC 734

Certain characteristics might vary from terminal to terminal. For example, print with no line advance does not work on mode 4 terminals. The user should consult site analysts for information on these characteristics.

Refer to the appropriate terminal operator's manual for information on terminal operation.

APPLICATION ORGANIZATION

An MCS application consists of the COBOL programs, terminal environment, message queues, and journal files that are defined through the Application Definition Language. One or more MCS applications can be active at any given time; however, communication between applications is not possible. One or more terminals can be connected to a single application; however, a single terminal cannot be connected to more than one application at a time. The association between physical terminals and the terminal environment described in the application definition is established at login time. One or more COBOL programs can be included in a given application; a COBOL program can be included in more than one application provided it is defined in each application definition that uses the program. The relationships of the elements of two MCS applications are illustrated in figure 1-2.

APPLICATION DEFINITION

To use the capabilities of MCS, the MCS Application Definition Language (ADL) must be used to establish an application definition. ADL performs the following functions:

- Defines which COBOL programs are to execute within the application environment.
- Defines the message queues that are to be used by the COBOL programs and their residency (central memory or disk).
- Defines which terminals can be connected to the application.
- Specifies the criteria to be used by MCS for routing messages between terminals and queues.
- Specifies supervisory processing for conditions such as accumulation of a given number of messages in a queue, or chronological events.
- Assigns user-defined names to programs, terminals, journal files, and queues.

Application Definition Language programs are compiled by the Application Definition Language processor, which produces a collection of tables for use by MCS. These tables, known collectively as the application definition, are then added to the application definition library, which is a collection of MCS application definitions. Each application definition resides on this library and is completely independent of other application definitions.

Figure 1-3 illustrates the elements of an MCS application defined in the application definition.

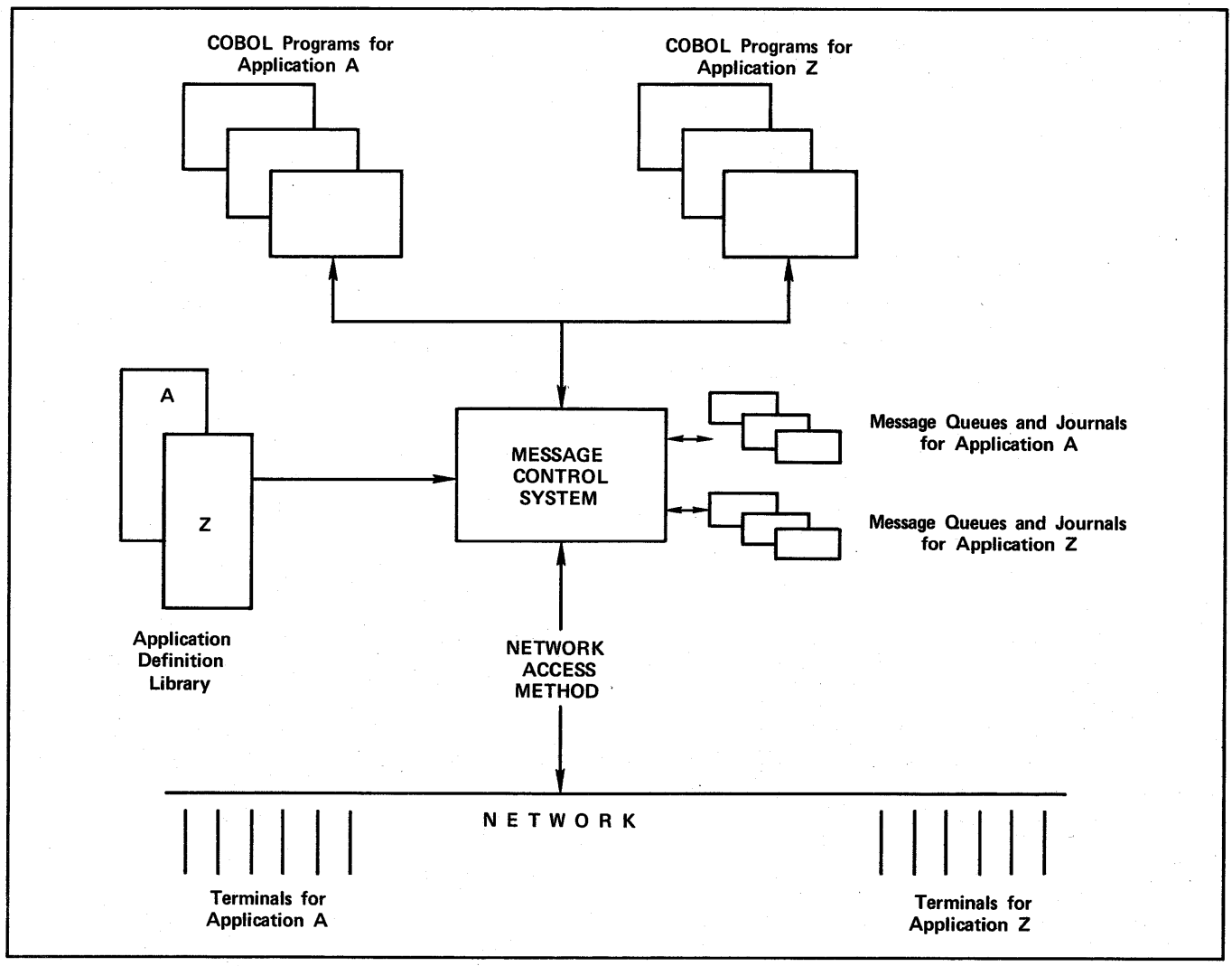


Figure 1-2. MCS Application Organization

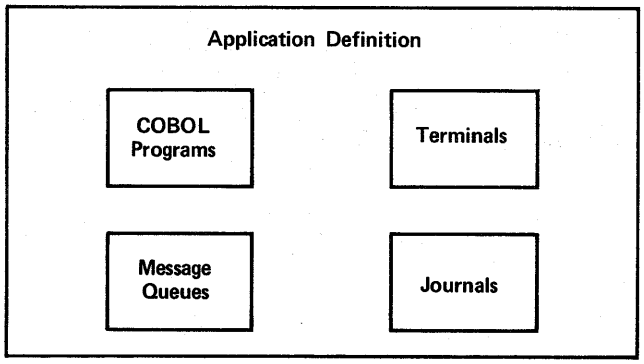


Figure 1-3. MCS Application Definition

MESSAGE FLOW

Under MCS, messages are routed between COBOL programs and terminals, and among COBOL programs. The mechanism that allows programs and terminals to communicate is the message queue. Messages are buffered through queues where they await delivery to the appropriate destination. The flow of messages is from terminals to input queues and then to programs, from programs to output queues and then to terminals, and from programs to interprogram queues and then to other programs. All queues, programs, and terminals are referenced by symbolic names defined in the application definition.

COBOL INTERFACE

COBOL programs perform the message processing functions of an MCS application. The COBOL 5 Communication Facility (CCF) enables COBOL programs to communicate with MCS. CCF provides language statements for interfacing with MCS. Following is a brief summary of these statements. Refer to the COBOL 5 reference manual for more detailed information.

RECEIVE

Acquires a message or part of a message from the named input queue, and optionally suspends the program containing the RECEIVE statement when no complete message is available in the named queue.

SEND

Releases a message or part of a message to MCS for transmission to the named destinations.

ACCEPT MESSAGE COUNT

Obtains the count of complete messages currently existing in the named input queue.

PURGE

Deletes incomplete messages released to MCS by the program issuing the PURGE statement.

DISABLE

Breaks the logical path between sources and input queues or between output queues and destinations.

ENABLE

Establishes the logical path between sources and input queues or between output queues and destinations.

In addition to the preceding statements, the STRING and UNSTRING statements in COBOL 5 can facilitate the character handling necessary for construction and analysis of messages.

TERMINALS

Each terminal that is to input data to, or receive data from, an MCS application is referenced by a symbolic name within the COBOL programs of that application. This name is not necessarily the same name that is used in network definition files and internally within the Network Access Method. At login time, MCS associates the symbolic name used in the COBOL programs with the name used in the Network Access Method.

There are two ways to establish this association between physical device or user and a symbolic name. The first is through the Application Definition Language, which provides syntax for defining this association. The second way is through a dialog between MCS and the terminal. In this second method, the symbolic name must still be declared in the application definition, but it is not permanently associated with a specific terminal or user.

A given physical terminal can be defined in more than one application definition. For this reason, when a terminal is first connected to MCS, the terminal user must indicate the application desired. The terminal user can then indicate, when appropriate, which symbolic name should be associated with the terminal.

Once the terminal identity is established, the terminal user can select the mode of operation. In data mode, all input from the terminal is routed to the appropriate input queue and is available to COBOL programs within the application. In command mode, the user can enter commands to control certain aspects of MCS operation. The initial mode is established in the application definition. The technique for switching modes is described in section 2, Terminal Access.

The initial connection between a terminal and MCS is either established as part of the login sequence, or is predefined through the Network Definition Language. When the connection is predefined, the login sequence is controlled by parameters specified in the application definition.

APPLICATION DEVELOPMENT

Detailed knowledge of the network definition is not necessary to develop an MCS application. However, terminals used by the application must be included in the network definition.

The following steps are necessary to develop and bring on-line an MCS application:

- Prepare the application definition using the Application Definition Language.
- Process the application definition using the Application Definition Language processor to create the application definition tables. These tables are added to the application definition library.
- Prepare the COBOL programs that are to communicate with MCS.
- Prepare any data bases and data base definitions required by the COBOL programs.
- Test the application definition and COBOL programs off-line using the test mode facilities provided by MCS.
- Prepare an operational guide for users of the application.
- Test the application on-line.

OPERATIONAL CONTROL

MCS is started, idled, and shut down by commands issued from the CYBER Operator's Console. Once MCS is running, control of individual applications is accomplished via commands issued from terminals that have connected to MCS. Commands that request status information or affect only the terminal issuing the command can be issued by any terminal that is connected to MCS. These commands enable the terminal user to:

- Display information about input and output queues.
- Disable, and subsequently enable, the terminal.
- Terminate the terminal's connection to MCS.
- Send messages to the application operator.

However, the full repertoire of commands is available only to a terminal user designated as the application operator (AOP). The additional commands allow the AOP to monitor and control the activities of an MCS application. The AOP commands provide the capability to:

- Initiate the application.
- Disable, and subsequently enable, any queue or terminal in the application.
- Idle, and subsequently resume, the application.

- Disconnect any terminal connected to the application (except the AOP terminal).
- Display information about the status of the application.
- Display the contents of message queues.
- Abort a program executing under MCS.
- Initiate execution of user COBOL programs.
- Reroute messages destined for terminals.
- Send messages to all terminals or selected terminals.
- Shut down the application.

The Application Definition Language provides syntax for naming the terminals eligible for AOP status and for defining a password to protect this capability. The application operator is then designated at login time by specifying the correct password. A given application cannot have more than one AOP at a given time; however, when an AOP logs off, another user can log in as the AOP.

The MCS commands are also available to COBOL programs. Messages sent to the application name are interpreted as MCS commands and are not enqueued for routing to a terminal or another program. MCS responses to these commands are enqueued in the program's response queue, which is defined in the application definition.

Before a terminal can communicate with MCS it must first gain access to the network and then indicate with which NAM application program it wishes to communicate (in this case MCS). This section describes the procedure used to connect the terminal with the network via the Network Validation Facility (NVF), access MCS and an MCS application, and disconnect the terminal from the network.

ACCESS PROCEDURES

An access procedure consists of connecting the terminal to NVF to begin a login procedure. The available login procedures are standard login and automatic login.

In the standard login procedure, the network prompts the user for the necessary information. An abbreviated form of the login procedure allowing the user to enter all of the login information on a single line instead of waiting for prompts is also available.

A terminal can be configured for automatic login; in which case NAM supplies all of the necessary information. Terminals can also be configured for a login procedure in which some of the information is supplied by the user, and the remainder is supplied by NAM.

When the terminal is dedicated to MCS in the network definition, the standard login procedure is not applicable.

STANDARD LOGIN PROCEDURE

The login sequence begins with the terminal displaying three lines. The first line is the date, time, and terminal name, in the following format:

```
yy/mm/dd.  hh.mm.ss.  xxxxxxx
```

The date is given by year/month/day and the time in hours.minutes.seconds. The terminal name is represented by xxxxxxx, a 1- to 7-character name that the network uses to identify the terminal. For example, if the login for terminal TERM201 began at 12 minutes and 44 seconds after 2 o'clock in the afternoon of the third day in December, 1979, the first line would read:

```
79/12/03.    14.12.44    TERM201
```

The second line is an identifying header that is defined by installation; it might give the company name, the operating system, and the version of the operating system, as shown in the following example:

```
CONTROL DATA CORP.        NOS 1
```

The third line is:

```
FAMILY:
```

This line requests the family name of the mass storage device that contains the terminal user's permanent files. For example, if AAA is an approved family name, enter:

```
AAA
```

The network software next requests the user name:

```
USER NAME:
```

The user name identifies the terminal user in the NOS validation files. This is the identifier by which the terminal user is known until a session is terminated by logoff. User name is made up of letters, digits, and the asterisk, in any combination.

The prompting line is responded to by entering a valid user name. For example, if 87X32 is the assigned user name, enter:

```
87X32
```

The network software then responds with a request for password:

```
PASSWORD:
   or
PASSWORD:
■ ■ ■ ■ ■ ■ ■
```

The password assigned to a terminal user to provide access security must be entered. For example, if the assigned password is PASS123, enter:

```
PASS123
```

For terminals capable of overstriking characters, the line of blackouts preserves password privacy after entry.

The next network software prompt is for the name of the application with which connection is desired:

```
xxxxxxx-APPLICATION:
```

where xxxxxxx is the 1- to 7-character name used to identify the terminal. To select MCS, enter:

```
MCS
```

MCS responds with the message:

```
MCS 1.x yy/mm/dd.  hh.mm.ss
```

The specific version of MCS being accessed is represented by x, and yy/mm/dd and hh.mm.ss indicate the date and time access begins.

This completes the login and establishes a connection to MCS.

Once a terminal has gained access to the network and established a connection to MCS, the terminal user must then specify the desired MCS application (unless the terminal is dedicated to a particular application in the application definition). MCS issues the prompt:

```
MCS APPLICATION ?
```

In response to this prompt, enter the name of the desired MCS application:

```
MCS APPLICATION ? application
```

When the application has not been initiated by the application operator (as described in section 8), the following message is displayed:

```
APPLICATION NOT RUNNING
```

When the application has been activated and is available to terminal users, MCS issues the following prompt.

```
SYMBOLIC NAME ?
```

In response to this prompt, enter either one symbolic name or two symbolic names separated by a space. Each terminal with which an MCS application communicates is referenced by a symbolic name within that application. This name is not necessarily the same as the name used by NAM to reference the terminal. MCS associates the symbolic name used in an application with the name used by NAM. Symbolic names must be defined in the application definition.

When a single name is entered, that name must be defined in the application definition as either a source name, a destination name, or an interactive name. When two names are specified, the first must be defined as a source name, and the second as a destination name. For example, to associate the symbolic name SOURCEB with the terminal, enter:

```
SOURCEB
```

When the terminal is dedicated to a particular symbolic name in the application definition, MCS does not issue the SYMBOLIC-NAME prompt. See section 4 under ALIAS clause for an explanation of dedicated.

MCS next issues the following message:

```
mmmm MODE
?
```

When the terminal is dedicated, MCS issues the following message:

```
DEDICATED TERMINAL
APPLICATION=xxxx
SYMBOLIC NAME=yyyy
STATUS=zzzz
mmmm MODE
?
```

In these examples, mmmm is either DATA or COMMAND (indicating the operating mode of the terminal as described in this section under Operating Modes) and ? is the MCS prompt for user input. In the second example, xxxx is the application name, yyyy is the terminal symbolic name as defined in the application definition, zzzz is ENABLED or DISABLED.

This completes the procedure for establishing a connection to MCS.

AUTOMATIC NETWORK APPLICATION PROGRAM SELECTION

The Network Validation Facility (NVF) is designed to perform automatic network application program selection whenever the terminal is allowed to access only one network application program. Automatic network application program selection is performed immediately following successful validation of the terminal. The APPLICATION prompt is not issued, and any network application program name entry is ignored.

Automatic network application program selection occurs only on the first login in a given terminal session (initial login); after that, MCS must be entered in response to the APPLICATION prompt. To repeat the automatic login procedure, the physical connection must be broken and reestablished.

ABBREVIATED LOGIN PROCEDURE

The login procedure can be shortened by entering all of the login information at once. For example:

```
FAMILY:
familyname,username,password,application
```

The information must be entered after an NVF prompt and must be separated by commas. These parameters are order-dependent, and any default or unused parameters must be indicated by separator commas. When the abbreviated login procedure is used, subsequent system prompts are suppressed.

For example, to login and connect to MCS with a family name of AAA, a terminal user's name of AB123, and a password of XYZ, enter:

```
AAA,AB123,XYZ,MCS
```

To use the default value for family name, enter:

```
,AB123,XYZ,MCS
```

AUTOMATIC LOGIN

When the network is configured so that NAM performs automatic login of terminals, NAM supplies the proper family name, user name, and network application program name to NVF. No login prompts are displayed at the terminal, and MCS is connected automatically.

If NVF is not provided with all of the required parameters, NVF issues prompts to the terminal for the missing parameters. The terminal user must then respond to the prompts by entering the requested parameter. No password parameter is used for terminals configured for automatic login.

AUTOMATIC MCS LOGIN

When a terminal is dedicated within MCS, MCS automatically selects the application and terminal symbolic name upon connection to MCS. The MCS APPLICATION and SYMBOLIC-NAME prompts are not displayed.

When a terminal is associated with (but not dedicated to) a specific user or terminal in the ALIAS clause of the application definition, MCS automatically assigns the terminal symbolic name when the user or terminal selects that MCS application. The SYMBOLIC NAME prompt is not displayed.

LOGIN DIAGNOSTICS

An unsuccessful login sequence is indicated by the diagnostic messages described in the following paragraphs.

If insufficient system resources are available to allow the terminal to gain access to the network, the following message is displayed:

```
LOGIN ABORTED, TRY LATER
```

If MCS is not running at a host computer control point, the following message indicates that the connection could not be made:

```
APPLICATION NOT PRESENT.
xxxxxxx-APPLICATION:
```

where xxxxxxx is the terminal name. The name of another network application program must be entered or a logoff procedure performed.

NVF can prevent more than one terminal with the same user name from being logged into MCS. The following message shows that login has been performed correctly and that an attempt has been made to connect to MCS, but connection is prohibited because another terminal user with the same family name and user name is currently connected.

```
CONNECTION PROHIBITED, TRY AGAIN LATER
xxxxxxx-APPLICATION:
```

where xxxxxxx is the terminal name. Although it is necessary to wait until the current user has finished before a new connection to MCS is permitted, another application can be requested.

In times of heavy usage or if available resources such as computer memory are limited, MCS can become saturated. If this occurs, MCS is unable to accept any additional connections. The following message is issued:

```
CONNECTION REJECTED
```

and the user must either select another network application or attempt an MCS connection at a later time.

An unacceptable family name, user name, or password is indicated by the message:

```
IMPROPER LOGIN, TRY AGAIN
FAMILY:
```

The login procedure must be restarted from the beginning, regardless of the step in the procedure at which the diagnostic occurred.

After four unsuccessful attempts to enter family name, user name, and password, the following message appears:

```
ILLEGAL USER
```

If this occurs for a switchable terminal, the terminal is disconnected from the network; further attempts to log in are not possible until the terminal is reconnected to NVF.

If the application name entered in response to the APPLICATION prompt is unknown to NVF, or the application is not available to the logged in user, the following message is displayed:

```
ILLEGAL APPLICATION, TRY AGAIN.
xxxxxxx-APPLICATION:
```

where xxxxxxx is the terminal name. After a fixed number (installation option with default of 4) of unsuccessful attempts to enter an application name, the following message is displayed at the terminal:

```
APPLICATION RETRY LIMIT
```

If the application retry limit is exceeded, the terminal is disconnected from the network.

Once a connection to MCS has been established, the terminal user is allowed a limited number of tries (installation option with a default of 3) to correctly select an MCS application name and symbolic terminal name. If an incorrect application name is entered in response to the MCS APPLICATION prompt, the following message is displayed:

```
UNKNOWN APPLICATION
```

and MCS reissues the MCS APPLICATION prompt.

If an incorrect symbolic name is entered in response to the SYMBOLIC-NAME prompt, the following message is displayed:

```
UNKNOWN DESTINATION
```

or

```
UNKNOWN SYMBOLIC NAME
```

or

```
INVALID SYMBOLIC NAME
```

and MCS reissues the SYMBOLIC-NAME prompt.

An attempt to log in under a symbolic name that is being used by another terminal user results in the message:

```
DESTINATION ALREADY IN USE
```

or

```
SOURCE ALREADY IN USE
```

depending on how the symbolic name is defined in the application definition. The user must enter a different symbolic name.

If the established limit for login attempts is exceeded, the following message appears:

```
SOLICITATION LIMIT EXCEEDED
```

and the terminal is logged out of MCS; the network reissues the APPLICATION prompt, and the user must select a new network application.

If a user attempts to connect to an application that has not been initiated, the following message is displayed:

```
APPLICATION NOT RUNNING
```

MCS reissues the MCS APPLICATION prompt, and the user must select a different MCS application.

If a user attempts to connect to an idle application, the following message is displayed:

APPLICATION IDLE

and the user must select a different MCS application.

An attempt to log into an application that is running in test mode is rejected.

A response to any non-MCS login prompt must be entered within a system-specified time interval (installation option with default of 2 minutes). If the terminal user takes too much time to respond to a prompting message, the following message is displayed:

TIMEOUT

If this occurs, the terminal is disconnected. No further login entries are possible at a dialup terminal until the telephone connection is broken and reestablished. The login sequence is restarted at a hardwired terminal by entering any character and pressing the transmission key.

If an internal error causes MCS to abort or the system operator disables MCS after the terminal has been connected, the following message is displayed:

APPLICATION FAILED.
MCS CONNECT TIME hh.mm.ss.
xxxxxxx-APPLICATION:

The parameters hh.mm.ss specify the length of time the terminal was connected to MCS; xxxxxxx represents the terminal name.

A complete list of terminal error messages, and an explanation of each message, is included in appendix B.

SWITCHING NETWORK APPLICATIONS

To transfer the terminal connection from MCS to another network application program, the terminal user must enter the LOGIN command from MCS command mode. A prompt for a new application name is eventually displayed as part of the responses to the LOGIN command.

After disconnection from MCS, NVF indicates that a connection switch is in progress by issuing the following message:

MCS CONNECT TIME hh.mm.ss.

where hh.mm.ss indicates the time elapsed since connection with MCS occurred.

After this message appears, another prompt for a network application program name is issued, unless the terminal is dedicated to MCS.

The command form:

LOGIN application

can also be used to switch network applications. This command is described in section 7.

DISCONNECT PROCEDURE

During the login procedure, two sequential logical connections are established within the network. The first connection is with NVF; this occurs when the family name, user name, and password are requested. These provide the security and accounting information needed for NVF to establish the second connection. The second connection is established (to a network application program such as MCS) when the response to the APPLICATION prompt is entered successfully. The logical connection to the network application program must be severed before the user can be logged off from the network.

EXIT FROM MCS

A terminal user ends the logical connection with MCS by entering an END, HELLO, LOGON, or LOGIN command (described in section 7). These commands disconnect the terminal from MCS and transfer control to NVF.

EXIT FROM THE NETWORK

An exit from the network can be accomplished by entering a LOGOUT, LOGOFF, BYE, or GOODBYE command. MCS interprets each of these commands as an alternate END command; MCS responds by sending LOGOUT to NVF without being prompted. Therefore, the terminal can be disconnected from both MCS and the network by the LOGOUT, LOGOFF, BYE, or GOODBYE command. When the terminal console connection to MCS is terminated but the terminal is not disconnected from the network, the following message is displayed:

MCS ENDED yy/mm/dd. hh.mm.ss.
MCS CONNECT TIME hh.mm.ss.

Actual disconnection can take up to several minutes; therefore, any entry that is made before disconnection causes the login procedure to be restarted. Once a dialup terminal has been disconnected from the network, it must be reconnected. The NVF logout procedure does not disconnect a hardwired terminal from the network. When the terminal is queued for disconnection, the following message appears:

LOGGED OUT.

LOGOUT WITHOUT DISCONNECTION

The terminal user can log out and initiate the login dialog without disconnecting the terminal from the network. When the user exits MCS by entering a LOGIN, LOGON, or HELLO command, or responds to the APPLICATION prompt with a HELLO or LOGIN command, the terminal is logged out and disassociated from its current user number, and the login procedure is restarted.

A sample dialog illustrating the login procedure is shown in figure 2-1.

OPERATING MODES

While a terminal is connected to MCS, it can operate in one of two modes: command mode or data mode. The Application Definition Language allows the user to specify the initial mode for each of the application's terminals.

The initial operating mode of a terminal is displayed immediately after the user enters the terminal symbolic name. For example:

SYMBOLIC NAME ?term12
COMMAND MODE

```
FAMILY:  myfam                                  } NVF has
USER NAME: myname                               } control
PASSWORD:  pwrd                                 }
TNAME  - APPLICATION: mcs                       }
MCS 1.0 79/10/05. 08.22.30.                     }
MCS APPLICATION ?app1                           }
SYMBOLIC-NAME ?term1                            }
COMMAND MODE                                    }
?data                                           }
DATA MODE                                       } MCS has
.                                               } control
.                                               }
.                                               }
<b2><cr>                                        }
COMMAND MODE                                    }
?end                                            }
MCS ENDED  79/10/05.  08.25.15.                 }
MCS     CONNECT TIME 00.32.52.                  } NVF has
LOGGED OUT.                                     } control
```

Figure 2-1. Sample Login Dialog

The initial mode of a terminal, when not otherwise specified in the application definition, is command mode. In either mode, the MCS prompt for user input is a question mark (?). (In data mode, the prompt is issued only when serial number echoing is selected in the application definition, as described in section 4.)

NOTE

MCS does not distinguish between uppercase and lowercase letters. Therefore, when entering commands or messages, either form can be used, and MCS maps to uppercase.

COMMAND MODE

When a terminal is in command mode, input messages are treated as MCS commands. No data messages are sent to the application while command mode is active. Terminals can be switched to command mode by entering the break-2 control sequence. MCS responds with the message:

```
COMMAND MODE
?
```

The question mark is the MCS prompt for user input. When a command is entered, MCS checks the command for correctness. If the command is unrecognizable, MCS issues the message:

```
UNKNOWN COMMAND
```

and the user can reenter the command.

DATA MODE

When a terminal is in data mode, messages entered at the terminal are sent to input queues, and application output messages are delivered to the terminal. Terminals can be switched to data mode by entering the command:

```
DATA
```

MCS responds with the message:

```
DATA MODE
?
```

When in data mode, MCS issues prompts only if serial number echoing is specified in the application definition (as described in section 4).

The routing procedures that determine the destinations of messages entered in data mode are defined in the application definition. For example, an application might route a message to the queue specified in the first field of the message, where the field is delimited by a comma. For this example, the following dialog switches the terminal to data mode and sends messages to queues Q6 and Q25:

```
.
.
.
?data
DATA MODE
?q25, today is tuesday
SERIAL NUMBER = 1
?q6, let's go home
SERIAL NUMBER = 2
```

End-of-segment, end-of-message, and end-of-group conventions for messages entered from a terminal are described in section 3.

BREAK SEQUENCES

MCS recognizes the break-1, break-2, and cancel (control x) terminal control sequences. A terminal in command mode can be switched to data mode by entering break-2.

Break-1 can be used to terminate output to the terminal. This feature is useful when user commands, such as DISPLAY, result in excessive output.

For asynchronous terminals (classes 1 through 8) only, cancel can be used to delete a partial message. This feature can be used to correct errors when entering messages in data mode. For example, where (cr) indicates carriage return and (lf) indicates line feed, the dialog:

```
?first message (cr)
first segment of second message (lf)
second segment of second message (lf)
```

enters a complete message followed by two segments of a second message. A cancel entered at this point would delete the two segments of the incomplete second message, allowing the user to reenter those segments.

COMMANDS

MCS provides a set of commands that allows users to control certain aspects of its operation. These commands are divided into two categories: user and AOP. User commands control the operation of a given terminal connected to MCS and are available to all MCS users. The user commands are described in section 7. AOP commands control the operation of an MCS application and are restricted to the Application Operator. The AOP commands are described in section 8.

The basic unit of data manipulated by the interface between MCS and a COBOL program is the message. Messages from a terminal to a program, messages from a program to a terminal, or messages from a program to another program are collected in message queues while awaiting processing or transmission. This section discusses the message concept as it relates to MCS and the queue structure used to accomplish message manipulation.

THE CONCEPT OF MESSAGES

A message is a string of characters with an implied beginning and end. A message can contain one or more physical lines to be displayed at a terminal.

Messages can be logically subdivided into smaller units of data called segments. The maximum length of an MCS message segment is 4100 characters. However, there are certain criteria affecting what MCS interprets as a message segment. The subsection on message transmission discusses these criteria.

MESSAGE INDICATORS

Message indicators delimit messages and notify MCS or a COBOL program that a specific condition exists. An end-of-segment indicator (ESI) delimits a message segment within a message. An end-of-message indicator (EMI) delimits a message, which can consist of one or more segments, from the next message. Similarly, an end-of-group indicator (EGI) logically separates a group of several messages from succeeding messages. An EGI implies an ESI and an EMI, because a group of messages contains both messages and message segments. An EMI implies an ESI, because a message contains message segments. Figure 3-1 illustrates the end indicator hierarchy.

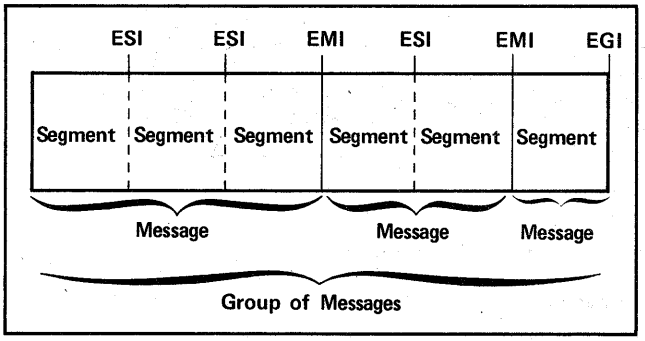


Figure 3-1. End Indicator Example

MESSAGE TRANSMISSION

On output, the COBOL program that uses MCS to send the message text to the terminal indicates which end indicator, if any, is associated with that text. At a terminal, MCS starts each message and each message segment sent to the terminal on a new line. When the last message in a group of messages is sent to a terminal, MCS signals the end of the group to the terminal. An EGI character is displayed at the terminal, followed by a carriage return. Section 4, Application Definition Language, describes the syntax to define an EGI.

On input, a combination of terminal user actions and MCS conventions determines where the end indicators are in the message text entered by the terminal user. Table 3-1 shows which end indicator MCS interprets after a particular terminal action. If a terminal user inputs text longer than the defined page width for that terminal, MCS inserts additional ESI indicators so that no message segment is longer than the defined page width.

TABLE 3-1. END INDICATORS INTERPRETED BY MCS

Terminal Type	Terminal Action	End Indicator†
Synchronous	New Line Carriage Return Send ETX	EMI
Asynchronous	Line Feed New Line ATTN	ESI
	Carriage Return	EMI

†A terminal action normally interpreted as an EMI is interpreted as an EGI when the character immediately preceding the terminal action is defined as an EGI in the application definition.

The communication description (CD) area of a COBOL program issuing a RECEIVE is updated by MCS to show which end indicator, if any, is associated with message text. See the COBOL 5 reference manual for detailed information. The COBOL programs of an application must perform any message processing functions that are required.

QUEUES

A queue is a storage area for messages awaiting delivery to a COBOL program or transmission to a terminal. A queue consists of the collection of related messages stored in it. Queuing messages provides for message buffering and message routing. One or more COBOL programs can communicate with one or more terminals through queues. Conversely, one or more terminals can communicate with one or more COBOL programs through queues. The process of placing messages in a queue is called enqueuing. Dequeuing is the process by which messages are removed from a queue. MCS can acquire

and enqueue messages from a terminal when no COBOL message processing programs are running. MCS can continue to send messages to terminals after the programs that generated the messages have terminated. MCS can be shut down with unprocessed messages in queues. When MCS is started again, messages in mass storage queues are still intact in the queues. Messages in central memory queues are discarded at application shutdown time. The user defines where message queues reside in the application definition. Figure 3-2 shows the flow of messages in an on-line data processing application using MCS.

INPUT QUEUES

Messages from terminals are placed into input queues by MCS while awaiting disposition by a COBOL program. Portions of messages are not available to the program; MCS does not enqueue an incoming message until all the segments of that message have been sent from the terminal. When MCS acquires a message from a terminal, it determines into which input queue to place the message based on the source of the message, the content of the message, the time of day, or any combination of these three. The application definition can specify that messages from one group of terminals be routed into one input queue, that messages from a second group of terminals be routed into a second input queue, and so forth. The same possibilities exist for routing messages based on content or time of day. Any COBOL program defined as part of the application can receive messages from any of the input queues. Section 4 describes the syntax for specifying message disposition.

There are three types of special-purpose input queues. These are the injection queue, the collection queue, and the response queue. The injection queue and the collection queue are used when an application is running in test mode. MCS enqueues messages from a test message generation program in the injection queue. MCS enqueues messages that are normally output to terminals in the collection queue. Test mode is explained in section 5. The response queue is a special destination used when a COBOL program sends commands to MCS. MCS responses to these commands are enqueued in the response queue. Section 4 describes the syntax for defining these special-purpose queues.

OUTPUT QUEUES

Messages sent from COBOL programs to destinations (terminals) are routed through output queues. Programs can send partial segments, partial messages, or message segments; incomplete messages are buffered by MCS until an EMI is sent by the program. When an EMI is received by MCS, the message is enqueued in an output queue prior to transmission to a terminal. Programs can send messages to terminals that are not presently connected or that are temporarily disabled. MCS accepts such messages and holds them in the output queues until the terminal is connected (logs in) or is reenabled. When a message is sent to a connected terminal, MCS routes the message according to routing information specified in the application definition.

INTERPROGRAM QUEUES

Interprogram message routing can be used in applications. Interprogram message queues are input queues that hold messages destined for other COBOL programs until the other programs request them.

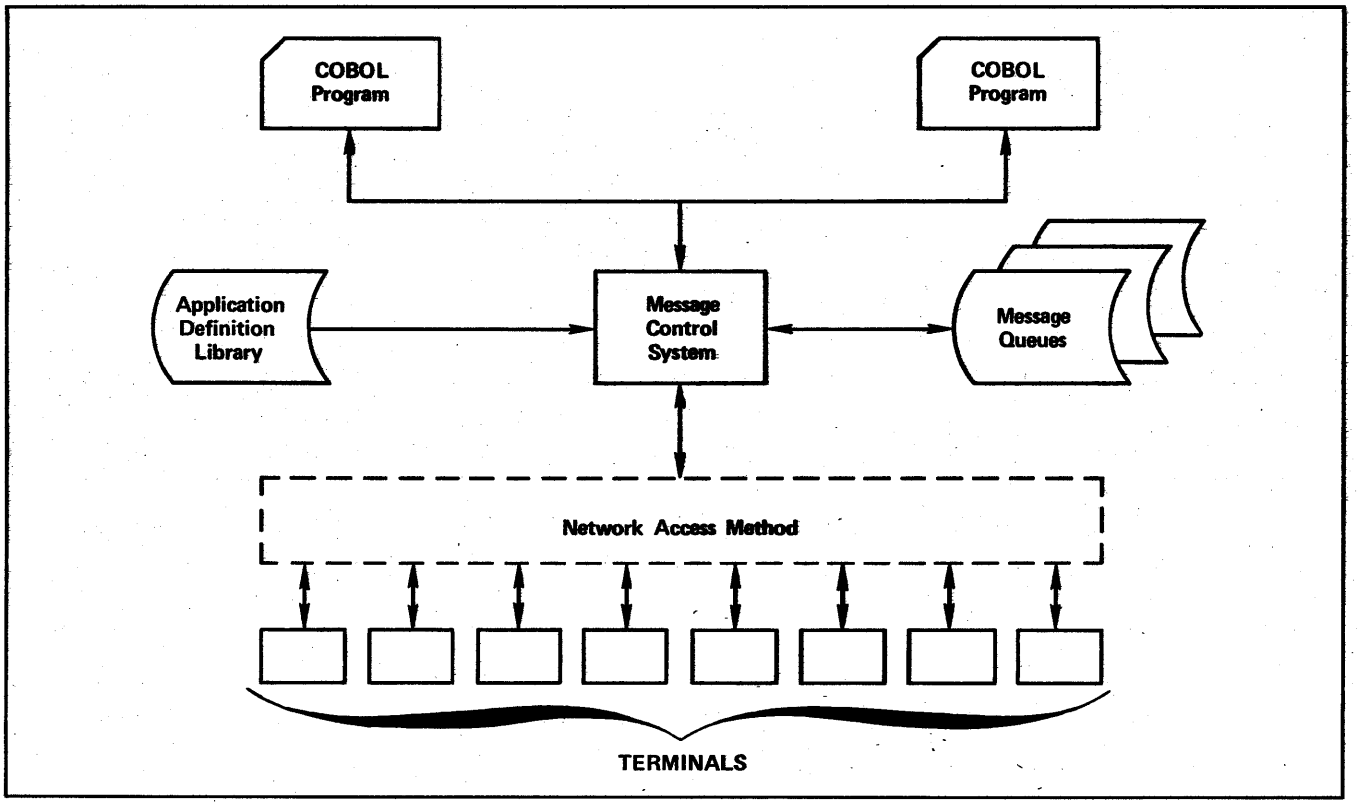


Figure 3-2. Message Flow Using MCS

QUEUE HIERARCHY

Four levels of input queues can be defined by the user to explicitly control the messages being enqueued and dequeued. These hierarchal divisions of queues are known as subqueues. The four levels are named queue, sub-queue-1, sub-queue-2, and sub-queue-3. The four-level structure allows specification of multiple conditions on which to base message routing. Figure 3-3 shows a four-level input queue structure. Any number of queues can be defined at any of the subqueue levels. For example, QA of figure 3-3 could include three sub-queue-1 level queues, each with three sub-queue-2 level queues, and each of these queues could have three sub-queue-3 level queues. An application can include zero, one, or more hierarchal input queue structures.

A queue with subqueues is a compound queue; a queue with no subqueues is a simple queue. In figure 3-3, queues A, B, C, D, E, F, and G are all compound queues, because they have queues below them in the hierarchy. Queues H, I, J, K, L, M, N, and O of figure 3-3 are simple queues, because they have no queues below them in the hierarchy. Any queue at the queue, sub-queue-1, or sub-queue-2 level can be either a compound or a simple queue depending on whether or not there are any queues below it. All sub-queue-3 level queues are simple queues, because there can be no lower-level queues.

Only simple queues store messages. An incoming message is tested at each level of the queue hierarchy, according to the routing conditions specified in the application definition, until the message arrives at a simple queue. Section 4 describes the syntax for defining routing conditions.

The message MCS releases to a COBOL program depends on the COBOL RECEIVE request. When a COBOL program requests a message from a compound queue, MCS searches the named compound queue from left to right until a nonempty simple queue is found and returns the next message in that queue. MCS also updates the COBOL program CD area to indicate from which subqueue the message was returned. When the next message in a given subqueue is requested by the COBOL program, the request must specify both the queue name and any subqueue names. It is possible for a partial message to remain in a queue if the COBOL program receiving area is not large enough to contain the entire message. A subsequent RECEIVE against the same queue and any subqueues returns the remainder of the message. Using figure 3-3, the following examples illustrate the messages returned when MCS gets a RECEIVE request from a COBOL program:

- The COBOL program requests a message from:

  Queue A
  MCS returns message 1 and updates the CD area

- The COBOL program requests a message from:

  Queue A
  Sub-queue-1 C
  MCS returns message 10 and updates the CD area

- The COBOL program requests a message from:

  Queue A
  Sub-queue-1 B
  Sub-queue-2 E
  MCS returns message 6 and updates the CD area

- The COBOL program requests a message from:

  Queue A
  Sub-queue-1 B
  Sub-queue-2 E
  Sub-queue-3 J
  MCS returns no message because message 6 was previously returned

- The COBOL program requests a message from:

  Queue A
  Sub-queue-1 C
  Sub-queue-2 G
  Sub-queue-3 N
  MCS returns message 13

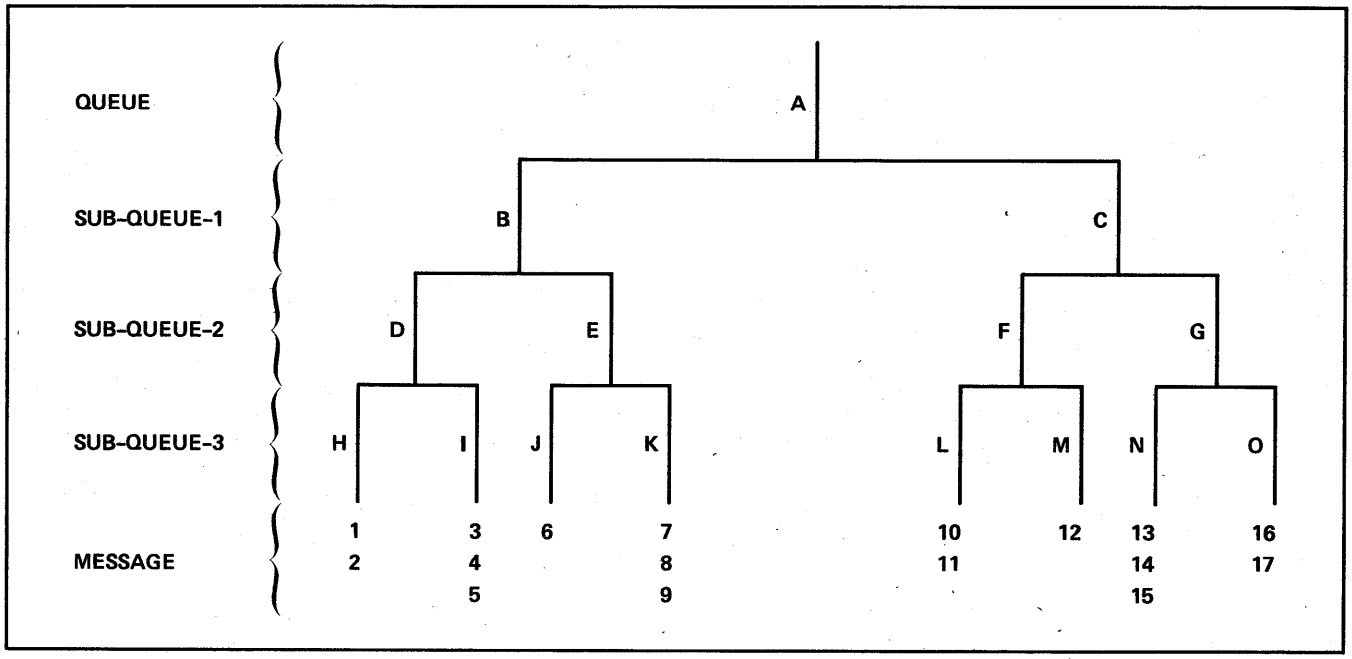


Figure 3-3. Input Queue Hierarchy Example

PRIORITY QUEUING

The queuing mechanism provides a way of assigning priorities in the processing of messages. Messages that the user considers high priority because of their source or content can be routed into one input queue by MCS, while messages considered lower priority can be routed into other input queues. High priority queues should be on the left in the queue structure, as shown in figure 3-4. See section 4 for syntax for assigning priorities.

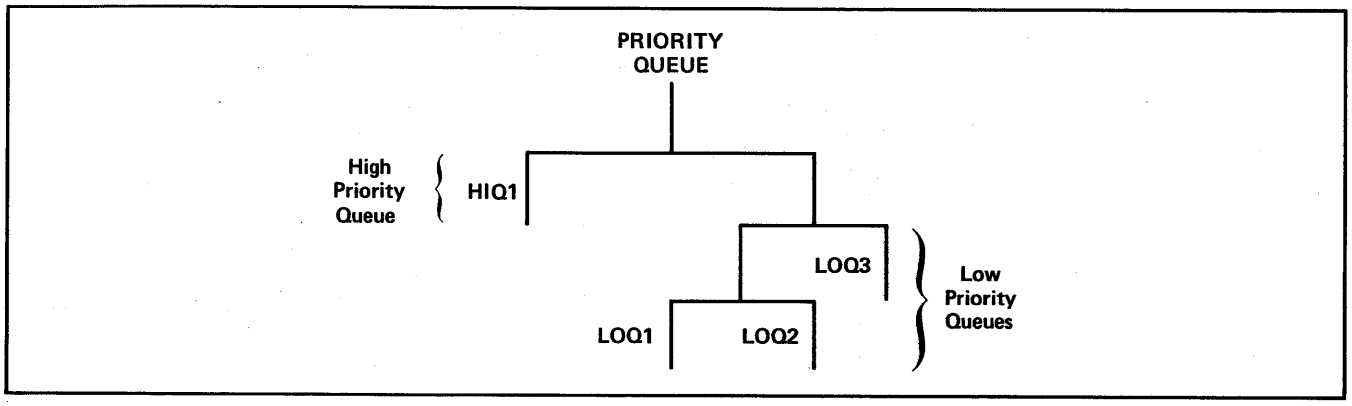


Figure 3-4. Priority Queue Example

APPLICATION DEFINITION LANGUAGE 4

Applications that use MCS are described by the Application Definition Language (ADL). In the context of MCS, an application is a collection of COBOL programs, queues, subqueues, communication devices (sources), output devices (destinations), and journal files. This section describes the functions, formats, and uses of ADL in specifying the various entities that make up a complete application.

LANGUAGE OVERVIEW

Three interfaces exist in the MCS environment. First is the MCS/COBOL program interface in which a COBOL program communicates with MCS. Second is the MCS/terminal user interface where a user directs MCS to send messages to a COBOL program. Third is the ADL/user interface that defines certain external aspects MCS uses in message routing. The external aspects include naming to MCS the COBOL programs included in the application, the message queues the application uses, and the sources and destinations to and from which messages are routed. ADL provides the syntax for defining these external aspects. Additionally, ADL is used to define the following:

- The association between a physical device (terminal) and the symbolic name an application uses to reference that terminal.
- The names of job invocation files and the conditions under which each job stream is submitted to the operating system by MCS.
- The criteria MCS uses in routing messages into input queues.
- The hierarchal structure of input queues.
- The terminals eligible for application operator (AOP) status and the password to protect this status.
- The queues, programs, and terminals that are monitored, the frequency with which monitored information is recorded, and the name of the file on which this information is recorded.
- The dump file, and how often or under what circumstances the copying is performed.

The various options available through ADL allow a user to tailor an application to a particular requirement, while providing flexibility in defining and modifying the external aspects of the application. ADL has a COBOL-like format and syntax enabling programmers familiar with COBOL to define an application easily. ADL uses a division, section, paragraph, and clause structure to allow a logical grouping of application entities, and it employs the COBOL coding format standards (discussed in the following subsection).

An ADL program has six divisions. All divisions are required, and they must appear in a specific order. The division names in the required order are as follows:

APPLICATION GLOBAL DIVISION

APPLICATION PROGRAM DIVISION

APPLICATION DATA DIVISION

SOURCE-DESTINATION DIVISION

QUEUE DIVISION

APPLICATION PROCESSING DIVISION

These divisions are each discussed separately later in this section.

LANGUAGE ELEMENTS

Several types of language elements are used in writing an application definition with ADL. Each type of language element has a specific use within an application definition.

RESERVED WORDS

A reserved word is an ADL word listed in appendix E that is used in ADL programs in a given context. Reserved words must not appear in an ADL program as user-defined names. Reserved words can be used only as specified in the ADL formats. Reserved words can be keywords or optional words in the formats.

Key Words

A keyword is a reserved word that is required when the format in which the word appears is used in an ADL program. Within each format, keywords are indicated by uppercase letters and underlining.

Optional Words

An optional word is a reserved word that is not required when the format in which the word appears is used in an ADL program. The presence of an optional word often adds to the understanding of the format, but does not affect execution. Within each format, uppercase letters without underlining indicate optional words.

USER-DEFINED NAMES

A user-defined name is a word that must be supplied by the programmer to satisfy the format of a paragraph or clause. User-defined names can be formed using the uppercase letters A through Z, the digits 0 through 9, and the hyphen. The maximum length of a user-defined name is 12 characters. The first character in the name must be alphabetic, and the last character of the name must not be a hyphen. All user-defined names must be unique within an application and cannot be reserved words.

Application Definition Language Names

ADL user-defined names are internal to an ADL program. The ADL names are data names and condition names.

Data Names

A data name is a name that represents a unit of storage that can assume different values. Data names are defined in the MESSAGE paragraphs of the Application Data Division.

Condition Names

A condition name is a name that assigns a specific value, set of values, or range of values to a data name. Condition names are defined in the CONDITION clauses of the Application Data Division.

MCS/COBOL Names

MCS/COBOL names represent elements that are known to both MCS and a COBOL program that uses MCS. This category of user-defined names includes queue names and symbolic names.

Queue Names

A queue name represents a message storage queue. Queue names are defined in the Queue Division.

Symbolic Names

A symbolic name represents a source (terminal), destination (output device), a group of sources or destinations, an interactive terminal that is both a source and a destination, an interprogram queue, or a journal file to which messages are written after they are enqueued and/or dequeued. Symbolic names are defined in the Source-Destination Division.

System Names

System names are used to communicate with the operating system. They can be from one to seven characters in length. They are formed using the uppercase letters A through Z, and the digits 0 through 9. The first character in the name must be alphabetic. System names cannot include a hyphen. User-defined system names are file names and routine names.

File Names

A file name represents a storage medium used by MCS in cooperation with the operating system to record relevant information.

Routine Names

A routine name represents an executable code module (program) external to MCS.

LITERALS

A literal is a character string whose value is implied by the ordered set of characters that make up the string. Three types of literals can be defined using ADL: integer literals, nonnumeric literals, and time literals.

Integers

An ADL integer is a numeric literal composed of a string of characters using the digits 0 through 9. ADL does not recognize signed numeric literals or those including a decimal point. The maximum value of an ADL integer is $2^{15}-1$ (32 767).

Nonnumeric Literals

A nonnumeric literal is a character string of 1 through 80 characters delimited on both ends by quotation marks. Any character in the 64-character display code set (see appendix A) can be part of a nonnumeric literal. Trailing blanks are not significant.

When a quotation mark is part of the nonnumeric literal, it must be represented as two contiguous quotation marks within the character string. Two quotation marks embedded between other characters produce a single quotation mark within the literal. An example of a nonnumeric literal with a quote is:

```
""EMBEDDED""""QUOTE""""
```

Produces the value "EMBEDDED" "QUOTE"

A character used in a source program as a quotation mark might appear on an output device as a different character, depending on the device and the character set. In particular, a quotation mark might appear as ≠ on a print file.

Time Literals

A time literal is a special type of nonnumeric literal that produces a value representing a chronological quantity. This chronological quantity is represented in the following form:

```
"hh.mm.ss"
```

where hh represents hours, mm represents minutes, and ss represents seconds. Time literals represent 24-hour clock time as opposed to 12-hour clock time. For example:

```
"23.23.23"
```

is 23 minutes and 23 seconds after 11 o'clock at night.

CODING FORMAT

Each line in an ADL program can be represented as an 80-column punch card. Columns define areas of varying significance within a line as follows:

Columns 73 through 80 contain optional program identification.

Columns 1 through 6 contain sequence numbers for the source line.

Columns 7 through 72 contain language statements. These columns are divided into three areas:

7 Indicator area

8 thru 11 Area A

12 thru 72 Area B

The indicator area can contain one of four characters that identify the type of line in which it occurs:

space	Normal line to be compiled.
-	Continuation line if a word, integer, or nonnumeric literal is split between lines.
*	Comment line listed as it is encountered.
/	Comment line that causes a page eject to occur for the source listing before the comment line is listed.

Area A is used to begin the following language elements:

Division header

Section header

Paragraph header

Area B is the starting area for:

Continuation of clauses

Clauses can begin in either area A or area B.

No punctuation, such as periods at the end of division headers, is allowed in ADL.

LANGUAGE STRUCTURE

Each of the six ADL divisions specifies a particular aspect of the application being defined. The remainder of this section is devoted to a complete description of each division.

APPLICATION GLOBAL DIVISION

The Application Global Division names an application and specifies certain aspects of it that apply to the entire application. This division provides for application validity checking, application dump files and monitor files, application initiation and testing parameters, and an application operator (AOP).

The division header is:

APPLICATION GLOBAL DIVISION

The header must appear on a separate line, beginning in area A. Figure 4-1 shows a skeleton of the Application Global Division. All paragraphs except the APPLICATION-NAME paragraph are optional. When any of the optional paragraphs of this division are used, they must appear in the order shown in figure 4-1.

```
APPLICATION GLOBAL DIVISION

APPLICATION-NAME paragraph

[SIGNATURE paragraph]

[DUMP-FILE paragraph]

[MONITOR-FILE paragraph]

[INJECTION-QUEUE paragraph]

[COLLECTION-QUEUE paragraph]

[OPERATOR paragraph]

[INITIATION paragraph]
```

Figure 4-1. Application Global Division Skeleton

APPLICATION-NAME Paragraph

The APPLICATION-NAME paragraph names the application. This paragraph must be present in an ADL program, and it must be the first paragraph in the Application Global Division. The APPLICATION-NAME paragraph must begin in area A. Figure 4-2 shows the format of the APPLICATION-NAME paragraph.

```
APPLICATION-NAME IS routine-name
```

Figure 4-2. APPLICATION-NAME Paragraph Format

Routine-name identifies the application to MCS during execution. Routine-name also identifies the application in a library containing numerous application definitions. Routine-name must be unique in a library of application definitions.

Routine-name can also be used as the name of a symbolic destination. This symbolic destination is used when a COBOL program sends commands to MCS. A program can send to MCS any of the commands listed in sections 7 and 8 except DATA, or the login or logout commands. MCS responses to these commands are enqueued in the response queue associated with the program (see the Application Program Division subsection for a discussion of the response queue).

SIGNATURE Paragraph

The SIGNATURE paragraph names a single password used for protection by the entire application. This paragraph is optional. When used, this paragraph must appear immediately after the APPLICATION-NAME paragraph. The SIGNATURE paragraph must begin in area A. Figure 4-3 shows the format of the SIGNATURE paragraph.

```
SIGNATURE IS nonnumeric-literal
```

Figure 4-3. SIGNATURE Paragraph Format

Nonnumeric-literal in this paragraph is the password for assigning application operator status and for verifying that an application can enable or disable a particular source, destination, or queue. Nonnumeric-literal can be a maximum of 10 characters in length.

When this paragraph is omitted, there is no password protection for the application except where a PASSWORD clause is present in the OPERATOR paragraph of the Application Global Division, in the Source-Destination Division, or in the Queue Division (see the subsections on these divisions). When the SIGNATURE paragraph is present and a PASSWORD clause is also present, the specific password from the PASSWORD clause is required to gain application operator status, or to enable or disable a particular source, destination, or queue.

DUMP-FILE Paragraph

The DUMP-FILE paragraph names a file to record application information about the status of internal tables used by MCS. See section 5 for more detailed information on the dump file.

This paragraph is optional. When used, it must appear in the order shown in figure 4-1. The DUMP-FILE paragraph must begin in area A. Figure 4-4 shows the format of the DUMP-FILE paragraph.

```
DUMP-FILE IS file-name

    [OWNER IS nonnumeric-literal]
```

Figure 4-4. DUMP-FILE Paragraph Format

When the DUMP-FILE paragraph is omitted, application dumps are not possible. When the OWNER phrase is present, nonnumeric-literal must be a system-validated user name. When the OWNER phrase is omitted, the system assumes MCS is the file owner; the system takes the user name from the MCS procedure file (see section 9 for information on the procedure file).

MONITOR-FILE Paragraph

The MONITOR-FILE paragraph names a file to which application status displays are written. The displays written to this file are the same as those transmitted to the application operator when the DISPLAY verb or command is used. The displays written to the monitor file include all terminals connected to the application, all input and/or output queues used by the application, and the status of all COBOL programs. See section 5 for more detailed information on the monitor file.

The MONITOR-FILE paragraph is optional. When used, it must appear in the order shown in figure 4-1. The MONITOR-FILE paragraph must begin in area A. Figure 4-5 shows the format of this paragraph.

```
MONITOR-FILE IS file-name

    [OWNER IS nonnumeric-literal]
```

Figure 4-5. MONITOR-FILE Paragraph Format

When the OWNER phrase is present, nonnumeric-literal must be a system-validated user name. When the OWNER phrase is omitted, the system assumes MCS is the file owner; the system takes the user name from the MCS procedure file (see section 9 for information on the procedure file).

INJECTION-QUEUE Paragraph

The INJECTION-QUEUE paragraph names a queue that is used for input messages when the application is in test mode. This paragraph is optional. When used, it must appear in the order shown in figure 4-1. The INJECTION-QUEUE paragraph must begin in area A. Figure 4-6 shows the format of this paragraph.

```
INJECTION-QUEUE IS queue-name
```

Figure 4-6. INJECTION-QUEUE Paragraph Format

Queue-name must refer to a simple input queue named in a QUEUE or SUB-QUEUE-n paragraph of the Input Section of the Queue Division. In test mode, MCS solicits the named queue for input messages. Thus, this queue is a substitute for input from a terminal. Messages stored in this queue are from a message generation program that sends messages to this queue. See section 5 for a discussion of test mode.

COLLECTION-QUEUE Paragraph

The COLLECTION-QUEUE paragraph names a queue for copying outbound messages when the application is in test mode. This paragraph is optional. When used, it must appear in the order shown in figure 4-1. The COLLECTION-QUEUE paragraph must begin in area A. Figure 4-7 shows the COLLECTION-QUEUE paragraph format.

```
COLLECTION-QUEUE IS queue-name
```

Figure 4-7. COLLECTION-QUEUE Paragraph Format

Queue-name must refer to a simple input queue named in a QUEUE or SUB-QUEUE-n paragraph of the Input Section of the Queue Division. In test mode, MCS copies messages bound for external destinations from output queues to the queue named in this paragraph. Thus, this queue is a substitute for a network terminal. An application output analysis program can be written to receive messages from this queue. See section 5 for a discussion of test mode.

OPERATOR Paragraph

The OPERATOR paragraph names a terminal or group of terminals that can be the application operator (AOP). The AOP is the controlling entity for the application; the AOP is the primary source of commands to MCS, and the primary destination of status messages from MCS. See section 8 for more information about the AOP.

The OPERATOR paragraph is optional. When used, it must appear in the order shown in figure 4-1. The OPERATOR paragraph must begin in area A. Figure 4-8 shows the OPERATOR paragraph format.

```
OPERATOR IS symbolic-name

[PASSWORD clause]
```

Figure 4-8. OPERATOR Paragraph Format

Symbolic-name is a source or destination named in a SYMBOLIC-NAME paragraph of the Source-Destination Division. Symbolic-name can also name a group of sources or destinations, as defined in an INVITATION-LIST or BROADCAST-LIST paragraph of the Source-Destination Division. Each of the named sources or destinations can be the AOP, but for each application there can be only one active AOP at a time. Symbolic-name cannot refer to a source or destination that is declared as dedicated in a SYMBOLIC-NAME paragraph of the Source-Destination Division.

PASSWORD Clause

The PASSWORD clause names a password used for validation of AOP status. The PASSWORD clause is optional. When the PASSWORD clause is omitted from the OPERATOR paragraph, the SIGNATURE paragraph must name an application password, or there can be no AOP for the application. See figure 4-9 for the format of this clause. Nonnumeric-literal can be a maximum of 10 characters in length.

```
PASSWORD IS nonnumeric-literal
```

Figure 4-9. PASSWORD Clause Format, Application Global Division

INITIATION Paragraph

The INITIATION paragraph defines how an application is started. This paragraph is optional. When used, it must be the last paragraph in the Application Global Division. The INITIATION paragraph must begin in area A. Figure 4-10 shows the format of this papagraph.

```
INITIATION IS { AUTOMATIC [TEST] }
              { EXPLICIT         }
```

Figure 4-10. INITIATION Paragraph Format

When the INITIATION paragraph specifies EXPLICIT, an AOP must be named in the OPERATOR paragraph. The application is then started when a terminal gains AOP status. The system console operator can also start the application when EXPLICIT is specified.

When the INITIATION paragraph specifies AUTOMATIC, the application is started at the same time that MCS is started. When this paragraph specifies AUTOMATIC TEST, the application is started in test mode at MCS startup time (see section 5 for a discussion of test mode).

When the INITIATION paragraph is omitted, the system assumes automatic. Explicit initiation is recommended, because active applications require central memory space at the MCS control point even when there are no application users connected.

Application Global Division Example

Figure 4-11 shows an example of the Application Global Division. The name of the application is EXAMPLE. MCSRM in the SIGNATURE paragraph is the password for verifying that EXAMPLE can enable or disable the sources and destinations named in the Source-Destination Division, and the queues named in the Queue Division. Password MCSRM does not assign AOP status, because there is an OPERATOR paragraph with a password.

```
APPLICATION GLOBAL DIVISION
APPLICATION-NAME IS EXAMPLE
SIGNATURE IS "MCSRM"
DUMP-FILE IS DMPF  OWNER IS "CDC123"
MONITOR-FILE IS MNTRF  OWNER IS "CDC123"
INJECTION-QUEUE IS INJECQ
COLLECTION-QUEUE IS COLLECQ
OPERATOR IS TERMS  PASSWORD IS "WRITER"
INITIATION IS EXPLICIT
```

Figure 4-11. Application Global Division Example

File DMPF is the file to which an application dump is written when the DUMP verb is used (see the Application Processing Division subsection for a discussion of the DUMP verb). CDC123 is a system-validated user name and names the owner of the dump file. This user name also appears in the MONITOR-FILE paragraph and names the owner of the monitor file. MNRTF is a file to which application status displays are written when the DISPLAY verb is used in the Application Processing Division.

The queue INJECQ is the input queue where incoming messages are enqueued when EXAMPLE is in test mode. INJECQ is defined in a QUEUE paragraph in the Queue Division of EXAMPLE. The queue COLLECQ is the input queue to which outbound messages are copied when EXAMPLE is in test mode. This queue is also defined in a QUEUE paragraph of the Queue Division.

The OPERATOR paragraph names TERMS as the group of terminals eligible for AOP status and WRITER as the password that must be used to gain this status. TERMS is defined as a group of terminals in an INVITATION-LIST paragraph in the Source-Destination Division of EXAMPLE. Any of the terminals in the group TERMS can be the AOP. MCS assigns AOP status to the first terminal from group TERMS that enters the password WRITER when selecting application EXAMPLE at MCS login time (see section 2 for login procedure). When one terminal from the group TERMS is active as the AOP, other terminals from the group can log into the application, but these other terminals cannot be the AOP. The AOP can control the application with any of the commands listed in section 8.

INITIATION IS EXPLICIT means that application EXAMPLE is started when a terminal from the group TERMS logs in with the password WRITER and becomes the AOP. The system console operator can also start EXAMPLE.

APPLICATION PROGRAM DIVISION

The Application Program Division names the COBOL programs in the application and specifies their interface with MCS. The division can be made up of a number of PROGRAM paragraphs, each PROGRAM paragraph naming a separate COBOL program. While the division is mandatory in an ADL program, the PROGRAM paragraph can be omitted. MCS supplies a default paragraph when the PROGRAM paragraph is omitted.

The division header is:

APPLICATION PROGRAM DIVISION

The header must appear on a separate line, beginning in area A. Figure 4-12 shows a skeleton of the Application Program Division.

```
APPLICATION PROGRAM DIVISION

[PROGRAM paragraph] . . .
```

Figure 4-12. Application Program Division Skeleton

PROGRAM Paragraph

The PROGRAM paragraph names a COBOL program. Every COBOL program that is part of the application must be named in a separate PROGRAM paragraph. If there are no programs in the application and, therefore, no PROGRAM paragraphs, MCS supplies a default program name that is the application name from the Application Global Division. The PROGRAM paragraph must begin in area A. Figure 4-13 shows the PROGRAM paragraph format. Routine-name must be the first seven characters of the COBOL program name from the COBOL PROGRAM-ID paragraph.

```
PROGRAM IS routine-name

    [INVOCATION-FILE clause]

    [RESPONSE-QUEUE clause]
```

Figure 4-13. PROGRAM Paragraph Format

INVOCATION-FILE Clause

The INVOCATION-FILE clause names a job submission file (invocation file) containing the control statements and job statements necessary to execute a COBOL program. This clause is optional. Figure 4-14 shows the format of the INVOCATION-FILE clause.

File-name must name an indirect access public permanent file. When the OWNER phrase is present, nonnumeric-literal must be a system-validated user name. When the OWNER phrase is omitted, the system assumes MCS is the file owner; the system takes the user name from the MCS procedure file (see section 9 for information on the procedure file).

```
INVOCATION-FILE IS file-name

    [OWNER IS nonnumeric-literal]
```

Figure 4-14. INVOCATION-FILE Clause Format

MCS submits the file denoted by file-name to the operating system job input queue when the COBOL program named in the associated PROGRAM paragraph is called by the INVOKE verb or the AOP INVOKE command. If this clause is omitted, the COBOL program named in the PROGRAM paragraph cannot be executed by use of the INVOKE verb or command. See section 5 for an invocation file example.

RESPONSE-QUEUE Clause

The RESPONSE-QUEUE clause names an input queue to store MCS responses to commands that a COBOL program can send to MCS. For example, the command DISPLAY ALL sent to MCS from a COBOL program causes the queue display described in section 7 in the subsection on the DISPLAY command to be stored in the queue named in this clause.

The RESPONSE-QUEUE clause is optional. Figure 4-15 shows the RESPONSE-QUEUE clause format. Queue-name must name a simple input queue defined in a QUEUE or SUB-QUEUE-n paragraph of the Input Section of the Queue Division.

```
RESPONSE-QUEUE IS queue-name
```

Figure 4-15. RESPONSE-QUEUE Clause Format

Application Program Division Example

Figure 4-16 shows an example of the Application Program Division. Application EXAMPLE of figure 4-11 has only one COBOL program associated with it, program MAWTEST. Program MAWTEST can receive messages for processing, and this program can send processed messages to a terminal using MCS. Execution of MAWTEST can be initiated by MCS. When INVOKE MAWTEST is used in the Application Processing Division, file INVF is submitted to the operating system job input queue. INVF contains control statements necessary to execute MAWTEST. CDC123 is a system-validated user name that names the invocation file owner.

```
APPLICATION PROGRAM DIVISION
PROGRAM IS MAWTEST
    INVOCATION-FILE IS INVF
        OWNER IS "CDC123"
    RESPONSE-QUEUE IS RESQ
```

Figure 4-16. Application Program Division Example

The queue RESQ is a simple input queue named in a QUEUE paragraph of the Queue Division. RESQ stores MCS responses to commands from MAWTEST.

APPLICATION DATA DIVISION

The Application Data Division describes input data that is enqueued based on the contents of the message, sets up a field within a message to store a number to identify each message, and provides for specification of an EGI. The division can consist of two types of paragraphs: the EGI paragraph and the MESSAGE paragraph. Both types of paragraphs are optional. When the EGI paragraph is present, it must appear before any MESSAGE paragraphs.

The division header is:

APPLICATION DATA DIVISION

The header must appear on a separate line, beginning in area A. Figure 4-17 shows a skeleton of the Application Data Division.

```
APPLICATION DATA DIVISION

[EGI paragraph]

[MESSAGE paragraph] . . .
```

Figure 4-17. Application Data Division Skeleton

EGI Paragraph

The EGI paragraph specifies an application end-of-group indicator that signals the end of a group of messages to MCS or to a COBOL program. This paragraph is optional. When used, the EGI paragraph must appear before any MESSAGE paragraphs, and it must begin in area A. If the EGI paragraph is omitted, messages input to a COBOL program cannot be associated with an end-of-group indicator, and messages that are output to a terminal with an end-of-group indicator are equivalent to messages output with an end-of-message indicator (EMI). Figure 4-18 shows the EGI paragraph format.

```
EGI IS nonnumeric-literal
```

Figure 4-18. EGI Paragraph Format

Nonnumeric-literal in this paragraph must be one character in length. Nonnumeric-literal can be any character in the 64-character display code set (see appendix A).

MESSAGE Paragraph

The MESSAGE paragraph describes the format of message data that is enqueued based on the contents of the message. Incoming message text is tested against the descriptions in MESSAGE paragraphs before being routed to the proper queue. This paragraph is also used to describe a field within a message to store an identifying number assigned for each message. There must be a separate MESSAGE paragraph for each type of application message that is enqueued based on contents. For example, an inventory control application might include one type of message updating number of parts on hand that is enqueued based on the first three characters of the part number, and another type of message tracking inventory at each of several locations that is enqueued based on a location code. This paragraph is optional. When used, the MESSAGE paragraph must begin in area A. Figure 4-19 shows the MESSAGE paragraph format.

```
MESSAGE IS data-name

[SERIAL-NUMBER clause]

[SEGMENT paragraph] . . .
```

Figure 4-19. MESSAGE Paragraph Format

SERIAL-NUMBER Clause

The SERIAL-NUMBER clause specifies how a serial number to identify each input message is assigned. This clause is optional. When the SERIAL-NUMBER clause is omitted, no serial numbers are assigned to messages. Figure 4-20 shows the SERIAL-NUMBER clause format.

```
SERIAL-NUMBER IS { SUPPLIED   }
                 { GENERATED  }

[IN data-name] [WITH ECHO]
```

Figure 4-20. SERIAL-NUMBER Clause Format

When the SERIAL-NUMBER clause specifies SUPPLIED, the message originator must include the serial number as part of the message text. The serial number must contain only the digits 0 through 9 and must not exceed $2^{30}-1$ (1 073 741 823). When SUPPLIED is specified, the IN phrase must be used. Data-name must be a field within the message text; the length of the field determines the largest serial number that can be supplied ($2^{30}-1$ requires a field 10 characters in length). MCS does not validate supplied serial numbers; two messages can have the same serial number.

When this clause specifies GENERATED, MCS assigns a serial number to the message. MCS assigns serial numbers on an application basis beginning with 1 for each application with a maximum of $2^{30}-1$. When GENERATED is specified, the IN phrase is optional. When this phrase is used, MCS stores the serial number in data-name; the data-name field must be large enough to contain the serial number. When the IN phrase is omitted, the serial number is not part of the message text and is not available to message recipients or a COBOL program.

The WITH ECHO phrase can be included in the SERIAL-NUMBER clause. When this phrase is used, the serial number assigned to the message is echoed back to the message source. No echoing occurs when this phrase is omitted.

SEGMENT Paragraph

The SEGMENT paragraph describes the data of a message segment. This paragraph is a subdivision of a MESSAGE paragraph, and a SEGMENT paragraph is associated with the preceding MESSAGE paragraph. The SEGMENT paragraph must begin in area A. There must be a separate SEGMENT paragraph for each segment of an input message that is enqueued based on contents. Figure 4-21 shows the SEGMENT paragraph format.

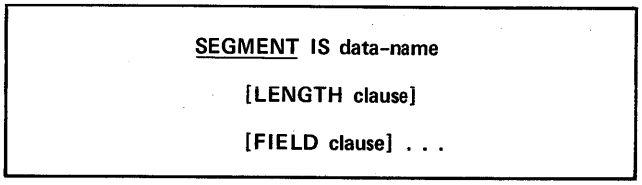


Figure 4-21. SEGMENT Paragraph Format

SEGMENT Paragraph Syntax

The SEGMENT paragraph can include three clauses that describe the format of the message segment and that assign values on which routing decisions are made. These clauses are the LENGTH clause, the FIELD clause, and the CONDITION clause.

The LENGTH clause defines the length of a message segment. This clause is optional. When the LENGTH clause is present, it provides a cross-check of the format and contents specified in FIELD and CONDITION clauses. Figure 4-22 shows the LENGTH clause format.

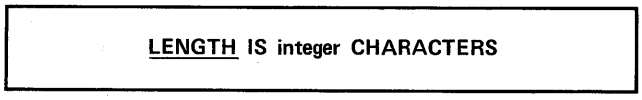


Figure 4-22. LENGTH Clause Format

The FIELD clause describes a field of a message segment. A FIELD clause is included for the data fields in a message segment that store serial numbers or contain values on which routing decisions are based. Figure 4-23 shows the FIELD clause format.

The STARTS phrase defines the beginning of the field within the message segment. When this phrase is omitted, MCS assumes the field begins at the first character of the message segment named in the associated SEGMENT paragraph. STARTS AT CHARACTER defines the field as beginning at the character position denoted by integer-1. Integer-1 cannot be greater than the length of the segment. For example:

```
FIELD IS FIELD1
STARTS AT CHARACTER 5
```

means that the field named FIELD1 begins at the fifth character of the message segment. The STARTS WITH or AFTER options define the field as beginning with or after the integer-2 occurrence of nonnumeric-literal-1 within the segment. Nonnumeric-literal-1 must be one character in length. For example:

```
FIELD IS FIELD2
STARTS WITH INSTANCE 1 OF ","
```

means that the field named FIELD2 begins with the first comma in the message segment, and

```
FIELD IS FIELD3
STARTS AFTER INSTANCE 2 OF ":"
```

means that the field named FIELD3 begins at the character after the second colon in the message segment.

The EXTENDS phrase defines the end of the field within the message segment. When this phrase is omitted, MCS assumes the field ends at the last character of the message segment named in the associated SEGMENT paragraph. EXTENDS FOR defines the field as having a length of integer-3 characters. For example:

```
FIELD IS DATAM
EXTENDS FOR 12 CHARACTERS
```

means that the field named DATAM is 12 characters in length. EXTENDS TO CHARACTER defines the field as ending at the character position denoted by integer-4. Integer-4 cannot be greater than the length of the segment. For example:

```
FIELD IS DATAN
EXTENDS TO CHARACTER 12
```

means that the field named DATAN ends at the twelfth character of the message segment.

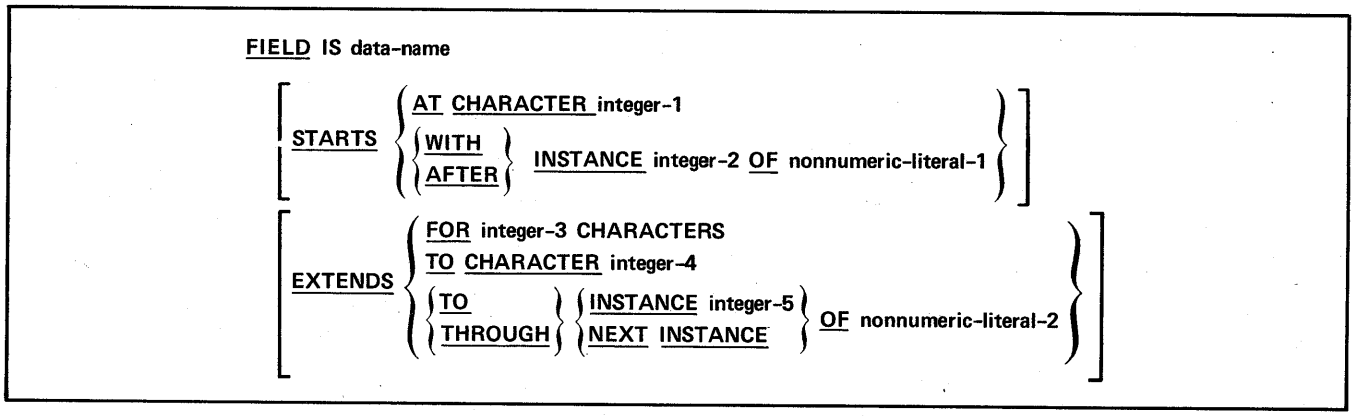


Figure 4-23. FIELD Clause Format

The TO or THROUGH options of the EXTENDS phrase define the field as ending at (TO) or after (THROUGH) the integer-5 occurrence of nonnumeric-literal-2 within the segment. Nonnumeric-literal-2 must be one character in length. For example:

```
FIELD IS DATAP
EXTENDS TO INSTANCE 1 OF ","
```

means that the field named DATAP ends at the first comma in the message segment, and

```
FIELD IS DATAR
EXTENDS THROUGH INSTANCE 2 OF "."
```

means that the field named DATAR ends after the second period in the message segment. NEXT defines the field as ending at or after the next occurrence of nonnumeric-literal-2 within the segment. For example:

```
FIELD IS DATAS
EXTENDS TO NEXT INSTANCE OF ","
```

means that the field named DATAS ends at the next comma in the segment, and

```
FIELD IS DATAT
EXTENDS THROUGH NEXT INSTANCE OF "."
```

means that the field named DATAT ends after the next period in the segment.

When both the STARTS phrase and the EXTENDS phrase include the INSTANCE option, and nonnumeric-literal-1 and nonnumeric-literal-2 are the same character, integer-2 must be less than integer-5. For example:

```
FIELD IS DATAZ
STARTS AFTER INSTANCE 1 OF ","
EXTENDS TO INSTANCE 3 OF ","
```

means that the field named DATAZ begins with the character after the first comma in the segment; the field ends at the third comma in the segment.

The CONDITION clause defines a value, set of values, or range of values against which a message field is tested. The clause is optional. When used, the clause defines a condition or conditions to test data-name of the preceding FIELD clause. Figure 4-24 shows the CONDITION clause format.

Each literal named in a CONDITION clause must be distinct. The literals can be nonnumeric when a value or set of values (not a range of values) is specified. The nonnumeric literals can be a maximum of 10 characters in length. When a range of values is specified by use of the THROUGH option, the literals must be integers, and the beginning of the range must be less than the end of the range (in figure 4-24, literal-2 must be less than literal-3, and literal-5 must be less than literal-6). The values specified in a CONDITION clause must fit in the message field named in the associated FIELD clause.

Comparison of the value of the FIELD clause data-name with the value or values defined in a CONDITION clause is based on actual algebraic value when integer values are specified in the CONDITION clause. When the CONDITION clause specifies nonnumeric values, the comparison is based on the installation-defined collating sequence (see appendix A). If data-name and the nonnumeric value are of unequal size, the shorter of the two is extended on the right by blanks.

Application Data Division Example

Figure 4-25 shows an example of the Application Data Division. This division is part of application EXAMPLE of figure 4-11.

The end indicator for a group of messages in this application is defined in the EGI paragraph as an asterisk (*). An asterisk in message text then signals the end of a group of messages to MCS or to the COBOL program MAWTEST.

Application EXAMPLE consists of two types of messages that are routed based on message content. Both MSG1 and MSG2 also contain fields to store serial numbers. MCS assigns serial numbers to messages associated with MSG1, and these numbers are stored in FIELD11. Messages associated with MSG2 are also assigned serial numbers by MCS, and these numbers are stored in FIELD21. In both cases, the serial numbers are echoed back to the source.

In the first MESSAGE paragraph in this example, segment SEG1 is defined as 15 characters in length. The first field in this segment is FIELD11, which begins at the first character of the segment and is 10 characters in length. MCS stores serial numbers assigned to MSG1 messages in FIELD11. The second field in SEG1 is named FIELD12, and it begins at the eleventh character of the segment and is five characters in length. FIELD12 has an associated CONDITION clause. Incoming FIELD12 message data is tested against the value FIVE5 of the CONDITION clause.

In the second MESSAGE paragraph in this example, segment SEG2 is defined as 20 characters in length. The first field in this segment is FIELD21, which begins at the first character of SEG2 and is 10 characters in length. MCS stores serial numbers assigned to MSG2 messages in FIELD21. The second field in SEG2 is FIELD22. This field begins at the eleventh character of the segment and ends after the first comma appearing in SEG2. Message data in FIELD22 is tested against the values 123 and 456, defined in the two condition clauses associated with this field. SEG2 has a third field, FIELD23, which begins with the character after the first comma in the segment. FIELD23 is six characters in length. FIELD23 message data is tested against the values assigned to condition name C-4 in the CONDITION clause.

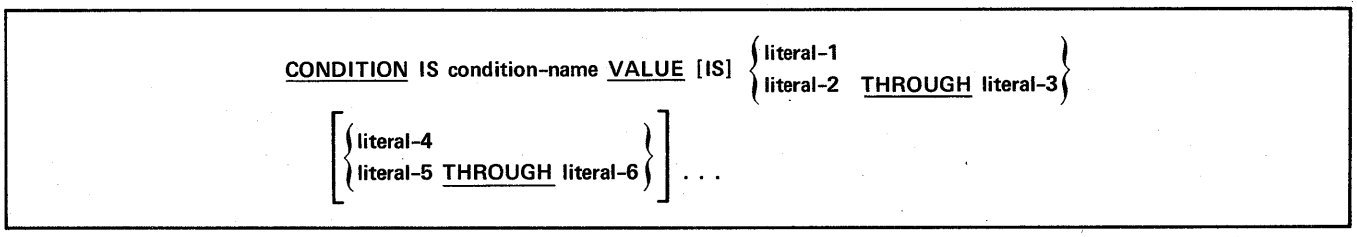


Figure 4-24. CONDITION Clause Format

```
APPLICATION DATA DIVISION
EGI IS "*"
MESSAGE IS MSG1
    SERIAL-NUMBER IS GENERATED IN FIELD11 WITH ECHO
SEGMENT IS SEG1
    LENGTH IS 15 CHARACTERS
    FIELD IS FIELD11
        STARTS AT CHARACTER 1
        EXTENDS FOR 10 CHARACTERS
    FIELD IS FIELD12
        STARTS AT CHARACTER 11
        EXTENDS FOR 5 CHARACTERS
        CONDITION IS C-NAME VALUE "FIVE5"
MESSAGE IS MSG2
        SERIAL-NUMBER IS GENERATED IN FIELD21 WITH ECHO
SEGMENT IS SEG2
    LENGTH IS 20 CHARACTERS
    FIELD IS FIELD21
        STARTS AT CHARACTER 1
        EXTENDS FOR 10 CHARACTERS
    FIELD IS FIELD22
        STARTS AT CHARACTER 11
        EXTENDS THROUGH INSTANCE 1 OF ","
        CONDITION IS C-2 VALUE "123,"
        CONDITION IS C-3 VALUE "456,"
    FIELD IS FIELD23
        STARTS AFTER INSTANCE 1 OF ","
        EXTENDS FOR 6 CHARACTERS
        CONDITION IS C-4 VALUE "AB/CD." "EF/GH."
```

Figure 4-25. Application Data Division Example

SOURCE-DESTINATION DIVISION

The Source-Destination Division specifies names that represent the sources (terminals), destinations (output devices), interprogram queues, and journal files an application uses. These names are called symbolic names by ADL. This division also describes the characteristics of the sources and destinations, and the MCS interface with them. This division is made up of three types of paragraphs: the SYMBOLIC-NAME, INVITATION-LIST, and BROADCAST-LIST paragraphs. At least one SYMBOLIC-NAME paragraph must appear in this division. Any other paragraphs are included at user option. The paragraphs can appear in any order.

The division header is:

SOURCE-DESTINATION DIVISION

The header must appear on a separate line, beginning in area A. Figure 4-26 shows a skeleton of the Source-Destination Division.

```
SOURCE-DESTINATION DIVISION

SYMBOLIC-NAME paragraph . . .

[INVITATION-LIST paragraph] . . .

[BROADCAST-LIST paragraph] . . .
```

Figure 4-26. Source-Destination Division Skeleton

SYMBOLIC-NAME Paragraph

The SYMBOLIC-NAME paragraph specifies symbolic names for the application sources, destinations, interprogram queues, and journal files. At least one SYMBOLIC-NAME paragraph must appear in an ADL program. The SYMBOLIC-NAME paragraph must begin in area A. Figure 4-27 shows the SYMBOLIC-NAME paragraph format.

```
SYMBOLIC-NAME IS symbolic-name

    [TYPE clause]

    [ALIAS clause]

    [MESSAGES clause]

    [MODE clause]

    [PASSWORD clause]

    [STATUS clause]
```

Figure 4-27. SYMBOLIC-NAME Paragraph Format

Symbolic-name must be a unique name within an application. The TYPE clause, if used, must appear first, but the other clauses can appear in any order. The clauses are described in this subsection following the paragraph descriptions.

INVITATION-LIST Paragraph

The INVITATION-LIST paragraph names a group of sources. A reference to an invitation list refers to all the sources named in the INVITATION-LIST paragraph. This paragraph provides a shorthand method of specifying characteristics for a number of symbolic sources. This paragraph also provides a way for MCS to reference a number of symbolic sources collectively. For example, MCS can reference a list in a USE paragraph in the Application Processing Division, and a command can reference a list; ENABLE list-name enables all the source terminals in the list.

This paragraph is optional. When used, it must begin in area A. Figure 4-28 shows the format of the INVITATION-LIST paragraph.

```
INVITATION-LIST IS symbolic-name-1

SOURCES ARE symbolic-name-2

                [AND symbolic-name-3] . . .

    [MESSAGES clause]

    [MODE clause]

    [PASSWORD clause]

    [STATUS clause]
```

Figure 4-28. INVITATION-LIST Paragraph Format

Symbolic-name-1 is the collective name used to reference all the sources in the list. Symbolic-name-2, symbolic-name-3, and so forth are the component sources of the list. Each component source must be defined as a source or as interactive in a separate SYMBOLIC-NAME paragraph.

The characteristics named in the clauses associated with an INVITATION-LIST paragraph apply to each of the component sources. The same type of clause cannot be used in an INVITATION-LIST paragraph and in the SYMBOLIC-NAME paragraphs that define symbolic-name-2, symbolic-name-3, and the other component sources of the list. For example, if the INVITATION-LIST paragraph has a STATUS clause associated with it, the SYMBOLIC-NAME paragraph that defines symbolic-name-2 cannot have a STATUS clause.

The clauses associated with an INVITATION-LIST paragraph can appear in any order. The clauses are described in this subsection following the paragraph descriptions.

BROADCAST-LIST Paragraph

The BROADCAST-LIST paragraph names a group of destinations. A reference to a broadcast list refers to all the destinations named in the BROADCAST-LIST paragraph. This paragraph provides a shorthand method of specifying characteristics for a number of symbolic destinations. This paragraph also provides a way for MCS or a COBOL program to reference a number of symbolic destinations collectively. For example, MCS can reference the list in a USE paragraph in the Application Processing Division, and a COBOL program SEND to a broadcast list delivers the message text to all logged in destinations included in the list.

This paragraph is optional. When used, it must begin in area A. Figure 4-29 shows the BROADCAST-LIST paragraph format.

```
BROADCAST-LIST IS symbolic-name-1

DESTINATIONS ARE symbolic-name-2

                [AND symbolic-name-3] . . .

    [MESSAGES clause]

    [MODE clause]

    [PASSWORD clause]

    [STATUS clause]
```

Figure 4-29. BROADCAST-LIST Paragraph Format

Symbolic-name-1 is the collective name used to reference all the destinations in the list. Symbolic-name-2, symbolic-name-3, and so forth are the component destinations of the list. Each component destination must be defined as a destination or as interactive in a separate SYMBOLIC-NAME paragraph.

The characteristics named in the clauses associated with a BROADCAST-LIST paragraph apply to each of the component destinations. The same type of clause cannot be used in a BROADCAST-LIST paragraph and in the SYMBOLIC-NAME paragraphs that define symbolic-name-2, symbolic-name-3, and the other component destinations of the list. For example, if the BROADCAST-LIST paragraph has a MODE clause associated with it, the SYMBOLIC-NAME paragraph that defines symbolic-name-2 cannot have a MODE clause.

The clauses associated with a BROADCAST-LIST paragraph can appear in any order. The clauses are described in the following paragraphs.

Source-Destination Division Clauses

The clauses described in this subsection are used in the Source-Destination Division paragraphs. The clauses are: TYPE clause, ALIAS clause, MESSAGES clause, MODE clause, PASSWORD clause, and STATUS clause. These clauses can appear in the Source-Destination Division paragraphs as shown in table 4-1.

TYPE Clause

The TYPE clause defines a symbolic name as representing a certain type of source or destination. A symbolic name can represent an external source (SOURCE), an external destination (DESTINATION), both an external source and an external destination (INTERACTIVE), an application journal (JOURNAL), or an interprogram queue (QUEUE).

TABLE 4-1. SOURCE-DESTINATION DIVISION CLAUSE USAGE

Clause	BROADCAST-LIST Paragraph	INVITATION-LIST Paragraph	SYMBOLIC-NAME Paragraph
TYPE Clause			X
ALIAS Clause			X
MESSAGES Clause	X	X	X
MODE Clause	X	X	X
PASSWORD Clause	X	X	X
STATUS Clause	X	X	X
X - Clause can appear in this paragraph			

The TYPE clause is associated with the SYMBOLIC-NAME paragraph. The clause is optional. When the TYPE clause is omitted, INTERACTIVE is assumed. When the TYPE clause is used, it must be the first clause in a SYMBOLIC-NAME paragraph. Figure 4-30 shows the TYPE clause format.

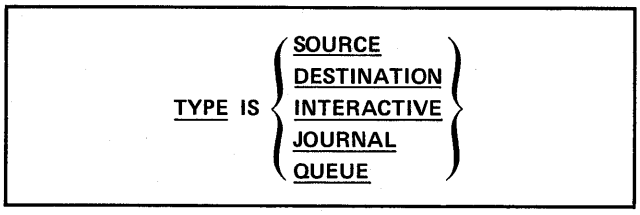


Figure 4-30. TYPE Clause Format

When the TYPE clause specifies SOURCE, symbolic-name of the associated SYMBOLIC-NAME paragraph represents a source external to the application that can input messages to the application. When this clause specifies DESTINATION, symbolic-name represents a destination external to the application that can receive messages from the application. When this clause specifies INTERACTIVE, symbolic-name represents a device external to the application that can both input messages to the application and receive messages from the application.

When the TYPE clause specifies JOURNAL, symbolic-name of the associated SYMBOLIC-NAME paragraph represents an application journal file to which messages are copied after they are enqueued or dequeued. Symbolic-name also represents a destination to which COBOL programs can send messages. When this clause specifies QUEUE, symbolic-name represents a destination that is an interprogram queue.

ALIAS Clause

The ALIAS clause relates symbolic-name of a SYMBOLIC-NAME paragraph to some other application entity by giving another name by which symbolic-name is known either to the application itself or to the network. Figure 4-31 shows the ALIAS clause format.

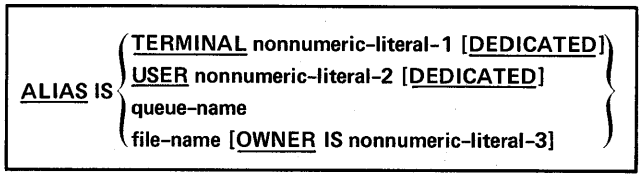


Figure 4-31. ALIAS Clause Format

The TERMINAL phrase of the ALIAS clause establishes a relationship between a symbolic name and a network terminal for automatic login purposes. The terminal option can be used only when the symbolic name is declared as SOURCE, DESTINATION, or INTERACTIVE in the TYPE clause. Nonnumeric-literal-1 must be a legitimate network terminal identification from the local configuration file. For example:

```
SYMBOLIC-NAME IS TERM1
TYPE IS INTERACTIVE
ALIAS IS TERMINAL TM104
```

means that symbolic name TERM1 represents the interactive network terminal named TM104 in the local configuration file. When a user of TM104 selects this application, MCS automatically assigns symbolic name TERM1. When DEDICATED is included in the TERMINAL phrase, this relationship between a symbolic name and a network terminal applies to all running applications in an application definition library. When a user selects network application MCS, MCS searches all running MCS applications for terminals declared as DEDICATED. When the name of the network terminal matches the terminal name in an ALIAS clause, MCS automatically logs the terminal into the corresponding application and assigns the symbolic name specified in the SYMBOLIC-NAME paragraph. A network terminal should not be declared as DEDICATED in more than one application; undesirable results may occur.

The USER phrase of the ALIAS clause establishes a relationship between a symbolic name and a user. The USER option can be used only when the symbolic name is declared as SOURCE, DESTINATION, or INTERACTIVE in

the TYPE clause. Nonnumeric-literal-2 must be a system-validated user name. For example:

```
SYMBOLIC-NAME IS TERM1
TYPE IS INTERACTIVE
ALIAS IS USER "MAW"
```

means that symbolic name TERM1 represents system user MAW. When user MAW logs in and selects network application MCS, MCS automatically logs MAW in as TERM1. This relationship applies only to the specific application selected at login time unless DEDICATED is included in the USER phrase. When DEDICATED is included, the relationship between a symbolic name and a system user applies to all running applications in an application definition library. When a user selects network application MCS, MCS searches all running MCS applications for users declared as DEDICATED. When the name of the user logging in matches the user name in an ALIAS clause, MCS automatically logs the user into the corresponding application and assigns the symbolic name specified in the SYMBOLIC-NAME paragraph.

The ALIAS IS queue-name option establishes a relationship between a symbolic name and an interprogram queue. The ALIAS clause must be used when the symbolic name is declared as QUEUE in the TYPE clause, because a symbolic name and a queue name cannot be the same within an application. Queue-name of this option must be a simple input queue defined in a QUEUE or SUB-QUEUE-n paragraph of the Input Section of the Queue Division. For example:

```
SYMBOLIC-NAME IS INTERQ
TYPE IS QUEUE
ALIAS IS PROGAQ
```

means that symbolic name INTERQ represents an interprogram queue that is named PROGAQ in the Queue Division.

The ALIAS IS file-name option establishes a relationship between a symbolic name and an application journal file to which messages are copied after they are enqueued or dequeued. The ALIAS clause must be used when the symbolic name is declared as JOURNAL in the TYPE clause to define the file MCS uses as the journal. When the OWNER phrase is present with this option, nonnumeric-literal-3 must be a system-validated user name. When the OWNER phrase is omitted, the system assumes MCS is the file owner; the system takes the user name from the MCS procedure file. An example of this option is:

```
SYMBOLIC-NAME IS JRNFL
TYPE IS JOURNAL
ALIAS IS FILE1
```

where symbolic name JRNFL represents a destination that is an application journal. The file name for this journal is FILE1.

The ALIAS clause can be omitted when the TYPE clause of the SYMBOLIC-NAME paragraph defines the symbolic name as SOURCE, DESTINATION, or INTERACTIVE. The symbolic name is then said to be transient. For example:

```
SYMBOLIC-NAME IS TERM1
TYPE IS INTERACTIVE
```

means that symbolic name TERM1 represents some interactive terminal that can both send and receive messages. No alias is given to associate the symbolic name TERM1 with an actual network terminal. Any terminal in the network can log in, say it is TERM1, and send or receive application messages.

MESSAGES Clause

The MESSAGES clause associates a particular source or destination with a message format defined in a MESSAGE paragraph of the Application Data Division. The MESSAGES clause is optional. When used, this clause can appear in a SYMBOLIC-NAME, INVITATION-LIST, or BROADCAST-LIST paragraph. Figure 4-32 shows the MESSAGES clause format.

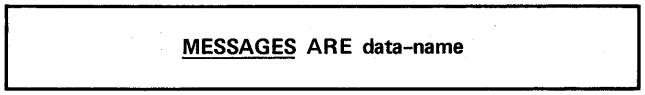


Figure 4-32. MESSAGES Clause Format

Data-name must be defined in a MESSAGE paragraph of the Application Data Division. The MESSAGES clause can be used only when the source or destination with which it is associated is declared as SOURCE, DESTINATION, or INTERACTIVE in the TYPE clause. The MESSAGES clause cannot be used when the source or destination is declared as JOURNAL or QUEUE in the TYPE clause.

MODE Clause

The MODE clause specifies the mode a terminal is in when login is complete. This clause is optional and can be used only when the symbolic name is declared as SOURCE, DESTINATION, or INTERACTIVE in the TYPE clause. This clause can appear in a SYMBOLIC-NAME, INVITATION-LIST, or BROADCAST-LIST paragraph. When the MODE clause is omitted, MCS assumes the terminal is in command mode after login is complete. Figure 4-33 shows the MODE clause format.

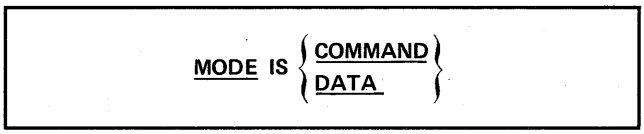


Figure 4-33. MODE Clause Format

MODE IS COMMAND means that MCS initially accepts as input any of the commands listed in section 7, or section 8 when the terminal is the AOP. MODE IS DATA means that MCS initially accepts as input actual message data that is routed to an input queue.

PASSWORD Clause

The PASSWORD clause names a password used for validation when a COBOL program attempts to enable or disable a source or destination. This clause is optional. When used, the PASSWORD clause can appear in a SYMBOLIC-NAME, INVITATION-LIST, or BROADCAST-LIST paragraph. When this clause is omitted, password checking does not take place unless there is a SIGNATURE paragraph in the Application Global Division. Figure 4-34 shows the PASSWORD clause format. Nonnumeric-literal can be a maximum of 10 characters in length.

```
PASSWORD IS nonnumeric-literal
```

Figure 4-34. PASSWORD Clause Format, Source-Destination Division

STATUS Clause

The STATUS clause specifies that a source or destination is enabled or disabled when the application is initiated. This clause is optional and can be used only when the symbolic name is declared as SOURCE, DESTINATION, or INTERACTIVE in the TYPE clause. The STATUS clause can appear in a SYMBOLIC-NAME, INVITATION-LIST, or BROADCAST-LIST paragraph. When the STATUS clause is omitted, MCS assumes the source or destination is enabled when the application is initiated. Figure 4-35 shows the STATUS clause format.

```
STATUS IS { ENABLED  }
          { DISABLED }
```

Figure 4-35. STATUS Clause Format, Source-Destination Division

STATUS IS ENABLED means that the source or destination is able to input and receive message data when the application is initiated. STATUS IS DISABLED means that the source or destination is not able to input or receive message data when the application is initiated.

Source-Destination Division Example

Figure 4-36 shows an example of the Source-Destination Division. This division is part of application EXAMPLE of figure 4-11.

```
SOURCE-DESTINATION DIVISION

INVITATION-LIST IS TERMS
SOURCES ARE TERM1 AND TERM2
    MODE IS COMMAND
    STATUS IS DISABLED
BROADCAST-LIST IS OUTPUTS
DESTINATIONS ARE TERM1 AND TERM2
SYMBOLIC-NAME IS TERM1
    TYPE IS INTERACTIVE
    ALIAS IS USER "CDC123"
SYMBOLIC-NAME IS TERM2
    TYPE IS INTERACTIVE
    ALIAS IS USER "CDC123"
SYMBOLIC-NAME IS JRNL1
    TYPE IS JOURNAL
    ALIAS IS FILE1  OWNER IS "CDC123"
SYMBOLIC-NAME IS JRNL2
    TYPE IS JOURNAL
    ALIAS IS FILE2  OWNER IS "CDC123"
```

Figure 4-36. Source-Destination Division Example

Application EXAMPLE can get message text from the terminals represented in the INVITATION-LIST paragraph. These terminals are given the collective name TERMS. The list TERMS is made up of two terminals, represented by the names TERM1 and TERM2. TERMS also represents the AOP, as defined in the OPERATOR paragraph of the Application Global Division. When either TERM1 or TERM2 logs into EXAMPLE with the password WRITER, this terminal is the AOP. TERM1 and TERM2 cannot both be the AOP at the same time. When application EXAMPLE is initiated, TERM1 and TERM2 are in command mode, and they are not able to input or receive message data (STATUS IS DISABLED).

Application EXAMPLE can deliver message text to the destinations represented in the BROADCAST-LIST paragraph. These terminals are given the collective name OUTPUTS. The list OUTPUTS is made up of two terminals, represented by the names TERM1 and TERM2. These two symbolic names are the same as those given the two terminals of the invitation list. TERM1 and TERM2 are defined as INTERACTIVE in the TYPE clause of the SYMBOLIC-NAME paragraphs that describe TERM1 and TERM2; TERM1 and TERM2 are destinations of messages as well as sources of messages. The clauses of the INVITATION-LIST paragraph also apply to the BROADCAST-LIST paragraph, because the same symbolic names are in the list.

In addition to describing the terminals represented by TERM1 and TERM2 as interactive, the SYMBOLIC-NAME paragraphs that define these terminals declare that TERM1 and TERM2 are also known to the application and the network as CDC123. This establishes a correspondence between the two symbolic names representing terminals and the user CDC123. When user CDC123 logs in, MCS assigns symbolic name TERM1. When a second user CDC123 logs in, MCS assigns symbolic name TERM2.

The other two SYMBOLIC-NAME paragraphs define JRNL1 and JRNL2 as journals onto which messages are written after they are enqueued or dequeued. The ALIAS clauses in these paragraphs define the file names for these journals as FILE1 and FILE2. The OWNER phrases specify the owner of FILE1 and FILE2 as CDC123.

QUEUE DIVISION

The Queue Division names and describes the input and output queues that an application uses. This division also specifies the criteria MCS uses for message disposition and message routing. The Queue Division is subdivided into three sections. The Input Section and the Output Section name the input and output queues, respectively, and specify the characteristics of these queues. The Routing Section specifies the relationships between input queues and the symbolic sources that input messages stored in these queues, and the relationships between output queues and the symbolic destinations that receive messages from these queues.

The division header is:

```
QUEUE DIVISION
```

The header must appear on a separate line, beginning in area A. Figure 4-37 shows a skeleton of the Queue Division.

```
QUEUE DIVISION

INPUT SECTION

OUTPUT SECTION

ROUTING SECTION
```

Figure 4-37. Queue Division Skeleton

The three sections of the Queue Division are required in an ADL program, and they must appear in the order shown in figure 4-37. At least one queue must be defined; a QUEUE paragraph must appear in either the Input Section or the Output Section.

Input Section

The Input Section names the input queues and specifies the characteristics of these queues. The section is made up of a QUEUE paragraph for each input queue. Each QUEUE paragraph can have any number of SUB-QUEUE-n paragraphs associated with it to define the input queue hierarchy.

The Input Section must begin in area A. Figure 4-38 shows the Input Section format.

```
INPUT SECTION

[ QUEUE paragraph                    ]
[ [SUB-QUEUE-n paragraph] . . .      ] . . .
```

Figure 4-38. Input Section Format

QUEUE Paragraph of the Input Section

The QUEUE paragraph of the Input Section names an input queue and specifies the queue characteristics. This paragraph is omitted if input queues are not used by the application. When present, the QUEUE paragraph must begin in area A. Figure 4-39 shows the QUEUE paragraph format.

```
QUEUE IS queue-name
    [JOURNAL clause]
    [MEDIUM clause]
    [PASSWORD clause]
    [RESIDENCY clause]
    [STATUS clause]
```

Figure 4-39. QUEUE Paragraph Format, Input Section

The clauses associated with a QUEUE paragraph can appear in any order. Clauses specified at the queue level apply to a subqueue of that queue only when the same clause is not specified for the subqueue. For example:

```
QUEUE IS A
STATUS IS ENABLED
SUB-QUEUE-1 IS SUB1A
```

names a queue and a subqueue of that queue, and the STATUS clause applies to both the queue and the subqueue, and

```
QUEUE IS A
STATUS IS ENABLED
SUB-QUEUE-1 IS SUB1A
STATUS IS DISABLED
```

names a queue and a subqueue of that queue, and STATUS IS ENABLED applies to the queue only. The subqueue is initially disabled. The clauses are discussed in this subsection following the paragraph descriptions.

SUB-QUEUE-n Paragraph

The SUB-QUEUE-n paragraph specifies the input queue hierarchy. Each input queue named in a QUEUE paragraph can have as many as three levels of subqueues associated with it, resulting in a four-level compound queue structure. Each subqueue must be named, and its characteristics defined, in a SUB-QUEUE-n paragraph. The n represents the subqueue level; n must be 1, 2, or 3. The SUB-QUEUE-n paragraph can be omitted if there are no subqueues. When used, this paragraph must begin in area A. Figure 4-40 shows the SUB-QUEUE-n paragraph format.

```
SUB-QUEUE-n IS queue-name
    [JOURNAL clause]
    [MEDIUM clause]
    [PASSWORD clause]
    [RESIDENCY clause]
    [STATUS clause]
```

Figure 4-40. SUB-QUEUE-n Paragraph Format

The clauses associated with a SUB-QUEUE-n paragraph can appear in any order. The clauses are discussed in this subsection following the paragraph descriptions.

Compound Queue Definition

Figure 4-41 shows three examples of compound queues the user can define. Example 1 shows a compound queue structure in which all simple queues are sub-queue-3 level queues. To write the QUEUE and SUB-QUEUE-n paragraphs that define the queues shown in Example 1, the user must write the appropriate level paragraphs following the numerical order of the queues. The ADL statements defining this queue structure are shown in Figure 4-42, Example 1.

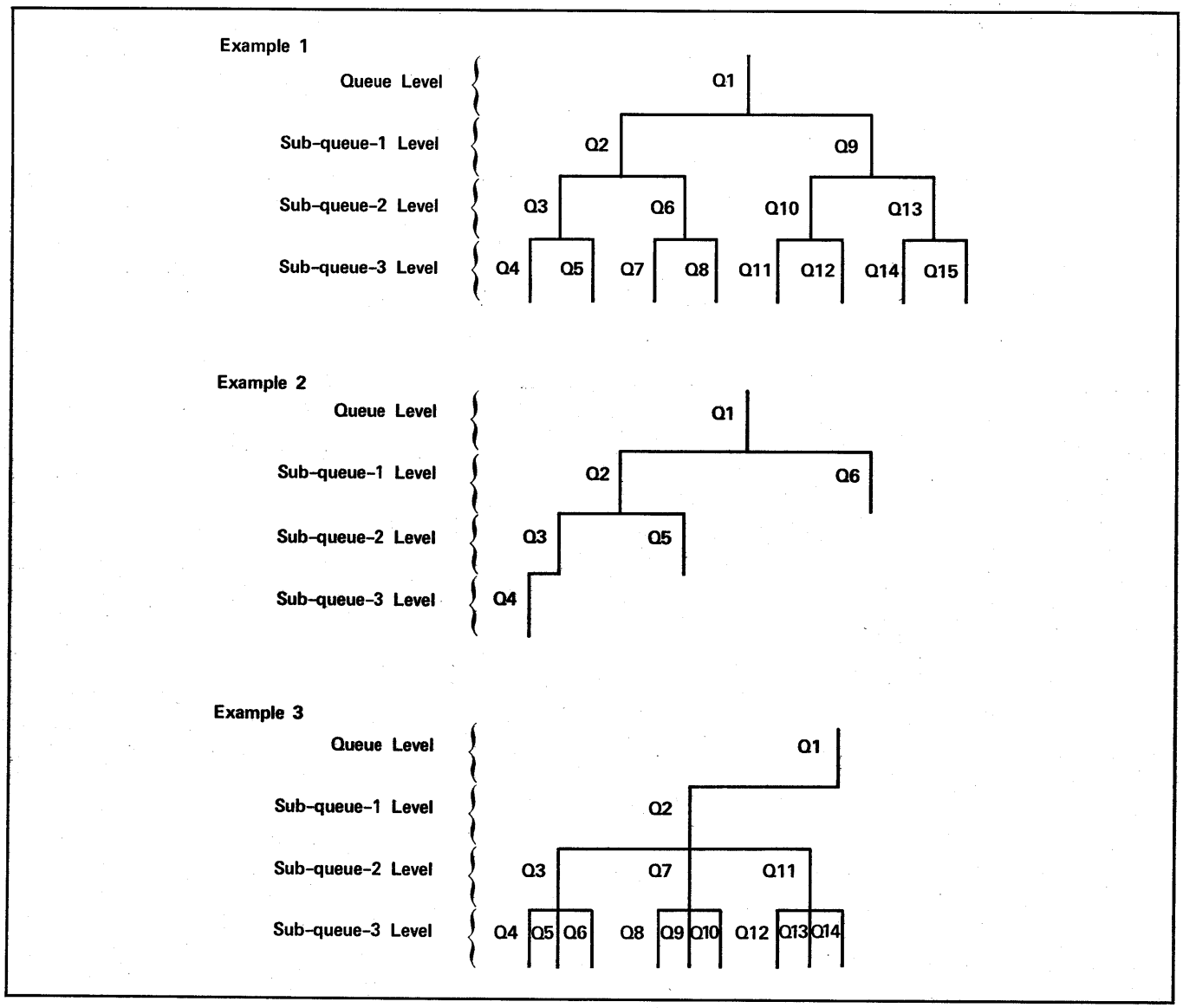


Figure 4-41. Compound Queue Structure Examples

Figure 4-41, Example 2 shows a compound queue that has one simple queue at each of the three subqueue levels. Q4, Q5, and Q6 are simple queues; these queues have no queues below them in the queue structure. To write the QUEUE and SUB-QUEUE-n paragraphs that define the queues shown in Example 2, the user must write the appropriate level paragraphs following the numerical order of the queues. The ADL statements defining this queue structure are shown in Figure 4-42, Example 2.

Figure 4-41, Example 3 shows a compound queue that has subqueues on only one side of the structure and more than two branches at subqueue levels 2 and 3. To write the QUEUE and SUB-QUEUE-n paragraphs that define the queues shown in Example 3, the user must write the appropriate level paragraphs following the numerical order of the queues. The ADL statements defining this queue structure are shown in figure 4-42, Example 3.

An application can use zero, one, or more compound queues. As as aid in writing the ADL statements defining a particular queue structure, the user should first draw the queue structure and number the queues as shown in figure 4-41. The user should then write the appropriate QUEUE and SUB-QUEUE-n paragraphs following the numerical order of the graphic representation.

Output Section

The Output Section names the output queues and specifies the characteristics of these queues. The section is made up of a QUEUE paragraph for each output queue used by the application.

The Output Section must begin in area A. Figure 4-43 shows the Output Section format.

Example 1

```
QUEUE IS Q1
 SUB-QUEUE-1 IS Q2
  SUB-QUEUE-2 IS Q3
   SUB-QUEUE-3 IS Q4
   SUB-QUEUE-3 IS Q5
  SUB-QUEUE-2 IS Q6
   SUB-QUEUE-3 IS Q7
   SUB-QUEUE-3 IS Q8
 SUB-QUEUE-1 IS Q9
  SUB-QUEUE-2 IS Q10
   SUB-QUEUE-3 IS Q11
   SUB-QUEUE-3 IS Q12
  SUB-QUEUE-2 IS Q13
   SUB-QUEUE-3 IS Q14
   SUB-QUEUE-3 IS Q15
```

Example 2

```
QUEUE IS Q1
 SUB-QUEUE-1 IS Q2
  SUB-QUEUE-2 IS Q3
   SUB-QUEUE-3 IS Q4
  SUB-QUEUE-2 IS Q5
 SUB-QUEUE-1 IS Q6
```

Example 3

```
QUEUE IS Q1
 SUB-QUEUE-1 IS Q2
  SUB-QUEUE-2 IS Q3
   SUB-QUEUE-3 IS Q4
   SUB-QUEUE-3 IS Q5
   SUB-QUEUE-3 IS Q6
  SUB-QUEUE-2 IS Q7
   SUB-QUEUE-3 IS Q8
   SUB-QUEUE-3 IS Q9
   SUB-QUEUE-3 IS Q10
  SUB-QUEUE-2 IS Q11
   SUB-QUEUE-3 IS Q12
   SUB-QUEUE-3 IS Q13
   SUB-QUEUE-3 IS Q14
```

Figure 4-42. ADL Compound Queue Definition Example

OUTPUT SECTION

[QUEUE paragraph] . . .

Figure 4-43. Output Section Format

QUEUE Paragraph of the Output Section

The QUEUE paragraph of the Output Section names an output queue and specifies the queue characteristics. This paragraph is omitted if output queues are not used by the application. When present, the QUEUE paragraph must begin in area A. Figure 4-44 shows the QUEUE paragraph format. Output queues are simple queues and have no subqueues.

The clauses associated with a QUEUE paragraph can appear in any order. The clauses are discussed in this subsection following the paragraph descriptions.

QUEUE IS queue-name

[JOURNAL clause]

[MEDIUM clause]

[RESIDENCY clause]

[STATUS clause]

Figure 4-44. QUEUE Paragraph Format, Output Section

Routing Section

The Routing Section specifies in which queue MCS stores a message. Routing to a specific queue is achieved by relating a symbolic name to a queue. An input queue is related to a symbolic source; an output queue is related to a symbolic destination. The relationship between a queue and a symbolic name can be based on message content, time of day, message source, message destination, or a combination of these criteria.

The Routing Section must begin in area A. Figure 4-45 shows the Routing Section format.

ROUTING SECTION

SELECT paragraph . . .

Figure 4-45. Routing Section Format

SELECT Paragraph

The SELECT paragraph specifies how MCS routes a message to a particular queue. An ADL program must include at least one SELECT paragraph, because at least one queue must be defined. An ADL program can include more than one SELECT paragraph, because a SELECT paragraph must appear for each type of queue (input and/or output), and for each nonsimple queue in the application. When more than one SELECT paragraph appears, these paragraphs can be in any order. The SELECT paragraph must begin in area A. Figure 4-46 shows the SELECT paragraph format.

When the Queue Division defines one or more input queues, a SELECT INPUT QUEUES paragraph must be included. Only one SELECT INPUT QUEUES paragraph is allowed.

When the Queue Division defines one or more compound queues, a SELECT SUB-QUEUES paragraph must be included. A SELECT SUB-QUEUES paragraph must appear for each nonsimple queue defined in the Input Section. Queue-name must be the name given the queue in the QUEUE or SUB-QUEUE-n paragraph of this division.

When the Queue Division defines one or more output queues, a SELECT OUTPUT QUEUES paragraph must be included. Only one SELECT OUTPUT QUEUES paragraph is allowed.

The BASED ON SOURCE, BASED ON TIME, BASED ON CONTENTS phrases can be used only in a SELECT INPUT QUEUES or a SELECT SUB-QUEUES paragraph. Data-name of the BASED ON CONTENTS option must be the name given in a FIELD clause of a SEGMENT paragraph of the Application Data Division. The BASED ON DESTINATION phrase can be used only in a SELECT OUTPUT QUEUES paragraph.

The ROUTE clause specifies in which queue MCS stores a particular message. This clause is discussed in the following subsection.

Queue Division Clauses

The Queue Division paragraphs can include the clauses discussed in this subsection. The clauses are: JOURNAL clause, MEDIUM clause, PASSWORD clause, RESIDENCY clause, ROUTE clause, and STATUS clause. These clauses can appear in the Queue Division paragraphs as shown in table 4-2.

JOURNAL Clause

Any QUEUE or SUB-QUEUE-n paragraph can include a JOURNAL clause. This clause names a file, or files, to which copies of messages are written after they are enqueued and/or dequeued. This journaling of messages provides a record of application message routing. Actual recording of messages occurs only for simple queues, because messages are not stored in nonsimple queues. When a JOURNAL clause appears in a paragraph naming a queue or subqueue that is a compound queue, the journal file is updated for any simple queues below the compound queue. For example, QA is a queue level queue with simple queues QB and QC below it at the sub-queue-1 level. A JOURNAL clause in the QUEUE paragraph defining QA names a file that records messages from QB and QC. The ADL statements for this example are as follows:

```
QUEUE IS QA
JOURNAL IS JRNL
SUB-QUEUE-1 IS QB
SUB-QUEUE-1 IS QC
```

When this clause is omitted from a QUEUE or SUB-QUEUE-n paragraph, no message recording is performed for any simple queues associated with the queue or subqueue. Figure 4-47 shows the JOURNAL clause format.

Symbolic-name-1 and symbolic-name-2 cannot be the same. Symbolic-name-1 and symbolic-name-2 must each be defined in SYMBOLIC-NAME paragraphs in the Source-Destination Division, and they must be declared as JOURNAL in the TYPE clause.

When neither the ON INPUT, nor the ON OUTPUT phrase is included in this clause, symbolic-name-1 records messages both after they are enqueued and after they are dequeued. For example:

```
QUEUE IS QA
JOURNAL IS FILE1
```

means that any message enqueued in or dequeued from QA, a simple queue, is recorded on FILE1.

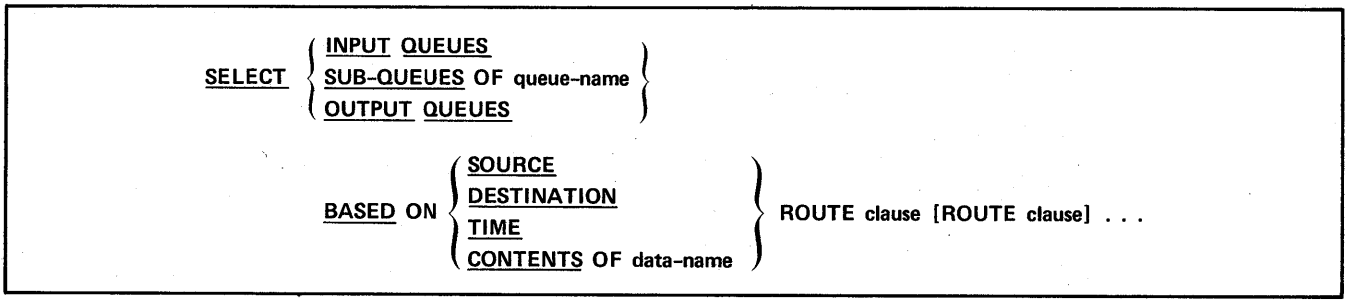


Figure 4-46. SELECT Paragraph Format

TABLE 4-2. QUEUE DIVISION CLAUSE USAGE

Clause	QUEUE Paragraph, Input Section	SUB-QUEUE-n Paragraph	QUEUE Paragraph, Output Section	SELECT Paragraph
JOURNAL Clause	X	X	X	
MEDIUM Clause	X	X	X	
PASSWORD Clause	X	X		
RESIDENCY Clause	X	X	X	
ROUTE Clause				X
STATUS Clause	X	X	X	

X - Clause can appear in this paragraph

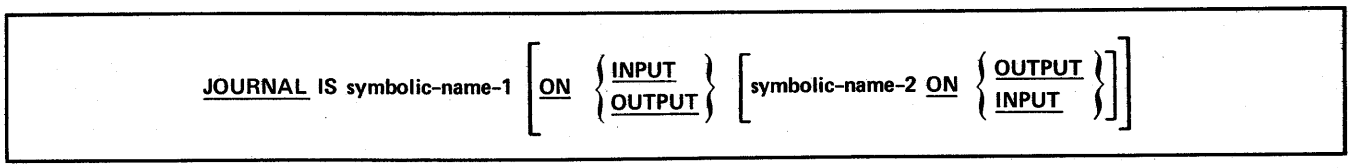


Figure 4-47. JOURNAL Clause Format

When the JOURNAL clause includes the ON INPUT phrase, but not the ON OUTPUT phrase, symbolic-name-1 records messages only after they are enqueued. For example:

```
QUEUE IS QA
JOURNAL IS FILE1 ON INPUT
```

means that FILE1 records messages after they are enqueued in QA, a simple queue; but messages are not recorded after they are dequeued.

When the JOURNAL clause includes the ON OUTPUT phrase, but not the ON INPUT phrase, symbolic-name-1 records messages only after they are dequeued. For example:

```
QUEUE IS QA
JOURNAL IS FILE2 ON OUTPUT
```

means that FILE2 records messages after they are dequeued from QA, a simple queue; but messages are not recorded after they are enqueued.

When the JOURNAL clause includes both the ON INPUT and the ON OUTPUT phrases, symbolic-name-1 records messages after they are enqueued, and symbolic-name-2 records messages after they are dequeued. For example:

```
QUEUE IS QA
JOURNAL IS FILE1 ON INPUT
    FILE2 ON OUTPUT
```

means that FILE1 records messages after they are enqueued in QA, a simple queue; and FILE2 records messages after they are dequeued from QA.

MEDIUM Clause

Any QUEUE or SUB-QUEUE-n paragraph can include a MEDIUM clause. This clause specifies where a queue is stored. Figure 4-48 shows the MEDIUM clause format.

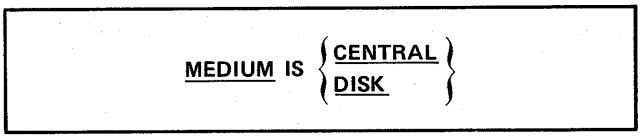


Figure 4-48. MEDIUM Clause Format

When MEDIUM IS CENTRAL is specified, the queue named in the associated QUEUE or SUB-QUEUE-n paragraph resides in central memory and is not permanent. When MEDIUM IS DISK is specified, the queue resides on mass storage. When this option is used, there must also be a RESIDENCY clause for each simple queue. When the MEDIUM clause is omitted, MCS assumes the queue resides in central memory.

PASSWORD Clause

Any QUEUE or SUB-QUEUE-n paragraph in the Input Section can include a PASSWORD clause. This clause names a password used for validation when a COBOL program enables or disables a queue. When this clause is omitted, password checking does not take place unless there is a SIGNATURE paragraph in the Application Global Division. Figure 4-49 shows the PASSWORD clause format. Nonnumeric-literal can be a maximum of 10 characters in length.

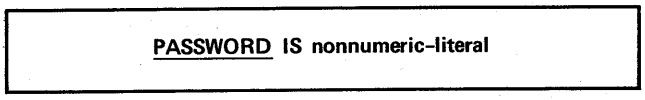


Figure 4-49. PASSWORD Clause Format, Queue Division

RESIDENCY Clause

Any QUEUE or SUB-QUEUE-n paragraph that names a simple queue (a queue with no subqueues) must include a RESIDENCY clause when the queue resides on mass storage (MEDIUM IS DISK). The RESIDENCY clause names the file on which messages are stored. Figure 4-50 shows the RESIDENCY clause format.

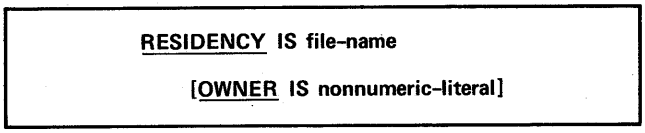


Figure 4-50. RESIDENCY Clause Format

File-name must be unique; each disk resident queue must be a separate file. When the OWNER phrase is used, nonnumeric-literal must be a system-validated user name. When the OWNER phrase is omitted, the system assumes MCS is the file owner; the system takes the user name from the MCS procedure file (see section 9 for information on the procedure file).

ROUTE Clause

The ROUTE clause associated with the SELECT paragraph specifies in which queue a message is stored. Several options on which to base the routing decision are available. At least one ROUTE clause must be associated with each SELECT paragraph. The same queue can be named in more than one ROUTE clause in the same SELECT paragraph. Figure 4-51 shows the ROUTE clause format.

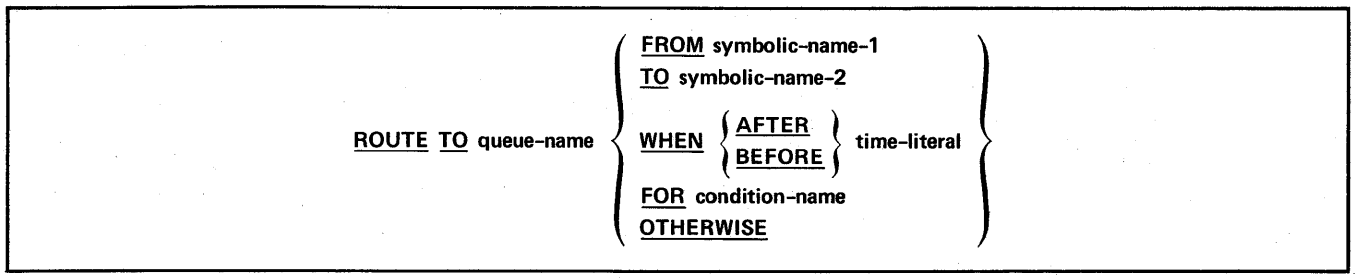


Figure 4-51. ROUTE Clause Format

When the ROUTE clause is in a SELECT INPUT QUEUES paragraph, queue-name must be defined as an input queue in a QUEUE paragraph of the Input Section. When the ROUTE clause is in a SELECT SUB-QUEUES paragraph, queue-name must be defined in a SUB-QUEUE-n paragraph of the Input Section. Queue-name must be a subqueue of the queue named in the SELECT SUB-QUEUES paragraph. For example, in the paragraph:

```
SELECT SUB-QUEUES OF QUE1 BASED ON SOURCE
    ROUTE TO SUBQUE1. . .
```

QUE1 is a compound input queue. Messages are routed to the queue named SUBQUE1 that is a subqueue of QUE1. When the ROUTE clause is in a SELECT OUTPUT QUEUES paragraph, queue-name must be defined as an output queue in a QUEUE paragraph of the Output Section.

The ROUTE clause in a SELECT INPUT QUEUES or SELECT SUB-QUEUES paragraph can include the FROM, WHEN, FOR, or OTHERWISE phrases. ROUTE TO queue-name FROM can be used only when the SELECT paragraph includes the BASED ON SOURCE phrase. Symbolic-name-1 must be defined as a source in a SYMBOLIC-NAME or INVITATION-LIST paragraph of the Source-Destination Division. Incoming messages arriving from symbolic-name-1 are enqueued in queue-name. For example:

```
SELECT INPUT QUEUES BASED ON SOURCE
    ROUTE TO QUE1 FROM TERM1
```

enqueues in QUE1 incoming messages from TERM1.

ROUTE TO queue-name WHEN can be used only when the SELECT paragraph includes the BASED ON TIME phrase. When the AFTER option is used, incoming messages arriving after time-literal are enqueued in queue-name. For example:

```
SELECT INPUT QUEUES BASED ON TIME
    ROUTE TO INQUA WHEN AFTER "09.00.00"
```

enqueues in INQUA incoming messages arriving after 9:00 a.m. When the BEFORE option is used, incoming messages arriving before time-literal are enqueued in queue-name. For example:

```
SELECT INPUT QUEUES BASED ON TIME
    ROUTE TO INQUA WHEN BEFORE "15.30.00"
```

enqueues in INQUA incoming messages arriving before 3:30 p.m.

ROUTE TO queue-name FOR can be used only when the SELECT paragraph includes the BASED ON CONTENTS phrase. Condition-name must be named in a CONDITION clause in a SEGMENT paragraph of the Application Data Division. Incoming messages with contents making the condition true are enqueued in queue-name. For example:

```
SELECT INPUT QUEUES BASED ON CONTENTS OF
    FIELD1 ROUTE TO OKQ FOR OKMSG
```

enqueues incoming messages in OKQ when the message field content is equal to the content of condition name OKMSG, defined in a CONDITION clause.

ROUTE TO queue-name OTHERWISE in a SELECT INPUT QUEUES or SELECT SUB-QUEUES paragraph specifies the default routing for incoming messages. ROUTE TO queue-name OTHERWISE must be specified last in a series of ROUTE clauses in a SELECT paragraph.

For incoming messages, the routing conditions specified in the SELECT INPUT QUEUES paragraph ROUTE clauses are tested in the order they appear until a condition is satisfied. When a routing condition is satisfied, the message is routed to the queue named in that ROUTE clause; the message is enqueued when the queue named is a simple queue. When the input queue is a compound queue, the SELECT SUB-QUEUES paragraph ROUTE clauses are tested in the same manner. A message is routed down the queue hierarchy until it arrives at a simple queue. For example, a message routed to a compound queue that has one queue at each of the three subqueue levels is tested three times before arriving at the sub-queue-3 level simple queue where it is enqueued.

The ROUTE clause in a SELECT OUTPUT QUEUES paragraph can include the TO or OTHERWISE phrases. ROUTE TO queue-name TO can be used only when the SELECT paragraph includes the BASED ON DESTINATION phrase. Symbolic-name-2 must be defined as a destination in a SYMBOLIC-NAME or BROADCAST-LIST paragraph of the Source-Destination Division. Outbound messages sent from a COBOL program to symbolic-name-2 are enqueued in queue-name while awaiting transmission. For example:

```
SELECT OUTPUT QUEUES BASED ON DESTINATION
    ROUTE TO OUTQ TO OUTERMS
```

enqueues in OUTQ messages sent from a COBOL program to destination OUTERMS.

ROUTE TO queue-name OTHERWISE in a SELECT OUTPUT QUEUES paragraph specifies the default routing for outbound messages; outbound messages are collected in a common queue, and the destination to which the data is sent is unpredictable. The OTHERWISE option should be used with caution in a SELECT OUTPUT QUEUES paragraph; if used, the OTHERWISE option must be the last clause in a SELECT OUTPUT QUEUES paragraph.

For outbound messages, the routing conditions specified in the ROUTE clauses are tested in the order specified until a condition is satisfied. When a routing condition is satisfied, the message is enqueued in the queue named in that ROUTE clause. Only one level of testing is necessary for outbound messages, because output queues are simple queues.

If the routing conditions of any SELECT paragraph have all been tested, and no condition is satisfied, the message is not enqueued. The message source is sent a diagnostic that the message is not enqueued (see appendix B).

STATUS Clause

Any QUEUE or SUB-QUEUE-n paragraph can include a STATUS clause. This clause specifies that a queue is enabled or disabled when the application is initiated. The STATUS clause is optional. When this clause is omitted, MCS assumes the queue is enabled when the application is initiated. Figure 4-52 shows the STATUS clause format.

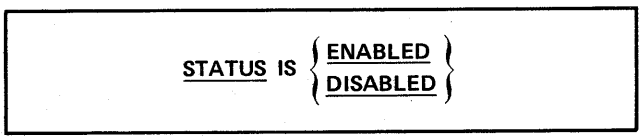


Figure 4-52. STATUS Clause Format, Queue Division

STATUS IS ENABLED means the queue is able to store message data at application initiation. STATUS IS DISABLED means the queue is not able to store message data at application initiation; no messages can be enqueued until the queue is enabled by the AOP ENABLE command (see section 8).

Queue Division Example

Figure 4-53 shows an example of the Queue Division. This division is part of application EXAMPLE of figure 4-11.

The Input Section defines one compound queue. The queue structure is shown in figure 4-54. The name of the queue is QA. Messages are recorded on journal file JRNL1 after they are enqueued in QA, and on journal file JRNL2 after they are dequeued from QA. JRNL1 and JRNL2 are defined as symbolic destinations of the type JOURNAL in SYMBOLIC-NAME paragraphs of the Source-Destination Division. QA resides on mass storage and is able to enqueue and dequeue messages when application EXAMPLE is started, because QA is initially enabled.

QA has six subqueues, each named in a separate SUB-QUEUE-n paragraph. Messages cannot be enqueued in or dequeued from any of these subqueues at application initiation (STATUS IS DISABLED in each SUB-QUEUE-n paragraph). When messages are enqueued in any of these subqueues, the messages are recorded on JRNL1; messages are recorded on JRNL2 as they are dequeued. The JOURNAL clause of the QUEUE paragraph applies to all the subqueues. All subqueues reside on mass storage, because the MEDIUM IS DISK clause in the QUEUE paragraph applies to all the subqueues.

QB, a sub-queue-1 level queue, has queues QC, QD, QE, and QF below it in the hierarchy, so it is not a simple queue. One other nonsimple queue appears in this example, and that is QC, a sub-queue-2 level queue that has QD and QE below it. QD and QE are simple queues, because they are sub-queue-3 level queues; this is the lowest level in the queue hierarchy. The SUB-QUEUE-3 paragraphs defining QD and QE include RESIDENCY clauses naming the mass storage files where these queues are stored. Queues QF and QG are also simple queues even though they are sub-queue-1 and sub-queue-2 level queues, respectively; they have no queues below them. Their SUB-QUEUE-n paragraphs include RESIDENCY clauses.

There are three other QUEUE paragraphs in the Input Section, each naming a special type of simple input queue. Queue INJECQ is the injection queue named in the INJECTION-QUEUE paragraph of the Application Global Division. Queue COLLECQ is the collection queue named in the COLLECTION-QUEUE paragraph of the Application Global Division. Queue RESQ is the response queue named in the RESPONSE-QUEUE clause in the PROGRAM paragraph of the Application Program Division.

The Output Section of figure 4-53 defines two output queues named OUTQ1 and OUTQ2. OUTQ1 and OUTQ2 reside on mass storage on files OUTFILE1 and OUTFILE2, respectively. Both output queues are able to enqueue and dequeue messages at application initiation (STATUS IS ENABLED).

The Routing Section of figure 4-53 includes five SELECT paragraphs. A SELECT INPUT QUEUES paragraph appears, because this division includes at least one input queue. Incoming messages from invitation list TERMS (composed of TERM1 and TERM2) are routed to QA, the queue level queue in this example.

The next SELECT paragraph is a SELECT SUB-QUEUES paragraph that is required because QA has subqueues. Messages from TERM2 are routed to QB. Messages that are not from TERM2 are routed to QG. The messages routed to QG are those from TERM1. These messages have now arrived at a simple queue and are enqueued in QG.

QB has queues below it and requires a SELECT SUB-QUEUES paragraph. The messages from TERM2 that are routed to QB are routed based on time. Messages arriving at QB before 9:00 a.m. are routed to QC. Messages arriving at QB after 9:00 a.m. are routed to QF. These messages are enqueued in QF, a simple queue.

QC has queues below it and requires a SELECT SUB-QUEUES paragraph. The messages originally from TERM2 that have been routed to QB and then QC are now routed based on message content. FIELD12 of each message is tested, and messages matching the contents of C-NAME are enqueued in QD, a simple queue. FIELD12 is defined in a SEGMENT paragraph of the Application Data Division. C-NAME is also defined in the Application Data Division, and is assigned the value FIVE5. Messages not matching this value are enqueued in QE, also a simple queue. All incoming messages have now been enqueued in a simple queue.

The last SELECT paragraph is a SELECT OUTPUT QUEUES paragraph that routes outbound messages from the output queue OUTQ1 to TERM1 and from the output queue OUTQ2 to TERM2.

```
QUEUE DIVISION
INPUT SECTION
QUEUE IS QA
    JOURNAL IS JRNL1 ON INPUT
        JRNL2 ON OUTPUT
    MEDIUM IS DISK
    STATUS IS ENABLED
 SUB-QUEUE-1 IS QB
    STATUS IS DISABLED
  SUB-QUEUE-2 IS QC
    STATUS IS DISABLED
   SUB-QUEUE-3 IS QD
    RESIDENCY IS QDFILE
    STATUS IS DISABLED
   SUB-QUEUE-3 IS QE
    RESIDENCY IS QEFILE
    STATUS IS DISABLED
  SUB-QUEUE-2 IS QF
    RESIDENCY IS QFFILE
    STATUS IS DISABLED
 SUB-QUEUE-1 IS QG
    RESIDENCY IS QGFILE
    STATUS IS DISABLED
QUEUE IS INJECQ
QUEUE IS COLLECQ
QUEUE IS RESQ

OUTPUT SECTION
QUEUE IS OUTQ1
    MEDIUM IS DISK
    RESIDENCY IS OUTFL1
    STATUS IS ENABLED
QUEUE IS OUTQ2
    MEDIUM IS DISK
    RESIDENCY IS OUTFL2
    STATUS IS ENABLED

ROUTING SECTION
SELECT INPUT QUEUES BASED ON SOURCE
    ROUTE TO QA FROM TERMS
SELECT SUB-QUEUES OF QA BASED ON SOURCE
    ROUTE TO QB FROM TERM2
    ROUTE TO QG OTHERWISE
SELECT SUB-QUEUES OF QB BASED ON TIME
    ROUTE TO QC WHEN BEFORE "09.00.00"
    ROUTE TO QF WHEN AFTER "09.00.00"
SELECT SUB-QUEUES OF QC BASED ON CONTENTS OF FIELD12
    ROUTE TO QD FOR C-NAME
    ROUTE TO QE OTHERWISE
SELECT OUTPUT QUEUES BASED ON DESTINATION
    ROUTE TO OUTQ1 TO TERM1
    ROUTE TO OUTQ2 TO TERM2
```

Figure 4-53. Queue Division Example

APPLICATION PROCESSING DIVISION

The Application Processing Division specifies events that MCS monitors, and it defines actions taken when an event occurs. The division is made up of a number of USE paragraphs, each specifying a particular event and one or more verbs that define the action MCS takes when the event occurs.

The division header is:

APPLICATION PROCESSING DIVISION

The header must appear on a separate line, beginning in area A. Figure 4-55 shows a skeleton of the Application Processing Division.

USE Paragraph

The USE paragraph names an event that MCS monitors. The paragraph includes a condition statement that defines the event and one or more verbs that execute when the condition is true. The USE paragraph can be omitted if there is no application event monitoring. When present, the USE paragraph must begin in area A. Figure 4-56 shows the USE paragraph format.

The USE EVERY option is used to specify recurring conditions, such as an amount of elapsed time. A USE paragraph with USE EVERY can be performed more than once.

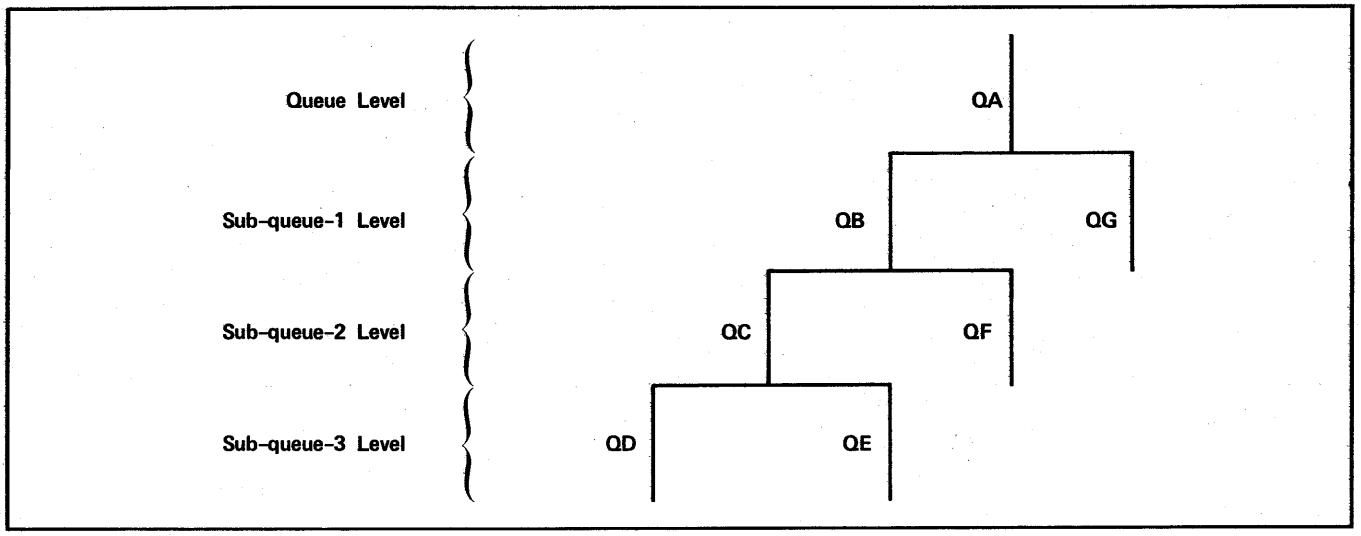


Figure 4-54. Application EXAMPLE Input Queue Structure

```
APPLICATION PROCESSING DIVISION
[USE paragraph] . . .
```

Figure 4-55. Application Processing Division Skeleton

```
USE { WHEN  } condition
    { EVERY }
verb-1
  [verb-2] . . .
```

Figure 4-56. USE Paragraph Format

The USE WHEN option specifies nonrecurring conditions, such as a specific system clock time. A USE paragraph with USE WHEN can be performed only once while an application is running. When an application is shut down and restarted, the condition is reset.

USE Paragraph Conditions

Various conditions define the events MCS monitors. When a condition references a symbolic name that is an invitation list or broadcast list, the condition applies to each component of the list. When a condition in a USE WHEN paragraph references a symbolic name that is an invitation list or broadcast list, the condition is true once for the entire list and not once for each component of the list. When a condition references a symbolic name that is an invitation list or broadcast list, and the same condition also references a symbolic name that is a component of the list, the component condition overrides the list condition. For example, TERM1 is a component of the broadcast list BLIST, and the following statements are defined:

```
USE WHEN condition TERM1
    verb
USE EVERY condition BLIST
    verb
```

The verb immediately following the USE WHEN paragraph is executed when condition is true for TERM1.

Conditions that can define an event are: CONNECT condition, CONNECTION-BROKEN condition, CONNECTION-INACTIVE condition, DISCONNECT condition, ELASPED-TIME condition, INITIATION condition, SIZE EXCEEDS condition, and TIME condition. Table 4-3 summarizes these conditions, which are discussed in the following paragraphs.

CONNECT Condition

Figure 4-57 shows the CONNECT condition format. This condition is true when the terminal or any of the group of terminals named by symbolic name establishes a logical connection with the application.

CONNECTION-BROKEN Condition

Figure 4-58 shows the CONNECTION-BROKEN condition format. This condition is true when the terminal or any of the group of terminals named by symbolic name is

involuntarily disconnected from the application. The execution of the DISCONNECT verb or the AOP DISCONNECT command does not make this condition true. This condition is true, for example, if a user at a dialup terminal hangs up the phone without logging off.

TABLE 4-3. USE PARAGRAPH CONDITIONS

Condition	Event Monitored
CONNECT	Logical connection between application and terminal or group of terminals.
CONNECTION-BROKEN	Terminal or group of terminals involuntarily disconnected from an application.
CONNECTION-INACTIVE	Terminal or group of terminals connected but not sending or receiving commands or messages.
DISCONNECT	Terminal or group of terminals voluntarily ending connection with an application.
ELAPSED-TIME	Length of time an application has been running.
INITIATION	Application initiation.
SIZE EXCEEDS	Number of messages in a queue greater than defined limit.
TIME	System clock time.

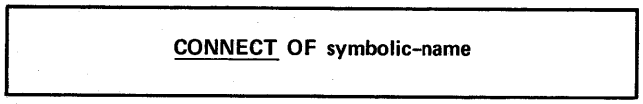


Figure 4-57. CONNECT Condition Format

Figure 4-58. CONNECTION-BROKEN Condition Format

CONNECTION-INACTIVE Condition

Figure 4-59 shows the CONNECTION-INACTIVE condition format. This condition is true when the terminal or any of the group of terminals named by symbolic name is inactive for an installation-defined period. An inactive terminal or group of terminals is connected to the application but is not sending or receiving any commands or messages.

Figure 4-59. CONNECTION-INACTIVE Condition Format

DISCONNECT Condition

Figure 4-60 shows the DISCONNECT condition format. This condition is true when the terminal or any of the group of terminals named by symbolic name voluntarily ends its logical connection with the application. A DISCONNECT condition occurs, for example, when a terminal user logs off.

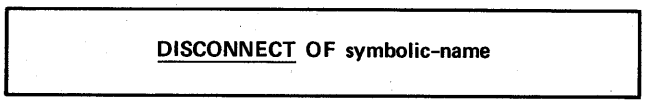


Figure 4-60. DISCONNECT Condition Format

ELAPSED-TIME Condition

Figure 4-61 shows the ELAPSED-TIME condition format. This condition is true when the MCS application elapsed time counter exceeds time-literal.

Figure 4-61. ELAPSED-TIME Condition Format

INITIATION Condition

Figure 4-62 shows the INITIATION condition format. This condition is true only when an application is initiated. Application initiation is discussed in the Application Global Division subsection.

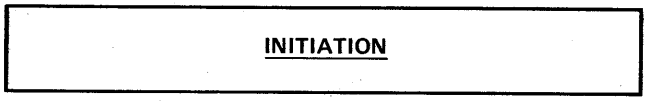


Figure 4-62. INITIATION Condition Format

SIZE EXCEEDS Condition

Figure 4-63 shows the SIZE EXCEEDS condition format. This condition is true when the message count for the named queue becomes greater than the specified integer.

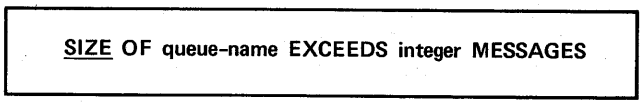


Figure 4-63. SIZE EXCEEDS Condition Format

TIME Condition

Figure 4-64 shows the TIME condition format. This condition is true when the system clock time reaches time-literal.

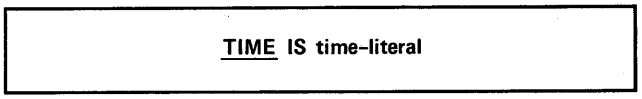


Figure 4-64. TIME Condition Format

USE Paragraph Verbs

The verbs discussed in the following paragraphs execute when the USE paragraph condition is true. A USE paragraph includes one or more verbs depending on the action or actions the user desires. Table 4-4 summarizes the verbs and their actions.

TABLE 4-4. USE PARAGRAPH VERBS

Verb	Description
DISABLE	Disables a terminal or group of terminals.
DISCONNECT	Disconnects a terminal from MCS.
DISPLAY	Displays status information.
DUMP	Dumps environmental information about the application to the dump file.
ENABLE	Enables a terminal or group of terminals.
IDLE	Halts application processing.
INVOKE	Initiates execution of the specified COBOL program.
MESSAGE	Sends a message string to a terminal or group of terminals.
PURGE	Deletes all partial messages sent to specified destination.
REROUTE	Reroutes output from one destination to another destination.
REVOKE	Terminates execution of the specified COBOL program.
SHUTDOWN	Terminates an application.

Any responses, including error responses, to USE paragraph verbs are sent to the AOP. If no AOP is logged in, the responses are discarded.

DISABLE Verb

When the DISABLE verb executes, a source, destination, list, or queue becomes temporarily inactive. Figure 4-65 shows the DISABLE verb format.

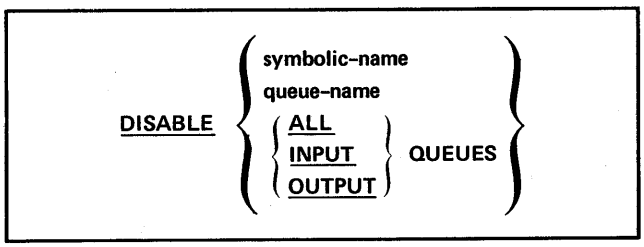


Figure 4-65. DISABLE Verb Format

Symbolic-name must be defined as a source or destination in a SYMBOLIC-NAME paragraph of the Source-Destination Division. Symbolic-name can be an invitation list or broadcast list. Queue-name must be defined as an input or output queue in a QUEUE or SUB-QUEUE-n paragraph of the Queue Division. When ALL is specified, all input and output queues are disabled. When INPUT is specified, all input queues are disabled. When OUTPUT is specified, all output queues are disabled. It is illegal to disable a collection queue, injection queue, or response queue. Disabling an already disabled source, destination, or queue results in a diagnostic message (see appendix B).

When a source is disabled, the source remains connected to the application, but no incoming data messages are accepted from this source. Incomplete messages from a source are purged from the system when the source is disabled. When a queue is disabled, no incoming or outbound data messages are enqueued in or dequeued from this queue. When a destination is disabled, it remains connected to the application, but no outbound data messages are sent to this destination. When symbolic-name is an invitation list or broadcast list, no data messages are accepted from or sent to any of the components of the list. A disabled source or destination can input commands to MCS.

DISCONNECT Verb

When the DISCONNECT verb executes, an external source or destination is disconnected from the application until the source or destination logs in again (see section 2 for login procedure). Figure 4-66 shows the DISCONNECT verb format. Symbolic-name must be defined as a source or destination in a SYMBOLIC-NAME paragraph of the Source-Destination Division.

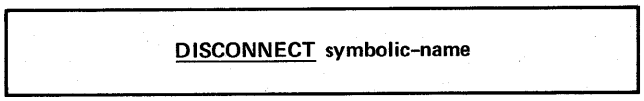


Figure 4-66. DISCONNECT Verb Format

When a source is disconnected, commands and data messages are not solicited from this source. DISCONNECT of a source causes all partial input messages from this source to be purged from the system. When a destination is disconnected, outbound messages are not sent to this destination. Messages sent to a destination that is disconnected remain in the output queue until the destination is connected. Disconnecting an already disconnected source or destination results in a diagnostic message (see appendix B).

DISPLAY Verb

When the DISPLAY verb executes, a display of application status information is transmitted to the AOP. This display can also be written to a file for later application analysis. Figure 4-67 shows the DISPLAY verb format.

When the MONITOR-FILE phrase is used, a monitor file must be named in a MONITOR-FILE paragraph of the Application Global Division. When the MONITOR-FILE phrase is omitted, an AOP must be named in an OPERATOR paragraph of the Application Global Division. When the MONITOR-FILE phrase is omitted, the display is transmitted to the AOP; if no AOP is connected, this verb has no effect.

The displays generated when this verb executes are the same as those generated in response to an AOP DISPLAY command. See section 8 for examples of the displays generated. When no display type is specified, a display of the current application status is generated. When the PROGRAMS option is used, a display of the current status of all COBOL programs is generated. When TERMINALS is specified, a display of the status of all application terminals is generated.

When the QUEUES option of the DISPLAY verb is used, and INPUT is specified, a display of the current status of all input queues is generated. When OUTPUT is specified, a display of the current status of all output queues is generated. When ALL is specified, a display of the current status of all input and output queues is generated.

DUMP Verb

When the DUMP verb executes, an application dump is performed and application environmental information is written to the dump file. A dump file must be named in a DUMP-FILE paragraph of the Application Global Division. Figure 4-68 shows the DUMP verb format. See section 5 for more information on the dump file.

ENABLE Verb

When the ENABLE verb executes, a source, destination, list, or queue becomes active; this verb causes the reverse of a DISABLE verb action. Figure 4-69 shows the ENABLE verb format.

Symbolic-name must be defined as a source or destination in a SYMBOLIC-NAME paragraph of the Source-Destination Division. Symbolic-name can be an invitation list or broadcast list. Queue-name must be defined as an input or output queue in a QUEUE or SUB-QUEUE-n paragraph of the Queue Division. When ALL is specified, all input and output queues are enabled. When INPUT is specified, all input queues are enabled. When OUTPUT is specified, all output queues are enabled. It is illegal to enable a collection queue, injection queue, or response queue. Enabling an already enabled source, destination, or queue results in a diagnostic message (see appendix B).

When a source is enabled, incoming data messages are accepted from this source. When a queue is enabled, incoming and outbound data messages are enqueued in and dequeued from this queue. When a destination is enabled, outbound data messages can be sent to this destination. When symbolic-name is an invitation list or broadcast list, data messages are accepted from or delivered to any of the components of the list.

IDLE Verb

When the IDLE verb executes, application processing halts temporarily. The application remains on-line, but MCS does not solicit input from the network or deliver output. MCS rejects any further COBOL program requests. However, the IDLE verb does not affect the AOP; the AOP is always active. The RESUME command reverses the IDLE verb (see section 8). Figure 4-70 shows the IDLE verb format.

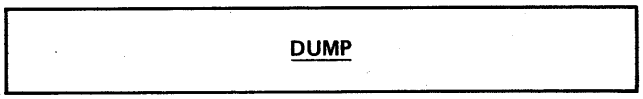


Figure 4-68. DUMP Verb Format

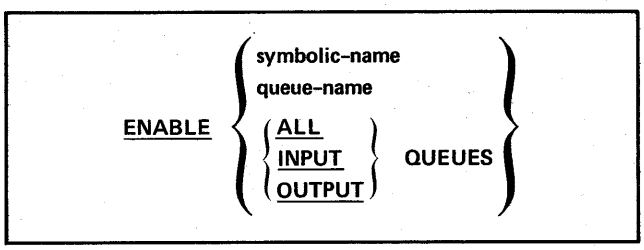


Figure 4-69. ENABLE Verb Format

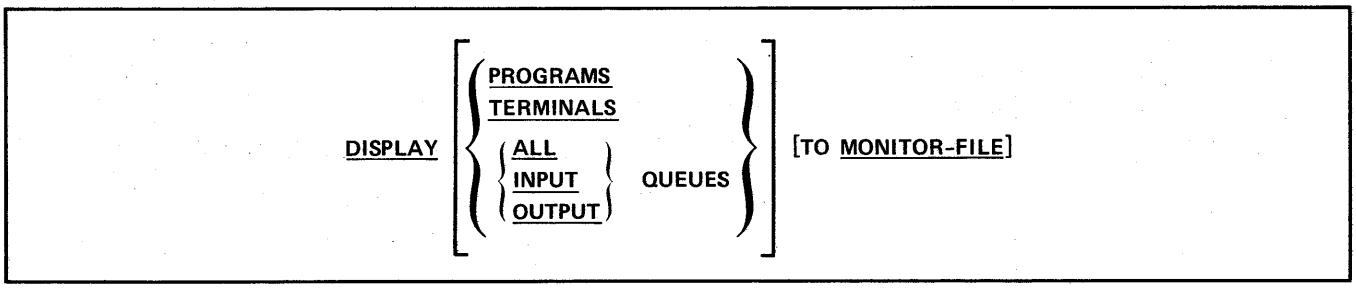


Figure 4-67. DISPLAY Verb Format

```
IDLE
```

Figure 4-70. IDLE Verb Format

INVOKE Verb

When the INVOKE verb executes, execution of a COBOL program is initiated. Figure 4-71 shows the INVOKE verb format.

```
INVOKE routine-name
```

Figure 4-71. INVOKE Verb Format

Routine-name must be a COBOL program name from a PROGRAM paragraph of the Application Program Division. An invocation file for this COBOL program must be named in an INVOCATION-FILE clause of the PROGRAM paragraph. The invocation file is submitted to the operating system job input queue when INVOKE executes. When the control statements of the invocation file execute, the COBOL program then executes.

MESSAGE Verb

When the MESSAGE verb executes, a message is sent to the AOP, all external destinations, or a named recipient. Figure 4-72 shows the MESSAGE verb format.

```
MESSAGE nonnumeric-literal

[TO symbolic-name]
```

Figure 4-72. MESSAGE Verb Format

When the TO phrase is used, symbolic-name must be named as a destination in a SYMBOLIC-NAME paragraph of the Source-Destination Division. Symbolic-name can be a broadcast list. When symbolic-name is a broadcast list, the message is sent to all currently connected components of the list. When the TO phrase is omitted, the message is sent to all currently connected application destinations. This verb has no effect if no destinations are connected. However, the message is delivered to a destination that is connected but disabled (not able to receive data messages).

PURGE Verb

When the PURGE verb executes, incomplete messages sent to an external destination or group of destinations are discarded. Complete messages are not affected. Figure 4-73 shows the PURGE verb format.

```
PURGE symbolic-name
```

Figure 4-73. PURGE Verb Format

Symbolic-name must be named as a destination in a SYMBOLIC-NAME paragraph of the Source-Destination Division. Symbolic-name can be a broadcast list.

REROUTE Verb

When the REROUTE verb executes, all output sent to an external destination or group of destinations is sent to an alternate destination. This verb allows alternate destinations for application output in case of a CONNECTION-BROKEN or DISCONNECT condition. Figure 4-74 shows the REROUTE verb format.

```
REROUTE symbolic-name-1 TO symbolic-name-2
```

Figure 4-74. REROUTE Verb Format

Symbolic-name-1 and symbolic-name-2 must be named as destinations in SYMBOLIC-NAME paragraphs of the Source-Destination Division. Symbolic-name-1 can be a broadcast list. A diagnostic message results if symbolic-name-2 is not connected to MCS (see appendix B). The alternate destination condition is no longer in effect when symbolic-name-1 reconnects to MCS.

REVOKE Verb

When the REVOKE verb executes, a COBOL program terminates. Figure 4-75 shows the REVOKE verb format.

```
REVOKE routine-name
```

Figure 4-75. REVOKE Verb Format

Routine-name must be a COBOL program name from a PROGRAM paragraph of the Application Program Division. The COBOL program aborts on the next program request to MCS. When the program is suspended waiting for data, it is aborted as soon as the REVOKE verb executes. The program can be ended gracefully by including an EXIT card in its invocation file. If the named COBOL program is not connected to MCS, the REVOKE verb has no effect.

SHUTDOWN Verb

When the SHUTDOWN verb executes, an application terminates. This verb must be the last verb in a series of verbs in a USE paragraph. Figure 4-76 shows the SHUTDOWN verb format. See section 8 for a list of the actions that occur following execution of the SHUTDOWN verb.

```
SHUTDOWN
```

Figure 4-76. SHUTDOWN Verb Format

Application Processing Division Example

Figure 4-77 shows an example of the Application Processing Division. This division is part of application EXAMPLE of figure 4-11.

In the first USE paragraph, when either TERM1 or TERM2 of invitation list TERMS connects to MCS, both TERM1 and TERM2 are enabled; they can input data messages and MCS accepts these messages. The status of terminals connected to EXAMPLE is then written to monitor file MNTRF. COBOL program MAWTEST is executed after the INVOKE verb causes the invocation file INVF to be submitted to the operating system job input queue.

In the second USE paragraph, every time there are more than 12 messages in QA, QA is disabled (can no longer accept incoming data messages). The input queue display is generated and sent to the AOP, and the message QA FULL is sent to the connected components of broadcast list OUTPUTS.

```
APPLICATION PROCESSING DIVISION
USE WHEN CONNECT OF TERMS
    ENABLE TERMS
    DISPLAY TERMINALS TO MONITOR-FILE
    INVOKE MAWTEST
USE EVERY SIZE OF QA EXCEEDS 12 MESSAGES
    DISABLE QA
    DISPLAY INPUT QUEUES
    MESSAGE "QA FULL" TO OUTPUTS
USE WHEN DISCONNECT OF OUTPUTS
    REVOKE MAWTEST
    PURGE OUTPUTS
    SHUTDOWN
```

Figure 4-77. Application Processing Division Example

In the last USE paragraph, when any component of the broadcast list OUTPUTS disconnects from MCS, COBOL program MAWTEST is revoked (terminated). Incomplete messages sent to OUTPUTS are discarded, and application EXAMPLE is then shutdown.

This section describes the MCS facilities for compiling a program written in ADL, for testing a newly developed MCS application, for monitoring the activities of an executing application, and for recovering an application after a shutdown.

COMPILATION

After an application definition has been written using ADL, it must then be compiled by the Application Definition Language processor (ADLP) before it can be used to control an MCS application. The ADL processor accepts as input a program written in ADL and produces as output an application definition library.

The ADL processor is a two-pass processor: pass 1 ensures that the ADL program is syntactically correct; pass 2 ensures that logic and usages are consistent within the program. If errors are encountered during either pass of ADL compilation, appropriate messages are written to an output file, and no application definition library is created. (Error messages are described in appendix B.) When the application definition is syntactically correct and logically consistent, the ADL processor transforms the definition into a collection of tables usable by MCS, and places the tables in the application definition library file.

APPLICATION DEFINITION LIBRARIES

Upon successful compilation, the application definition tables produced by the ADL processor are added to the new application definition library specified on the ADLP control statement. The application definition library is a file containing a collection of compiled MCS application definitions. Each application definition in a library file constitutes a single record and is independent of all other definitions in the library.

When MCS is initially started, the system operator or MCS procedure file must specify the name of the library containing the definitions of the applications to be executed. The applications available for initiation by an application operator at login time are the applications contained in the library. Only one library can be active at any given time. However, any number of applications within a library can be active. The applications that have been initiated by an application operator are available to terminal users at login time.

ADLP CONTROL STATEMENT

The ADL processor is invoked by the ADLP system control statement. Various processing options can be controlled by parameters specified on the control statement. These parameters have one of the following forms:

```
parameter=opt
parameter=opt1 [opt2 [opt3]]
parameter=opt1 [/opt2] . . .
```

When a parameter is omitted, a default value is assumed by the processor. Parameters can appear in any order and are separated by commas. Duplicate specification of parameters is not allowed.

The ADLP control statement has the forms shown in figure 5-1. The form:

```
ADLP.
```

causes all parameters to assume default values. Note that specification of the parameter without an option causes the I parameter to assume a second default value. For all other parameters, specification of parameter without an option is not allowed. For example:

```
ADLP(I=INFILE,LO=SRL,OLD=0,NEW=NLIB)
```

compiles the ADL source program stored on file INFILE. A source listing, reference map, and library maintenance listing are produced and written to file OUTPUT (L parameter omitted). A previous application library does not exist. A new library, NLIB, is created and the output application definition tables are stored in this library. The form:

```
ADLP(I=0,OLD=LIB6,NEW=LIB7,D=APP1/APP2,LO=L)
```

deletes application definitions APP1 and APP2 from old library LIB6, and creates a new library LIB7. No new application definition tables are produced. A library maintenance listing is written to file OUTPUT.

COMPILATION LISTINGS

The ADL processor produces three listings that can provide assistance in debugging ADL programs and in maintaining application definition libraries; these are: source listing, cross reference listing, and library maintenance listing. The processor produces any combination of these listings, or none of them, as determined by ADLP control statement parameters. Examples of these listings are described in the following paragraphs.

Source Listing

The ADLP source listing is specified by the LO=S control statement parameter. This listing includes all source lines submitted as input to the ADL processor. A header line at the top of each page contains the processor version number, date, time, and page number.

The source program is listed 60 lines per page. A line number is printed at the beginning of each line. These line numbers are used in the error messages and cross reference listing.

Wherever possible, error messages generated during pass 1 of an ADL compilation appear immediately before the statement containing the error; pass 2 error messages appear immediately after the last statement of the source listing. A complete list of ADL processor error messages and an explanation of each message is included in appendix B.

ADLP.

ADLP,$param_1$[,$param_2$] . . .

ADLP($param_1$[,$param_2$] . . .)

$param_n$ is one of the following parameters:

I — Name of input file containing ADL source program to be processed. Valid values are:

omitted	ADL source program is on file INPUT.
I	ADL source program is on file COMPILE.
I=0	No source file is input.
I=lfn	ADL source program is on file lfn.

I=0 should be specified for a delete-only run.

L — Name of output listing file. Valid values are:

omitted	Output listing is written to file OUTPUT.
L=0	Output listing is written to file OUTPUT and contains only error messages.
L=lfn	Output listing is written to file lfn.

LO — Output listing options. Any combination of S, R, or L can be specified. Error messages are produced regardless of LO specifications. Valid values are:

omitted	Same as LO=S if OLD=0. Same as LO=SL otherwise.
LO=0	Output listing contains only error messages.

LO=V_1 [[V_2]V_3]

where V_n has one of the following values, and $V_1 \neq V_2 \neq V_3$:

S	ADL source listing is produced as part of the output listing.
R	A cross reference map is produced as part of the output listing.
L	A library maintenance listing is produced as part of the output listing.

OLD — Name of file containing the old application definition library. Applications are copied from this file to the new application definition library and are added, deleted, or replaced, as directed. Valid values are:

omitted	Same as OLD=OLDLIB.
OLD=0	No application definition library currently exists.
OLD=lfn	Old application definition library is on file lfn.

NEW — Name of file to contain the new application definition library. This file is either an updated version of the old application definition library or a newly created library. Applications are copied from the old application definition library to the new application definition library and are added, deleted, or replaced, as directed. The new library is accessed by MCS during startup and can be used as input for other ADLP runs. Valid values are:

omitted	Same as NEW=ADLLIB.
NEW=0	No new application definition library is created. (No application definition tables are produced.)
NEW=lfn	The updated application definition library is written to file lfn.

D — Applications to be deleted from the old application definition library. Valid values are:

omitted	No application definitions are deleted from the old application definition library.
D=appl[/appl] . . .	The specified applications are deleted.

For a delete only run, I=0 should be specified so that no input file is expected.

Figure 5-1. ADLP Control Statement

A compilation summary appearing at the end of the source listing includes the following messages:

nnnnnnnnn DIAGNOSTIC MESSAGES

indicates the total number of error messages appearing in the source listing.

nnnnnnnnnB CM REQUIRED

indicates the total number (octal) of central memory words required for the compilation.

nnnnnn.nnn SECONDS CP TIME

indicates the total time (in central processor seconds) required for compilation.

nnnnnnnnnB TOTAL TABLES SIZE

indicates the number (octal) of central memory words required for the tables produced by the ADL processor.

Figure 5-2 illustrates a source listing for a simple ADL program. Figure 5-3 illustrates the same listing, except that an error has been introduced. The misspelling of FIELD1 in line 12 has caused a fatal error.

Cross Reference Listing

The cross reference listing is specified by the LO=R control statement parameter. This listing includes an alphabetical list of all symbolic names appearing in the source program together with certain attributes of each name. These attributes include the name type, the number of the source line in which the name is defined, and the numbers of all source lines in which the name is referenced. The cross reference map follows the source listing in the output file.

A symbolic name is defined when it is assigned to a specific application entity, such as a queue or terminal. A symbolic name that is referenced, but not defined in an ADL program, constitutes a fatal error. An undefined name is indicated by the characters UNDEF in the TYPE

```
                    * SOURCE LISTING *

 1                APPLICATION GLOBAL DIVISION
 2                 APPLICATION-NAME IS AD2
 3
 4                APPLICATION PROGRAM DIVISION
 5                 PROGRAM IS PROG1
 6
 7                APPLICATION DATA DIVISION
 8                 EGI IS "/"
 9                 MESSAGE IS MSG
10                   SERIAL-NUMBER IS GENERATED IN FIELD1
11                   SEGMENT IS SEG1  LENGTH IS 80 CHARACTERS
12                  FIELD IS FIELD1  STARTS AT CHARACTER 1
13                   EXTENDS FOR 3 CHARACTERS
14
15                SOURCE-DESTINATION DIVISION
16                 SYMBOLIC-NAME IS TERM1
17                 TYPE IS INTERACTIVE
18                 MODE IS DATA
19                 MESSAGES ARE MSG
20
21                QUEUE DIVISION
22                 INPUT SECTION
23                   QUEUE IS INQ
24                 OUTPUT SECTION
25                   QUEUE IS OUTQ
26                 ROUTING SECTION
27                   SELECT INPUT QUEUES
28                     BASED ON SOURCE
29                     ROUTE TO INQ FROM TERM1
30                   SELECT OUTPUT QUEUES
31                     BASED ON DESTINATION
32                     ROUTE TO OUTQ TO TERM1
33
34                APPLICATION PROCESSING DIVISION
```

Figure 5-2. Source Listing

column of the reference listing. The cross reference listing is, thus, a useful debugging tool, and should always be examined after an ADL processor compilation.

An example of a cross reference listing is illustrated in figure 5-4. Figure 5-5 shows a cross reference listing generated by a program that contains an error. FIELD1 is flagged as an undefined name because it was misspelled when defined in line 12.

Library Maintenance Listing

The library maintenance listing is specified by the LO=L parameter. This listing includes the name of each application definition in the new application definition library, the length of each application definition in physical record units (PRUs), and the creation date of each application definition. Application definitions are listed in order of creation date, with the most recently created application definition appearing first. If fatal compilation errors occur, a new library is not created, and the library maintenance listing is not produced.

An example of a library maintenance listing is illustrated in figure 5-6.

APPLICATION DEFINITION LIBRARY MAINTENANCE

After an application definition library has been created, application definitions can be added to, deleted from, or replaced in the library by specifying the appropriate ADLP control statement parameters. The following paragraphs illustrate system control statements and ADLP parameters necessary to perform these operations. Refer to the NOS reference manual for detailed information on system control statements.

CREATING AN APPLICATION DEFINITION LIBRARY

A new application definition library is created each time an application definition is successfully compiled. If a library does not currently exist, the application definition becomes the first one in the new library.

Figure 5-7 illustrates a job structure that compiles an application definition and creates a new application definition library. The DEFINE control statement is included to assign permanent file status to the library. The OLD=0 parameter on the ADLP control statement specifies that no old library exists. The NEW=ALIB parameter specifies the name of the new library that is to contain the application definition.

```
                          * SOURCE LISTING *

          1               APPLICATION GLOBAL DIVISION
          2                APPLICATION-NAME IS AD2
          3
          4               APPLICATION PROGRAM DIVISION
          5                PROGRAM IS PROG1
          6
          7               APPLICATION DATA DIVISION
          8                EGI IS "/"
          9                MESSAGE IS MSG
         10                  SERIAL-NUMBER IS GENERATED IN FIELD1
         11                  SEGMENT IS SEG1  LENGTH IS 80 CHARACTERS
         12                 FIELD IS (FELD1) STARTS AT CHARACTER 1
         13                  EXTENDS FOR 3 CHARACTERS
         14
         15              SOURCE-DESTINATION DIVISION
         16               SYMBOLIC-NAME IS TERM1
         17               TYPE IS INTERACTIVE
         18               MODE IS DATA
         19               MESSAGES ARE MSG
         20
         21              QUEUE DIVISION
         22               INPUT SECTION
         23                 QUEUE IS INQ
         24               OUTPUT SECTION
         25                 QUEUE IS OUTQ
         26               ROUTING SECTION
         27                 SELECT INPUT QUEUES
         28                   BASED ON SOURCE
         29                   ROUTE TO INQ FROM TERM1
         30                 SELECT OUTPUT QUEUES
         31                   BASED ON DESTINATION
         32                   ROUTE TO OUTQ TO TERM1
         33
         34              APPLICATION PROCESSING DIVISION

PASS 2 DIAGNOSTICS -
       0 * 061 F  SERIAL NUMBER DATANAME INVALID
```

Figure 5-3. Source Listing Containing Error

```
                        * CROSS REFERENCE *

      NAME                TYPE          DEF LINE  REFERENCES

      AD2                 SYSTEM           2
      FIELD1              DATA            12        10
      INQ                 QUEUE           23        29
      MSG                 DATA             9        19
      OUTQ                QUEUE           25        32
      PROG1               ROUTINE          5
      SEG1                DATA            11
      TERM1               SYMBOLIC        16        29        32

    NO DIAGNOSTIC MESSAGES
 16600B CM REQUIRED
  0.365 SECONDS CP TIME
   236B TOTAL TABLES SIZE
```

Figure 5-4. Cross Reference Listing

```
                          * CROSS REFERENCE *

NAME                      TYPE          DEF LINE   REFERENCES

AD2                       SYSTEM           2
FELD1                     DATA            12
FIELD1                  * UNDEF *          0         10
INQ                       QUEUE           23         29
MSG                       DATA             9         19
OUTQ                      QUEUE           25         32
PROG1                     ROUTINE          5
SEG1                      DATA            11
TERM1                     SYMBOLIC        16         29         32

        1 DIAGNOSTIC MESSAGES
* FATAL ERRORS - NO TABLES *
```

Figure 5-5. Cross Reference Listing Showing Undefined Name

```
                          NEW LIBRARY CONTENTS

ADLP,I=COMPILE,OLD=OLDLIB,NEW=ADLLIB,L=OUTPUT.

NAME                 SIZE/PRUS      CREATION DATE

APPL04                   3            79/07/30.
APPL03                   3            79/07/30.
APPL02F                  3            79/07/30.
APPL02E                  3            79/07/30.
APPL02D                  3            79/07/30.
APPL02C                  3            79/07/30.
APPL023                  3            79/07/30.
APPL02A                  3            79/07/30.
APPL02                   4            79/07/30.
APPL01                   3            79/07/30.
```

Figure 5-6. Library Maintenance Listing

```
JOB1.
USER statement
CHARGE statement
DEFINE(ALIB/PW=AAA,CT=PU,M=W)
ADLP(OLD=0,NEW=ALIB)
7/8/9  in column 1
       ADL source program
6/7/8/9  in column 1
```

Figure 5-7. Creating an Application Definition Library

ADDING APPLICATIONS TO AN APPLICATION DEFINITION LIBRARY

Once an application definition library has been created, new application definitions can be added to the library. Application definitions in the existing library are copied to a new library, and the new application difinition is added to the new library. The name of each application definition in the library must be unique. If an application definition to be added to the library has the same name as an existing one, the new application definition replaces the old one in the new library.

Figure 5-8 illustrates a job structure that compiles an application definition and adds it to a library. Control statements are included to attach the old library, to allocate permanent file space for the new library, and to purge the old library after the new one has been created. The OLD and NEW parameters on the ADLP statement specify the old and new library names respectively.

```
JOB2.
USER statement
CHARGE statement
ATTACH(ALIB)
DEFINE(BLIB/PW=AAA,CT=PU,M=W)
ADLP(OLD=ALIB,NEW=BLIB,LO=SRL,L=OUTLIST)
PURGE(ALIB)
7/8/9  in column 1
       ADL source program
6/7/8/9  in column 1
```

Figure 5-8. Adding an Application Definition to a Library

DELETING APPLICATION DEFINITIONS FROM A LIBRARY

To prevent a library from attaining an excessive size, unused applications should be deleted from the library. Applications can be deleted from an application definition library by specifying the D parameter on the ADLP control statement. Applications can be deleted during a compilation run or in a run in which no applications are compiled. A new library is created, and the applications from the old library are copied to the new library, except for those being deleted.

Figure 5-9 illustrates a delete-only run. The old library is attached and direct access permanent file status is assigned to the new library. The I=0 parameter on the ADLP control statement specifies that no input source file is to be read. The OLD and NEW parameters specify the names of the old and new libraries, respectively. Two application definitions, AD1 and AD2, are deleted from the library.

```
JOB3.
USER statement
CHARGE statement
ATTACH(BLIB)
DEFINE(CLIB/PW=AAA,CT=PU,M=W)
ADLP(OLD=BLIB,NEW=CLIB,I=0,D=AD1/AD2)
6/7/8/9 in column 1
```

Figure 5-9. Deleting Application Definitions from a Library

APPLICATION TESTING

MCS provides a mode of execution, called test mode, in which MCS applications execute independently of the network. In test mode, MCS does not poll the network for messages. Instead, messages are delivered to special queues called collection queues and received from special queues called injection queues, which function as substitutes for terminals. Thus, test mode provides a means of testing message routing and application program logic before the terminals defined in the application are included in the network or in the network definition files.

Individual MCS applications within an active library can execute in test mode while others execute in normal mode, or the entire library can be initiated in global test mode. In the latter case, all applications within the active library execute in test mode, and the network need not be on-line.

To execute an application in test mode, the application developer must write one or more COBOL programs to generate messages for input to the application and to receive and analyze messages output by the application. These programs interface with MCS through the COBOL Communication Facility in the same manner as other COBOL programs. The message generation and analysis programs send messages to the injection queue, and receive messages from the collection queue. The symbolic source and destination names referenced in the communication description areas of these programs are associated with the collection and injection queues defined in the application definition.

The message flow in test mode is illustrated in figure 5-10.

Test mode can be established by specifying the test option in the INITIATION paragraph of the application definition. Note that AUTOMATIC initiation is required. This paragraph has the form:

INITIATION IS AUTOMATIC TEST

In this case, one or more applications can execute in test mode while others execute in normal mode. To switch back to normal mode, however, it is necessary to remove the TEST parameter from the INITIATION paragraph in the application definition, and recompile. Test mode can also be established by system operator command for all applications or for a single application when MCS is initiated (as described in section 9).

COLLECTION QUEUES

In normal execution, MCS output messages are queued in output queues and then delivered to the network. In test mode, output messages are routed from the output queues to a special queue called the collection queue. A collection queue is a simple input queue that is defined in the input section of the queue division and declared in the COLLECTION-QUEUE paragraph of the Application Global Division in the application definition. The collection queue serves as a substitute for the destinations defined in the application definition. Messages in the collection queue can then be received by the message analysis program, which specifies the collection queue in the program's input communication description area.

To establish a collection queue, the queue name must be declared in the COLLECTION-QUEUE paragraph of the Application Global Division. For example:

COLLECTION-QUEUE IS CQ

establishes CQ as the collection queue.

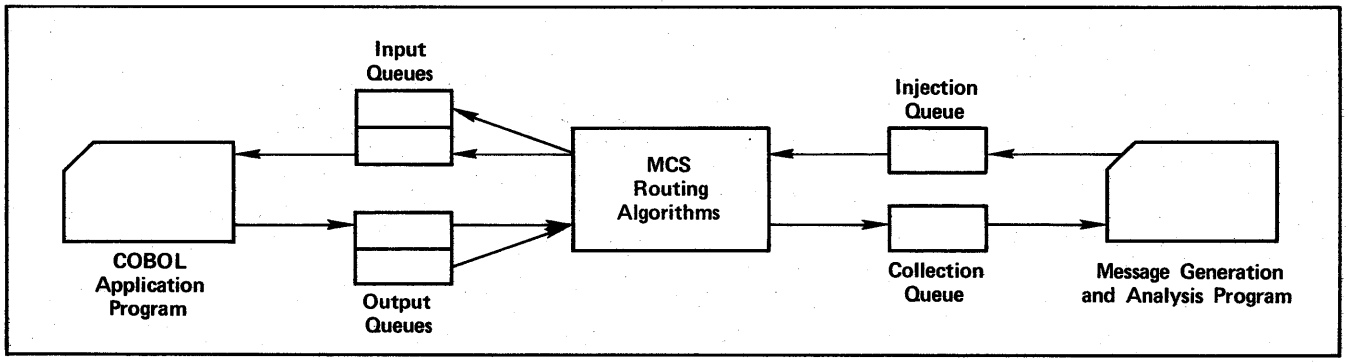


Figure 5-10. Test Mode Message Flow

The queue name must then be declared in the Input Section of the Queue Division. For example, the paragraph:

```
INPUT SECTION
QUEUE IS CQ
```

defines CQ to be a simple input queue. If CQ is a collection queue, all application messages to external destinations are sent to CQ.

Messages are routed to the collection queue according to the procedures defined in the Routing Section of the Queue Division. For example, the paragraph:

```
SELECT OUTPUT QUEUES
BASED ON DESTINATION
ROUTE TO OUTQ TO TERMB
```

appearing in the Routing Section routes application messages from output queue OUTQ to the destination TERMB. In normal mode, the name TERMB would be associated with a terminal. In test mode, messages sent to TERMB are actually sent to the collection queue.

The program to receive messages from the collection queue must specify the collection queue name in the appropriate field of the input communication description area.

INJECTION QUEUES

In normal execution, messages originating at terminals are routed to input queues, and then delivered to COBOL programs according to the procedures defined in the Routing Section. In test mode, messages originate in the message generation program, are queued in a special queue called the injection queue, and are then received by the COBOL programs.

An injection queue is a queue that is defined in the Input Section of the Queue Division and declared in the INJECTION-QUEUE paragraph of the Application Global Division. The injection queue is used as a source to simulate terminal input. Messages are stored in the injection queue by the message generation program and then routed to input queues where they are received by COBOL programs.

The message generation program can store messages in the injection queue in the same manner as for output queues, except that the first segment of each message must consist of the symbolic source name being represented. This establishes the source of the message and enables MCS to route the message to the approriate input queue according to the procedures set forth in the Routing Section. The first segment of the message is then discarded by MCS and does not form part of the message text.

To establish an injection queue, the queue name must be declared in the INJECTION-QUEUE paragraph of the Application Global Division. For example, the paragraph:

```
INJECTION-QUEUE IS INJQ
```

appearing in the Application Global Division establishes INJQ as the injection queue. The injection queue must then be assigned a symbolic name in the Source-Destination Division. The message generation program references this name in the destination field of the output communication description area. The symbolic name is of type QUEUE, and the alias is the name declared in the INJECTION-QUEUE paragraph. For example:

```
SYMBOLIC-NAME IS DESTB
TYPE IS QUEUE
ALIAS IS INJQ
```

associates the symbolic name DESTB with injection queue INJQ. All messages sent to DESTB are stored in INJQ, and all input from external sources is solicited from INJQ.

The injection queue must also be declared in the Input Section of the Queue Division. For example:

```
INPUT SECTION
QUEUE IS INJQ
```

defines injection queue INJQ to be an input queue.

Messages sent to the injection queue in normal mode are rejected.

When a source is defined in the application definition as initially in command mode, messages attributable to that source are treated as commands. Any of the commands described in sections 7 and 8 can be issued, with the exception of DISCONNECT and the login and logout commands. MCS responses to these commands are stored in the collection queue. However, once the source is switched to data mode, there is no provision for switching back to command mode; that is, the break-2 sequence cannot be entered in test mode.

APPLICATION DEFINITION FOR TEST MODE OPERATION

The application definition of an application to be executed in test mode must specify the injection queue, collection queue, and message generation and analysis programs.

Figure 5-11 illustrates a simple application definition for an MCS application to be executed in test mode. In the Application Global Division, the TEST option is specified so that when the application is initiated it executes in test mode. An injection queue (INJECQ) and a collection queue (COLLQ) are defined. In the Application Data Division, the area of memory to contain application messages is defined and assigned the name MSG. In the Source-Destination Division, terminal TERMA is defined as interactive (capable of both sending and receiving messages). The injection queue is assigned the symbolic name QTERM. The terminal is initiated in command mode. Messages sent to TERMA are contained in MSG. In the Queue Division, COLLQ and INJECQ are defined as input queues. In addition, an input queue (INQ) and an output queue (OUTQ) are defined. The Routing Section specifies that messages from TERMA are to be routed to INQ. In test mode, messages in the injection queue that contain the name TERMA in the first segment are delivered to INQ. Messages sent to TERMA are delivered to the collection queue COLLQ. The Application Program Division specifies two COBOL programs, APPROG and MSGPROG, to process application messages and to send messages to the collection queue and receive messages from the injection queue.

To execute this MCS application in normal mode, it is necessary to remove the TEST option from the INITIATION paragraph, and recompile. The application then interfaces with the network software.

```
                    * SOURCE LISTING *

 1              APPLICATION GLOBAL DIVISION
 2               APPLICATION-NAME IS APPY
 3               INJECTION-QUEUE IS INJECQ
 4               COLLECTION-QUEUE IS COLLQ
 5               INITIATION IS AUTOMATIC TEST
 6
 7              APPLICATION PROGRAM DIVISION
 8               PROGRAM IS APPROG
 9                INVOCATION-FILE IS INVFILE
10               PROGRAM IS MSGPROG
11
12              APPLICATION DATA DIVISION
13               MESSAGE IS MSG
14                SERIAL-NUMBER IS SUPPLIED IN FIELD1
15               SEGMENT IS S1 LENGTH IS 100 CHARACTERS
16               FIELD IS FIELD1 STARTS AT CHARACTER 1
17                EXTENDS FOR 3 CHARACTERS
18
19              SOURCE-DESTINATION DIVISION
20               SYMBOLIC-NAME IS TERMA
21                TYPE IS INTERACTIVE
22                MODE IS COMMAND
23                MESSAGES ARE MSG
24               SYMBOLIC-NAME IS QTERM
25                TYPE IS QUEUE
26                ALIAS IS INJECQ
27
28              QUEUE DIVISION
29               INPUT SECTION
30                QUEUE IS INQ
31                QUEUE IS COLLQ
32                QUEUE IS INJECQ
33               OUTPUT SECTION
34                QUEUE IS OUTQ
35               ROUTING SECTION
36                SELECT INPUT QUEUES BASED ON SOURCE
37                 ROUTE TO INQ FROM TERMA
38                SELECT OUTPUT QUEUES BASED ON DESTINATION
39                 ROUTE TO OUTQ TO TERMA
40
41              APPLICATION PROCESSING DIVISION
```

Figure 5-11. Application Definition for Test Mode Execution

MCS EVENTS

MCS provides syntax for defining processing to be performed on the occurrence of certain events. The events to be monitored and processing to be performed are defined in the Application Processing Division of the application definition, as described in section 4. The MCS event conditions include:

- Involuntary breaking of a terminal connection
- Connection inactive for a defined period
- Disconnecting a terminal
- Elapsed time exceeding a specified value
- Application initiation
- Message count for a specified queue exceeding a specified value
- System clock time reaching a specified value

Processing options described in this section that can be controlled by the preceding events are COBOL program invocation, and the writing of monitor files and dump files.

APPLICATION PROGRAM EXECUTION

Execution of COBOL programs that communicate with MCS can be initiated either through batch job submission or by MCS command. The MCS command to initiate program execution can be issued from a terminal, by another application program or by MCS itself, according to an application definition specification. For either method of initiation, the name of the program to be executed must be specified in the Application Program Division of the application definition.

BATCH JOB SUBMISSION

Batch-initiated job streams can contain programs that communicate with MCS. The job stream containing accounting statements, system control statements, and

data required by the program, is prepared in a normal manner, as described in the COBOL reference manual. The only special requirement is that the execution control statement (typically LGO) must contain a parameter of the form:

*APPL=name

where name is the name of an MCS application. This parameter identifies to MCS the particular MCS application with which the program is to communicate. The job can then be submitted as a card deck or stored on a file and submitted via the appropriate system command.

An example of a batch job deck is illustrated in figure 5-12. In this example, control statements are included to attach an input file to be read by the COBOL program and to define permanent storage space for an output file to be written by the program. The execution control statement (LGO) specifies the MCS application APPZ. The COBOL Communication Facility requests communicate with this application. Following the COBOL source statements are data records to be read by the program.

MCS-SUBMITTED JOBS

Program execution can be initiated by an INVOKE command issued from a terminal (as described in section 8) or from another application program, or by execution of an INVOKE verb in the Application Processing Division of the application definition. The INVOKE verb enables MCS to initiate program execution when specified conditions are satisfied.

In either case, the user must prepare a file, called the invocation file, associated with the program to be executed. The syntax for defining this file is described in section 4 under Application Program Division. The invocation file can contain any valid system control statement, and must contain the control statements necessary for execution of the COBOL application program. An invocation file must be an indirect access permanent file. When the INVOKE is issued, the invocation file is submitted to the operating system for execution. For example, the job stream in figure 5-12 could be written to an indirect access permanent file and used as an invocation file.

Figure 5-13 illustrates another example of an invocation file. In this example, the COBOL program is read from the input file called SRCE, and EXIT error processing is included. The COBOL program and invocation file name must be declared in the Application Program Division of the application definition.

The association between the COBOL program and its invocation file is established in the Application Program Division. For example:

```
APPLICATION PROGRAM DIVISION
PROGRAM IS PROGA
INVOCATION-FILE IS INVFIL
```

defines program PROGA to be included in the application, and declares that INVFIL is the invocation file associated with PROGA.

```
JOBX.
USER(DEF,PWRD,FAM305)   } <-- Accounting statements
CHARGE(4321,1098AAA)    }
ATTACH(INFILE)  <-- Attach data file to be read by COBOL program
DEFINE(OUTFILE) <-- Define permanent file space for file to be written by COBOL program
COBOL5.         <-- Call COBOL 5 compiler
LGO(*APPL=APPZ) <-- Initiate execution of COBOL program with MCS application APPZ
7/8/9  in column 1
       COBOL source program with COBOL Communication Facility statements
7/8/9  in column 1
       Data to be read by COBOL program
6/7/8/9 in column 1
```

Figure 5-12. Sample Job Stream

```
JOB.
USER(ABC,PASSW,FAM1)
CHARGE(1234,5678901)
ATTACH(SRCE)
COBOL5(I=SRCE)  <-- Compile COBOL program contained in file SRCE
LGO(*APPL=APPZ)
EXIT.
DMP.
7/8/9  in column 1
       Data to be read by COBOL program
6/7/8/9  in column 1
```

Figure 5-13. Invocation File Example

When the program name is specified in an INVOKE command or verb, the invocation file is submitted to the operating system for execution. For example:

```
?invoke,proga
```

when issued from a terminal, submits the invocation file associated with program PROGA to the operating system.

ADL provides for initiating program execution on the occurrence of MCS events. For example:

```
APPLICATION PROCESSING DIVISION
USE WHEN TIME IS "12.00.00"
INVOKE PROGA
```

initiates execution of the invocation file associated with program PROGA at noon system clock time.

APPLICATION MONITORING

MCS provides facilities for writing a formatted summary of application status information to a special file, called a monitor file, on the occurrence of an MCS event. This file can be used to analyze the activities of the application. ADL provides syntax for describing information to be monitored, the frequency with which information is written, and the name of the monitor file on which information is written. Information can also be written to the monitor file by direct terminal command, as described in section 7.

The name of the monitor file must be declared in the MONITOR-FILE paragraph of the Application Global Division. For example:

```
APPLICATION GLOBAL DIVISION
.
.
.
MONITOR-FILE IS MFILE
OWNER IS "USER1"
```

defines a monitor file called MFILE under user name USER1.

The information to be monitored and the conditions under which it is written are defined in the Application Processing Division by a statement of the form:

```
USE condition
DISPLAY item TO MONITOR-FILE
```

where condition is the condition that must be satisfied for DISPLAY to be executed, and item specifies the information to be written. For example:

```
APPLICATION PROCESSING DIVISION
USE EVERY ELAPSED-TIME IS "00.05.00"
DISPLAY ALL QUEUES TO MONITOR-FILE
DISPLAY TERMINALS TO MONITOR-FILE
```

writes queue and terminal status information to the monitor file at 5 minute intervals.

The terminal command to write to the monitor file is:

```
DISPLAY item TO MONITOR-FILE
```

where item can have any of the values described in section 7 for the DISPLAY command.

The format of the information written to the monitor file is identical to that described in section 7 for the DISPLAY command.

The monitor file is a direct access file. Each DISPLAY constitutes one record on the monitor file. In a subsequent job or terminal session, the user can attach and display or print the contents of this file. For example:

```
ATTACH,MFILE/UN=USER1.
COPYSBF,MFILE,OUTPUT.
```

attaches file MFILE and copies it to file OUTPUT, with right shift for carriage control, to produce a printed copy of the monitor file.

RECOVERY

MCS provides facilities that can be used to aid in the recovery and restart of MCS applications. These facilities provide information about the status of all aspects of an application including the queues, terminals, programs, and internal tables that comprise the application. MCS produces this information at selected times, on the occurrence of selected events, or through direct terminal command.

It is important to note that when an MCS session is to be continued after an application restart, the MEDIUM IS DISK clause must be specified for each queue in the application in order to preserve the contents of the queues; otherwise, queue contents are lost when the shutdown occurs.

DUMP FILE

An application dump file containing essential application status information can be created either according to procedures established in the application definition or by a terminal command entered by the application operator. This dump contains detailed information about the status of the internal tables used by MCS and is intended for use mainly by a system analyst in determining the cause of internal MCS errors.

The name of the dump file must be declared in the DUMP-FILE paragraph of the Application Global Division. For example:

```
APPLICATION GLOBAL DIVISION
.
.
.
DUMP-FILE IS DFILE OWNER IS "ABC"
```

declares file DFILE to be an application dump file under user name ABC.

Application dumps can be written according to conditions established in the Application Processing Division of the application definition. When the specified MCS event occurs, the dump is initiated. For example:

```
APPLICATION PROCESSING DIVISION
USE EVERY ELAPSED TIME IS "00.02.00"
DUMP
```

writes dumps to the application dump file at 2-minute intervals.

Application dumps can also be initiated by the application operator through the DUMP command described in section 8.

Each time a DUMP verb or command is executed, a complete application dump is written to the dump file. Each dump constitutes one record of the file.

QUEUE RECOVERY

MCS provides facilities that can be used to recover the contents of application queues in the event of an application shutdown and subsequent restart. These facilities include message serial numbers, a command to retrieve messages in queues, and journal files on which messages are recorded after they are enqueued and dequeued.

It is important to note that when an MCS session is to be continued after an application restart, the MEDIUM IS DISK clause must be specified for each queue in the application in order to preserve the contents of the queues; otherwise, queue contents are lost when the shutdown occurs.

Message Serial Numbers

Serial numbers can be assigned to input messages either by MCS or by the terminal user when the message is entered. ADL provides statements for specifying:

- The location of the serial number
- Whether the serial number is supplied by MCS or by the terminal that originates the message
- Whether the serial number is echoed to the source of the message

Details for defining these message characteristics are described in section 4 under Application Data Division.

MCS assigns serial numbers in ascending order, starting with 1. When the terminal user assigns serial numbers, this same convention should be followed because MCS assumes that the largest serial number is associated with the last message entered. ADL provides an option that causes serial numbers to be echoed to the source terminal. This enables a terminal user to know which numbers MCS has assigned to messages. For example:

```
APPLICATION DATA DIVISION
MESSAGE IS MSG
SERIAL-NUMBER IS GENERATED IN F1 WITH ECHO
SEGMENT IS S1 LENGTH IS 100 CHARACTERS
FIELD IS F1 STARTS AT CHARACTER 1
EXTENDS FOR 10 CHARACTERS
```

specifies that messages contained in MSG have serial numbers generated by MCS in character positions 1 through 10 of the first message segment. The serial number is echoed to the terminal after a message is entered.

When an application shutdown and subsequent restart occur, the user can determine the serial number of the last message entered from the terminal by entering a DISPLAY LAST command. For example, assume that terminals T1 and T2 are connected to an MCS application and that T1 enters message 1, T2 enters messages 2 and 3, and T1 enters message 4. When the application is shut down and later restarted, the terminal users can determine which messages were the last messages entered. When T1 enters DISPLAY LAST, MCS responds:

```
LAST SERIAL NUMBER = 4
```

When T2 enters DISPLAY LAST, MCS responds:

```
LAST SERIAL NUMBER = 3
```

The users can then continue the session at the point of interruption.

When GENERATED IN FIELD or SUPPLIED IN FIELD is specified in the application definition, then serial numbers known to the terminal user are also contained in the message text. The user can then determine if any messages have been lost by comparing the last number issued from the terminal with the last number received by the application program.

RETRIEVE Command

The RETRIEVE command, described in section 8, retrieves and displays messages from a specified queue. This command is used to determine the contents of application queues, and can be helpful in determining queue status following a shutdown and restart. The MEDIUM IS DISK clause must be specified for a queue if its contents are to be preserved following an application restart; otherwise, queue contents are lost when the shutdown occurs.

Journal Files

Journal files are files to which formatted copies of messages being enqueued or dequeued are written. These files can be used to maintain a record of queue activity for a given application. Journal files contain formatted information and can be printed, displayed at a terminal, or input to a user-written program for the purpose of analyzing the file contents. Separate journals should be used for input and output queues.

The symbolic name of a journal file must be declared in the Source-Destination Division to be of type JOURNAL and must be assigned an alias. The alias is the logical file name of the journal file. For example:

```
SOURCE-DESTINATION DIVISION
SYMBOLIC-NAME IS JOURN
TYPE IS JOURNAL
ALIAS IS JFILE
```

defines a journal file with internal symbolic name JOURN and file name JFILE.

Input and output messages for a given queue can be written to the same journal or to separate journals. Journals are associated with specific queues in the Queue Division. For example:

```
QUEUE DIVISION
INPUT SECTION
QUEUE IS QXYZ
JOURNAL IS JOURN ON INPUT
QOUT ON OUTPUT
```

specifies that messages enqueued in QXYZ are to be copied to journal JOURN, and messages dequeued from QXYZ are to be copied to journal QOUT.

Application MAILBOX is a mail service for four users named Bob G., Sue C., Tom R., and Leo W. The four users are at four different locations where each has access to an interactive terminal. The users can leave messages (mail) for each other, and they can receive messages from the other users. For purposes of this example, only one user can access MAILBOX at any one time.

This section includes the ADLP source listing of MAILBOX (figure 6-1) with comments explaining each division. The compound input queue structure for this application is shown in figure 6-2. MAILBOX has two COBOL programs: MSGDROP and TOOMANY. Both of these programs are invoked using MCS. The invocation file for MSGDROP is named MSGFILE and is shown in figure 6-3. To save time when MSGDROP is invoked, this program is compiled and saved on the binary file named MDBIFIL. The invocation file for TOOMANY is named TOOFILE and is shown in figure 6-4. TOOMANY is also compiled and saved on the binary file named TMBIFIL. The COBOL source listing of MSGDROP is shown in figure 6-5; figure 6-6 shows the COBOL source listing of TOOMANY.

A terminal session in which one user leaves messages for the other users and collects messages that have been left for him is shown in figure 6-7. Program TOOMANY is invoked when a user's queue contains more than 12 messages. Figure 6-8 shows an input queue display with 12 messages in SUECQ. Figure 6-9 shows the output from program TOOMANY.

```
 1
 2
 3                APPLICATION GLOBAL DIVISION
 4               *
 5               * THE NAME OF THIS APPLICATION IS MAILBOX.
 6               * THE PASSWORD REQUIRED WHEN A COBOL PROGRAM ENABLES OR DISABLES
 7               * A SOURCE, DESTINATION, OR QUEUE IS MAIL.
 8               * ANY TERMINAL FROM INVITATION LIST TERMS CAN BE THE AOP
 9               * BY ENTERING IAMAOP AT LOGIN.
10               * THIS APPLICATION IS INITIATED WHEN ONE OF THE TERMINALS
11               * IN LIST TERMS LOGS IN AS THE AOP.
12               *
13                APPLICATION-NAME IS MAILBOX
14                SIGNATURE IS "MAIL"
15                DUMP-FILE IS DMPFL  OWNER IS "MCS0469"
16                MONITOR-FILE IS MNTRFL  OWNER IS "MCS0469"
17                OPERATOR IS TERMS  PASSWORD IS "IAMAOP"
18                INITIATION IS EXPLICIT
19
20
21                APPLICATION PROGRAM DIVISION
22               *
23               * THERE ARE 2 COBOL PROGRAMS IN THIS APPLICATION -
24               *      MSGDROP AND TOOMANY.
25               * THEIR INVOCATION FILES ARE NAMED MSGFILE AND TOOFILE.
26               * THIS DIVISION ALSO NAMES THE PROGRAMS' RESPONSE QUEUES.
27               *
28                PROGRAM IS MSGDROP
29                    INVOCATION-FILE IS MSGFILE  OWNER IS "MCS0469"
30                    RESPONSE-QUEUE IS QRESP1
31                PROGRAM IS TOOMANY
32                    INVOCATION-FILE IS TOOFILE  OWNER IS "MCS0469"
33                    RESPONSE-QUEUE IS QRESP2
34
35
36                APPLICATION DATA DIVISION
37               *
38               * INPUT MESSAGES ARE ASSIGNED A SERIAL NUMBER BY MCS -
39               * IT IS ECHOED BACK TO THE SOURCE.
40               * INPUT MESSAGES ARE ENQUEUED BASED ON THE CONTENTS OF THE
41               * FIRST SEVEN CHARACTERS.
42               *
```

Figure 6-1. ADLP Source Listing of Application MAILBOX (Sheet 1 of 4)

```
 43                MESSAGE IS MSG1
 44                    SERIAL-NUMBER IS GENERATED WITH ECHO
 45                SEGMENT IS SEG1
 46                    LENGTH IS 60 CHARACTERS
 47                    FIELD IS FIELD1 STARTS AT CHARACTER 1
 48                    EXTENDS FOR 7 CHARACTERS
 49                    CONDITION IS COND1 VALUE "TO BOBG"
 50                    CONDITION IS COND2 VALUE "TO SUEC"
 51                    CONDITION IS COND3 VALUE "TO TOMR"
 52                    CONDITION IS COND4 VALUE "TO LEOW"
 53                    CONDITION IS COND5
 54                        VALUE "BOBG" "SUEC" "TOMR" "LEOW"
 55
 56
 57                SOURCE-DESTINATION DIVISION
 58               *
 59               * INVITATION LIST TERMS CONSISTS OF FOUR INTERACTIVE TERMINALS,
 60               * WHICH ARE ALSO INCLUDED IN BROADCAST LIST OUTPUTS.
 61               * INQLOG, OUTQLOG, AND NAMQLOG ARE JOURNALS.
 62               *
 63                INVITATION-LIST IS TERMS
 64                    SOURCES ARE TERM1 AND TERM2
 65                        AND TERM3 AND TERM4
 66                    MESSAGES ARE MSG1
 67                    MODE IS COMMAND
 68                    STATUS IS ENABLED
 69                BROADCAST-LIST IS OUTPUTS
 70                    DESTINATIONS ARE TERM1 AND TERM2
 71                        AND TERM3 AND TERM4
 72
 73                SYMBOLIC-NAME IS TERM1
 74                    TYPE IS INTERACTIVE
 75                SYMBOLIC-NAME IS TERM2
 76                    TYPE IS INTERACTIVE
 77                SYMBOLIC-NAME IS TERM3
 78                    TYPE IS INTERACTIVE
 79                SYMBOLIC-NAME IS TERM4
 80                    TYPE IS INTERACTIVE
 81                SYMBOLIC-NAME IS INQLOG
 82                    TYPE IS JOURNAL
 83                    ALIAS IS FILE1
 84                SYMBOLIC-NAME IS OUTQLOG
 85                    TYPE IS JOURNAL
 86                    ALIAS IS FILE2
 87                SYMBOLIC-NAME IS NAMQLOG
 88                    TYPE IS JOURNAL
 89                    ALIAS IS FILE3
 90
 91
 92                QUEUE DIVISION
 93                INPUT SECTION
 94               *
 95               * DATA IN INPUT QUEUE NAMEQ IS USED BY MSGDROP TO DETERMINE
 96               * WHICH USER'S MESSAGES TO PROCESS.
 97               * MESSAGES FOR USERS ARE STORED IN DISK RESIDENT QUEUES
 98               * BOBGQ, SUECQ, TOMRQ, LEOWQ.
 99               * THE COBOL PROGRAMS' RESPONSE QUEUES ARE DEFINED IN THIS SECTION.
100               *
101                QUEUE IS NAMEQ
102                    JOURNAL IS NAMQLOG
103                    STATUS IS ENABLED
```

Figure 6-1. ADLP Source Listing of Application MAILBOX (Sheet 2 of 4)

```
104                 QUEUE IS INQ
105                     JOURNAL IS INQLOG ON INPUT  OUTQLOG ON OUTPUT
106                     MEDIUM IS DISK
107                     STATUS IS ENABLED
108                  SUB-QUEUE-1 IS INQ1
109                   SUB-QUEUE-2 IS INQ12
110                    SUB-QUEUE-3 IS BOBGQ
111                     RESIDENCY IS BGFILE
112                    SUB-QUEUE-3 IS SUECQ
113                     RESIDENCY IS SCFILE
114                   SUB-QUEUE-2 IS TOMRQ
115                     RESIDENCY IS TRFILE
116                  SUB-QUEUE-1 IS LEOWQ
117                     RESIDENCY IS LWFILE
118                 QUEUE IS QRESP1
119                 QUEUE IS QRESP2
120
121                 OUTPUT SECTION
122                *
123                * EACH USER'S MESSAGES ARE SENT TO A TERMINAL THROUGH
124                * OUTPUT QUEUES BOUTQ, SOUTQ, TOUTQ, LOUTQ.
125                * THE OTHER OUTPUT QUEUE IS USED WHEN MSGDROP RECEIVES AN
126                * INVALID MESSAGE, INITIATES PROCESSING, AND
127                * TERMINATES PROCESSING.
128                *
129                 QUEUE IS BOUTQ
130                     STATUS IS ENABLED
131                 QUEUE IS SOUTQ
132                     STATUS IS ENABLED
133                 QUEUE IS TOUTQ
134                     STATUS IS ENABLED
135                 QUEUE IS LOUTQ
136                     STATUS IS ENABLED
137                 QUEUE IS OUTQ
138                     STATUS IS ENABLED
139
140                 ROUTING SECTION
141                *
142                * INPUT MESSAGES ARE ENQUEUED BASED ON THE CONTENTS OF THE
143                * FIRST SEVEN CHARACTERS.
144                * MESSAGES CONSISTING OF A USER'S NAME FOLLOWED BY THREE BLANKS
145                * ARE ROUTED TO NAMEQ AND ENQUEUED.
146                * OTHER MESSAGES ARE ROUTED TO INQ AND ENQUEUED IN THE SIMPLE
147                * QUEUES ASSOCIATED WITH EACH USER.
148                * MESSAGES NOT MATCHING ONE OF THE FIVE CONDITIONS
149                * ARE DISCARDED.
150                * OUTBOUND MESSAGES ARE ROUTED THROUGH THE OUTPUT QUEUES
151                * TO EACH TERMINAL OR TO THE LIST OUTPUTS.
152                *
153                 SELECT INPUT QUEUES BASED ON CONTENTS OF FIELD1
154                     ROUTE TO NAMEQ FOR COND5
155                     ROUTE TO INQ OTHERWISE
156                 SELECT SUB-QUEUES OF INQ BASED ON CONTENTS OF FIELD1
157                     ROUTE TO LEOWQ FOR COND4
158                     ROUTE TO INQ1 OTHERWISE
159                 SELECT SUB-QUEUES OF INQ1 BASED ON CONTENTS OF FIELD1
160                     ROUTE TO TOMRQ FOR COND3
161                     ROUTE TO INQ12 OTHERWISE
162                 SELECT SUB-QUEUES OF INQ12 BASED ON CONTENTS OF FIELD1
163                     ROUTE TO SUECQ FOR COND2
164                     ROUTE TO BOBGQ FOR COND1
165                 SELECT OUTPUT QUEUES BASED ON DESTINATION
166                     ROUTE TO BOUTQ TO TERM1
167                     ROUTE TO SOUTQ TO TERM2
168                     ROUTE TO TOUTQ TO TERM3
169                     ROUTE TO LOUTQ TO TERM4
170                     ROUTE TO OUTQ TO OUTPUTS
```

Figure 6-1. ADLP Source Listing of Application MAILBOX (Sheet 3 of 4)

```
171
172
173                APPLICATION PROCESSING DIVISION
174               *
175               * PROGRAM TOOMANY IS INVOKED WHEN ANY USER HAS MORE THAN
176               * 12 MESSAGES IN A QUEUE.
177               *
178                USE WHEN SIZE OF BOBGQ EXCEEDS 12 MESSAGES
179                    INVOKE TOOMANY
180                USE WHEN SIZE OF SUECQ EXCEEDS 12 MESSAGES
181                    INVOKE TOOMANY
182                USE WHEN SIZE OF TOMRQ EXCEEDS 12 MESSAGES
183                    INVOKE TOOMANY
184                USE WHEN SIZE OF LEOWQ EXCEEDS 12 MESSAGES
185                    INVOKE TOOMANY
```

Figure 6-1. ADLP Source Listing of Application MAILBOX (Sheet 4 of 4)

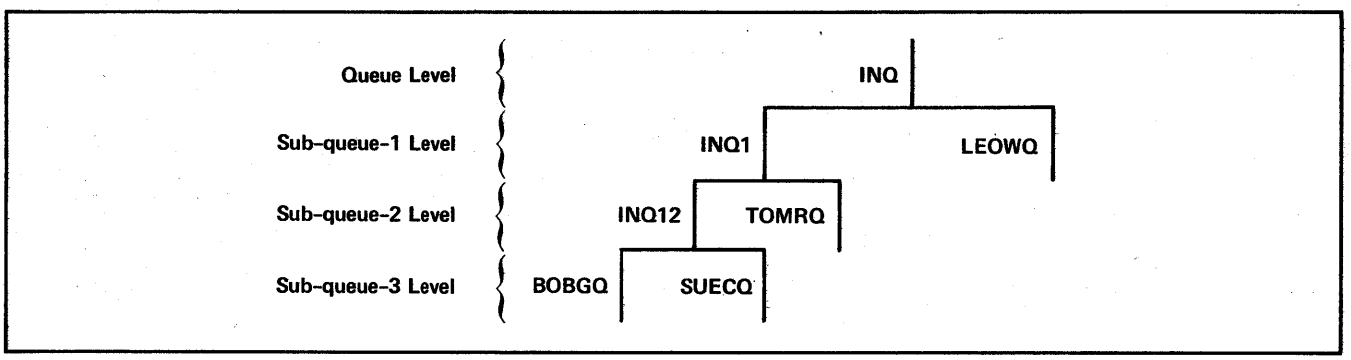


Figure 6-2. Compound Input Queue Structure for Application MAILBOX

```
JOB Statement.
USER Statement
PACKNAM,MCS.        <-- Directs permanent file requests to MCS mass storage device.
GET,MDBIFIL.        <-- Get binary file on which program MSGDROP is compiled.
MDBIFIL,*APPL=MAILBOX.  <-- Initiate execution of MSGDROP and call MCS application MAILBOX.
--EOR--
END OF FILE
```

Figure 6-3. Invocation File MSFFILE

```
JOB Statement.
USER Statement.
PACKNAM,MCS.        <-- Directs permanent file requests to MCS mass storage device.
GET,TMBIFIL.        <-- Get binary file on which program TOOMANY is compiled.
TMBIFIL,*APPL=MAILBOX.  <-- Initiate execution of TOOMANY and call MCS application MAILBOX.
--EOR--
END OF FILE
```

Figure 6-4. Invocation File TOOFILE

```
1
2
3               IDENTIFICATION DIVISION.
4               PROGRAM-ID.  MSGDROP.
5              *
6              * PROGRAM MSGDROP IS INVOKED FROM A TERMINAL BY USE OF THE
7              * AOP INVOKE COMMAND.
8              *
9               ENVIRONMENT DIVISION.
10              CONFIGURATION SECTION.
11              SOURCE-COMPUTER.  CYBER.
12              OBJECT-COMPUTER.  CYBER.
13
14              DATA DIVISION.
15              WORKING-STORAGE SECTION.
16             *
17             * SETS UP STORAGE AREAS TO STORE VARIOUS MESSAGES AND TO
18             * SEND A MESSAGE WHEN AN ERROR OCCURS.
19             *
20              01  MSG-SLOT         PIC  X(80)  VALUE SPACES.
21              01  SHORT-SLOT       PIC  X(20)  VALUE SPACES.
22              01  NAME             PIC  X(05)  VALUE SPACES.
23              01  I                PIC  9(01)  VALUE ZERO.
24              01  ERR-MSG.
25                  05 FILLER        PIC  X(49)
26                     VALUE "ERROR DURING EXECUTE OF LAST SEND, STATUS KEY IS ".
27                  05 E-STAT-KEY    PIC  X(02)  VALUE SPACES.
28                  05 FILLER        PIC  X(15)  VALUE ", ERROR KEY IS ".
29                  05 E-ERR-KEY     PIC  X(01)  VALUE SPACE.
30
31              COMMUNICATION SECTION.
32             *
33             * THE AREA PROGRAM MSGDROP USES TO COMMUNICATE WITH MCS.
34             * IN THE INPUT CD AREA, THE SYMBOLIC QUEUE FIELDS MUST BE SET
35             * BEFORE A MESSAGE IS RECEIVED.
36             * MCS UPDATES THE OTHER INPUT CD AREA FIELDS
37             * WHEN RECEIVE EXECUTES.
38             *
39              CD  IN-MSG; FOR INPUT
40                  SYMBOLIC QUEUE IS INPUT-Q
41                  SYMBOLIC SUB-QUEUE-1 IS SUB-Q1
42                  SYMBOLIC SUB-QUEUE-2 IS SUB-Q2
43                  SYMBOLIC SUB-QUEUE-3 IS SUB-Q3
44                  MESSAGE DATE IS IN-DATE
45                  MESSAGE TIME IS IN-TIME
46                  SYMBOLIC SOURCE IS IN-SOURCE
47                  TEXT LENGTH IS IN-LENGTH
48                  END KEY IS IN-KEY
49                  STATUS KEY IS IN-STATUS
50                  MESSAGE COUNT IS IN-COUNT.
51             *
52             * THE AREA MSGDROP USES WHEN SENDING MESSAGES TO TERMINALS
53             * THROUGH MCS QUEUES.
54             * DESTINATION COUNT, TEXT LENGTH, AND SYMBOLIC DESTINATION
55             * FIELDS MUST BE SET - ERROR KEY AND STATUS KEY FIELDS
56             * ARE UPDATED BY MCS WHEN SEND EXECUTES.
57             *
58              CD  OUT-MSG; FOR OUTPUT
59                  DESTINATION COUNT IS D-COUNT
60                  TEXT LENGTH IS OUT-LENGTH
61                  STATUS KEY IS OUT-STATUS
62                  DESTINATION TABLE OCCURS 4 TIMES
63                      INDEXED BY D-TABLE
64                  ERROR KEY IS OUT-KEY
65                  SYMBOLIC DESTINATION IS OUT-DEST.
66
```

Figure 6-5. Source Listing of Program MSGDROP (Sheet 1 of 4)

```
 PROCEDURE DIVISION.

 PROMPT-TERM.
*
* SET THE OUTPUT CD AREA.
* SEND THE PROMPT "NAME?" TO THE LOGGED IN TERMINAL
* THAT INVOKED MSGDROP.
*
     PERFORM SET-OUTCD
     MOVE "OUTPUTS" TO OUT-DEST (1)
     MOVE "NAME?" TO SHORT-SLOT
     MOVE 6 TO OUT-LENGTH
     SEND OUT-MSG FROM SHORT-SLOT WITH EMI.

 GET-NAME.
*
* MOVE SPACES TO INPUT CD AREA.
* SET INPUT CD AREA TO RECEIVE A MESSAGE FROM NAMEQ.
* RECEIVE LOGGED IN USER'S NAME.
* GO TO THE PROCEDURE TO PROCESS THE USER'S MESSAGES.
* WHEN THE USER'S NAME IS NOT VALID, GO TO
* INVALID NAME PROCEDURE.
*
     PERFORM CLEAR-INCD
     MOVE "NAMEQ" TO INPUT-Q
     RECEIVE IN-MSG MESSAGE INTO NAME.
     IF NAME EQUALS "BOBG" PERFORM BOBG-MSG
       ELSE IF NAME EQUALS "SUEC" PERFORM SUEC-MSG
         ELSE IF NAME EQUALS "TOMR" PERFORM TOMR-MSG
           ELSE IF NAME EQUALS "LEOW" PERFORM LEOW-MSG
             ELSE PERFORM INVALID-NAME.

 WRAP-UP.
*
* ALL MESSAGES HAVE BEEN PROCESSED AND SENT TO A TERMINAL.
* SET THE OUTPUT CD AREA AND SEND A MESSAGE NOTIFYING
* USER THAT PROCESSING IS COMPLETE.
*
     PERFORM SET-OUTCD
     MOVE "OUTPUTS" TO OUT-DEST (1)
     MOVE "END OF RUN" TO SHORT-SLOT
     MOVE 11 TO OUT-LENGTH
     SEND OUT-MSG FROM SHORT-SLOT WITH EMI
       AFTER ADVANCING 1 LINES
     STOP RUN.

 BOBG-MSG.
*
* MOVE SPACES TO THE INPUT CD AREA, AND THEN MOVE IN
* QUEUE NAMES TO RECEIVE MESSAGES FROM BOBGQ.
* SET THE OUTPUT CD AREA TO SEND BOB'S MESSAGES TO HIM AT TERM1.
* GET A COUNT OF THE NUMBER OF MESSAGES TO PROCESS.
*
     PERFORM CLEAR-INCD
     MOVE "INQ" TO INPUT-Q
     MOVE "INQ1" TO SUB-Q1
     MOVE "INQ12" TO SUB-Q2
     MOVE "BOBGQ" TO SUB-Q3
     PERFORM SET-OUTCD
     MOVE "TERM1" TO OUT-DEST (1)
     ACCEPT IN-MSG MESSAGE COUNT
     PERFORM MOVE-MSG IN-COUNT TIMES.
```

Figure 6-5. Source Listing of Program MSGDROP (Sheet 2 of 4)

```
130              SUEC-MSG.
131             *
132             * MOVE SPACES TO THE INPUT CD AREA, AND THEN MOVE IN
133             * QUEUE NAMES TO RECEIVE MESSAGES FROM SUECQ.
134             * SET THE OUTPUT CD AREA TO SEND SUE'S MESSAGES TO HER AT TERM2.
135             * GET A COUNT OF THE NUMBER OF MESSAGES TO PROCESS.
136             *
137                  PERFORM CLEAR-INCD
138                  MOVE "INQ" TO INPUT-Q
139                  MOVE "INQ1" TO SUB-Q1
140                  MOVE "INQ12" TO SUB-Q2
141                  MOVE "SUECQ" TO SUB-Q3
142                  PERFORM SET-OUTCD
143                  MOVE "TERM2" TO OUT-DEST (1)
144                  ACCEPT IN-MSG MESSAGE COUNT
145                  PERFORM MOVE-MSG IN-COUNT TIMES.
146
147              TOMR-MSG.
148             *
149             * MOVE SPACES TO THE INPUT CD AREA, AND THEN MOVE IN
150             * QUEUE NAMES TO RECEIVE MESSAGES FROM TOMRQ.
151             * SET THE OUTPUT CD AREA TO SEND TOM'S MESSAGES TO HIM AT TERM3.
152             * GET A COUNT OF THE NUMBER OF MESSAGES TO PROCESS.
153             *
154                  PERFORM CLEAR-INCD
155                  MOVE "INQ" TO INPUT-Q
156                  MOVE "INQ1" TO SUB-Q1
157                  MOVE "TOMRQ" TO SUB-Q2
158                  MOVE "TERM3" TO OUT-DEST (1)
159                  ACCEPT IN-MSG MESSAGE COUNT
160                  PERFORM MOVE-MSG IN-COUNT TIMES.
161
162              LEOW-MSG.
163             *
164             * MOVE SPACES TO THE INPUT CD AREA, AND THEN MOVE IN
165             * QUEUE NAMES TO RECEIVE MESSAGES FROM LEOWQ.
166             * SET THE OUTPUT CD AREA TO SEND LEO'S MESSAGES TO HIM AT TERM4.
167             * GET A COUNT OF THE NUMBER OF MESSAGES TO PROCESS.
168             *
169                  PERFORM CLEAR-INCD
170                  MOVE "INQ" TO INPUT-Q
171                  MOVE "LEOWQ" TO SUB-Q1
172                  PERFORM SET-OUTCD
173                  MOVE "TERM4" TO OUT-DEST (1)
174                  ACCEPT IN-MSG MESSAGE COUNT
175                  PERFORM MOVE-MSG IN-COUNT TIMES.
176
177              INVALID-NAME.
178             *
179             * WHEN AN INVALID NAME IS ENTERED FROM A TERMINAL,
180             * SEND A MESSAGE TO THE TERMINAL AND TERMINATE THE PROGRAM.
181             *
182                  PERFORM SET-OUTCD
183                  MOVE "OUTPUTS" TO OUT-DEST (1)
184                  MOVE "INVALID NAME, PROGRAM TERMINATING" TO MSG-SLOT
185                  MOVE 33 TO OUT-LENGTH
186                  SEND OUT-MSG FROM MSG-SLOT WITH EMI
187                      AFTER ADVANCING 1 LINES
188                  PERFORM WRAP-UP.
189
```

Figure 6-5. Source Listing of Program MSGDROP (Sheet 3 of 4)

```
190              CLEAR-INCD.
191             *
192             * THIS PROCEDURE MOVES SPACES TO THE INPUT CD AREA PRIOR TO
193             * SETTING IT WITH QUEUE AND SUBQUEUE NAMES.
194             *
195                  MOVE SPACES TO INPUT-Q
196                  MOVE SPACES TO SUB-Q1
197                  MOVE SPACES TO SUB-Q2
198                  MOVE SPACES TO SUB-Q3
199                  MOVE ZEROES TO IN-DATE
200                  MOVE ZEROES TO IN-TIME
201                  MOVE SPACES TO IN-SOURCE
202                  MOVE ZEROES TO IN-LENGTH
203                  MOVE SPACES TO IN-KEY
204                  MOVE SPACES TO IN-STATUS
205                  MOVE ZEROES TO IN-COUNT.
206
207              SET-TABLE.
208             *
209             * THIS PROCEDURE CLEARS OUT THE TABLE THAT
210             * REFERENCES SYMBOLIC DESTINATIONS.
211             *
212                  MOVE SPACES TO OUT-KEY (I)
213                  MOVE SPACES TO OUT-DEST (I)
214                  ADD 1 TO I.
215
216              SET-OUTCD.
217             *
218             * THIS PROCEDURE PREPARES THE OUTPUT CD AREA PRIOR TO
219             * SENDING A MESSAGE.
220             *
221                  MOVE 1 TO D-COUNT
222                  MOVE SPACES TO OUT-STATUS
223                  MOVE 1 TO I
224                  PERFORM SET-TABLE 4 TIMES.
225
226              ERR-RTN.
227             *
228             * WHEN THE OUTPUT CD AREA STATUS KEY OR ERROR KEY
229             * INDICATES AN ERROR, SEND A MESSAGE TO THE TERMINAL.
230             *
231                  MOVE OUT-STATUS TO E-STAT-KEY
232                  MOVE OUT-KEY (1) TO E-ERR-KEY
233                  MOVE 77 TO OUT-LENGTH
234                  SEND OUT-MSG FROM ERR-MSG WITH EMI
235                      AFTER ADVANCING 1 LINES.
236
237              MOVE-MSG.
238             *
239             * RECEIVE A USER'S MESSAGES FROM A QUEUE,
240             * AND THEN SEND THE MESSAGES TO THE USER AT THE TERMINAL.
241             * CHECK TO SEE IF ERROR MESSAGE SHOULD BE SENT.
242             *
243                  MOVE SPACES TO MSG-SLOT
244                  RECEIVE IN-MSG MESSAGE INTO MSG-SLOT
245                  MOVE IN-LENGTH TO OUT-LENGTH
246                  SEND OUT-MSG FROM MSG-SLOT WITH EMI
247                      AFTER ADVANCING 1 LINES.
248                  IF OUT-STATUS IS NOT EQUAL TO "00"
249                      OR OUT-KEY (1) IS NOT EQUAL TO "0"
250                      PERFORM ERR-RTN.
251
252              END-MSGDROP.
```

Figure 6-5. Source Listing of Program MSGDROP (Sheet 4 of 4)

```
1
2
3               IDENTIFICATION DIVISION.
4               PROGRAM-ID.  TOOMANY.
5
6               ENVIRONMENT DIVISION.
7               CONFIGURATION SECTION.
8               SOURCE-COMPUTER.  CYBER.
9               OBJECT-COMPUTER.  CYBER.
10              INPUT-OUTPUT SECTION.
11              FILE-CONTROL.
12                  SELECT PRINT-FILE ASSIGN TO "OUTPUT".
13
14              DATA DIVISION.
15              FILE SECTION.
16             *
17             * DEFINES THE FILE TO WHICH MESSAGES ARE WRITTEN.
18             *
19              FD  PRINT-FILE
20                  LABEL RECORDS ARE OMITTED.
21              01  PRINT-REC       PIC  X(133).
22
23              WORKING-STORAGE SECTION.
24             *
25             * SETS UP A STORAGE AREA TO RECEIVE MESSAGES FROM A QUEUE.
26             *
27              01  STORE-IT        PIC  X(80).
28
29              COMMUNICATION SECTION.
30             *
31             * THE AREA PROGRAM TOOMANY USES TO COMMUNICATE WITH MCS.
32             * TOOMANY IS INVOKED BY MCS WHEN THE NUMBER OF MESSAGES IN
33             * A QUEUE IS GREATER THAN 12.
34             * THE QUEUE AND SUBQUEUE NAMES OF THE QUEUES REACHING MESSAGE
35             * THRESHOLD ARE ENTERED IN THE SYMBOLIC QUEUE FIELDS BY MCS.
36             * NO OUTPUT CD IS DEFINED BECAUSE TOOMANY ONLY RECEIVES
37             * MESSAGES FROM QUEUES - IT DOES NOT SEND MESSAGES TO QUEUES.
38             *
39              CD  IN-COMING; FOR INITIAL INPUT
40                  SYMBOLIC QUEUE IS INIT-Q
41                  SYMBOLIC SUB-QUEUE-1 IS INIT-SQ1
42                  SYMBOLIC SUB-QUEUE-2 IS INIT-SQ2
43                  SYMBOLIC SUB-QUEUE-3 IS INIT-SQ3
44                  MESSAGE DATE IS MSG-DATE
45                  MESSAGE TIME IS MSG-TIME
46                  SYMBOLIC SOURCE IS MSG-SOURCE
47                  TEXT LENGTH IS MSG-LEN
48                  END KEY IS E-KEY
49                  STATUS KEY IS S-KEY
50                  MESSAGE COUNT IS MSG-CNT.
51
52              PROCEDURE DIVISION.
53              BEGIN.
54                  OPEN OUTPUT PRINT-FILE.
55
56              GET-MSG.
57             *
58             * TOOMANY RECEIVES MESSAGES AND PRINTS THEM OUT.
59             * PROCESSING CONTINUES UNTIL ALL MESSAGES HAVE BEEN
60             * PRINTED OUT.
61             *
62                  MOVE SPACES TO STORE-IT.
63                  RECEIVE IN-COMING MESSAGE INTO STORE-IT
64                      NO DATA PERFORM END-IT.
65                  MOVE SPACES TO PRINT-REC.
66                  MOVE STORE-IT TO PRINT-REC.
67                  WRITE PRINT-REC AFTER 2.
68                  GO TO GET-MSG.
69
70              END-IT.
71                  CLOSE PRINT-FILE
72                  STOP RUN.
```

Figure 6-6. Source Listing of Program TOOMANY

```
FAMILY:  family name, user name, password, mcs
MCS 1.0 79/10/12. 15.19.07.
MCS APPLICATION ?mailbox iamaop
APPLICATION INITIATED
SYMBOLIC-NAME ?term1
COMMAND MODE
HELLO - YOU'RE THE AOP
?data
DATA MODE
?to suec have you started work on the new project yet? bob
SERIAL NUMBER =          10
?to suec the repairman will be there to fix the printer on tues. bob
SERIAL NUMBER =          11
?to suec our machine time on thursday is now 3:00. bob
SERIAL NUMBER =          12
?to suec the bill from the phone co. is $135. please pay it. bob
SERIAL NUMBER =          13
?to loew from bob - the journal with the paper in it is the jan. 79 issue.
ROUTING FAILURE - MESSAGE DISCARDED
?to leow from bob - the journal with the paper in it is the jan. 79 issue.
SERIAL NUMBER =          14
?)
COMMAND MODE
?invoke msgdrop
?data
DATA MODE
?
NAME? bobg
SERIAL NUMBER =          15
?
TO BOBG THE CLASS ON WED. HAS BEEN CHANGED TO RM. 112. LEO
TO BOBG I HAVE SENT YOUR BOOKS VIA AIRMAIL. TOM 10/11, 11:00.
TO BOBG OUR PLANE RESERVATIONS ARE NOW ON FARAWAY, FLIGHT 17. LEO
TO BOBG THE CATALOG YOU ORDERED WILL BE SENT MONDAY. SUE
TO BOBG THE NEW CODE CORRECTS THAT PROBLEM WE FOUND. SUE
END OF RUN
?)
COMMAND MODE
?hello
MCS ENDED  79/10/12.  15.37.37.MCS      CONNECT TIME 00.18.41.
```

Terminal user completes login to MCS, switches to data mode, and enters several messages.

Typing error causes a routing failure.

Break-2 character to switch to command mode.

Execution of program MSGDROP initiated.

Terminal switched to data mode to receive program output.

Message from program indicating all of user's messages processed.

Terminal user switches to command mode, ends connection to MCS, and prepares to log into another network application program.

Figure 6-7. Application MAILBOX Terminal User Session

```
INPUT QUEUE DISPLAY   15.32.46.
QUEUE NAME      SUB QUEUE 1     SUB QUEUE 2     SUB QUEUE 3      NUMBER LAST
NAMEQ.........E                                                       0     0
INQ...........E                                                      21    21
INQ...........E INQ1..........E INQ12.........E BOBGQ.........E      5     5
INQ...........E INQ1..........E INQ12.........E SUECQ.........E     12    17
INQ...........E INQ1..........E TOMRQ.........E                     2    19
INQ...........E LEOWQ.........E                                     2    21
QRESP1........E                                                       0     0
QRESP2........E                                                       0     0
?
```

Figure 6-8. Application MAILBOX Input Queue Display

```
TO SUEC PLEASE SEND ME THE MINUTES OF LAST WEEK'S MEETING. TOM
TO SUEC THE CLASS ON WED. HAS BEEN CHANGED TO RM. 112. LEO
TO SUEC OUR MACHINE TIME ON THURSDAY IS NOW 3:00. BOB
TO SUEC CAN YOU MAIL THE REGISTRATION FOR THE SEMINAR FOR US? LEO
TO SUEC HAVE YOU STARTED WORK ON THE NEW PROJECT YET? BOB
TO SUEC THE REPAIRMAN WILL BE THERE TO FIX THE PRINTER ON TUES. BOB
TO SUEC YOU CAN PICK UP THE TAPES YOU ORDERED ANY TIME AFTER 10/11. TOM
TO SUEC FROM LEO - ARE YOU AVAILABLE FOR A MEETING THURS. AT 5:00?
TO SUEC LUNCH TUESDAY AT THE PINES IS OK. TOM
TO SUEC THE BILL FROM THE PHONE CO. IS $135. PLEASE PAY IT. BOB
TO SUEC THE AAC MEETING HAS BEEN POSTPONED UNTIL FURTHER NOTICE. LEO
TO SUEC CALL ME BEFORE NOON TODAY (MON.). I HAVE SOME NEWS. LEO
TO SUEC HOW COME I HAVEN'T HEARD FROM YOU? TOM
```

Figure 6-9. Output From Program TOOMANY

MCS provides a set of commands that allows terminal users to control the operation of terminals connected to an MCS application. These commands can be entered only when the terminal is in command mode (as described in section 2). The user commands are summarized in table 7-1.

TABLE 7-1. USER COMMANDS

Command	Description
DATA	Switches terminal from command mode to data mode.
DISABLE	Disables the terminal.
DISPLAY INPUT	Displays information about input queues.
DISPLAY OUTPUT	Displays information about output queues.
DISPLAY LAST	Displays serial number of last message enqueued from terminal.
DISPLAY ALL	Displays information about input and output queues.
DISPLAY name	Displays information about the specified queue.
ENABLE	Enables the terminal.
END	Terminates the terminal's connection to MCS.
LOGIN, LOGON, HELLO	Begins LOGIN dialog.
LOGOUT, LOGOFF, GOODBYE, BYE	Terminates the connection to MCS and exits the network.
MESSAGE "string"	Sends a message to the AOP.

The MCS commands consist of a verb followed by one or more keywords; certain commands can also include one or more user-supplied names or literals. Each verb, keyword, name, or literal in a command must be separated by one or more blanks.

The commands described in this section, with the exception of DATA, END, and the login and logout commands, are also available to the COBOL programs that are included in an MCS application. Messages treated as commands must be sent to a special destination. The name of this destination is the MCS application name. Thus, a SEND to the application name sends the message text to MCS where it is processed as a command. MCS responses to these commands are enqueued in the program's response queue. An illegal or unrecognizable command results in an error message. A complete list of error messages appears in appendix B.

DATA COMMAND

The DATA command switches the terminal from command mode to data mode. This command has the form:

```
DATA
```

When a terminal is in data mode, messages entered at the terminal are routed to an input queue, and output messages are sent to the terminal. The terminal can be returned to command mode by entering break-2. The following example illustrates the use of this command:

```
MCS 1.0  79/07/05.  15:45:30
MCS APPLICATION ?  able
SYMBOLIC NAME ?  xyz
COMMAND MODE
?data
DATA MODE
```

DISABLE COMMAND

The DISABLE command breaks the logical path between an MCS application and the terminal. This command has the form:

```
DISABLE
```

While a terminal is disabled, it remains connected to the MCS application. When the terminal is defined in the application definition as a source, it cannot send messages to the application. When the terminal is defined as a destination, it does not receive messages from the application. When the terminal is defined as interactive (both a source and a destination), it neither sends nor receives application messages. A disabled terminal can, however, perform all command mode activities.

DISPLAY COMMAND

The DISPLAY command displays information about the the current status of input and output queues defined for an application. The status of the queue (enabled or disabled), the number of messages in the queue, and the serial number of the last message enqueued can be displayed. This command has the following forms:

- DISPLAY LAST

  displays the serial number of the last message received from the terminal.

- DISPLAY INPUT

  displays information about the status of all the input queues. The format of this display is shown in figure 7-1.

- DISPLAY OUTPUT

  displays information about the status of all the output queues. The format of this display is shown in figure 7-2.

INPUT QUEUE DISPLAY hh.mm.ss.

QUEUE NAME		SUB QUEUE 1		SUB QUEUE 2		SUB QUEUE 3		NUMBER	LAST
aaaaaaaa	s	bbbbbbbb	s	cccccccc	s	dddddddd	s	nn	kk
aaaaaaaa	s	bbbbbbbb	s	cccccccc	s	dddddddd	s	nn	kk
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
aaaaaaaa	s	bbbbbbbb	s	cccccccc	s	dddddddd	s	nn	kk

hh.mm.ss	Time (hour.minute.second) command was executed
aaaaaaaa	Queue name
bbbbbbbb	Name of level 1 subqueue
cccccccc	Name of level 2 subqueue
dddddddd	Name of level 3 subqueue
s	Queue status; one of the following:
	E enabled
	D disabled
nn	Number of complete messages in queue
kk	Serial number of last message enqueued

Figure 7-1. Input Queue Display Format

OUTPUT QUEUE DISPLAY hh.mm.ss.

QUEUE NAME		NUMBER
aaaaaaaa	s	nn
aaaaaaaa	s	nn
.	.	.
.	.	.
.	.	.
aaaaaaaa	s	nn

hh.mm.ss	Time (hour.minute.second) command was executed
aaaaaaaa	Queue name
s	Queue status; one of the following:
	E enabled
	D disabled
nn	Number of complete messages in queue

Figure 7-2. Output Queue Display Format

- DISPLAY ALL

  displays information about the status of both the input and output queues. This command combines the displays shown in figures 7-1 and 7-2.

- DISPLAY qname

  where qname is the name of an input or output queue defined in the application definition, displays information about the specified queue. When an input queue is specified, the display shown in figure 7-1 is generated. When an output queue is specified, the display shown in figure 7-2 is generated.

Any of the preceding DISPLAY commands can be used to write information to a monitor file. The command form:

DISPLAY name TO MONITOR-FILE

where name is LAST, INPUT, OUTPUT, ALL, or a queue name, writes the specified display to the monitor file. The syntax for defining a monitor file is described in section 4.

An example of the DISPLAY command is illustrated in figure 7-3.

ENABLE COMMAND

The ENABLE command reestablishes the logical path between MCS and the terminal after it has been broken by a DISABLE command. This command has the form:

ENABLE

While a terminal is enabled, it can send and receive application messages.

END COMMAND

The END command terminates the terminal's connection to MCS. This command has the form:

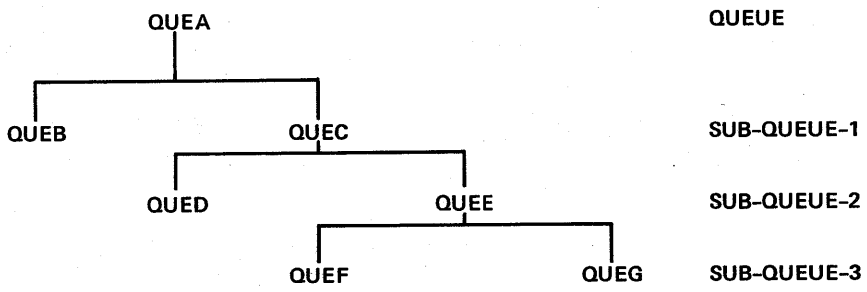
END [application]

where application is a network application program name. When an END command is entered without the application parameter, MCS operations are terminated, control is transferred to NVF, and the following message and prompt are displayed:

```
MCS ENDED yy/mm/dd. hh.mm.ss.
MCS CONNECT TIME hh.mm.ss.
xxxxxxx - APPLICATION:
```

The MCS connect time is the total time that has elapsed since the connection was established. In response to the APPLICATION prompt, the terminal user can enter an application name or log out.

Structure of Queue QUEA:



Terminal Display:

```
display quea
INPUT QUEUE DISPLAY   07.46.42
QUEUE NAME     SUB QUEUE 1    SUB QUEUE 2    SUB QUEUE 3    NUMBER LAST
QUEA.........E                                                  15   104
QUEA.........E QUEB.........E                                    8   104
QUEA.........E QUEC.........E                                    7   103
QUEA.........E QUEC.........E QUED.........E                     4   103
QUEA.........E QUEC.........E QUEE.........E                     3   102
QUEA.........E QUEC .........E QUEE.........E QUEF.........E     2   102
QUEA.........E QUEC.........E QUEE.........E QUEG.........E     1   101
```

Figure 7-3. Input Queue Display Example

Between the time the END command is entered and the APPLICATION prompt is displayed, the terminal is in a transitional state while network applications are switched. Commands entered during this transitional state are ignored.

When the END application form is used, the terminal is automatically connected to the specified application. For example:

```
END RBF
```

disconnects the terminal from MCS and establishes a connection to the Remote Batch Facility.

LOGIN COMMANDS

The login commands terminate the terminal's connection to MCS and initiate the login dialog. These commands have the forms:

```
LOGIN [application]

LOGON [application]

HELLO [application]
```

When any of the preceding forms are entered without the application parameter, MCS operations are terminated (as with the END command) and control is transferred to NVF, which begins the login dialog. When an application name is specified, the terminal is connected to the requested network application program. For example:

```
login
MCS ENDED 79/07/13. 08.44.29.
MCS CONNECT TIME 00.32.14.
CONTROL DATA CORP. NOS 1
FAMILY: fam1
USER NAME: xyz
PASSWORD: pass2
APPLICATION: iaf
```

transfers control from MCS, and establishes a connection to the Interactive Facility.

LOGOUT COMMANDS

The logout commands disconnect the terminal from MCS and from the network. These commands have the forms:

```
LOGOUT [application]

LOGOFF [application]

BYE [application]

GOODBYE [application]
```

When one of the preceding forms is entered, MCS operations are terminated as described under END Command, and control is transferred to NVF for the logout procedure described in section 2 under Disconnect Procedure. When an application is specified, the terminal is then connected to the requested network application program. For example:

```
logout
MCS ENDED 79/08/02. 14.02.25.
MCS CONNECT TIME 01.29.54.
```

terminates MCS and transfers control to NVF.

MESSAGE Command

The MESSAGE command sends a message to the application operator. This command has the form:

```
MESSAGE "string"
```

where string is a character string of 80 characters or less. When a MESSAGE command is entered, the specified character string is displayed at the application operator's console. An example of this command is:

```
message "from joe at term1: when is lunch?"
FROM AOP - LUNCH AT 2 OCLOCK
```

APPLICATION OPERATOR 8

The application operator (AOP) for an MCS application is the operator of a terminal that is logged in to the application as the AOP terminal. The AOP is provided with a set of commands in addition to those described in section 7. These commands allow the AOP to control various aspects of application operation including initiating, idling, and shutting down the application.

ADL includes syntax for defining the terminals eligible for AOP status and for defining a password to protect this status.

DEFINING AOP ELIGIBILITY

The application definition for an MCS application that is to have an AOP must include an OPERATOR paragraph and either a SIGNATURE paragraph or a PASSWORD clause in the Application Global Division.

The OPERATOR paragraph specifies the terminals that are eligible for designation as AOP terminals at login time. A single symbolic name, or a broadcast list or invitation list, can be declared; however, only a single AOP can exist at a given time for a given application.

Password protection for AOP status is provided by the SIGNATURE paragraph or PASSWORD clause. The SIGNATURE paragraph provides a password to be used for all application activities requiring one; the PASSWORD clause in the OPERATOR paragraph specifies a password to be used only for AOP designation.

Refer to section 4 for syntax descriptions and examples of the OPERATOR and SIGNATURE paragraphs, and the PASSWORD clause.

AOP LOGIN PROCEDURE

The terminal designated as the AOP terminal is logged in to the network in the same manner as other terminals, as described in section 2; however, the procedure for connecting to MCS differs slightly.

In response to the MCS APPLICATION prompt, enter the application name and password, separated by one or more blanks, as follows:

```
MCS APPLICATION ? application password
```

This password is the one defined in the application definition, and is distinct from the password specified when logging in to the network.

When a valid application name and password are entered, and when the requested application is already running, MCS issues the next prompt. When the requested application is not currently running, MCS initiates the application and displays the message:

```
APPLICATION INITIATED
```

MCS then issues the SYMBOLIC-NAME prompt. In response to this prompt, enter either one or two symbolic names, as described in section 2. However, these symbolic names must be eligible for AOP status (that is, they must be declared in the OPERATOR paragraph). When a valid symbolic name is entered, MCS displays the messages described in section 2. In addition, the following message signifies that the user has successfully logged in as the AOP:

```
HELLO - YOU'RE THE AOP
```

This completes the AOP login procedure.

AOP LOGIN DIAGNOSTICS

The user attempting to log in as the AOP can receive any of the diagnostic messages described in section 2. In addition, invalid entries can result in the messages described in the following paragraphs.

As with other login entries, the user is allowed a limited number of tries (installation option with a default of 3) to correctly enter the password. Each time an incorrect password is entered, MCS issues the message:

```
INVALID PASSWORD
```

and reissues the APPLICATION prompt. When the limit is exceeded, the message:

```
SOLICITATION LIMIT EXCEEDED
```

appears, and control returns to NVF, which reissues the APPLICATION prompt.

Only one AOP can be active at a time for a given MCS application. If a user enters a valid password, but an application operator is already active for the requested application, MCS issues the message:

```
APPLICATION OPERATOR ACTIVE
```

and reissues the APPLICATION prompt. The user must then log in as a non-AOP or wait until the current AOP logs out.

If the user enters a valid password and then, in response to the SYMBOLIC-NAME prompt, enters a symbolic name that is not eligible for AOP status (that is, a name not defined in the OPERATOR paragraph), MCS issues the message:

```
NOT VALIDATED AS AOP, NON-AOP ASSUMED
```

and the user is connected to the application as a non-AOP.

AOP COMMANDS

AOP commands can be issued only by the application operator (AOP). These commands provide the AOP with control over an application and the terminals connected to the application. AOP commands, which are summarized in table 8-1, have the same format as the user commands described in section 7.

TABLE 8-1. AOP COMMANDS

Command	Description	Command	Description
DISABLE	Disables the AOP terminal.	ENABLE INPUT	Enables all of the application's input queues.
DISABLE INPUT	Disables all of an application's input queues.	ENABLE OUTPUT	Enables all of the application's output queues.
DISABLE OUTPUT	Disables all of an application's output queues.	ENABLE ALL	Enables all of the application's input and output queues.
DISABLE ALL	Disables all of an application's input and output queues.	ENABLE name	Enables the specified input queue, output queue, source, or destination.
DISABLE name	Disables the specified source, destination, input queue, output queue, or list.	IDLE	Causes the application to suspend processing.
DISCONNECT name	Disconnects a terminal (other than AOP) from MCS.	INVOKE name	Initiates execution of the specified application program.
DISPLAY	Displays application status information.	MESSAGE "string"	Sends a message to all terminals connected to the application.
DISPLAY LAST	Displays serial number of last message received from terminal.	MESSAGE "string" TO name	Sends a message to the specified terminal.
DISPLAY INPUT	Displays input queue status information.	PURGE name	Deletes all partial messages from the specified destination.
DISPLAY OUTPUT	Displays output queue status information.	REROUTE name1 TO name2	Reroutes output destined for the terminal specified by name1 to the terminal specified by name2.
DISPLAY ALL	Displays input and output queue status information.	RESUME	Resumes processing of an idled application.
DISPLAY TERMINALS	Displays terminal status information.	RETRIEVE name number	Retrieves (displays) a message or group of messages from a queue.
DISPLAY PROGRAMS	Displays application program status information.	REVOKE name	Terminates execution of the specified application program.
DISPLAY name	Displays status information for the specified input queue, output queue, source, destination, or list.	SHUTDOWN	Disconnects terminals from MCS and immediately deactivates the application.
DUMP	Dumps environmental information about the MCS application to the application dump file.		
ENABLE	Enables the AOP terminal.		

The commands described in this section, with the exception of the login and logout commands, DATA, END, RETRIEVE, and RESUME are also available to COBOL programs that are part of an MCS application. Messages treated as commands must be sent to a special destination. The name of this destination is the MCS application name. Thus, a SEND to the application name sends the message text to MCS where it is processed as a command. Responses to MCS commands are enqueued in the program's response queue. An illegal or unrecognizable command results in an error message. A complete list of error messages appears in appendix B.

DISABLE COMMAND

The DISABLE command breaks the logical connection between MCS and a source, destination, invitation list, broadcast list, or queue. When a source or input queue is disabled, MCS does not accept application messages from the source or queue. When a destination or output queue is disabled, MCS does not deliver application messages to the destination or queue. The DISABLE command has the following forms:

- DISABLE

  disables the AOP terminal. When the AOP terminal is disabled, it cannot communicate with MCS. It can, however, continue to perform command mode activities.

- DISABLE INPUT

  disables all of the application's input queues.

- DISABLE OUTPUT

  disables all of the application's input and output queues.

- DISABLE ALL

  disables all of the application's input and output queues.

- DISABLE name

  disables the source, destination, broadcast list, invitation list, input queue, or output queue identified by the specified name; name must be a valid source, destination, list, input queue, or output queue name defined in the application definition. Note that when a terminal is identified by separate source and destination names, the terminal can be enabled as a destination and disabled as a source, and vice-versa. For example, if a terminal is associated with the source name SRC1 and the destination name DST1, and the AOP enters:

  DISABLE DST1

  the terminal can send, but cannot receive, application data messages.

Injection queues, collection queues, and response queues cannot be disabled.

DISCONNECT COMMAND

The DISCONNECT command disconnects a terminal (other than the AOP terminal) from MCS. This command has the form:

DISCONNECT name

where name is the symbolic name associated with a terminal. The effect of a DISCONNECT command is the same as if the specified terminal had entered an END command. When the AOP enters a DISCONNECT command, the following message is displayed at the disconnected terminal:

DISCONNECTED BY AOP

Disconnecting a terminal causes any partial message entered by that terminal to be purged. For example, when the AOP enters disconnect t11 the following message is displayed at terminal T11:

```
DISCONNECTED BY AOP
MCS ENDED 79/11/25. 10.18.06.
MCS CONNECT TIME 00.15.36.
TERM201-APPLICATION:
```

DISPLAY COMMAND

The DISPLAY command displays information about the status of the MCS application. This command has the following forms:

- DISPLAY LAST

  displays the serial number of the last message received from the AOP terminal. (Message serial numbers are described in section 4.)

- DISPLAY

  displays the following information about the current status of the MCS application:

  State of the application (RUN or IDLE)

  Number of messages in input queues

  Number of messages in output queues

  Number of terminals connected

  Number of programs executing within the MCS application

  The format of the display generated by the DISPLAY command is shown in figure 8-1.

- DISPLAY INPUT

  displays information about the status of all of the input queues defined for the current MCS application. The format of the display is identical to that of the DISPLAY INPUT command described in section 7.

- DISPLAY OUTPUT

  displays information about the status of all of the output queues defined for the current MCS application. The format of the display is identical to that generated by the DISPLAY OUTPUT command described in section 7.

- DISPLAY ALL

  displays information about the status of the input and output queues defined for the current MCS application. This command produces a DISPLAY INPUT and a DISPLAY OUTPUT display.

- DISPLAY TERMINALS

  displays information about the status of all terminals connected to the current MCS application. This information includes:

  Source name

  Destination name

APPLICATION DISPLAY hh.mm.ss. STATE = sss

ii INPUTS, jj OUTPUTS, mm TERMINALS, nn PROGRAMS

hh.mm.ss	Time (hour.minute.second) command was executed
sss	Application status; one of the following:
	RUN application running
	IDLE application idle
ii	Total number of messages in all input queues
jj	Total number of messages in all output queues
mm	Number of terminals connected to this application
nn	Number of programs currently executing (running or idle)

Figure 8-1. Application Status Display Format

Network terminal name defined by NAM

User name defined in MCS application definition

Operating mode (command mode or data mode)

Operating state (enabled or disabled)

Connection status (connected or disconnected)

The format of the display produced by the DISPLAY TERMINALS command is shown in figure 8-2.

- DISPLAY PROGRAMS

displays information about the status of all COBOL programs within the environment of the current MCS application. This information includes the following:

Program name

Program status (running, suspended, stopping, terminated, or inactive)

Number of times the program has been invoked by MCS

Number of send requests issued by the program

Number of receive requests issued by the program

The display generated by DISPLAY PROGRAMS has the format shown in figure 8-3.

- DISPLAY name

displays information about the status of the specified input queue, output queue, source, destination, broadcast list, or invitation list. When an input queue is specified, a DISPLAY INPUT display is generated. When an output queue is specified, a DISPLAY OUTPUT display is generated. When a source, destination, or list is specified, a DISPLAY TERMINALS display is generated.

Any of the preceding DISPLAY commands can be used to write information to a monitor file. The command to write to a monitor file has the form:

DISPLAY name TO MONITOR-FILE

where name is INPUT, OUTPUT, ALL, TERMINALS, PROGRAMS, a queue name, source name, destination name, broadcast list, or invitation list. This command writes the specified display to the monitor file. The display has the same format as that described in the preceding paragraphs. The syntax for defining a monitor file is described in section 4.

An example of a DISPLAY command is illustrated in figure 8-4.

DUMP COMMAND

The DUMP command dumps application status information to the application dump file (described in section 5). The information contained in the application dump file can be used for error analysis. The DUMP command has the form:

DUMP

The DUMP command provides the AOP with the capability of producing a snapshot of application activity to be used for tracing MCS errors.

ENABLE COMMAND

The ENABLE command reestablishes the logical connection between MCS and a source, destination, list, or queue after it has been broken by a DISABLE command. When an input queue or source is enabled, MCS accepts messages from the queue or source. When an output

TERMINALS DISPLAY hh.mm.ss.

SOURCE	DESTINATION	NAM	USER	MODE	STATE	CONN
ssssssss	dddddddd	ttt	uuu	mmm	ffff	cccc
ssssssss	dddddddd	ttt	uuu	mmm	ffff	cccc
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
ssssssss	dddddddd	ttt	uuu	mmm	ffff	cccc

hh.mm.ss	Time (hour.minute.second) command was executed
ssssssss	Symbolic source name
dddddddd	Symbolic destination name
ttt	Network terminal name assigned by ALIAS clause in application definition if dedicated, or network terminal name in local configuration file if connected to MCS; blank if terminal not connected to this application
uuu	User name associated with terminal; blank if no user name
mmm	Terminal mode; one of the following:
	COMD command mode
	DATA data mode
ffff	Terminal status; one of the following:
	ENABLED
	DISABLED
cccc	Connection status; one of the following:
	CONN terminal is connected to this application
	DISC terminal is not connected to this application

Figure 8-2. Terminal Status Display Format

queue or destination is enabled, MCS delivers messages to the queue or destination. The ENABLE command has the following forms:

- ENABLE

  enables the AOP terminal. When the AOP terminal is enabled, it can send and receive application data messages.

- ENABLE INPUT

  enables all of the MCS application's input queues.

- ENABLE OUTPUT

  enables all of the MCS application's output queues.

- ENABLE ALL

  enables all of the MCS application's input and output queues.

- ENABLE name

  enables the specified source, destination, input queue, output queue, invitation list, or broadcast list.

IDLE COMMAND

The IDLE command causes the MCS application to cease processing. The application remains active, but no processing occurs. This command has the form:

IDLE

PROGRAMS DISPLAY hh.mm.ss.

PROGRAM	STATE	INVOKE	SEND	RECEIVE
ppppppp	sss	ii	mm	nn
ppppppp	sss	ii	mm	nn
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
ppppppp	sss	ii	mm	nn

hh.mm.ss	Time (hour.minute.second) command was executed
ppppppp	Program name
sss	Program status; one of the following:
	RUNNING
	SUSPENDED
	STOPPING
	TERMINATED
	INACTIVE
ii	Number of times program has been invoked
mm	Number of SEND requests issued by program
nn	Number of RECEIVE requests issued by program

Figure 8-3. COBOL Program Status Display Format

```
?display terminals
TERMINALS DISPLAY   07.48.57.
  SOURCE      DESTINATION      NAM       USER     MODE   STATE     CONN
TERM1          TERM1          TM101A   MCS0463   COMD   ENABLED   CONN
               TERM2                             COMD   DISABLED  DISC
TERM3                                            DATA   DISABLED  DISC
TERM4          TERM4                             COMD   ENABLED   DISC
?
```

Figure 8-4. Terminal Status Display Example

When an IDLE command is entered, the following message is sent to all terminals connected to the MCS application:

APPLICATION IDLE

When an application is idle, only the AOP terminal is polled by MCS. All commands, data, requests from non-AOP terminals, and COBOL Communication Facility requests are rejected. All output to the terminals connected to the application is halted until the application is resumed by the AOP.

INVOKE COMMAND

The INVOKE command initiates execution of a specified program. This command has the form:

INVOKE pname

where pname is the name of a COBOL program. The program must be defined in the application definition along with a job invocation file that is submitted to the operating system when the program is invoked. The job invocation file contains appropriate control statements for executing the program. Refer to section 5 for a description of this file.

MESSAGE COMMAND

The MESSAGE command sends a message to a terminal or group of terminals. This command has the following forms:

- MESSAGE "string"

  sends the specified character string to all terminals connected to the application.

- MESSAGE "string" TO sname

  sends the specified character string to the terminal or list of terminals identified by sname; sname must be a symbolic source name, destination name, interactive name, invitation list, or broadcast list defined in the application definition.

The maximum length of the message string is 80 characters. An example of this command is:

```
From terminal T11:   HOW ARE YOU?
                     ?message "i am fine" to t11
```

The MESSAGE command can send messages to terminals that have been disabled.

PURGE COMMAND

The PURGE command deletes all partial messages (messages that have been partially transmitted) from a specified destination or broadcast list. Complete messages (as indicated by an end-of-message indicator or end-of-group indicator) and message segments are not affected. This command has the form:

```
PURGE sname
```

where sname is a symbolic destination or interactive name defined in the application definition.

REROUTE COMMAND

The REROUTE command reroutes the output for a destination to another destination. The form of this command is:

```
REROUTE sname1 TO sname2
```

where sname1 and sname2 are destination names or interactive names; sname2 can be a broadcast list. After a REROUTE is entered, all output for the destination identified by sname1 is delivered to the destination identified by sname2. Destination sname2 must be connected to MCS. When sname1 is the same as sname2, the rerouting is cleared. This command can be used as follows:

```
REROUTE T11 TO T12
```

reroutes output destined for terminal T11 to terminal T12.

```
REROUTE T11 TO T11
```

clears the rerouting; output is again sent to terminal T11.

RESUME COMMAND

The RESUME command allows an MCS application to resume processing after an idle. This command has the form:

```
RESUME
```

When a RESUME is issued, the following message is sent to all terminals connected to the application:

```
APPLICATION RESUMED
```

All terminals connected to the application are again prompted by MCS for input, and output to the terminals is resumed.

RETRIEVE COMMAND

The RETRIEVE command retrieves and displays a message or group of messages from a specified queue. The retrieved messages remain in the queue, and the queue is in no way altered. The transition is not journaled. This command has the form:

```
RETRIEVE qname number
```

where qname is a queue name defined in the application definition, and number is the number of messages to be displayed. An example of this command is as follows:

```
?retrieve Q25 5
THIS IS THE FIRST MESSAGE IN QUEUE Q25
THIS IS THE SECOND MESSAGE IN QUEUE Q25
THIS IS THE THIRD MESSAGE IN QUEUE Q25
THIS IS THE FOURTH MESSAGE IN QUEUE Q25
THIS IS THE FIFTH MESSAGE IN QUEUE Q25
```

If a number greater than the number of messages in a queue is entered, all messages in the queue are retrieved.

Messages are retrieved from the subqueues of a compound queue in the default search order. Thus, the messages might not be displayed in the order they were entered. For example, assume a compound queue, Q, has two level 1 subqueues, SQ11 and SQ22. If there are three messages in both SQ11 and SQ22, then the command:

```
RETRIEVE Q 5
```

retrieves three messages from SQ11 and two messages from SQ22, regardless of the order in which the messages were entered.

REVOKE COMMAND

The REVOKE command revokes (stops execution of) a specified program. This command has the form:

```
REVOKE name
```

where name is the name of a COBOL program executing under MCS. When a REVOKE command is issued, execution of the specified program terminates, not immediately, but when the next COBOL Communication Facility statement in the program is executed. If the program is waiting to receive data, it is immediately aborted.

An EXIT statement can be included in the job invocation file to specify processing to occur after the REVOKE. Refer to the NOS reference manual for a description of the EXIT statement.

SHUTDOWN COMMAND

The SHUTDOWN command logs out all terminals connected to the MCS application, including the AOP terminal, and takes the application off-line. This command has the form:

```
SHUTDOWN
```

When a SHUTDOWN command is issued, the following message is sent to all terminals connected to the current application:

```
APPLICATION SHUTDOWN
```

A SHUTDOWN command causes the following events to occur for the application:

- All terminals connected to the application are disconnected.
- All MCS operations in progress are completed.
- Central memory queues are released.
- Attached files are returned.
- Empty queue files are purged.
- Application table space is released.
- Further application COBOL program requests are rejected.

When all application activity has ceased, application execution is terminated. The following example illustrates this command:

```
?message "***shutdown in 5 min.***"
    .
    .
    .
?shutdown
APPLICATION SHUTDOWN
```

To connect a given terminal to an application that has been shut down, the AOP or system operator must reactivate the application, and the terminal user must repeat the login dialog.

This section describes the procedure files used to initiate MCS, and the system console commands available to the system operator for controlling MCS operation.

INITIATION PROCEDURE FILE

MCS is initiated by a procedure file invoked by the system operator. The procedure file contains control statements specifying various parameters necessary to initiate MCS. This file can be created by the system operator or site analyst.

Information required by the operating system to initiate MCS includes a user name, family name, and password; the name of the application definition library file containing the applications to be executed; and the mode of MCS operation (test mode or normal mode). Also required in the procedure file are a header statement, an RFL statement, and an MCS call statement. Certain of these statements are optional; when they are omitted, the system assumes default values for parameters specified on these statements. If errors occur during procedure file processing, a message is displayed on the system console and is written to the MCS dayfile. A complete list of MCS error messages is included in appendix B.

The following paragraphs describe the procedure file control statements, and default values where applicable. Refer to the NOS reference manual for more information on procedure files.

HEADER STATEMENT

The header statement specifies the name used in the procedure file call. This statement has the form:

```
MCS ffff
```

where ffff is one through four nonblank characters. This statement is required, and must be the first statement in the procedure file.

USER STATEMENT

The USER statement must specify a valid user name, password, and family name. This statement has the form:

```
USER(,usernam,passwrd,familyname)
```

The USER statement can be omitted; in which case, the default user name SYSTEMX is used.

NOTE

In order to execute the procedure file at a system control point, the user name must have system origin access word privileges. Refer to the NOS Maintenance reference manual for more information on the system origin access word.

RFL STATEMENT

The RFL statement specifies the initial field length to be occupied by MCS. This statement has the form:

```
RFL(n)
```

where n must be equal to or greater than 30 000. This statement is required in the initiation procedure file.

ONSW STATEMENT

The ONSW statement initiates MCS in test mode. This statement has the form:

```
ONSW(1)
```

This statement is optional. When it is present, all applications in the attached library execute in test mode. (In test mode, MCS executes independently of the network. Refer to section 5 for information on test mode operation.) When the ONSW statement is omitted, MCS communicates with the network in a normal manner.

ATTACH OR GET STATEMENT

The ATTACH or GET statement attaches the application definition library file containing the MCS applications to be executed. These statements have the forms:

```
ATTACH(ADLLIB=pfn/param-list)
GET(ADLLIB=pfn/param-list)
```

where pfn is the permanent file name of the application definition library file to be initiated, and param-list specifies additional file information. Refer to the NOS reference manual for more information on these statements.

The ATTACH or GET statement is optional. When it is omitted, the system searches for a local file named ADLLIB; when this file is not found locally, the system attempts to attach the file. If file ADLLIB cannot be found, an error message is displayed on the system console and the operator can then enter a library name as described under System Console Commands.

MCS CALL STATEMENT

The MCS call statement initiates MCS. This statement has the forms:

```
MCS.
MCS(GO)
```

When the form MCS(GO) is specified, MCS is initiated without operator interaction. When the form MCS. is specified, MCS pauses before initiation. The system operator can then enter commands to change certain parameters in the procedure file before resuming execution of MCS. These commands are described in this section under System Console Commands.

SYSTEM-SUPPLIED PROCEDURE FILE

The default procedure file is established when MCS is installed. A suggested version is shown in figure 9-1. The last two statements in this example provide for reprieve processing in the event of an MCS abort; the NAM trace file ZZZZZDN is formatted and printed on file OUTPUT by the DLFP statement, as described in the NAM reference manual.

```
'MCS
 RFL(30000)
 MCS(GO)
 EXIT.
 DLFP(I=0)
```

Figure 9-1. System-Default Procedure File

PROCEDURE FILE CALL

The MCS procedure file is called by a console command of the form:

n.MCS ffff .

where MCSffff is the MCS procedure file name. When the characters ffff are omitted, the system default file is used. When this command is entered, the operating system searches first for the procedure file in the system library, then for an indirect access file under user name SYSTEMX. When the file is found, the procedure is started at control point n with subsystem privileges.

PROCEDURE FILE EXAMPLE

Figure 9-2 illustrates an example of an MCS initiation procedure file.

```
MCSTEST
USER(AAA789,,SYS172)
RFL(30000)
ONSW(1)
ATTACH(ADLLIB=APPLS/UN=MCS123)
MCS.
```

Figure 9-2. Procedure File Example

In this example, MCS is initiated in test mode. The name of the application definition library is APPLS, under user name MSC123. All files created by MCS have a default user name of AAA789 and reside on family SYS172. Since the GO parameter is omitted from the MCS call, execution of MCS is suspended before initiation, allowing the system operator to change parameter values. This procedure file is called by the command:

n.MCSTEST.

SYSTEM CONSOLE COMMANDS

MCS provides a set of system console commands which allows the system operator to:

- Initiate MCS (MCS command)
- Change parameters specified in the initiation procedure file (CFO.ADL, ONSW, OFFSW commands)
- Resume execution of MCS (CFO.GO command)
- Initiate an inactive MCS application (CFO.START command)
- Idle MCS (CFO.IDLE command)
- Shut down MCS (CFO.DISABLE command)

These commands are described in the following paragraphs. The parameter n. prefixing each command is the number of the control point where MCS is initiated.

If an invalid command is entered, an error message is displayed on the system console. Refer to appendix B for a complete list of MCS error messages.

MCS

The MCS command initiates execution of the specified procedure file. This command has the form:

n.MCS ffff .

where MCSffff is the name appearing on the header statement of the initiation procedure file. The procedure file is executed at control point n. When the characters ffff are omitted, the system-supplied default file is used.

CFO.ADL

The CFO.ADL command specifies the name of the application definition library to be initiated. This command has the form:

n.CFO.ADL,pfn,un,pw.

where pfn is the permanent file name of the application definition library, un is the user name, and pw is the password. The file must be public. If un or pw are omitted, the system default user name and password are assumed.

The CFO.ADL command can be entered only when MCS execution is suspended prior to initiation, as described in the preceding subsection. The parameters specified on this command override corresponding parameters on an ATTACH or GET statement included in the procedure file.

ONSW1

The ONSW1 command sets sense switch 1 so that MCS executes in test mode. This command has the form:

n.ONSW1.

The ONSW1 command can be entered only when execution of MCS is suspended prior to initiation.

OFFSW1

The OFFSW1 command turns off sense switch 1. This command has the form:

n.OFFSW1.

The OFFSW1 command has effect only when MCS is suspended prior to initiation. When an ONSW statement has been included in the procedure file, the OFFSW1 command overrides this statement; when MCS is resumed, it is initiated in normal mode.

CFO.GO

The CFO.GO command resumes execution of MCS. This command has the form:

n.CFO.GO.

The CFO.ADL, ONSW1, and OFFSW1 commands are ignored if entered after CFO.GO.

CFO.START

The CFO.START command initiates an inactive MCS application. This command has the form:

n.CFO.START,appl ,TEST

where appl is the application name as specified in the application definition, and TEST causes the application to execute in test mode. The specified application must reside in an application definition library that has been initiated as described in the preceding subsection.

CFO.IDLE

The CFO.IDLE performs a graceful shutdown of MCS. This command has the form:

n.CFO.IDLE.

All applications in the active library are shut down, and connected terminals are logged out of MCS and the network. COBOL programs executing in the MCS environment are allowed to complete an MCS request in progress, and can continue to execute non-MCS logic, but all subsequent MCS requests are rejected. MCS performs a NETOFF, and the message:

LOGGED OUT

is displayed at connected terminals.

CFO.DISABLE

The CFO.DISABLE command immediately drops MCS from the control point. This command has the form:

n.CFO.DISABLE.

All connected terminals are logged out, and executing COBOL programs are immediately aborted. The message:

APPLICATION FAILED

is displayed at connected terminals.

NOTE

Since the DISABLE command causes an immediate drop, the IDLE command is recommended for normal MCS shutdowns.

STANDARD CHARACTER SETS A

OPERATING SYSTEM CHARACTER SETS

Control Data operating systems offer the following variations of a basic character set:

CDC 63-character set
CDC 64-character set
ASCII 63-character set
ASCII 64-character set

The set in use at a particular installation is specified when the operating system is installed or deadstarted.

Depending on another installation option, the system assumes an input deck has been punched in either 026 or 029 mode (regardless of the character set in use).

The alternate mode can be specified by a 26 or 29 punched in columns 79 and 80 of any 6/7/9 card or 7/8/9 card. The specified mode remains in effect through the end of the job unless it is reset by specification of the alternate mode on a subsequent 7/8/9 card or 6/7/9 card. In addition, 026 mode can be specified by a card with 5/7/9 multipunched in column 1, and 029 mode can be specified by a card with 5/7/9 multipunched in column 1 and a 9 punched in column 2.

Graphic character representation appearing at a terminal or printer depends on the installation character set and the terminal type. Characters shown in the CDC Graphic column of table A-1 are applicable to BCD terminals; ASCII graphic characters are applicable to ASCII-CRT and ASCII-TTY terminals.

128-CHARACTER ASCII SET

Table A-4 contains the 128-character ASCII set supported by the Network Access Method (NAM). A 96-character subset consists of the rightmost six columns; a 64-character subset consists of the middle four columns. Note that display code equivalents exist for the characters in this 64-character subset only.

TABLE A-1. APPLICATION DEFINITION LANGUAGE (ADL) AND STANDARD CHARACTER SETS

ADL	Display Code (octal)	CDC			ASCII		
		Graphic	Hollerith Punch (026)	External BCD Code	Graphic Subset	Punch (029)	Code (octal)
	00†	: (colon)††	8-2	00	: (colon) ††	8-2	072
A	01	A	12-1	61	A	12-1	101
B	02	B	12-2	62	B	12-2	102
C	03	C	12-3	63	C	12-3	103
D	04	D	12-4	64	D	12-4	104
E	05	E	12-5	65	E	12-5	105
F	06	F	12-6	66	F	12-6	106
G	07	G	12-7	67	G	12-7	107
H	10	H	12-8	70	H	12-8	110
I	11	I	12-9	71	I	12-9	111
J	12	J	11-1	41	J	11-1	112
K	13	K	11-2	42	K	11-2	113
L	14	L	11-3	43	L	11-3	114
M	15	M	11-4	44	M	11-4	115
N	16	N	11-5	45	N	11-5	116
O	17	O	11-6	46	O	11-6	117
P	20	P	11-7	47	P	11-7	120
Q	21	Q	11-8	50	Q	11-8	121
R	22	R	11-9	51	R	11-9	122
S	23	S	0-2	22	S	0-2	123
T	24	T	0-3	23	T	0-3	124
U	25	U	0-4	24	U	0-4	125
V	26	V	0-5	25	V	0-5	126
W	27	W	0-6	26	W	0-6	127
X	30	X	0-7	27	X	0-7	130
Y	31	Y	0-8	30	Y	0-8	131
Z	32	Z	0-9	31	Z	0-9	132
0	33	0	0	12	0	0	060
1	34	1	1	01	1	1	061
2	35	2	2	02	2	2	062
3	36	3	3	03	3	3	063
4	37	4	4	04	4	4	064
5	40	5	5	05	5	5	065
6	41	6	6	06	6	6	066
7	42	7	7	07	7	7	067
8	43	8	8	10	8	8	070
9	44	9	9	11	9	9	071
+	45	+	12	60	+	12-8-6	053
-	46	-	11	40	-	11	055
*	47	*	11-8-4	54	*	11-8-4	052
/	50	/	0-1	21	/	0-1	057
	51	(	0-8-4	34	(	12-8-5	050
	52	)	12-8-4	74	)	11-8-5	051
	53	$	11-8-3	53	$	11-8-3	044
	54	=	8-3	13	=	8-6	075
blank	55	blank	no punch	20	blank	no punch	040
	56	, (comma)	0-8-3	33	, (comma)	0-8-3	054
	57	. (period)	12-8-3	73	. (period)	12-8-3	056
" (quote)	60	≡	0-8-6	36	#	8-3	043
	61	[	8-7	17	[	12-8-2	133
	62	]	0-8-2	32	]	11-8-2	135
	63	% ††	8-6	16	% ††	0-8-4	045
	64	≠	8-4	14	" (quote)	8-7	042
	65	→	0-8-5	35	_ (underline)	0-8-5	137
	66	∨	11-0 or 11-8-2†††	52	!	12-8-7 or 11-0†††	041
	67	∧	0-8-7	37	&	12	046
	70	↑	11-8-5	55	' (apostrophe)	8-5	047
	71	↓	11-8-6	56	?	0-8-7	077
	72	<	12-0 or 12-8-2†††	72	<	12-8-4 or 12-0†††	074
	73	>	11-8-7	57	>	0-8-6	076
	74	≤	8-5	15	@	8-4	100
	75	≥	12-8-5	75	\	0-8-2	134
	76	¬	12-8-6	76	~ (circumflex)	11-8-7	136
	77	; (semicolon)	12-8-7	77	; (semicolon)	11-8-6	073

†Twelve zero bits at the end of a 60-bit word in a zero byte record are an end of record mark rather than two colons.
††In installations using a 63-graphic set, display code 00 has no associated graphic or card code; display code 63 is the colon (8-2 punch).
The % graphic and related card codes do not exist and translations yield a blank (55_8).
†††The alternate Hollerith (026) and ASCII (029) punches are accepted for input only.

TABLE A-2. CDC CHARACTER SET COLLATING SEQUENCE

Collating Sequence Decimal	Collating Sequence Octal	CDC Graphic	Display Code	External BCD	Collating Sequence Decimal	Collating Sequence Octal	CDC Graphic	Display Code	External BCD
00	00	blank	55	20	32	40	H	10	70
01	01	≤	74	15	33	41	I	11	71
02	02	%	63 †	16 †	34	42	∨	66	52
03	03	[	61	17	35	43	J	12	41
04	04	→	65	35	36	44	K	13	42
05	05	≡	60	36	37	45	L	14	43
06	06	∧	67	37	38	46	M	15	44
07	07	↑	70	55	39	47	N	16	45
08	10	↓	71	56	40	50	O	17	46
09	11	>	73	57	41	51	P	20	47
10	12	≥	75	75	42	52	Q	21	50
11	13	¬	76	76	43	53	R	22	51
12	14	.	57	73	44	54	]	62	32
13	15	)	52	74	45	55	S	23	22
14	16	;	77	77	46	56	T	24	23
15	17	+	45	60	47	57	U	25	24
16	20	$	53	53	48	60	V	26	25
17	21	*	47	54	49	61	W	27	26
18	22	-	46	40	50	62	X	30	27
19	23	/	50	21	51	63	Y	31	30
20	24	,	56	33	52	64	Z	32	31
21	25	(	51	34	53	65	:	00 †	none†
22	26	=	54	13	54	66	0	33	12
23	27	≠	64	14	55	67	1	34	01
24	30	<	72	72	56	70	2	35	02
25	31	A	01	61	57	71	3	36	03
26	32	B	02	62	58	72	4	37	04
27	33	C	03	63	59	73	5	40	05
28	34	D	04	64	60	74	6	41	06
29	35	E	05	65	61	75	7	42	07
30	36	F	06	66	62	76	8	43	10
31	37	G	07	67	63	77	9	44	11

†In installations using the 63-graphic set, the % graphic does not exist. The : graphic is display code 63, External BCD code 16.

TABLE A-3. ASCII CHARACTER SET COLLATING SEQUENCE

Collating Sequence Decimal	Collating Sequence Octal	ASCII Graphic Subset	Display Code	ASCII Code	Collating Sequence Decimal	Collating Sequence Octal	ASCII Graphic Subset	Display Code	ASCII Code
00	00	blank	55	20	32	40	@	74	40
01	01	!	66	21	33	41	A	01	41
02	02	"	64	22	34	42	B	02	42
03	03	#	60	23	35	43	C	03	43
04	04	$	53	24	36	44	D	04	44
05	05	%	63†	25	37	45	E	05	45
06	06	&	67	26	38	46	F	06	46
07	07	'	70	27	39	47	G	07	47
08	10	(	51	28	40	50	H	10	48
09	11	)	52	29	41	51	I	11	49
10	12	*	47	2A	42	52	J	12	4A
11	13	+	45	2B	43	53	K	13	4B
12	14	,	56	2C	44	54	L	14	4C
13	15	-	46	2D	45	55	M	15	4D
14	16	.	57	2E	46	56	N	16	4E
15	17	/	50	2F	47	57	O	17	4F
16	20	0	33	30	48	60	P	20	50
17	21	1	34	31	49	61	Q	21	51
18	22	2	35	32	50	62	R	22	52
19	23	3	36	33	51	63	S	23	53
20	24	4	37	34	52	64	T	24	54
21	25	5	40	35	53	65	U	25	55
22	26	6	41	36	54	66	V	26	56
23	27	7	42	37	55	67	W	27	57
24	30	8	43	38	56	70	X	30	58
25	31	9	44	39	57	71	Y	31	59
26	32	:	00†	3A	58	72	Z	32	5A
27	33	;	77	3B	59	73	[	61	5B
28	34	<	72	3C	60	74	\	75	5C
29	35	=	54	3D	61	75	]	62	5D
30	36	>	73	3E	62	76	^	76	5E
31	37	?	71	3F	63	77	_	65	5F

†In installations using a 63-graphic set, the % graphic does not exist. The : graphic is display code 63.

TABLE A-4. FULL ASCII CHARACTER SET

128-Character Set

96-Character Subset

64-Character Subset

b_7			0	0	0	0	1	1	1	1
b_6			0	0	1	1	0	0	1	1
b_5			0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1	ROW / COLUMN →	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	¦
1	1	0	1	D	CR	GS	-	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	^	n	~
1	1	1	1	F	SI	US	/	?	O	_	o	DEL

MCS issues diagnostic messages indicating errors occurring during application definition compilation and during execution of an application.

ADLP ERROR MESSAGES

Error messages issued by ADLP are listed in table B-1. These messages inform the application developer of errors occurring during compilation of an application definition. ADLP error messages are written to the output listing file specified by the L parameter on the ADLP control statement.

Each error message is preceded by a number indicating the source line in which the error occurred. The message text is preceded by a 3-digit number and a 1-letter severity level indicator. The error number can be used to locate the error message in the table. The severity level indicator is either T, signifying a trivial error, or F, signifying a fatal error. If fatal errors occur during compilation, no application definition tables are produced. Trivial errors result in informative error messages; if only trivial errors occur, compilation completes successfully and valid tables are produced.

ADLP is a two-pass processor. The first pass checks for syntactical errors; the second pass checks for errors in logic and consistency of usage. The numbers of first and second pass errors detected during compilation are printed in an error summary that appears at the end of the source listing. If fatal errors occur on the first pass, no second pass scanning is performed. Certain fatal errors result in immediate termination of compilation, while others allow compilation to continue. If fatal errors occur, no application definition tables are produced.

After compilation has completed, ADLP writes a compilation summary consisting of one or more of the messages listed in table B-2. These messages appear on the output listing file immediately after the source listing.

ADLP DAYFILE MESSAGES

The messages listed in table B-3 appear in the job dayfile produced by an ADLP run. ADLP generates these messages when a control statement error or an internal error causes compilation to terminate prematurely.

EXECUTION ERROR MESSAGES

Error messages issued during the login procedure and during execution of an MCS application appear at the terminal. These messages are listed alphabetically in table B-4. Certain of the execution messages are issued by a network software component other than MCS. The issuing routine is indicated in the table. The following abbreviations are used:

NVF	Network Validation Facility
CCP	Communications Control Program
TIP	Terminal Interface Program

Refer to the Network Access Method reference manual for more information on these programs.

SYSTEM CONSOLE AND DAYFILE MESSAGES

The messages listed in table B-5 appear on the system operator's console and on the system dayfile. The messages issued by MCS are generated during startup or shutdown, or as a result of an invalid operator command. The messages issued by ADLP are generated as a result of errors on the ADLP control statement or errors occurring during ADL compilation.

TABLE B-1. ADLP ERROR MESSAGES

Message	Significance	Action
000 F ERROR LIMIT EXCEEDED	Too many errors occurred so compilation was aborted.	Correct as many errors as possible and rerun.
001 F EMPTY SOURCE FILE	The file input to ADLP contained no information.	The input file is bad. Create a new input source file.
002 F SOURCE WORD LONGER THAN 80 CHARACTERS	Names used in the application definition must not exceed 80 characters.	Replace the name with a shorter name. Note that different types of names have different maximum lengths.
003 F APPLICATION GLOBAL DIVISION MISSING	An Application Global Division must be included in an application definition.	Check for missing header. Ensure that the necessary Application Global Division statements are present. Ensure that the Application Global Division appears in the proper sequence.

TABLE B-1. ADLP ERROR MESSAGES (Contd)

Message	Significance	Action
004 F APPLICATION-NAME PARAGRAPH MISSING	The Application-Name paragraph must be included in the Application Global Division.	Ensure that a correct Application-Name paragraph is present and that it appears in the correct sequence.
005 F INVALID CHARACTERS IN FILE NAME	A file name can contain only the characters A through Z, 0 through 9.	Replace the file name with a valid name.
006 F INVALID CHARACTERS IN DATA NAME	The name of the data item can contain only the characters A through Z, 0 through 9, and - . The last character must not be - .	Replace the data name with a valid name.
007 F INVALID CHARACTERS IN SYMBOLIC NAME	The symbolic name of a source or destination can contain only the characters A through Z, 0 through 9, and - . The last character must not be - .	Replace the symbolic name with a valid name.
008 F INVALID CHARACTERS IN ROUTINE NAME	The name of an external routine can contain only the characters A through Z and 0 through 9.	Replace the routine name with a valid name.
009 F INVALID CHARACTERS IN CONDITION NAME	Names defined in CONDITION clauses can contain only the characters A through Z, 0 through 9, and - . The last character must not be - .	Replace the condition name with a valid name.
010 F INVALID CHARACTERS IN QUEUE NAME	Names assigned to queues can contain only the characters A through Z, 0 through 9, and - . The last character must not be - .	Replace the queue name with a valid name.
011 F USER-DEFINED NAME TOO LONG FOR ITS TYPE	A name supplied by the user contains more than the allowable number of characters. The limit is different for different types of names.	Refer to section 4 for maximum length and shorten name.
012 F INVALID NON-NUMERIC LITERAL	A non-numeric literal was not supplied where one was required.	Supply a valid non-numeric literal.
013 F NON-NUMERIC LITERAL TOO LONG	A non-numeric literal exceeds 80 characters.	Shorten the non-numeric literal.
014 F INITIATION MUST SPECIFY AUTOMATIC OR EXPLICIT	Required syntax is INITIATION IS AUTOMATIC or INITIATION IS EXPLICIT.	Supply the correct INITIATION paragraph.
015 F TEST MAY ONLY BE USED WITH AUTOMATIC	In the INITIATION paragraph, the TEST phrase can be specified only with the AUTOMATIC phrase.	Correct the INITIATION paragraph.
016 F APPLICATION PROGRAM DIVISION MISSING	An application definition requires an Application Program Division, even if no application programs are included.	Supply Application Program Division. Check for missing header or header in incorrect sequence.
017 F SEARCHING FOR NEXT AREA A STATEMENT	ADLP expected a division or section header or a paragraph.	Check for missing statements or area A statements that do not begin in column 8 through 11.

TABLE B-1. ADLP ERROR MESSAGES (Contd)

Message	Significance	Action
018 F UNEXPECTED EOF ON SOURCE FILE	ADLP encountered an end-of-file indicator while reading the input source file.	The file is probably bad. It might be necessary to recreate the source file.
019 F DIVISION, SECTION, PARA BEGIN COLS 8-11	A division header, section header, or paragraph, was encountered that did not begin in columns 8 through 11.	Ensure that all division headers, section headers, and paragraphs begin in columns 8 through 11.
020 F DUPLICATE USER-DEFINED NAME	Each user-supplied name in the application definition must be unique.	Check the cross reference listing for assistance in locating the duplicate name. Change to a unique name.
021 F OPERATOR CLAUSE REQUIRES SIGNATURE OR PASSWORD	If the PASSWORD clause is omitted from the OPERATOR paragraph of the Application Global Division, a SIGNATURE paragraph must be supplied.	Include the PASSWORD clause in the OPERATOR paragraph or supply a SIGNATURE paragraph.
022 F APPLICATION DATA DIVISION HEADER MISSING	An Application Data Division header statement is required and it must follow the last statement of the Application Program Division.	Supply the missing header and ensure that the Application Data Division is in the proper sequence.
023 F SERIAL NUMBER REQUIRES SUPPLIED OR GENERATED	A SERIAL NUMBER clause must contain a SUPPLIED or a GENERATED phrase.	Supply the missing phrase.
024 F SUPPLIED SERIAL NUMBER REQUIRES IN PHRASE	The correct form is SERIAL-NUMBER SUPPLIED IN data-name.	Correct the SERIAL-NUMBER clause.
025 F REQUIRED KEYWORD MISSING	ADLP expected a keyword.	Determine missing keyword from context and supply.
026 F INVALID INTEGER LITERAL	An integer literal consists of digits 0 through 9.	Supply a correct integer literal.
027 F MAXIMUM INTEGER VALUE IS 32767	An integer literal exceeded the maximum allowable value.	Supply a smaller integer literal.
028 F STARTS PHRASE REQUIRES AT, WITH OR AFTER	The phrase FIELD IS data-name STARTS must be followed by AT, WITH, or AFTER.	Supply the missing syntax.
029 F STARTING CHARACTER BEYOND SEGMENT	The first character of a field is defined to occur beyond the end of a segment.	Check LENGTH clause in SEGMENT paragraph and STARTS phrase in associated FIELD clause. The field must begin within the segment.
030 F FIELD LENGTH BEYOND SEGMENT	A field within a segment is defined to extend beyond the end of the segment.	Check LENGTH clause in SEGMENT paragraph and EXTENDS phrase in associated FIELD clause. The field must be contained within the segment.
031 F EXTENDS REQUIRES FOR, TO OR THROUGH	The EXTENDS phrase of a FIELD clause must specify FOR, TO, or THROUGH	Supply the correct syntax.
032 F LITERAL MUST BE INTEGER OR NONNUMERIC	The specified literal is invalid or is of wrong type.	Supply a valid integer or nonnumeric literal.

TABLE B-1. ADLP ERROR MESSAGES (Contd)

Message	Significance	Action
033 F SOURCE-DESTINATION DIVISION HEADER MISSING	All application definitions require a Source-Destination Division header statement. The Source-Destination Division must follow the Application-Data Division and precede the Queue Division.	Supply the header statement.
034 F INVALID SYMBOLIC NAME TYPE	A symbolic name is limited to 12 characters selected from A through Z, 0 through 9, and - . The last character must not be a - .	Supply a valid symbolic name.
035 F QUEUE DIVISION HEADER MISSING	All application definitions require a Queue Division header statement. The Queue Division must follow the Source-Destination Division and precede the Application Processing Division.	Supply the header statement.
036 F STARTS/EXTENDS INSTANCE RANGE ERROR	STARTS INSTANCE must be less than EXTENDS INSTANCE.	Supply correct values.
037 F SELECT MUST SPECIFY INPUT, OUTPUT OR SUBQUEUES	A SELECT clause must include an INPUT-QUEUES, OUTPUT-QUEUES, or SUB-QUEUES phrase.	Supply the correct phrase.
038 F BASED REQUIRES SOURCE/ DESTINATION/CONTENTS/TIME	The BASED phrase in a SELECT paragraph must include a SOURCE, DESTINATION, CONTENTS, or TIME phrase.	Supply the correct phrase.
039 F SELECT REQUIRES ROUTE CLAUSE	A SELECT paragraph must include at least one ROUTE clause.	Supply the correct clause.
040 F INVALID TIME LITERAL	The syntax requires a time literal. Correct form is "hh.mm.ss".	Supply a valid time literal.
041 F MEDIUM REQUIRES CENTRAL OR DISK	A MEDIUM clause in a SELECT paragraph must include a CENTRAL or DISK phrase.	Supply the correct phrase.
042 F STATUS REQUIRES ENABLED OR DISABLED	A STATUS clause in a SELECT paragraph must include an ENABLED or DISABLED phrase.	Supply the correct phrase.
043 F APPLICATION PROCESSING DIVISION HEADER MISSING	All application definitions require an Application Processing Division header statement. This statement must follow the Queue Division.	Supply the header statement.
044 F USE REQUIRES WHEN OR EVERY	A USE statement must specify a WHEN or EVERY phrase.	Supply the correct phrase.
045 F SELECT PARAGRAPH MISSING	SELECT INPUT QUEUES or SELECT INPUT SUB-QUEUES must be specified if input queues are defined, SELECT OUTPUT QUEUES must be specified if output queues are defined.	Supply the correct SELECT paragraph.
046 F INVALID NAME TYPE	The user-supplied name is invalid in this context.	Refer to section 4 to determine correct name type.

TABLE B-1. ADLP ERROR MESSAGES (Contd)

Message	Significance	Action
047 F INITIATION MAY BE SPECIFIED ONLY ONCE	An application definition cannot contain more than one INITIATION paragraph.	Check for multiple occurrences of SELECT paragraph and remove all but one.
048 F INVALID SYMBOLIC NAME OR TYPE	ADLP expected a symbolic source or destination name. Symbolic names are restricted to 12 characters selected from A through Z, 0 through 9, and - . The last character must not be - .	Supply a valid symbolic name.
049 F INVALID QUEUE NAME OR TYPE	ADLP expected a queue name. Queue names are limited to 12 characters selected from A through Z, 0 through 9, and - . The last character must not be - .	Supply a valid queue name.
050 F SIZE MAY BE SPECIFIED ONLY ONCE PER QUEUE	In a USE WHEN or USE EVERY statement in the Application Processing Division, a SIZE OF queue-name EXCEEDS integer MESSAGES phrase cannot be specified more than once.	Check for multiple occurrences of SIZE phrase in a USE WHEN or USE EVERY statement.
051 F INVALID NAME IN COMMAND	An ADL verb specified a name that was of the wrong type for that verb.	Refer to section 4 to determine valid name types for the verb and supply a correct name.
052 F ROUTING ALREADY SPECIFIED FOR QUEUE NAME	The same queue name was specified in more than one ROUTE phrase in different SELECT paragraphs.	Specify correct routing.
053 F CONTENTS REQUIRES FIELD NAME	The CONTENTS phrase of a SELECT paragraph must specify the name of a data field. The specified field must be defined in the Application Data Division.	Supply a valid field name.
054 F QUEUE IN ROUTE IS NOT SUBQUEUE OF SELECT	The queue specified in the ROUTE clause must be defined as a sub-queue of the queue specified in the associated SELECT paragraph.	Define the correct queues and sub-queues in the INPUT section.
055 F FROM USED ONLY WITH SOURCE CRITERION	In the phrase FROM name, name must be defined as a source in SYMBOLIC-NAME paragraph of the Source-Destination Division.	Either supply a defined source name in the FROM phrase, or define the specified name in the Source-Destination Division.
056 F TO USED ONLY WITH DESTINATION CRITERION	In a TO phrase, the specified name must be defined as a destination in a SYMBOLIC-NAME paragraph of the Source-Destination Division.	Either specify a defined destination name in the TO phrase, or define the specified name as a destination.
057 F BEFORE-AFTER USED ONLY WITH TIME CRITERION	The name specified in a BEFORE or AFTER phrase must be a valid time literal.	Supply a valid time literal.
058 F INVALID CONDITION NAME	The name specified in a FOR phrase must be defined in a CONDITION clause in a MESSAGE paragraph of the Application Data Division.	Specify a valid condition name.

TABLE B-1. ADLP ERROR MESSAGES (Contd)

Message	Significance	Action
059 F FOR USED ONLY WITH CONTENTS CRITERION	A ROUTE FOR clause can be specified only if the BASED ON CONTENTS phrase was used in the SELECT paragraph.	Either eliminate the FOR clause or specify BASED ON CONTENTS in the SELECT paragraph.
060 F CONDITION NAME NOT DEFINED IN BASED PHRASE	If a field is tested for a condition in the Routing Section of the Queue Division, the condition name and field name must be associated in a SEGMENT paragraph of the Application Data Division.	Supply correct condition name and field name in Application Data Division and Queue Division.
061 F SERIAL NUMBER DATA NAME INVALID	The IN phrase of a SERIAL-NUMBER clause must specify a valid data name. The data name must be defined as a message field in a SEGMENT paragraph.	Ensure that data name is a valid name as described in section 4. Ensure that the name is correctly defined in a SEGMENT paragraph.
062 F ON INPUT, ON OUTPUT LIMITED TO ONE EACH	In a JOURNAL clause in the QUEUE paragraph of the Queue Division, at most one ON INPUT and one ON OUTPUT phrase can appear.	Correct the JOURNAL clause.
063 F INJECTION/COLLECTION/RESPONSE QUEUE INVALID	An invalid name was used for an injection, collection, or response queue.	Check for usage conflict, e.g., a queue name being used in another context as a symbolic name.
064 F INVALID OPERATOR NAME	The name specified in the OPERATOR IS paragraph of the Application Global Division cannot be used as an operator name.	Check ALIAS clause in Source-Destination Division. Operator cannot be dedicated.
065 F RANGE VALUES MUST BE INTEGER	The names specified in the VALUE phrase of a CONDITION clause must be valid integers.	Refer to section 4 for integer format and supply valid integers.
066 F INVALID DATA NAME OR TYPE	A user-supplied name does not conform to the rules for data names, or a conflict in name usage has occurred.	Ensure that the data name is valid as described in section 4. Check for usage conflicts, e.g., a name defined as a queue name being used in a context where a data name is required.
067 F ALIAS REQUIRED FOR TYPE QUEUE OR JOURNAL	If the SYMBOLIC-NAME paragraph of the Source-Destination Division specifies a queue name or a journal name, an ALIAS clause must be included.	Supply an ALIAS clause.
068 F NO DESTINATIONS FOR OTHERWISE ROUTING	The OTHERWISE phrase in a SELECT paragraph will never be executed because all routing possibilities are covered by previous ROUTE clauses in the same SELECT paragraph.	Delete the OTHERWISE phrase.
069 F DUPLICATE OR INVALID USE OF TYPE	A user-supplied name was used in an invalid context.	Check cross reference listing and check all usages of the name. Ensure that the name is unique and that no conflicts occur, e.g., name used as data name and symbolic name.

TABLE B-1. ADLP ERROR MESSAGES (Contd)

Message	Significance	Action
070 F EXPLICIT INITIATION REQUIRES OPERATOR	If INITIATION IS EXPLICIT is used, the OPERATOR paragraph must be specified.	Specify the OPERATOR paragraph.
071 F INVALID LIST COMPONENT TYPE	Names specified in a BROADCAST-LIST or INVITATION-LIST paragraph must be symbolic names in SYMBOLIC-NAME paragraphs.	Ensure that the necessary SYMBOLIC-NAME paragraphs are present. Check for conflicts in name usages.
072 F DUPLICATE MESSAGES CLAUSE-SYMB NAME AND LIST	A MESSAGE clause appears in both a SYMBOLIC-NAME paragraph and in an INVITATION-LIST or BROADCAST-LIST paragraph which includes the symbolic name.	Delete one of the MESSAGE clauses.
073 F DUPLICATE MADE CLAUSE-SYMBOLIC NAME AND LIST	A MODE clause appears in both a SYMBOLIC-NAME paragraph and in an INVITATION-LIST or BROADCAST-LIST paragraph which includes the symbolic name.	Delete one of the MODE clauses.
074 F DUPLICATE PASSWORD CLAUSE-SYMB NAME AND LIST	A PASSWORD clause appears in both a SYMBOLIC-NAME paragraph and in an INVITATION-LIST or BROADCAST-LIST paragraph which includes the symbolic name.	Delete one of the PASSWORD clauses.
075 F DUPLICATE STATE CLAUSE-SYMBOLIC NAME AND LIST	A STATE clause appears in both a SYMBOLIC-NAME paragraph and in an INVITATION-LIST or BROADCAST-LIST paragraph which includes the symbolic name.	Delete one of the STATE clauses.
076 F OUTPUT BASED ON DESTINATION ONLY	The BASED ON DESTINATION phrase can be used only in a SELECT OUTPUT QUEUES paragraph.	Correct the SELECT paragraph.
077 F THROUGH VALUE < or = START VALUE	In the clause CONDITION IS VALUE n1 THROUGH n2, n2 must be greater than n1.	Supply correct values.
078 F MEDIUM DISK REQUIRES RESIDENCY	If a MEDIUM IS DISK clause is specified in a QUEUE paragraph of the Queue Division, the RESIDENCY clause must also be specified.	Supply a RESIDENCE clause.
079 F INVOKE REQUIRES INVOCATION FILE	If an INVOKE verb is used in the Application Processing Division, an invocation file must be defined for the invoked program. The invocation file name must be declared in the Application Program Division.	Refer to the discussion of Program Initiation in section 5.
080 F NO STATUS/MESSAGES FOR TYPE QUEUE OR JOURNAL	The STATUS and MESSAGE clauses cannot appear with a TYPE IS QUEUE or TYPE IS JOURNAL clause in a SYMBOLIC-NAME paragraph of the Source-Destination Division.	Delete the STATUS or MESSAGE clause.
081 F OPERATOR MAY NOT BE DEDICATED	A terminal declared in the OPERATOR paragraph of the Application Global Division cannot be declared DEDICATED in a SYMBOLIC-NAME paragraph of the Source-Destination Division.	Delete the DEDICATED phrase or define a different operator terminal.

TABLE B-1. ADLP ERROR MESSAGES (Contd)

Message	Significance	Action
082 F INVALID OR MISPLACED SYNTAX	Determine from context.	Correct the error.
083 F ALIAS IS NAME ONLY WITH TYPE QUEUE OR JOURNAL	In a SYMBOLIC-NAME paragraph, ALIAS IS name can be specified only if name is type QUEUE or JOURNAL. If symbolic name type is SOURCE, DESTINATION, or INTERACTIVE, the correct form is ALIAS IS TERMINAL or ALIAS IS USER.	Correct the SYMBOLIC-NAME paragraph.
084 F AT LEAST ONE SYMBOLIC NAME REQUIRED	At least one SYMBOLIC-NAME paragraph must appear in the Source-Destination Division.	Supply a SYMBOLIC-NAME paragraph.
085 F AT LEAST ONE QUEUE REQUIRED	The Queue Division must contain at least one QUEUE paragraph.	Supply a QUEUE paragraph.
086 F NO RESIDENCY FOR COMPOUND QUEUE	A RESIDENCY clause can be specified only for simple queues.	Correct the RESIDENCY clause.
087 F VALUE NOT REPRESENTABLE IN FIELD	A literal is too long for a field.	Supply a valid literal or change field definition.

TABLE B-2. COMPILATION SUMMARY MESSAGES

Message	Significance	Action
nnnnnnnnnn DIAGNOSTIC MESSAGES	The indicated number of diagnostic messages was issued during compilation.	If messages were issued, refer to table B-1 for significance and action.
nnnnnnnnnB C M REQUIRED	The indicated octal number of central memory words was required for the compilation.	None.
nnnnnn.nnn SECONDS CP TIME	The indicated number (decimal) of central processor seconds was required for compilation.	None.
nnnnnnnnnB TOTAL TABLES SIZE	The indicated octal number of central memory words was required for the tables generated by ADLP.	None.
* FATAL ERRORS - NO TABLES *	Because of fatal errors detected by ADLP during compilation, no application definition tables were produced.	Correct the errors and recompile.

TABLE B-3. ADLP DAYFILE MESSAGES

Message	Significance	Action
54 TABLE NOT PRESENT	Internal ADLP error.	Follow site-defined procedures for reporting software problems.
ADLP ABORTED	Internal ADLP error.	Follow site-defined procedures for reporting software problems.
BAD OLD LIBRARY HEADER	The library file header is not properly formatted.	The old library cannot be recovered. Recreate the library.

TABLE B-3. ADLP DAYFILE MESSAGES (Contd)

Message	Significance	Action
CALL STATEMENT ERROR	One or more ADLP control statement parameters contains an error.	Correct the ADLP control statement.
COMPILER ERROR - STD STACK OVERFLOW	Internal ADLP error.	Follow site-defined procedures for reporting software problems.
COMPILER ERROR - STD STACK UNDERFLOW	Internal ADLP error.	Follow site-defined procedures for reporting software problems.
DELETE WITHOUT OLD AND NEW LIBS	In order to delete files from an old library, the old library name (OLD parameter) and new library name (NEW parameter) must be specified on the ADLP control statement.	Specify OLD=oldname and NEW=newname on the ALDP control statement.
NO SOURCE, OLD, OR NEW LIBRARY	ADLP could not find the source file, old library, or new library specified on the ADLP control statement.	Correct the ADLP control statement.
OLD LIBRARY NOT FOUND	ADLP could not find the old library specified on the ADLP control statement.	Correct the ADLP control statement.
OLD LIB-DIRECTORY DO NOT MATCH	Old library directory does not match file structure.	Recreate old library.
TSB LOGICAL ERROR	Internal ADLP error.	Follow site-defined procedures for reporting software problems.
* FATAL ERRORS - NO TABLES *	Because of fatal errors in the ADL program, no application definition tables are produced.	Correct the errors listed in the ADLP source listing and rerun.
xxxxxxxxxx OVERLAY LOAD ERROR	The named MCS overlay could not be loaded so program aborted.	Follow site-defined procedures for reporting software problems.

TABLE B-4. MCS AND NETWORK ERROR MESSAGES

Message	Significance	Action	Issued By
APPLICATION FAILED. MCS CONNECT TIME hh.mm.ss xxxxxxx - APPLICATION:	MCS has failed: hh.mm.ss specifies the length of time terminal xxxxxxx was connected to MCS.	MCS is not available. Enter another application name or wait a short time and enter MCS.	NVF
APPLICATION FAILED. aaaaaaa CONNECT TIME hh.mm.ss xxxxxxx-APPLICATION:	The application has failed. The parameter aaaaaaa is the name of the failed application. The parameter hh.mm.ss specifies the length of time terminal xxxxxxx was connected to the application.	Enter the name of another application.	NVF
APPLICATION HAS NO DUMP FILE DEFINED	For a DUMP command, a dump file must be defined in the application definition.	Specify a DUMP-FILE paragraph in the Application Global Division.	MCS
APPLICATION IDLE	Attempt was made to log in to an idled application. Also issued following an IDLE command.	The application operator must RESUME the application.	MCS

TABLE B-4. MCS AND NETWORK ERROR MESSAGES (Contd)

Message	Significance	Action	Issued By
APPLICATION INITIATED	Informative. Issued after the application operator initiates an application.	None.	MCS
APPLICATION NOT PRESENT xxxxxxx-APPLICATION:	The requested network application program is not executing at a control point. xxxxxxx represents the terminal name.	Enter the name of another application, or try again later.	NVF
APPLICATION NOT RUNNING	An attempt was made to log in to an MCS application that has not been initiated.	The application operator must initiate the application.	MCS
APPLICATION OPERATOR ACTIVE	The user cannot log in as the application operator because one is already active.	Log in as non-application operator.	MCS
APPLICATION RESUMED	Informative. Issued following a RESUME command.	None.	MCS
APPLICATION RETRY LIMIT	Four unsuccessful attempts to enter a legal network application name have been made.	Hang up and get help.	NVF
APPLICATION SHUTDOWN	An attempt was made to log in to an MCS application that has been shut down by the application operator. Also issued following a SHUTDOWN command.	The application must be reinitiated by the AOP or console operator.	MCS
APPLICATION START FAILED	An internal error occurred when the application operator attempted to initiate the application.	Consult system analyst.	MCS
COMMAND INVALID FROM PROGRAM	DATA, END, LOGIN, LOGOUT, RETRIEVE, RESUME cannot be entered from a user program.	Correct program.	MCS
COMMAND MODE	Informative. The terminal is in command mode.	None.	MCS
CONNECTION PROHIBITED, TRY AGAIN LATER xxxxxxx - APPLICATION:	Another terminal user with the same user name is currently logged into MCS.	Enter the name of another application, or try again later.	NVF
CONNECTION REJECTED. xxxxxxx - APPLICATION	MCS has refused connection with the terminal xxxxxxx.	MCS is saturated (all resources in maximum usage). Try again later.	NVF
DATA MODE	Informative. The terminal is in data mode.	None.	MCS
DEDICATED TERMINAL APPLICATION=xxxxxxx SYMBOLIC NAME=xxxxxxxxxxx STATUS=xxxxxx xxxxxxx MODE	Informative. The terminal is dedicated to application xxxxxxx, or to symbolic name xxxxxxxxxxx for this application.	None.	MCS
DESTINATION ALREADY IN USE	The symbolic destination name specified in login sequence is being used by another terminal.	Specify a different destination name.	MCS
ERROR IN ROUTE, PROGRAM NOT INVOKED	MCS was unable to initiate execution of the specified program.	Invocation file might be direct access; if so, change to indirect.	MCS

TABLE B-4. MCS AND NETWORK ERROR MESSAGES (Contd)

Message	Significance	Action	Issued By
ERR..	Unrecognized TIP command syntax.	Reenter command correctly.	TIP
EXPECTING MONITOR FILE	This form of the DISPLAY command must specify a monitor file name.	Specify a monitor file.	MCS
FAMILY:	Normal system prompt for family name.	Enter a valid NOS family name.	NVF
FROM EVENT TABLE - string	Informative. The string was sent by MCS when an event described in the application definition occurred.	None.	MCS
FROM LOP...(string)	A message from the local operator is displayed as (string).	None.	TIP
FROM PROG xxxxxxx - string	Informative. The string was sent from program xxxxxxx, via MESSAGE command.	None.	MCS
FROM TERMINAL UNKNOWN	The source terminal specified in a REROUTE command is not known to the application.	Specify a valid source terminal.	MCS
FROM THE AOP - string	Informative. The string was sent by the application operator.	None.	MCS
HELLO - YOU'RE THE AOP	Informative. The correct password was specified in login sequence, and the terminal is now the AOP terminal.	None.	MCS
HOST UNAVAILABLE	Unable to communicate with host computer.	None.	CCP
ILLEGAL APPLICATION, TRY AGAIN xxxxxxx-APPLICATION:	The name used to request a network application program is unknown to NVF.	Correct the entry, or enter the name of another application program.	NVF
ILLEGAL USER	Four unsuccessful login attempts have been made.	Hang up and get help.	NVF
IMPROPER LOGIN, TRY AGAIN FAMILY:	An unacceptable family name, user name, or file access password has been entered.	Restart the login procedure by entering a valid family name.	NVF
INCORRECT SYMBOLIC NAME	The name specified in a PURGE command is not defined in the application definition as a symbolic destination or interactive name.	Specify a valid symbolic destination or interactive name.	MCS
INVALID COMMAND	MCS could not recognize the command, or an AOP command was entered from a non-AOP terminal.	Enter a valid command.	MCS
INVALID JOB FILE	The invocation file associated with the program specified in the INVOKE command contains a job statement error.	Correct the invocation file.	MCS

TABLE B-4. MCS AND NETWORK ERROR MESSAGES (Contd)

Message	Significance	Action	Issued By
INVALID KEYWORD	MCS could not recognize a required word in the command.	Enter the correct keyword.	MCS
INVALID PARAMETER	MCS could not recognize a parameter in the command.	Enter the correct command.	MCS
INVALID PASSWORD	The specified MCS application password does not match the one defined in the application definition.	Enter the correct password.	MCS
INVALID RETRIEVE COUNT	The message count specified in the RETRIEVE command was an invalid number.	Enter a valid retrieve count.	MCS
INVOCATION FILE NOT FOUND	MCS could not find the invocation file associated with the program specified in the INVOKE command.	Create an invocation file and ensure that it is properly defined in the application definition and that it is a public indirect access file.	MCS
LOGIN ABORTED, TRY LATER	Insufficient system resources are available to allow the terminal to gain access to the network.	Wait a short time for system resources to become available, then reinitiate login.	NVF
MCS 1.x date time	Informative.	None.	MCS
MCS APPLICATION ?	Prompt requesting MCS application name.	Respond with name of desired application.	MCS
MCS ENDED dd/mm/yy hh.mm.ss	Informative. Issued after terminal's connection to MCS is broken. Indicates date and time of disconnect.	None.	MCS
NEXT BLOCK MAY BE OUT OF SEQUENCE	A message output transmission error has occurred.	If error recurs, follow site-defined procedure for reporting software problems.	MCS
NO MONITOR FILE DEFINED	Information cannot be written to a monitor file because none is defined in the application definition.	No user action. The application definition must be rewritten to include a monitor file definition.	MCS
NOT VALIDATED AS AOP, NON AOP ASSUMED	The user entered an AOP password, but the symbolic name was not validated in the application definition as an AOP name.	None.	MCS
OVER..	Page wait has occurred. More output is available.	Enter an empty line.	TIP
INVALID SYMBOLIC NAME	An invalid name was entered in response to the SYMBOLIC-NAME prompt.	Enter a valid symbolic name.	MCS
PARITY ERROR IN MESSAGE	MCS could not transmit the message because of an internal error.	Reenter the message. If error recurs, follow site-defined procedures for reporting software problems.	MCS
PASSWORD:	Normal system prompt for file access password.	Enter a valid password.	NVF

TABLE B-4. MCS AND NETWORK ERROR MESSAGES (Contd)

Message	Significance	Action	Issued By
PROGRAM ALREADY RUNNING	The program specified in the INVOKE command is already executing.	None.	MCS
PROGRAM HAS NO INVOCATION FILE	The program cannot be initiated by an INVOKE command because no invocation file is defined in the application definition.	No user action. The application definition must be rewritten to include an invocation file definition for the program.	MCS
PROGRAM NAME NOT KNOWN	The program name specified in the INVOKE command is not defined in the application definition.	Ensure that correct program name is specified. Otherwise, no user action. Application definition must be rewritten to include program name.	MCS
PROGxxx - CHECKPOINT TAKEN AT hh.mm.ss	Informative. The COBOL program has initiated a checkpoint dump.	No action.	MCS
QUEUE DISABLED - MESSAGE DISCARDED	Issued in data mode. The message could not be delivered because the input queue is disabled.	The application operator must enable the queue.	MCS
REPEAT..	Due to a temporary overload condition, the last logical line of input has been discarded.	Repeat input.	TIP
ROUTING FAILURE - MESSAGE DISCARDED	Message entered did not meet any of the routing criteria specified in the application definition.	Use correct message format.	MCS
SERIAL NUMBER = xxxxxxxxxx	Informative. The message just entered was assigned serial number xxxxxxxxxx.	No action.	MCS
SERIAL NUMBER FIELD MISSING	The message description in the application definition requires that input messages contain a serial number.	Supply a serial number in the appropriate field of the input message.	MCS
SERIAL NUMBER INVALID	Serial numbers must be valid integers and must not exceed 1 073 741 823.	Supply a valid serial number.	MCS
SERIAL NUMBER NOT FOUND	The serial number field in the message could not be found.	Use correct serial number format as defined in the application definition.	MCS
SERIAL NUMBER TOO LARGE	The maximum value of message serial numbers is 1 073 741 823.	Supply a smaller serial number.	MCS
SOLICITATION LIMIT EXCEEDED	The limit on the number of attempts for entering a valid application name or symbolic name has been exceeded.	The login procedure must be repeated.	MCS
SOURCE ALREADY IN USE	The symbolic source name is being used by another terminal.	Enter a different source name.	MCS
SYMBOLIC-NAME ?	MCS prompt requesting symbolic name. The symbolic name is the name to be associated with the terminal in the application definition.	Enter either one or two symbolic names.	MCS

TABLE B-4. MCS AND NETWORK ERROR MESSAGES (Contd)

Message	Significance	Action	Issued By
SYNTAX ERROR	MCS could not recognize the command.	Enter a correct command.	MCS
TERMINAL DESTINATION ONLY - MESSAGE DISCARDED	The symbolic name under which the terminal is logged in is defined in the application definition to be a destination. Therefore, the terminal cannot send messages.	Repeat login procedure and select a symbolic name defined as source or interactive.	MCS
TERMINAL DISABLED - MESSAGE DISCARDED	The terminal has been disabled by the application operator and therefore cannot receive messages.	Enable the terminal in command mode.	MCS
TERMINAL NOT CONNECTED	The terminal specified in the MESSAGE command is not connected to the application.	Wait for terminal to connect.	MCS
TIMEOUT	The most recent prompting message was not responded to within 2 minutes. Terminal disconnect is requested.	If the terminal disconnects, a new telephone connection must be established before login can begin.	NVF
TO KEYWORD EXPECTED	The keyword TO was omitted from the MESSAGE command.	Enter the correct command.	MCS
TO TERMINAL MUST BE CONNECTED	The terminal to which the message was rerouted is not connected to the application.	Send message to a connected terminal	MCS
TO TERMINAL UNKNOWN	The symbolic name to which the message was rerouted is not defined in the application definition.	Specify a defined symbolic name.	MCS
UNKNOWN APPLICATION	MCS could not find the specified application. The application name might be spelled incorrectly or it might not reside on the currently active library.	Enter a valid application name.	MCS
UNKNOWN COMMAND	MCS could not recognize the command.	Enter a correct command.	MCS
UNKNOWN DESTINATION	The symbolic destination name (second name if two names entered) is not defined in the application definition.	Enter a valid destination name.	MCS
UNKNOWN PARAMETER	The DISPLAY option contains an error.	Enter the correct DISPLAY command.	MCS
UNKNOWN PROGRAM NAME	The program name specified in the REVOKE command is not defined in the application definition.	Specify a defined program name.	MCS
UNKNOWN QUEUE NAME	The specified queue name is not defined in the application definition.	Ensure that queue name is spelled correctly.	MCS
UNKNOWN SYMBOLIC NAME	The specified symbolic name is not defined in the application definition.	Enter a defined symbolic name.	MCS
UNKNOWN TERMINAL NAME	The terminal name specified in the MESSAGE command is not defined in the application definition.	Specify a defined symbolic name.	MCS

TABLE B-4. MCS AND NETWORK ERROR MESSAGES (Contd)

Message	Significance	Action	Issued By
USER NAME:	Normal system prompt for user name, issued during login.	Enter the assigned user name.	NVF
YOU CAN'T DISCONNECT YOURSELF	AOP DISCONNECT command cannot specify AOP terminal.	None.	MCS
aaaaaaa CONNECT TIME hh.mm.ss xxxxxxx-APPLICATION:	Control of the terminal has been returned to NVF from an application. The parameter aaaaaaa is the name of the application previously selected. The parameters hh.mm.ss indicate the length of time terminal xxxxxxx was connected to the application.	Enter the name of another application.	NVF
xxxxxxx-APPLICATION:	Prompt for application program. xxxxxxx represents the terminal name.	Enter name of desired application program.	NVF
xxxxxxx - INVALID TEST MODE SOURCE	For applications in test mode, the first segment of a message in the injection queue must be a valid symbolic name.	Correct program that loads injection queue.	MCS

TABLE B-5. SYSTEM CONSOLE AND DAYFILE MESSAGES

Message	Significance	Action	Issued By
ADL ASSIGNED PFN= xxxxxx UN=xxxxxx.	Informative - indicates Application Definition file attached by MCS.	None.	MCS
ADL CREATED YY/MM/DD. HH.MM.SS	Informative - creation date and time of ADL file.	None.	MCS
ADL NOT AVAILABLE PFN=xxxxxxx, UN=xxxxxxx	System could not attach ADL file.	Assign correct ADL file.	MCS
ADLP ABORTED.	The ADLP control statement or the input source file contains errors.	Correct error and rerun job.	ADLP
APPL - xxxxxxx INITIALIZED.	Informative - indicates application was started and is now active.	None.	MCS
APPL - xxxxxxx JOURNAL yyyyyyy DISABLED	Journaling was turned off due to CIO errors or bad owner name in ADL definition of the file.	Correct owner if appropriate.	MCS
APPL - xxxxxxx MONITOR yyyyyyy DISABLED	Monitor file is not used due to CIO errors or bad owner name in ADL definition of the file.	Correct owner if appropriate.	MCS
APPL - xxxxxxx PROG yyyyyyy CONNECTED.	Informative - a test mode program has connected to MCS.	None.	MCS
APPL - xxxxxxx PROG yyyyyyy DISCONNECT.	Informative - a test mode program has disconnected from MCS.	None.	MCS
APPL - xxxxxxx PROG yyyyyyy REVOKED.	MCS has revoked (aborted) the named program.	None.	MCS

TABLE B-5. SYSTEM CONSOLE AND DAYFILE MESSAGES (Contd)

Message	Significance	Action	Issued By
APPL - xxxxxxx Q yyyyyyyyyyyy PURGED.	The named queue file was purged because it could not be verified upon recovery.	None.	MCS
APPL - xxxxxxx QUEUE yyyyyyyyyyyy IN CM	Named disk queue was moved to CM due to bad owner name in ADL definition of the file.	Correct owner name in ADL.	MCS
APPL - xxxxxxx RECOVERED FILE yyyyyyy.	Informative - displayed for each file when application is initiated.	None.	MCS
APPL - xxxxxxx SHUTDOWN.	The application was successfully shutdown.	None.	MCS
APPL - xxxxxxx START FAILED, NO MEMORY	No memory is available to start the application.	Try again later.	MCS
APPL - xxxxxxx START FAILED, I/O ERROR.	Errors occurred while trying to read ADL for the application.	Recreate ADL file.	MCS
APPL - xxxxxxx START FAILED, FILE BUSY.	The application file is busy so application startup was aborted. This message is preceded by file busy message specifying the name of the busy file.	Return the busy file and try start up again.	MCS
APPL - xxxxxxx Q yyyyyyyyyyyy FLUSHED.	Memory saturation forced the indicated queue to disk.	None.	MCS
APPLICATION ALREADY RUNNING.	An attempt was made to start an already active application.	None.	MCS
BAD DIRECTORY ON ADL.	Bad ADL file during MCS start up.	Recreate ADL file.	MCS
BAD OLD LIBRARY HEADER	Old ADL file header is not properly formatted.	Recreate old ADL library	ADLP
BAD VERIFICATION RECORD ON ADL.	Bad ADL file during MCS start up.	Recreate ADL file.	MCS
CALL STATEMENT ERROR.	One or more of the ADLP parameters is invalid.	Correct ADLP call statement.	ADLP
COMMAND ILLEGAL AFTER GO.	The console command entered is illegal after n.CFO.GO.	None.	MCS
DELETE WITHOUT OLD AND NEW LIBS	Cannot delete an ADL file from an old library unless both library files are specified.	Add OLD and NEW parameters to ADLP call.	ADLP
* FATAL ERRORS - NO TABLES *	Due to fatal errors no new ADL library was written.	Correct errors and rerun.	ADLP
FILE BUSY PFN = xxxxxxx UN = xxxxxxx.	Informative - MCS could not attach the named file.	None.	MCS
GO ALREADY RECEIVED.	Informative.	None.	MCS
GO RECEIVED.	Informative.	None.	MCS
ILLEGAL COMMAND.	Informative.	Reenter the correct command.	MCS
IO ERROR xxxx ON ROLLOUT.	Rollout aborted due to IO errors. Further roll outs will not be attempted.	Consult system analyst.	MCS

TABLE B-5. SYSTEM CONSOLE AND DAYFILE MESSAGES (Contd)

Message	Significance	Action	Issued By
IO ERROR xxxx ON yyyyyyy.	CIO error xxxx encountered on file yyyyyyy.	An additional message will provide the disposition of file. Consult system analyst.	MCS
MCS DISABLED BY NETWORK.	MCS cannot NETON to NAM.	Enter LOP command to enable MCS in the network.	MCS
MCS IDLE DOWN STARTED.	Informative - operator entered CFO.IDLE command.	None.	MCS
MCS INITIATED INCORRECTLY - TRY N.MCS	Operator entered X.MCS command.	Enter n.MCS	MCS
MCS NETON COMPLETE.	Informative.	None.	MCS
MCS REPRIEVE.	Fatal error encountered by MCS.	Inform site analyst.	MCS
MCS SHUTDOWN COMPLETE.	Informative.	None.	MCS
NO SOURCE, OLD OR NEW LIBRARY.	All of the files mentioned could not be found.	Correct ADLP call statement.	ADLP
OLD LIBRARY NOT FOUND.	Old library was specified but could not be found.	Correct ADLP call statement.	ADLP
OLD LIB-DIRECTORY DO NOT MATCH	Old library directory does not match file structure.	Recreate old library file.	ADLP
PFM ERROR xx PFN = xxxxxxx UN = xxxxxxx.	MCS could not attach file as indicated.	Refer to NOS reference manual for error codes.	MCS
REASSIGN ADL.	Informative message that follows ADL NOT AVAILABLE message.	Enter n.CFO.ADL command.	MCS
STRING TOO LONG.	CFO command contains string longer than 7 characters.	Reenter command.	MCS
TEST MODE, NETWORK NOT USED.	MCS was started in global test mode.	None.	MCS
UNKNOWN APPLICATION.	Invalid application name was specified on a n.CFO.START.	Enter correct name.	MCS
WAITING FOR CFO.GO.	MCS processing is suspended until GO is entered.	Enter n.CFO.GO.	MCS
WAITING FOR NETWORK.	NAM was not active when MCS tried to NETON.	Bring NAM up.	MCS
xxxxxxxxxx OVERLAY LOAD ERROR	Overlay could not be loaded so program aborted.	Consult system analyst.	MCS, ADLP

GLOSSARY C

Alias -
Another name by which a symbolic name is known, either to the MCS application or to the operating system. Aliases are defined in the ALIAS clause of the Source-Destination Division. See Symbolic Name.

Application -
See MCS Application and Network Application.

Application Data Division -
The ADL division that describes the format of application message text that is enqueued based on contents or contains a field to store serial numbers.

Application Definition Language (ADL) -
The language that defines and describes all application components to MCS.

Application Definition Language Processor -
The software that compiles ADL programs and produces as output an application definition.

Application Definition Library -
The collection of application definitions that have been compiled by the Application Definition Language processor. The library consists of a multi-record file plus a directory.

Application Global Division -
The ADL division that names an application and specifies certain aspects of it that apply to the entire application.

Application Operator (AOP) -
The terminal or group of terminals that can control an application. The AOP is the primary source of commands to MCS, and the primary destination of status messages from MCS. Only one AOP can be active at a time for an application.

Application Processing Division -
The ADL division that specifies application events that MCS monitors and defines actions MCS takes when the events occur.

Application Program Division -
The ADL division that names the COBOL programs in the application and specifies their interface with MCS.

Application Signature -
The password that provides access security for the entire application. This password is specified in the SIGNATURE paragraph of the Application Global Division. See Password.

Automatic Login -
The process whereby one or more of the Network Validation Facility login dialog parameters are supplied to NVF from the local configuration file. Parameters supplied through automatic login suppress prompting for the corresponding dialog entries and override any entries made from the terminal. Also, MCS automatic login where MCS supplies the symbolic name when a network terminal or user is named in an ALIAS clause; MCS supplies the application name in addition to the symbolic name when the terminal or user is declared as DEDICATED in the ALIAS clause. See Login.

Break-1 -
A character used by a terminal operator to terminate terminal output.

Break-2 -
A character used by a terminal operator to switch a terminal in data mode to command mode.

Broadcast List -
A group of destinations of messages referred to collectively by MCS or a COBOL program.

COBOL Communication Facility (CCF) -
The COBOL language feature that enables COBOL programs to use MCS to receive messages for processing and to send processed messages to a terminal.

Collection Queue -
The network destination substitute that stores messages copied from output queues when an MCS application is in test mode.

Command Mode -
The mode of operation of a terminal that accepts as input MCS user commands or AOP commands and accepts as output MCS responses to these commands.

Comment Line -
A source program line with an asterisk or a slash in column 7 and any characters from the 64-character display code set in area A and area B of that line. The line is documentary. A slash in column 7 ejects the page before the comment line is listed.

Communication Description Area -
The area of a COBOL program that interfaces with MCS; part of the COBOL Communication Facility.

Compound Queue -
A queue that has subqueues. A hierarchal structure of queues. See Queue.

Condition Name -
A user-defined name that is assigned a specific value, set of values, or range of values within a complete set of values that a data name can assume. See Data Name.

Data Mode -
The mode of operation of a terminal that accepts as input data messages for processing and accepts as output processed data messages.

Data Name -
A user-defined name that represents a unit of storage that can assume different values. See User-Defined Name.

Dedicated -
A relationship between a symbolic name and a terminal or a user applying to all running applications in an application definition library. The relationship is defined in the ALIAS clause of the Source-Destination Division. See Symbolic Name.

Dequeue -
The process of removing a message from a queue.

Destination -
The terminal or group of terminals to which application messages can be sent.

Disable -
The process of deactivating the logical connection between an MCS application and a terminal. A disabled terminal cannot enter or receive data messages. See Logical Connection.

Disconnect -
The process of ending a logical connection between a terminal and an MCS application. See Logical Connection.

Division -
One of the six required parts of an ADL program. The divisions are the Application Global Division, Application Program Division, Application Data Division, Source-Destination Division, Queue Division, and Application Processing Division.

Enable -
The process of activating or reactivating the logical connection between an MCS application and a terminal. An enabled terminal can enter and receive data messages. See Logical Connection.

End-of-Group Indicator (EGI) -
A character that is defined in the application definition and logically separates a group of several messages from succeeding messages and signals the end of a group of messages to MCS or a COBOL program.

End-of-Message Indicator (EMI) -
A conceptual indicator delimiting one message from the next message and notifying MCS or a COBOL program that the end of message condition exists.

End-of-Segment Indicator (ESI) -
A conceptual indicator delimiting one segment within a message from the next segment within the message and notifying MCS or a COBOL program that the end of segment condition exists.

Enqueue -
The process of storing a message in a queue.

Echo -
The process of returning the serial number assigned to a message to the message source.

Event -
An application occurrence that MCS monitors; specified in the Application Processing Division.

File Name -
A user-defined name that represents a storage medium used by MCS in cooperation with the operating system to record relevant information. See User-Defined Name.

Initiation -
The manner in which an MCS application is started. Specified in the INITIATION paragraph of the Application Global Division. Initiation can be defined as automatic or explicit. When automatic initiation is specified, the application is started in normal mode or test mode at MCS startup time. When explicit is specified, the application is started in normal mode when a terminal gains AOP status.

Injection Queue -
The network source substitute that stores incoming messages from a COBOL message generation program when an MCS application is in test mode.

Input Queue -
A storage area for incoming messages awaiting disposition by a COBOL message processing program.

Integer -
A numeric literal that does not include any character positions to the right of the assumed decimal point; must not be signed. See Numeric Literal.

Interprogram Queue -
An input queue that stores messages destined for other COBOL programs until the other programs request them.

Invitation List -
A group of sources of messages referred to collectively by MCS or a COBOL program.

Invocation File -
A job submission file containing control statements and job statements necessary to execute a COBOL program; must be an indirect access public permanent file. This file is submitted to the operating system job input queue when the COBOL program is called by the INVOKE verb or the AOP INVOKE command.

Journal File -
A file onto which messages are copied after they are enqueued or dequeued.

Journaling -
The process of copying a message onto a journal file.

Keyword -
An ADL reserved word that is required when the format in which the word appears is used in an ADL program. See Reserved Word.

Literal -
A character string whose value is implied by the ordered set of characters that make up the string.

Local Configuration File -
A file in the host computer system that contains information on the physical and logical makeup of the communication elements in the system. The file contains a list of network application programs available for execution in the host computer, and lines and terminals that can access it.

Logical Connection -
A logical message path established between a COBOL program and an MCS application or between a terminal and an MCS application. Until terminated or deactivated, the logical connection allows messages to pass between the two entities.

Login -
The procedure used to connect a terminal with the network and access MCS. Parameters must be entered by the user in response to prompts from the Network Validation Facility. See Network Validation Facility.

Mass Storage -
A disk pack or other rotating mass storage device; not a magnetic tape.

MCS Application -
The COBOL programs, sources, destinations, queues, and journals required to accomplish message routing between the COBOL programs and the user. A complete application is defined in one Application Definition Language program. In this manual, use of the term application implies MCS application. Contrast with Network Application.

Message -
A string of characters with an implied beginning and end. To MCS, a message is the data associated with an EMI or EGI.

Message Indicators -
See End-of-Group Indicator, End-of-Message Indicator, End-of-Segment Indicator.

Message Segment -
Data that forms a logical subdivision of a message.

Monitor File -
A file onto which application status displays are written. The displays written to this file are those generated when the DISPLAY verb or command is used. This file is named in the MONITOR-FILE paragraph of the Application Global Division.

Network Access Method (NAM) -
A software package of interface routines that MCS uses for shared access to a network of terminals.

Network Application -
In the context of network software, a program resident in a host computer that is a terminal servicing facility and provides a specific processing capability such as remote job entry or transaction processing. MCS is a network application program. Contrast with MCS Application.

Network Definition File -
An NDL program output file that determines the configuration of the network; a local configuration file.

Network Definition Language (NDL) -
The compiler-level language used to define the local configuration file contents.

Network Processing Unit (NPU) -
The collection of hardware and software that switches, buffers, and transmits data between terminals and host computers.

Network Validation Facility (NVF) -
The portion of the network software that provides login access security processing.

Nonnumeric Literal -
A literal bounded by quotation marks. Can include any character in the 64-character display code set. To represent a single quotation mark within a nonnumeric literal, two contiguous quotation marks must be used. See Literal.

Numeric Literal -
A literal composed of one or more numeric characters; can contain a decimal point, unary sign or both. See Literal.

Off-Line -
Not interacting directly with the network or not connected to the network.

On-Line -
Interacting with the network; connected to the network.

Output Queue -
A storage area for outbound messages awaiting transmission to a terminal.

Password -
A word chosen by the user to provide access security for an application. Passwords can be specified in an ADL program in the Application Global Division to provide access security for the AOP, in the Source-Destination Division to provide access security when a COBOL program enables or disables a source or destination, and in the Queue Division to provide access security when a COBOL program enables or disables an input queue. One password to provide all of the access security can be specified in the SIGNATURE paragraph of the Application Global Division. Also, a word assigned to a terminal user by a site administrator to provide access security for login to the network. This password is a NOS validation file parameter.

Physical Record Unit (PRU) -
The amount of information transmitted by a single physical operation of a specified device. MCS application definitions reside on mass storage; a PRU is sixty-four 60-bit words.

Purge -
The process of discarding incomplete messages sent to an external destination.

Queue -
A storage area for messages; resides in central memory or on mass storage. See Compound Queue, Input Queue, Interprogram Queue, Output Queue, and Simple Queue.

Queue Division -
The ADL division that names and describes the input and output queues that an application uses. This division also specifies the criteria MCS uses for message disposition and message routing.

Reserved Word -
An ADL word listed in appendix E that can be used in a source ADL program, but which must not appear in a program as a user-defined name.

Response Queue -
A simple input queue defined in the RESPONSE-QUEUE clause of a PROGRAM paragraph. This queue stores MCS responses to commands that a COBOL program can send to MCS.

Routine Name -
A user-defined name that represents an executable code module (program) external to MCS. See User-Defined Name.

Serial Number -
A number identifying each input data message. Serial numbers can be supplied by the user or generated by MCS. The SERIAL-NUMBER clause of the Application Data Division specifies how serial numbers are assigned.

Simple Queue -
A queue with no subqueues. Contrast with Compound Queue.

Source -
A terminal from which messages can be received by an MCS application.

Source-Destination Division -
The ADL division that specifies symbolic names that represent the sources and destinations of an MCS application, and describes the characteristics of the sources and destinations.

Source Program -
A set of ADL statements beginning with an Application Global Division and ending with an Application Processing Division.

Subqueue -
A subdivision of a compound queue; a lower level in the queue hierarchy. See Compound Queue.

Symbolic Name -
A user-defined name representing a source, destination, interprogram queue, or journal file. Symbolic names are defined in the SYMBOLIC-NAME paragraphs of the Source-Destination Division. See User-Defined Name.

System Name -
A user-defined name used to communicate with the operating system. File names and routine names are system names. See User-Defined Name.

System Operator -
The computer operator who controls operations at the host computer console. This operator initializes the MCS subsystem and sends certain commands to MCS.

Test Mode -
An MCS mode of execution in which MCS applications execute independently of the network. This provides a means of testing message routing.

Time Literal -
A special type of nonnumeric literal that represents a chronological quantity. See Nonnumeric Literal.

Transient Terminal -
A terminal that is not associated with a particular symbolic name in an MCS application; a terminal not named in an ALIAS clause in a SYMBOLIC-NAME paragraph. See Alias.

User-Defined Name -
A word supplied by the programmer to satisfy the ADL format of a paragraph or clause.

User Name -
A NOS validation file parameter identifying a valid system user.

APPLICATION DEFINITION LANGUAGE SUMMARY D

A summary of Application Definition Language formats appears in this appendix. Detailed information for each format is referenced by page number. The following elements are alphabetized in one list:

Division structure, by division name
Section structure, by section name
Paragraph header and contents, by paragraph name
Clauses, by clause name
Verbs, by verb name
Conditions, by condition name

Application Processing Division

4-22

APPLICATION PROCESSING DIVISION

[USE paragraph] ...

Application Program Division

4-6

APPLICATION PROGRAM DIVISION

[PROGRAM paragraph] ...

BROADCAST-LIST Paragraph

4-11

BROADCAST-LIST IS symbolic-name-1

DESTINATIONS ARE symbolic-name-2 [AND symbolic-name-3] ...

[MESSAGES clause]

[MODE clause]

[PASSWORD clause]

[STATUS clause]

COLLECTION-QUEUE Paragraph

4-4

COLLECTION-QUEUE IS queue-name

CONDITION Clause

4-9

CONDITION IS condition-name VALUE [IS] { literal-1 / literal-2 THROUGH literal-3 }

[{ literal-4 / literal-5 THROUGH literal-6 }] ...

CONNECT Condition

4-23

CONNECT OF symbolic-name

CONNECTION-BROKEN Condition

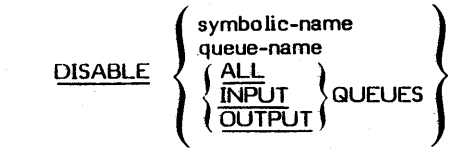
4-23

CONNECTION-BROKEN FOR symbolic-name

CONNECTION-INACTIVE Condition 4-24

CONNECTION-INACTIVE FOR symbolic-name

DISABLE Verb 4-25



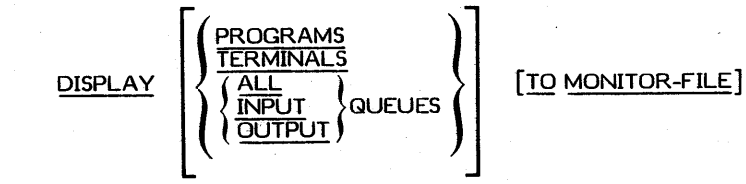
DISCONNECT Condition 4-24

DISCONNECT OF symbolic-name

DISCONNECT Verb 4-25

DISCONNECT symbolic-name

DISPLAY Verb 4-26



DUMP-FILE Paragraph 4-4

DUMP-FILE IS file-name [OWNER IS nonnumeric-literal]

DUMP Verb 4-26

DUMP

EGI Paragraph 4-7

EGI IS nonnumeric-literal

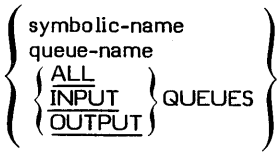
ELAPSED-TIME Condition

4-24

ELAPSED-TIME IS time-literal

ENABLE Verb

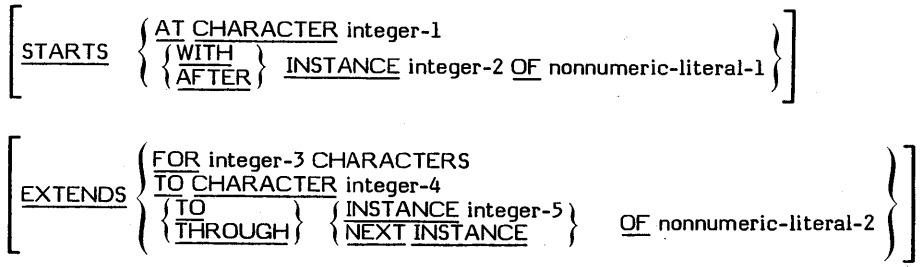
4-26



FIELD Clause

4-8

FIELD IS data-name



[CONDITION clause] ...

IDLE Verb

4-26

IDLE

INITIATION Condition

4-24

INITIATION

INITIATION Paragraph

4-5

INITIATION IS { AUTOMATIC [TEST] / EXPLICIT }

INJECTION-QUEUE Paragraph

4-4

INJECTION-QUEUE IS queue-name

Input Section of Queue Division

4-15

INPUT SECTION

[QUEUE paragraph
[SUB-QUEUE-n paragraph] ...] ...

INVITATION-LIST Paragraph

4-11

INVITATION-LIST IS symbolic-name-1

SOURCES ARE symbolic-name-2 [AND symbolic-name-3] ...

[MESSAGES clause]

[MODE Clause]

[PASSWORD clause]

[STATUS clause]

INVOCATION-FILE Clause

4-6

INVOCATION-FILE IS file-name [OWNER IS nonnumeric-literal]

INVOKE Verb

4-27

INVOKE routine-name

JOURNAL Clause

4-18

JOURNAL IS symbolic-name-1 [ON { INPUT / OUTPUT } [symbolic-name-2 ON { OUTPUT / INPUT }]]

LENGTH Clause

4-8

LENGTH IS integer CHARACTERS

MEDIUM Clause

4-19

MEDIUM IS { CENTRAL / DISK }

MESSAGE thru PASSWORD

MESSAGE Paragraph

4-7

MESSAGE IS data-name

[SERIAL-NUMBER clause]

[SEGMENT paragraph] ...

MESSAGE Verb

4-27

MESSAGE nonnumeric-literal [TO symbolic-name]

MESSAGES Clause

4-13

MESSAGES ARE data-name

MODE Clause

4-13

MODE IS { COMMAND / DATA }

MONITOR-FILE Paragraph

4-4

MONITOR-FILE IS file-name [OWNER IS nonnumeric-literal]

Output Section of Queue Division

4-16

OUTPUT SECTION

[QUEUE paragraph] ...

OPERATOR Paragraph

4-4

OPERATOR IS symbolic-name [PASSWORD IS nonnumeric-literal]

PASSWORD Clause

4-5, 4-13, 4-19

PASSWORD IS nonnumeric-literal

PROGRAM Paragraph 4-6

PROGRAM IS routine-name

[INVOCATION-FILE clause]

[RESPONSE-QUEUE clause]

PURGE Verb 4-27

PURGE symbolic-name

Queue Division 4-14

QUEUE DIVISION

INPUT SECTION

OUTPUT SECTION

ROUTING SECTION

QUEUE Paragraph 4-15, 4-17

QUEUE IS queue-name

[JOURNAL clause]

[MEDIUM clause]

[PASSWORD clause]

[RESIDENCY clause]

[STATUS clause]

REROUTE Verb 4-27

REROUTE symbolic-name-1 TO symbolic-name-2

RESIDENCY Clause 4-19

RESIDENCY IS file-name [OWNER IS nonnumeric-literal]

RESPONSE-QUEUE Clause 4-6

RESPONSE-QUEUE IS queue-name

REVOKE Verb

4-27

REVOKE routine-name

ROUTE Clause

4-19

ROUTE TO queue-name { FROM symbolic-name-1 / TO symbolic-name-2 / WHEN { AFTER / BEFORE } time-literal / FOR condition-name / OTHERWISE }

Routing Section of Queue Division

4-17

ROUTING SECTION

SELECT paragraph ...

SEGMENT Paragraph

4-8

SEGMENT IS data-name

[LENGTH clause]

[FIELD clause] ...

SELECT Paragraph

4-17

SELECT { INPUT QUEUES / SUB-QUEUES OF queue-name / OUTPUT QUEUES } BASED ON { SOURCE / DESTINATION / TIME / CONTENTS OF data-name }

ROUTE clause-1

[ROUTE clause-2] ...

SERIAL-NUMBER Clause

4-7

SERIAL-NUMBER IS { SUPPLIED / GENERATED } [IN data-name] [WITH ECHO]

SHUTDOWN Verb

4-27

SHUTDOWN

SIGNATURE Paragraph 4-3

SIGNATURE IS nonnumeric-literal

SIZE EXCEEDS Condition 4-24

SIZE OF queue-name EXCEEDS integer MESSAGES

Source-Destination Division 4-10

SOURCE-DESTINATION DIVISION

[SYMBOLIC-NAME paragraph] ...

[INVITATION-LIST paragraph] ...

[BROADCAST-LIST paragraph] ...

STATUS Clause 4-14, 4-21

STATUS IS { ENABLED / DISABLED }

SUB-QUEUE-n Paragraph 4-15

SUB-QUEUE-n IS queue-name

[JOURNAL clause]

[MEDIUM clause]

[PASSWORD clause]

[RESIDENCY clause]

[STATUS clause]

SYMBOLIC-NAME Paragraph 4-10

SYMBOLIC-NAME IS symbolic-name

[TYPE clause]

[ALIAS clause]

[MESSAGES clause]

[MODE clause]

[PASSWORD clause]

[STATUS clause]

TIME Condition

TIME IS time-literal 4-25

TYPE Clause 4-2

TYPE IS { SOURCE / DESTINATION / INTERACTIVE / JOURNAL / QUEUE }

USE Paragraph 4-22

USE { WHEN / EVERY } condition verb-1 [verb-2] ...

APPLICATION DEFINITION LANGUAGE RESERVED WORDS E

This appendix contains a list of reserved words. Reserved words must be spelled correctly including any hyphens. Reserved words cannot be used in an ADL program except as specified in the language format.

AFTER
ALIAS
ALL
AND
APPLICATION
APPLICATION-NAME
ARE
AT
AUTOMATIC
BASED
BEFORE
BROADCAST-LIST
CENTRAL
CHARACTER
CHARACTERS
COLLECTION-QUEUE
COMMAND
CONDITION
CONNECT
CONNECTION-BROKEN
CONNECTION-INACTIVE
CONTENTS
DATA
DEDICATED
DESTINATION
DESTINATIONS
DISABLE
DISABLED
DISCONNECT
DISK
DISPLAY
DIVISION
DUMP
DUMP-FILE
ECHO
EGI
ELAPSED-TIME
ENABLE
ENABLED
EVERY
EXCEEDS
EXPLICIT
EXTENDS
FIELD
FOR
FROM
GENERATED
GLOBAL
IDLE
IN
INITIATION
INJECTION-QUEUE
INPUT
INSTANCE
INTERACTIVE
INTO
INVITATION-LIST
INVOCATION-FILE
INVOKE
IS
JOURNAL
LENGTH
MEDIUM
MESSAGE
MESSAGES
MODE
MONITOR-FILE
NEXT
OF
ON
OPERATOR
OTHERWISE
OUTPUT
OWNER
PASSWORD
PROCESSING
PROGRAM
PROGRAMS
PURGE
QUEUE
QUEUES
REROUTE
RESIDENCY
RESPONSE-QUEUE
REVOKE
ROUTE
ROUTING
SECTION
SEGMENT
SELECT
SERIAL-NUMBER
SHUTDOWN
SIGNATURE
SIZE
SOURCE
SOURCES
SOURCE-DESTINATION
STARTS
STATUS
SUB-QUEUES
SUB-QUEUE-1
SUB-QUEUE-2
SUB-QUEUE-3
SUPPLIED
SYMBOLIC-NAME
TERMINAL
TERMINALS
TEST
THROUGH
TIME
TO
TYPE
USE
USER
VALUE
WHEN
WITH

INDEX

CONTROL DATA CORPORATION

COMMENT SHEET

MANUAL TITLE: Message Control System Version 1 Reference Manual

PUBLICATION NO.: 60480300 **REVISION:** A

NAME: _____

COMPANY: _____

STREET ADDRESS: _____

CITY: _____ **STATE:** _____ **ZIP CODE:** _____

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

CUT ALONG LINE

AA3419 REV. 4/79 PRINTED IN U.S.A.

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

TAPE

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division
215 Moffett Park Drive
Sunnyvale, California 94086

CUT ALONG LINE

FOLD

FOLD

TAPE

USER COMMANDS

AOP COMMANDS

CORPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO I

CONTROL DATA CORPORATION