Burroughs

Reference Manual

B 1000 Series Generalized Message Control System (GEMCOS)

(Relative to the 7.0 Software Release)

Priced Item
Printed in U.S.A.
February 1985

1163920

Burroughs

Reference Manual

B 1000 Series Generalized Message Control System (GEMCOS)

(Relative to the 7.0 Software Release)

Priced Item
Printed in U.S.A.
February 1985

1163920

Burroughs cannot accept any financial or other responsibilities that may be the result of your use of this information or software material, including direct, indirect, special or consequential damages. There are no warranties extended or granted by this document or software material.

You should be very careful to ensure that the use of this software material and/or information complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Comments or suggestions regarding this document should be submitted on a Field Communication Form (FCF) with the Class specified as "2" (System Software), the Type specified as "1" (F.T.R.), and the Product specified as the seven-digit form number of the manual (for example, 1163920) The FCF should be sent to the following address:

Burroughs Corporation
Product Assurance and Support
3519 W. Warner Avenue
Santa Ana, CA 92704

TABLE OF CONTENTS

INTRODUCTION

This document is a reference manual for users of the Burroughs B 1000 GEneralized MEssage COntrol System (GEMCOS).

SUMMARY OF SECTIONS

Section 1, the Overview, discusses general features of GEMCOS and gives further information on these features.

Section 2 gives a detailed description of the Transaction Control Language (TCL) using railroad diagrams. (For instruction on how to use the railroad diagrams, see Appendix G.) Section 2 also discusses compiling the TCL and executing the MCS.

Section 3 presents the Network Control Commands.

Section 4 discusses how to format and route messages, while Section 5 describes the interface between application programs and GEMCOS port files.

Security (access control) is discussed in Section 6. Section 7 gives information on selecting recovery options.

Section 8 presents information on testing, patching, using the timimg mechanism, and debugging GEMCOS.

Section 9 discusses how to use station options, including routeheaders. It also discusses Burroughs Network Architecture (BNA) station transfer. Section 10 gives information on how to use the conversational feature.

The Appendices summarize Network Control commands, files, hardware requirements, error messages, and other reference data.

The style identification numbers for the GEMCOS product are: MCB700 (Basic Version), MCA700 (Advanced Version), MCT700 (Total Version).

HELPFUL DOCUMENTS

The following manuals contain additional information about GEMCOS:

GEMCOS MANUALS

1. Formatting Guide, B 1000 Generalized Message Control System (GEMCOS), form 1106531.

2. Capabilities Manual, B 1000 Generalized Message Control System (GEMCOS), form 1164001.

3. User's Guide, B 1000 Generalized Message Control System (GEMCOS) Format Generator, form 1164019.

RELATED MANUALS

1. B 1000 Systems Network Definition Language (NDL) Reference Manual, form 1152014.

2. B 1000 Systems SDL/UPL Reference Manual, form 1137833.

3. B 1000 Systems COBOL Reference Manual, form 1057197.

4. B 1000 Systems Report Programming Language Reference Manual, form 1057189.

5. Burroughs Network Architecture, Architectural Description Reference Manual, form 1132171.

6. Burroughs Network Architecture, Network Control Reference Manual, form 113180.

7. B 1000 Burroughs Network Architecture Installation and Operations Manual, form 1151874.

8. Subordinate Message Control System (SMCS) Manual, form 1152279.

SECTION 1

SYSTEM OVERVIEW

The B 1000 Series Generalized Message Control System (GEMCOS) is a system of programs and files which creates and supports a Message Control System (MCS). An MCS manages the flow of messages between the Network Controller (NC) and application programs that process messages to and from remote terminals.

B 1000 GEMCOS is a software package which allows users to tailor their MCS to meet the specific requirements of their installation. The GEMCOS MCS is both flexible and efficient. It includes these major features:

- The Transaction Control Language (TCL), which allows users to write their own specifications for the MCS.
- Network Control and Administration. GEMCOS provides orderly communication between the hardware devices in the network and adapts the message flow to changing conditions. It also allows remote stations to execute programs, transfers files between computers (through the routeheader capability), supervises other MCSs, logs messages, gathers statistics, and handles errors.
- Interface with Application Programs. GEMCOS executes and terminates application programs, allows multiple programs to run in parallel, prevents two programs from updating a record simultaneously, and attaches remote stations. It simplifies the work of the application programmer by supplying hardware codes and handling error conditions.
- Access Control (Security). GEMCOS prevents unauthorized access to programs and data bases.
- Audit and Recovery. GEMCOS recovers messages, transactions from application programs, and data bases. GEMCOS provides orderly shutdown for the entire network and for network recovery.
- Testing and Debugging. GEMCOS provides off-line testing with its MCSSIM program, patching with its MCSFIX program, and debugging with the GEMCOS data dump and Monitor Trace.

GEMCOS comes in several versions, which are discussed in the following subsections.

VERSIONS OF GEMCOS

GEMCOS is available in three versions in order to accommodate several levels of operating complexity. These versions are the Basic version, the Advanced version, and the Total version. The major capabilities of each of these versions follow:

1. Basic version GEMCOS:

    a. Transaction Control Language (TCL).
    b. Network control.
    c. Message routing.
    d. Nonparticipating MCS.
    e. Access control.
    f. Message auditing.
    g. Message recovery.

2. Advanced version GEMCOS:

    a. All Basic version capabilities.
    b. Message formatting.

3. Total version GEMCOS:

    a. All Advanced version capabilities.
    b. Data base and synchronized recovery.

This manual discusses the features of all versions of GEMCOS.

TRANSACTION CONTROL LANGUAGE

The Transaction Control Language (TCL) is used to select MCS options. It sets up message formatting and routing, and security. The TCL is also used to choose audit and recovery options for user programs and data bases.

The TCL is free form in structure. Key words describe the network environment and user requirements. When the TCL is compiled, customized tables are generated. The MCS then interprets these tables. Because the TCL for B 1000 GEMCOS is similar to the TCL for Large Systems GEMCOS, migration to Burroughs Large Systems is simplified. Compiling the TCL is discussed further in Section 2.

In addition, Mergeable External Source Statements (MESS) allow the user to write special requirements which differ from standard GEMCOS logic. These procedures are also discussed in Section 2.

NETWORK CONTROL AND ADMINISTRATION

GEMCOS and the Network Controller work together to control the network. GEMCOS configures the network, including types of terminals used. The user can alter the hardware and software configuration as needed. Figure 1-1 shows the system structure of B 1000 GEMCOS, with the relationship between the Network Controller, the GEMCOS compiler (MCSTIC), the Control station, the audit file, and application programs.

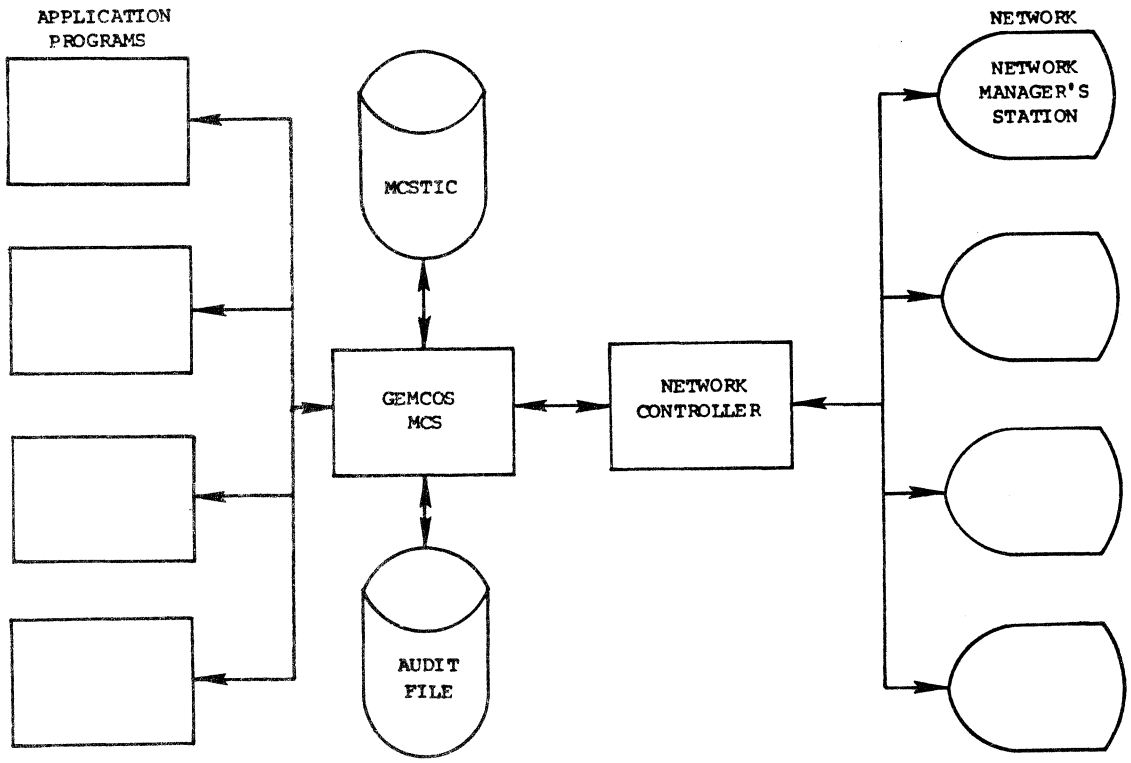


Figure 1-1. System Structure

GEMCOS permits communication between computers through the routeheader capability. It also permits communication with different devices such as matrix printers and modular terminals. These capabilities are discussed in Section 9.

In addition, GEMCOS can act as a supervisory MCS. For example, the station operator can use the Command and Edit Language (CANDE), the On-Line Data Entry System (ODESY), or the Subordinate Message Control System (SMCS), and switch between them as needed. Without GEMCOS, the subordinate MCS would have to be shut down and restarted to switch stations. But with GEMCOS, station operators can use Network Control Commands to switch from one subordinate MCS to another without interrupting other operators. This topic is discussed further at the end of Section 4.

Network restoration is also provided by GEMCOS. Network restoration updates the Network Controller on network status. To perform this task, GEMCOS uses the Table Information Control File, part of the TCL compiler. The MCSTIC file is discussed further in Section 2.

Among its other functions, GEMCOS gathers statistics about stations, programs, and the MCS. The user can obtain information about peak-loads, network use, and response time. It further detects and diagnoses errors and recovery from errors. It also retransmits output as needed, and provides controlled system shutdown when this is required.

NETWORK CONTROL COMMANDS

The following discusses the functions of Network Control Commands and gives information on Control stations.

Functions of Network Control Commands

Network Control Commands (NCCs) perform several functions. They handle security, attachment of stations, MCS control, program execution and termination, message routing and retrieval, changing station status, and reporting program status.

The user decides which Network Control Comands are needed and uses TCL parameters to specify these commands. Therefore, the MCS does not need to contain the logic to execute all of the Network Control Commands.

Control Stations

Operators enter Network Control Commands (NCCs) from Control stations. Control stations are declared in the TCL. A few of the NCCs (such as the sign-on and sign-off commands), can be entered from any station, but most of the Network Control Commands can only be entered from a Control station.

An operator can communicate with the MCS from a Control station. The MCS tells these stations about exceptional conditions in the network. In turn, the operator can ask the MCS about its status and dynamically alter its features.

The supervisory console can also be used as a Control station. All of the Network Control Commands can be entered from the supervisory console.

ERROR HANDLING

The GEMCOS error handling subsystem provides the logic needed to handle error conditions not directly related to applications tasks. GEMCOS automatically takes action to keep the system running and communicates the error condition to an operator. For additional information on error conditions, see Appendix D.

INTERFACE TO APPLICATION PROGRAMS

GEMCOS provides many helpful features for applications programmers. It allows an application program to have parallel processing of a wide variety of transactions. GEMCOS controls the flow of multiple messages and program execution. This improves response time for the application user. Application programs can be written in high-level languages such as COBOL and COBOL74.

The GEMCOS formatting capability means that application programmers do not need to know hardware device codes. This makes the application programs independent of the hardware devices. Message formatting and routing are discussed in Section 4.

The programmer also does not need to write the logic to handle error conditions. This logic is contained in the GEMCOS subsystem. Error conditions are discussed further in Appendix D.

GEMCOS further provides several recovery options, so that the user can choose those options which suit the needs of particular application programs. Recovery is discussed in Section 7.

MESSAGE FORMATTING AND ROUTING

Another important feature of GEMCOS is its ability to perform message formatting and routing.

MESSAGE FORMATTING

The application programmer does not need to know hardware-control and device codes or the buffer capacity for terminals. These are described in the TCL specifications for the MCS. GEMCOS formatting features helpful to the programmer include:

- Forms retrieval.
- Enhancment of the readability of message-text for the user of applications programs.
- Modification of message format without compiling or interrupting application programs.
- Screen wraparound. If a message is too long for a station's buffer, the MCS breaks the message into two or more transmissions.

Information on message formatting is presented in Section 4.

MESSAGE ROUTING

GEMCOS routes messages to and from stations and programs. It also handles communication between programs, and between computers, using the routeheader function. Message routing is discussed in Section 4. The routeheader function is discussed in Section 9.

ACCESS CONTROL (SECURITY)

GEMCOS provides both access security and process security. Access security prevents unauthorized persons from using the system. Process security limits the functions authorized persons are allowed to perform. A specific MCS may be generated as a GEMCOS subsystem. This MCS can have the logic for access security alone, or for both access and process security. Section 6 has further information on access control (security).

AUDIT

GEMCOS keeps an audit trail of all messages sent to an application program or to a data base. The audit trail is written to a disk file called an audit file. See Section 7 for more information on audit files.

RECOVERY

GEMCOS also provides several types of recovery. The types of recovery range from checkpoint recovery to complex data base rollback and synchronized recovery. The user can analyze application programs and select the recovery options which meet the needs of those programs. In addition, GEMCOS also has a procedure for controlled shutdown. Additional information on recovery is available in Section 7.

TESTING, PATCHING, TIMING, AND DEBUGGING

Testing, patching, and timing are accomplished through auxiliary programs. Debugging is done through a data dump and a logic flow monitor. These are controlled by parameters set in the TCL. Each of these features is discussed in detail in Section 8.

TCL COMPILER AND UTILITY PROGRAMS

Specific information on the TCL compiler is given at the beginning of Section 2.

The GEMCOS system also contains four utility programs:

1. MCSRECALL is used for recalling audited messages and is discussed in Section 2.

2. MCSFILXFER is used for transferring disk files between computers. See Section 9 for more information.

3. MCSSIN is used for testing and is discussed in Section 8.

4. MCSFIX is used for patching. It is also discussed in Section 8.

More information on the TCL compiler and on TCL statements is found in Section 2. At the end of Section 2, information is given on how to execute the MCS.

Each of the features of GEMCOS is discussed further in succeeding sections.

SECTION 2

TRANSACTION CONTROL LANGUAGE COMPILER AND STATEMENTS

The B 1000 Transaction Control Language (TCL) is a high-level, descriptive language which enables the user to select required functions and to describe on-line system relationships.

This section discusses the TCL compiler and the files related to it, gives the steps in executing the TCL compiler, presents TCL statements, and explains how to execute the MCS.

TRANSACTION CONTROL LANGUAGE COMPILER (MCSTCL) AND RELATED FILES

The TCL compiler is on the GEMCOS release tape in a file called MCSTCL. When the compiler is executed, it produces a Message Control System (MCS) composed of GEMCOS intrinsics and a data file consisting of on-line relationships. (See Section 1 for additonal information on the functions of an MCS.)

If the users need to change the MCS or the relationships with which it operates, they can easily do this by recompiling.

To enable the user to check the system, MCSTCL produces an optional hard-copy listing of the user's data communication system (MCSLST). It also provides extensive data syntax checking to ensure that the MCS is properly defined. MCSLST is discussed in more detail later in this section.

The limits on the size of the TCL compiler are given in Appendix C.

The following files, the Table Information Control file and the Functions and Formats file, are used with the TCL compiler (MCSTCL).

TABLE INFORMATION CONTROL FILE (MCSTIC)

The MCS uses the Table Informaton Control file (MCSTIC) to store some of its important variables and parameters. By storing these parameters in a disk file, they are preserved from one execution of the MCS to another. They are also protected in case the MCS is terminated unconventionally. When systems relationships change, the TCL compiler rewrites the MCSTIC file.

In addition, the MCSTIC file is used in network restoration. The purpose of network restoration is to update the Network Controller with MCS data on network status. Network status consists of information on the current physical status of a station, such as whether a station is enabled or disabled. Network restoration occurs automatically through the information stored in the MCSTIC file.

The MCSTIC file is also used with the Network Controller. Each time the MCS is executed, it uses status data from the MCSTIC file to generate commands for the Network Controller. The Network Controller uses these files to update its tables in main memory. See the MCSTIC FILE NAME statement in this section for additional information.

FUNCTIONS AND FORMATS FILE (MCSFORMATS)

In the advanced and total versions of GEMCOS, all functions and formats created by the TCL compiler are stored in a separate disk file called MCSFORMATS. See the FORMAT FILE NAME statement in this section for additional information.

SUMMARY OF STEPS TO EXECUTE THE TCL COMPILER

The first step in executing the TCL compiler is to analyze which programs, stations, or features are needed in an MCS. To do this, read through the TCL statements in this section and study any other sections which might be helpful.

The next step is to create the TCL source image, using either cards or the Commmand and Edit Language (CANDE) to create a disk file. The TCL source image is called MCSIN, and consists of the Control Statement, the Global Section, and the Definition Section of the TCL.

To write the source image refer to the Control statement, the syntax of the source image (deck description), the Global Section, and the Definition Section of the TCL. These are all explained in this section.

The third step is to load several files and programs from the B 1000 GEMCOS release tape. These files and programs are discussed in detail later in this section. Also see Figures 2-1 through 2-3 following the CONTROL statement. GEMCOS system files are summarized in Appendix B.

After creating the source image and loading GEMCOS system files, the user is ready to execute the TCL compiler (MCSTCL) and then to use the hard-copy listing to check any syntax errors.

After the TCL compiler has been executed, the user can execute the MCS produced when the TCL is compiled. Instructions on executing the MCS are given at the end of this section, following the TCL statements.

CREATING THE TCL SOURCE IMAGE (MCSIN) (CARD DECK OR CANDE FILE)

To create either a card deck or a CANDE source file, the user first needs to understand the following:

1. The syntax of the source image (deck description).
2. The Control Statement.
3. The Global and Definition Sections of the TCL.
4. The rules for writing the source image.
5. Optionally, how to use the Library statement.

Once the TCL source image (card deck or CANDE file) has been written, the TCL compiler can be executed.

SYNTAX AND COMPONENTS OF THE SOURCE IMAGE

Users need to be familiar with the syntax of the source image, which is given at the beginning of the the Control Statement. They should also understand the other parts of the source image, the Control Statement, and the Global and Definition Sections of the TCL.

Once users understand these components, they can use the following rules to write the source image.

RULES FOR THE SOURCE IMAGE

The cards which compose the TCL source image (deck) are similar to those of a User Programming Language (UPL) source deck:

1. Columns 73 through 80 are reserved for sequence numbers.

2. Comments may occur on any card following a "%".

3. Statements may begin in any column.

4. Any source image that contains the string "&PAGE" or "& PAGE" starting in column 1 causes the listing to be advanced to channel 1 before the printing continues.

5. A source image that contains "$NO LIST" in column 1 causes only errors to be listed. When a source image that contains "$LIST" in column 1 is read, both source records and errors are listed. "$LIST" is the default at the start of MCSTCL.

6. The TCL compiler does not permit a continuation from card to the next. If a string is begun on a card, it must end on that card.

USING THE LIBRARY STATEMENT

Users can merge disk files (libraries) into the main TCL specification with the LIBRARY OPTION STATEMENT. The syntax for this statement is:

$LIBRARY <file-ID>.

or:

$ LIBRARY <file-ID>.

Any source image that starts with $LIBRARY, or $LIBRARY followed by a valid disk file name, causes MCSTCL to merge that file into the main TCL specification deck. A LIBRARY statement can occur anywhere within the TCL specifications, but not within a library. The dollar sign ($) must be in column 1. <file ID> is a valid B 1000 file identifier. The statement must terminate with a period.

Whenever the TCL compiler encounters a valid LIBRARY statement, and the specified <file-ID> is on disk, the contents of this file is compiled at the point of occurrence as if the contents actually were contained

within the main TCL deck. If the <file ID> is not on disk, the statement is simply ignored.

The listing produced by the TCL compiler reflects any files merged into the main deck. The merged lines are marked with an "L" in column 1 of the listing. Any number of LIBRARY statements can occur within a TCL specifications deck. The following are two examples of these statements.

```
$LIBRARY GEMCOS/FORMATS.

$ LIBRARY GEMCOS/"NEW.FORMS".
```

EXECUTING THE TCL COMPILER (MCSTCL)

After the TCL source image has been written, the TCL compiler can be executed. The first step in doing this is to load GEMCOS system programs and files.

LOADING GEMCOS SYSTEM FILES

Load the following files and programs from the GEMCOS release tape:

1. The TCL compiler, MCSTCL.
2. The file GEMCOS/MCSGTS, which is the master source file. GEMCOS/MCSGTS is input to GEMOCS/MCSGO.
3. The program GEMCOS/MCSGO, which uses GEMCOS/MCSGTS to build user source code from the master source code.
4. The User Programming Language 2 (UPL2) compiler, which is only used for GEMCOS compiles.
5. Optionally, the file MCSERROR, which contains standard GEMCOS error messages that are modifiable, may also be loaded.

Modifying MCSGO

The user can place GEMCOS/MCSGTS on a different pack, and/or modify its name. To change the external file name of MCSGTS, enter the following statement:

PID USER

```
MODIFY GEMCOS/MCSGO FILE MCSGTS NAME <file name>;
```

The following gives information on using standard GEMCOS error messages.

Using Standard Error Messages

Users have the option of using standard GEMCOS error messages or messages of their choice. To use standard output messages, no change from the present procedures is required. The standard messages are contained in the TCL compiler.

Modifiable error messages are contained in a CANDE-compatible file which is input to MCSTCL. This file is called GEMCOS/MCSERROR, and can be modified with CANDE. It is a sequential file with the following structure:

1. The first record in the file is the control record. The current GEMCOS release level must appear in the first 5 bytes. The remaining records contain the messages.

2. There can be 35 bytes maximum for each output message (in columns 1 to 35).

Users need to be careful not to add or delete any records from the error file. Records can only be changed. If GEMCOS/MCSERROR is not present when MCSTCL is executed, the standard output messages are used.

EXECUTION USING A CARD DECK

Once the system programs and files are loaded, read in the TCL source image (either a card deck or CANDE source file.) The card deck is constructed as follows:

```
?EX MCSTCL
?DATA MCSIN
      .
      .
      .
<Deck description>
      .
      .
      .
?END
```

EXECUTION USING A CANDE SOURCE FILE

Alternatively, users can create and maintain a TCL source file with CANDE instead of a card file. The TCL source file is named MCSIN <user's CANDE file name>. The CANDE default file type should be used. To execute the TCL compiler with a CANDE source file, enter the following:

```
EX MCSTCL FILE MCSIN NAME <user's CANDE file name>
   DSK DEF;
```

CHECKING FOR SYNTAX ERRORS

As soon as compilation begins, the compiler (MCSTCL) reads MCSIN (the TCL source image) and then writes MCSLST to a line printer. MCSLST lists the TCL source deck (or source file) and gives any syntax errors. Another listing, called MCSERRLST, prints out any errors or warnings separately.

Users can decide whether to print the entire listing, or just the syntax errors. They do this by specifying "$LIST" or "$NO LIST" in column 1 of a source record. If "$LIST" is specified, MCSLST is printed and both the source record and errors are listed. If "$NO LIST" is specified, only MCSERRLST prints, which gives any errors or warnings. The default setting is $LIST.

The user can use MCSLST and MCSERRLST to check and correct any syntax errors in the TCL source image.

SAMPLE TCL SOURCE IMAGE (DECK OR FILE)

The following is a sample TCL source image used to create an MCS when the TCL is compiled.

```
?EX MCSTCL
?DATA MCSIN
  CONTROL = GENERATE, LIST, COMPILE.
  GLOBAL:
    MONITORTRACE = TRUE.
    NCCOKRESPONSE = "$OK$".
    STATUSREPORTS = TRUE.
  BEGIN
  PROGRAM PAYROLL USER:
    TITLE = PAYROLL.
    TRANCODE = UPDATE(1,1).
    TRANCODE = INQ(1,2).
  PROGRAM INVENTORY USER:
    TITLE = INVNT.
    TRANCODE = RCV (2,1), SHIP (2,2).
  PROGRAM GAME UTILITY:
    TITLE = MAZE/GAME.
    INTERFACE = NONPARTICIPATION.
  STATION TD800A:
  CONTROLSTATION = TRUE.
  MONITORSTATION = TRUE.
  STATION TD800B:
  STATION TD800C:
  STATION TD700A:
  STATIC DECLARATIONS:
    RECORD 01 MESS_STRUCTURE        CHARACTER(5),
    02 MESS_ITEM_1                  CHARACTER(3),
    02 MESS_ITEM_2                  CHARACTER(2);
  DECLARE ME                        MESS_STRUCTURE;
  ENDSOURCECODE.
  PROCEDURE SETVALUES:
    PROCEDURE MESS_SET_VALUES;%
         ME.MESS_ITEM_1 := "AAA";%
         ME.MESS_ITEM_2 := "BB";%
         END MESS_SET_VALUES;%
  ENDSOURCECODE.
  END.
?END.
```

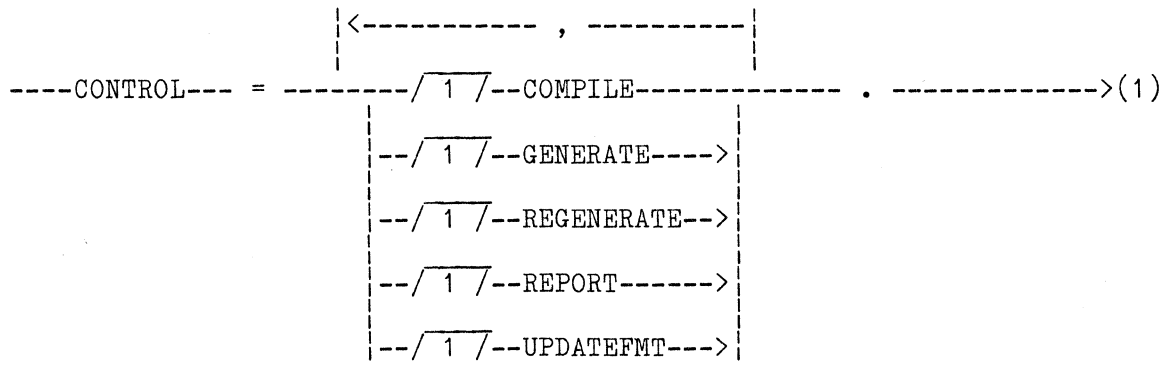
For information on how to execute the MCS created when the TCL is compiled, see the end of Section 2, following the individual TCL statements. Presentation of the individual TCL statements begins with a discussion of the Control statement.

CONTROL STATEMENT

The following diagram shows the syntax of the TCL source image (deck description). The CONTROL statement forms the first part of this source image.

Syntax:

<source image>



Semantics:

The CONTROL statement defines the task(s) to be performed during a run of MCSTCL. The control list defines the individual task or combination of tasks.

REPORT causes a hard-copy record of the user's data communication system description to be written to a line printer. The listing is labeled MCSRPT. If REPORT is the only option in the control list, the Global section and the Definition section are not required. The MCSTIC file, however, must be available to MCSTCL.

GENERATE causes MCSTCL to create a disk file labeled MCSTMP and ZIP MCSGO. MCSGO uses MCSTMP and MCSGTS to create MCSSRC, the user's MCS source-code file. In addition, when GENERATE appears in the

CONTROL statement, a disk file labeled MCSTIC is written. MCSTIC contains customized tables consisting of the user's data communication system network relationships. The MCSTIC file must be present when executing a B 1700 GEMCOS-generated MCS. When GENERATE is in the control list, both the Global section and Definition section must be present.

REGENERATE causes MCSTCL to create a new MCSTIC file from an old one. This option should be used if a station, transaction, program, or access key is to be added or changed. REGENERATE neither writes MCSCRD nor ZIP-executes MCSGO, thus saving machine time. When REGENERATE is in the control list, both the Global section and the Definition section must be present. If MCSTCL determines (while modifying an existing MCSTIC file) that the MCS code file is no longer compatible, it produces a syntax error and the regeneration does not occur. This happens when, for example, AUDIT was not specified in the original GENERATE run, but appears in the REGENERATE run.

NOTE

> During a REGENERATE run, the station network control information, which is used to bring stations back to their last running state, is not copied from the old MCSTIC file to the new one. Therefore, after a regeneration, stations in the network have those attributes specified in the NDL which do not reflect the accumulated changes caused by GEMCOS Network Control Commands. Moreover, the audit file number is reset to zero; all existing audit files are no longer valid.

COMPILE causes MCSTCL to instruct MCSGO to ZIP-execute the UPL compiler to create MCSSRC/object from MCSSRC. If COMPILE appears in the control list, GENERATE must also appear.

UPDATEFMT facilitates recompilation of the TCL Format section without requiring generation or regeneration. The Format section can be recompiled while the MCS is operating and without affecting the MCSTIC file. Only previously compiled functions and formats can be modified. The recompiled functions and formats are copied into the format file, MCSFORMATS. Programs and stations have access to the new copy of the format through the *UPD network control command, entered from the control station or the SPO.

Examples:

```
CONTROL = REPORT.
CONTROL = REGENERATE, REPORT.
CONTROL = GENERATE, LIST, COMPILE.
CONTROL = UPDATEFMT.
```

Figures 2-1 through 2-3 illustrate which files are created and accessed by the TCL compiler (MCSTCL) when the previously listed sample CONTROL statements are present.

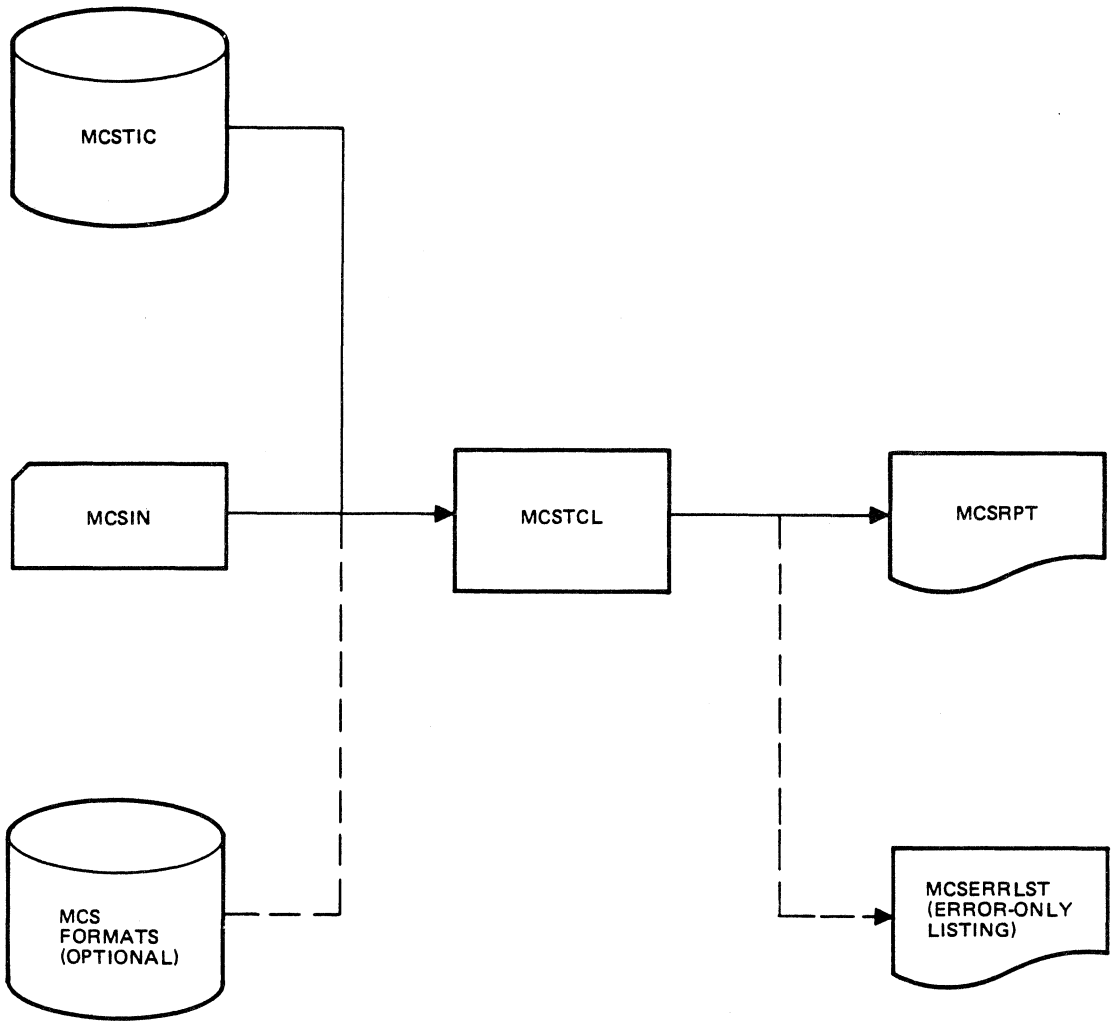


Figure 2-1. Files Created by TCL Compiler When CONTROL = REPORT

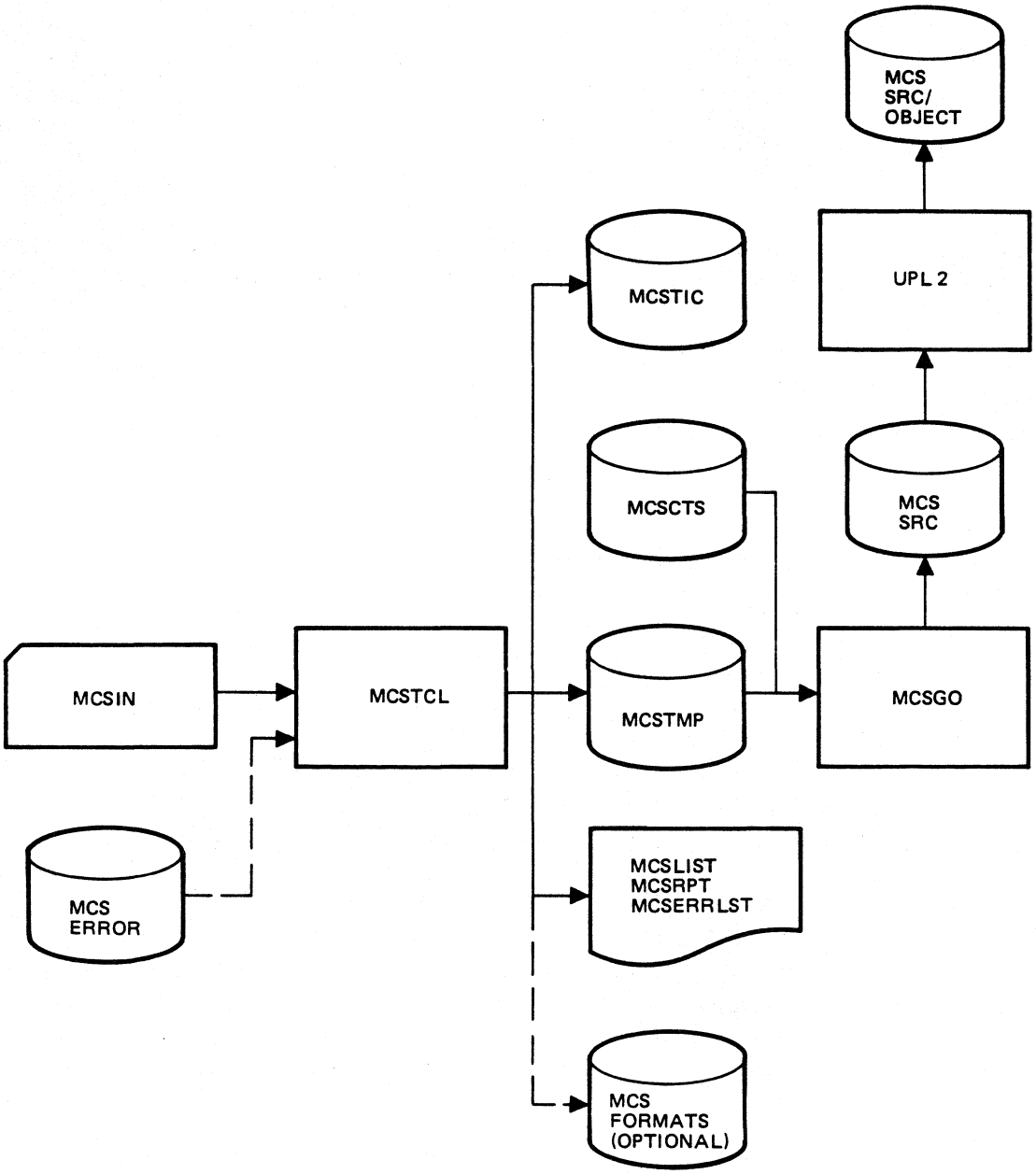


Figure 2-2. Files Created by TCL Compiler When CONTROL = GENERATE, REPORT, or COMPILE

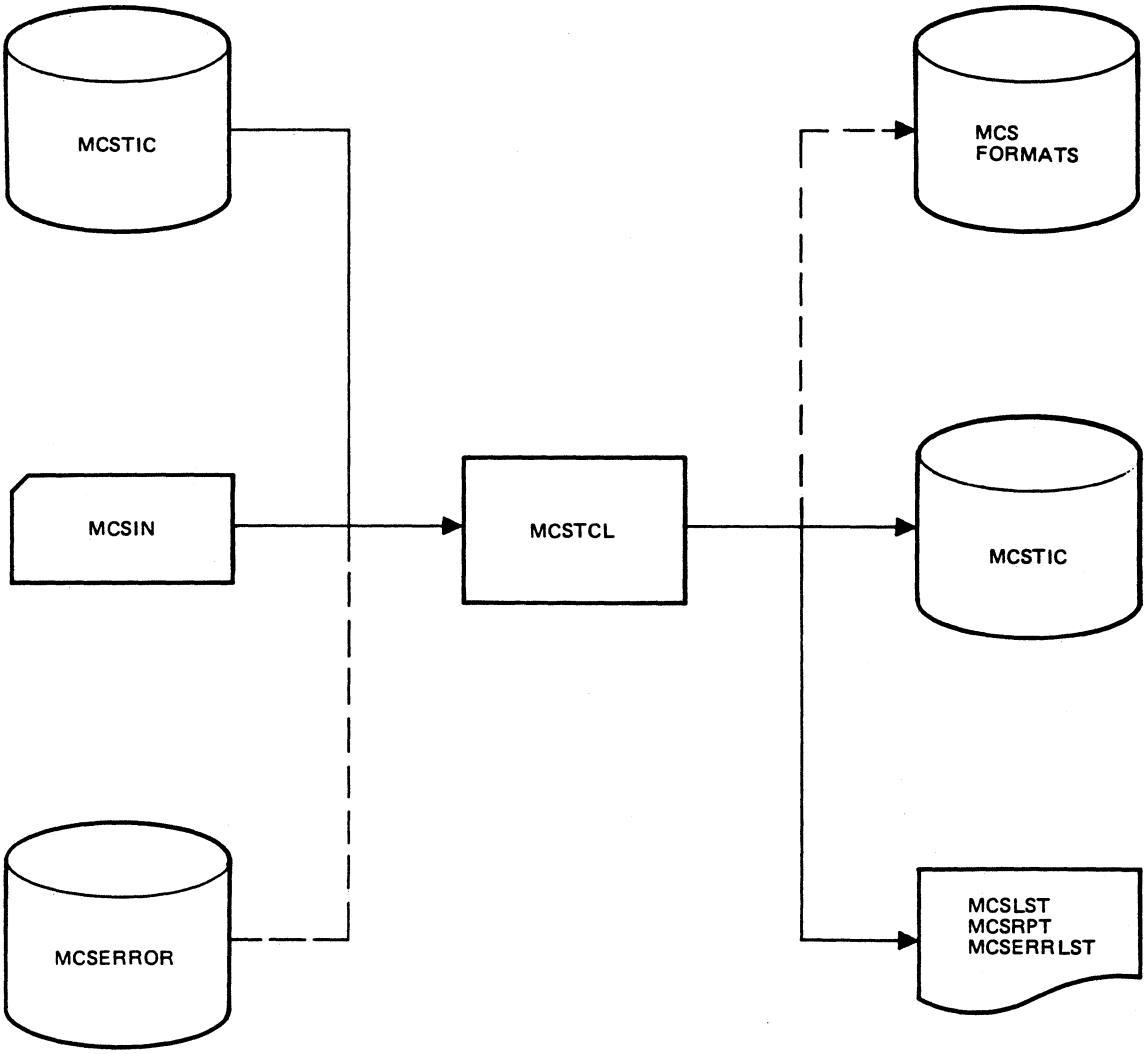
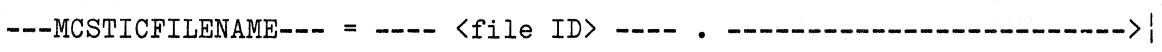


Figure 2-3. Files Created by TCL Compiler When CONTROL = REGENERATE or REPORT

MCSTIC FILE NAME STATEMENT

Syntax:

<MCSTIC FILE NAME statement>



Semantics:

The MCSTIC FILE NAME statement allows for the specification of the MCSTIC file name. The MCSTIC FILE NAME statement, if present, must appear after the CONTROL statement and before the Global section. <File ID> is a B 1000 file identifier. By default, MCSTICFILENAME is MCSTIC.

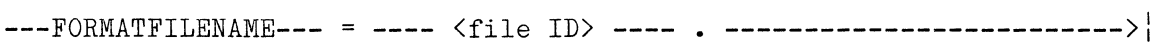
Examples:

```
MCSTICFILENAME = MYMCSTIC.
MCSTICFILENAME = TEST/MYMCSTIC.
MCSTICFILENAME = PACKB/TEST/MYMCSTIC.
```

FORMAT FILE NAME STATEMENT

Syntax:

<FORMAT FILE NAME statement>



Semantics:

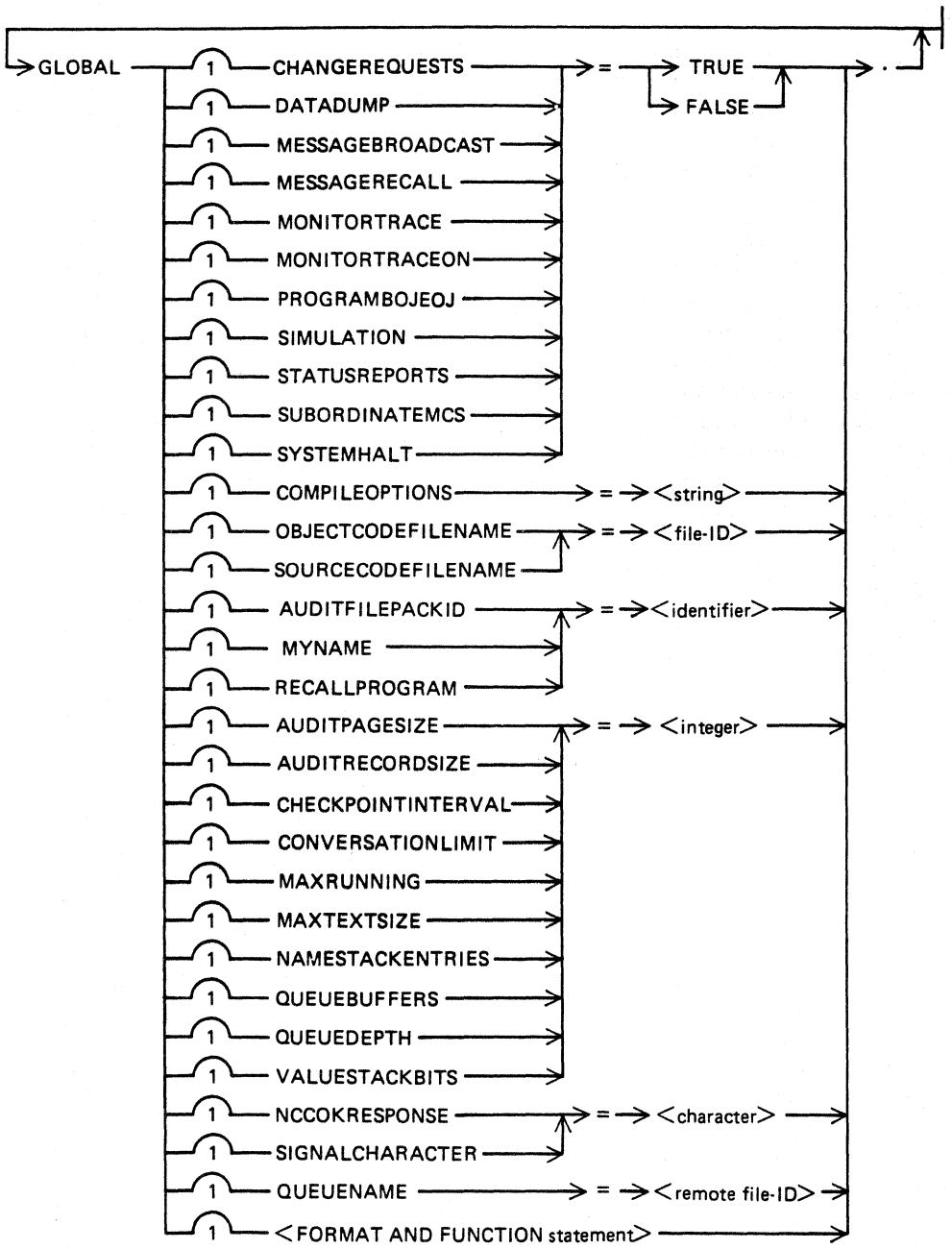
The FORMAT FILE NAME statement is used to change the name of the MCSFORMATS file. This statement only functions in the Advanced and Total Versions of GEMCOS. It must immediately follow the MCSTIC FILE NAME statement and precede the Global section. <File ID> is a B 1000 file identifier. By default, FORMATFILENAME is GEMCOS/MCSFORMATS.

Examples:

```
FORMATFILENAME = ALLFORMATS.
FORMATFILENAME = TEST/FORMATS.
FORMATFILENAME = GEMPAC/LIVE/FORMATS.
```

GLOBAL SECTION

See the following syntax diagram for an explanation of the Global Section. Please refer to this diagram when reading the succeeding statements, which are part of the Global section.



Semantics:

The Global section is composed of two types of GLOBAL statements: CODE GENERATION statements and MISCELLANEOUS PARAMETER statements. Any given GLOBAL statement may occur only once in the Global section, except in the FORMAT and FUNCTION statements.

There are two types of CODE GENERATION statements. The first type of CODE GENERATION statement causes optional MCS intrinsics to be generated into the MCS source file. This type can take on a TRUE or FALSE value.

These optional MCS intrinsics include code to support the following:

1. Change commands.
2. The data dump command.
3. Message broadcast.
4. Message recall.
5. Program control commands.
6. Monitor trace.
7. Status commands.
8. System shutdown.
9. Audit.
10. Output audit.
11. Queue restoration.

The second type of CODE GENERATION statement controls the names of GEMCOS files, UPL2 compiler options, and object code memory size requirements.

It is important to note that both types of CODE GENERATION statements directly affect the MCS source and/or object code files. Therefore, if a CODE GENERATION statement is modified, GENERATE and COMPILE should appear in the CONTROL statement since new source and object code files are required. Otherwise, MCSTCL detects an object code file, MCSTIC file incompatibility error.

MISCELLANEOUS PARAMETER statements specify various attributes of a running GEMCOS MCS, such as the signal character or Network Control Command response. Except for the AUDIT PAGE SIZE statement and the AUDIT RECORD SIZE statement, MISCELLANEOUS PARAMETER statements may be safely changed in REGENERATE MCSTCL runs.

Example:

```
GLOBAL:
  PROGRAMBOJEOJ = TRUE.
  MONITORTRACE = FALSE.
  COMPILEOPTIONS = "LIST SUMMARY".
  QUEUEBUFFERS = 3.
  DATADUMP = TRUE.
```

AUDIT FILE FAMILY ID STATEMENT

Syntax:

<AUDIT FILE FAMILY ID statement>

Semantics:

The AUDIT FILE FAMILY ID statement allows MCS audit files to contain a user-specified family portion of the file name (multi-file-ID). This option is particularly useful whenever more than one GEMCOS MCS performing audit and recovery are run on the same site. The length of the identifier must be 10 characters or less. By default, the multi-file-ID portion of the audit file is MCSAUDIT.

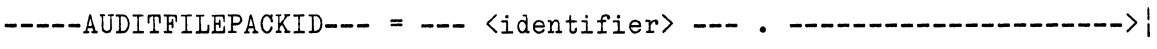
Examples:

```
AUDITFILEFAMILYID = A.
AUDITFILEFAMILYID = DEMOAUDIT.
```

AUDIT FILE PACK ID STATEMENT

Syntax:

<AUDIT FILE PACK ID statement>



Semantics:

The AUDIT FILE PACK ID statement allows MCS audit files to reside on other than the system pack. It is recommended that audit files reside on a user pack to increase throughput and decrease the time spent in audit and recovery. The identifier must be 10 characters or less in length. By default, audit files reside on the system pack.

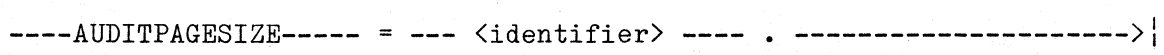
Examples:

```
AUDITFILEPACKID = MCSPACK.
AUDITFILEPACKID = AUDITPACK.
```

AUDIT PAGE SIZE STATEMENT

Syntax:

<AUDIT PAGE SIZE STATEMENT>



Semantics:

The AUDIT PAGE SIZE statement controls the size of the audit files by specifying the number of records in each page (that is, the area). There are always 40 pages. By default, AUDITPAGESIZE equals 1000.

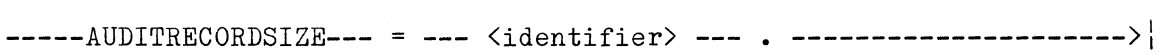
Examples:

```
AUDITPAGESIZE = 500.
AUDITPAGESIZE = 2000.
```

AUDIT RECORD SIZE STATEMENT

Syntax:

<AUDIT RECORD SIZE statement>



Semantics:

The AUDIT RECORD SIZE statement controls the size of the audit record by specifying the number of bytes in each record. Increments of 180 are the only allowable values. When a value other than an increment of 180 is specified, a warning is issued and the next highest increment of 180 is selected. By default, AUDITRECORDSIZE equals 180.

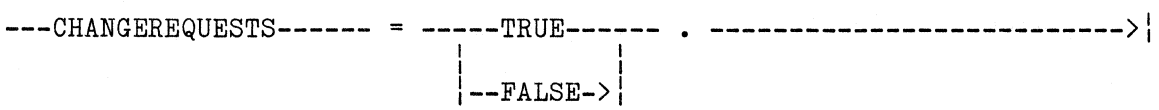
Examples:

```
AUDITRECORDSIZE = 180.
AUDITRECORDSIZE = 540.
```

CHANGE REQUESTS STATEMENT

Syntax:

<CHANGE REQUESTS statement>



Semantics:

The CHANGE REQUESTS statement determines whether the GEMCOS MCS is to contain the logic to support the following seven Network Control Command change requests:

1. CHANGE STATION ADDRESS (CSA).
2. CHANGE STATION DIAGNOSTIC (CSD).
3. CHANGE STATION FREQUENCY (CSF).
4. CHANGE STATION MAXIMUM RETRY (CSM).
5. CHANGE STATION QUEUE (CSQ).
6. CHANGE STATION READY (CSR).
7. CHANGE STATION TRANSMISSION NUMBER (CST).

When MONITORTRACE equals TRUE, the CHANGE MONITOR FLAG (CMF) command becomes the eighth change request, and CHANGEREQUEST code will be generated automatically (for internal use). However, users will not be able to access the seven Network Control Command change requests unless they set CHANGEREQUESTS to TRUE in the TCL.

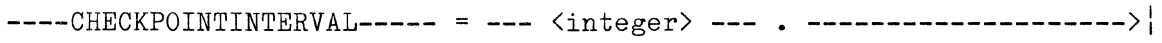
Example:

```
CHANGEREQUESTS = TRUE.
```

CHECKPOINT INTERVAL STATEMENT

Syntax:

<CHECK POINT INTERVAL statement>



Semantics:

The CHECKPOINT INTERVAL statement determines the length of time between checkpoints taken by the MCS during auditing. Specifying too small a number causes the MCS to do an excessive number of I/Os, thereby reducing throughput. By default, CHECKPOINTINTERVAL equals 60 (seconds).

Examples:

```
CHECKPOINTINTERVAL = 30.
CHECKPOINTINTERVAL = 90.
```

COMPILE OPTIONS STATEMENT

Syntax:

<COMPILE OPTIONS statement>

Semantics:

The COMPILE OPTIONS statement allows for the specification of UPL compiler control statements when COMPILE appears in the CONTROL statement (refer to DOCUMENT/SDL2 on the 6.00 or later GEMCOS release tape for a complete description of available options). <String> must begin and end with a quote and must not exceed 65 characters. By default COMPILEOPTIONS is set to NO LIST SUMMARY.

Examples:

```
COMPILEOPTIONS = "LIST CODE".
COMPILEOPTIONS = "LIST XREF".
```

CONVERSATION LIMIT STATEMENT

Syntax:

<CONVERSATION LIMIT statement>

```
---CONVERSATIONLIMIT---- = --- <integer> --- . ---------------------->
```

Semantics:

The CONVERSATION LIMIT statement allows the user to specify the maximum number of stations that may converse concurrently. The integer specified must not exceed the number of stations declared in the TCL. The maximum limit allowed by GEMCOS is 64. When there are no CONVERSATION SIZE statements declared for programs in the TCL, the default value is zero, that is, no conversation capability exists in the MCS. When conversational programs are present, the default value is the number of stations declared in the TCL.

This statement establishes the number of reserved conversation areas. The number of areas is reserved by powers of 2. When the limit is declared, the nearest 2 to the nth power that is greater than or equal to the limit is the number of areas reserved. Even if the reserved area is larger than the limit, the maximum number of concurrent conversations may not exceed the specified limit. If the limit needs to be increased and the new limit exceeds the number of reserved areas, a GENERATE and re-COMPILE would be required.

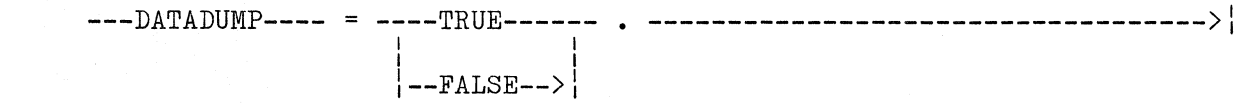
Examples:

```
CONVERSATIONLIMIT = 8.
CONVERSATIONLIMIT = 5.
```

DATA DUMP STATEMENT

Syntax:

<DATA DUMP statement>



Semantics:

The DATA DUMP statement indicates whether the code to create a dump of internal MCS variables is present. When DATADUMP equals TRUE, the REPORT DATA DUMP (RDM) command is recognized. By default, DATADUMP equals FALSE.

Example:

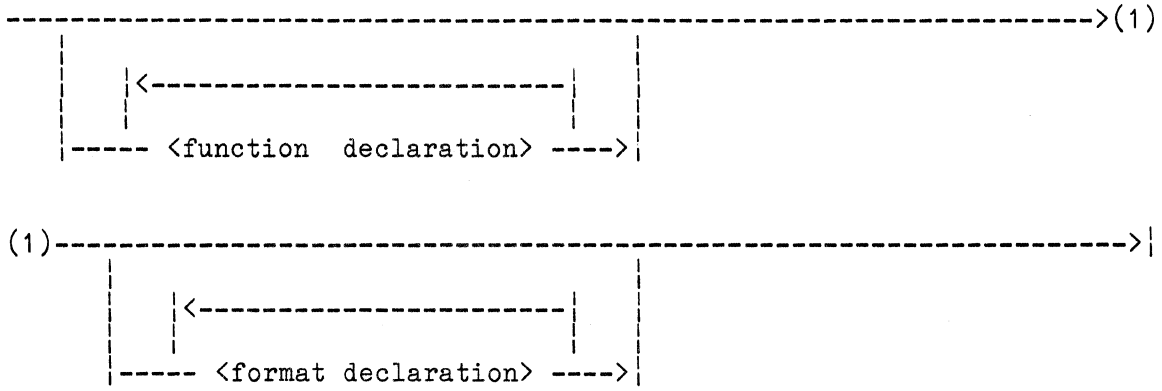
```
DATADUMP = FALSE.
```

FORMAT AND FUNCTION STATEMENT LIST

Syntax:

The following diagram shows the syntax of the FUNCTION and FORMAT statement.

<FUNCTION and FORMAT statement>



Semantics:

In addition to the functional capabilities of the Basic version of B 1000 GEMCOS, the Advanced version includes a Message Formatting module. The Message Formatting module can be used to support forms requests, modify the text of messages, and/or ensure application program device independence. Users of the Basic version will find that an attempt to invoke the formatting capabilities of GEMCOS results in a syntax error.

The Forms Request function provides station operators with the ability to enter a message-ID (refer to Device section below) and to receive in return a formatted screen with blank data fields. Application programs may also invoke the Forms Request function, causing formatted screens with blank data fields to be displayed at stations in the network.

The text of messages entered at stations can be modified, re-arranged and/or supplemented prior to being routed to the appropriate application program. This process is referred to as input formatting. The text of messages written by application programs can be modified, re-arranged and/or placed into data fields of formatted screens before being sent to stations; this is referred to as output formatting.

When a network is comprised of two or more types of terminal devices, the stations may be grouped into several device classifications in the Device section. A set of formats is defined for each device classification. When invoked, the formatting module recognizes the device classification of the station involved, and applies a format from the set associated with that classification. As a result, messages sent or received by application programs can have a standard record layout regardless of the device type of the destination/source station. Moreover, the application program does not need to be affected by the different control characteristics of different devices.

There are two areas of the TCL which relate to formatting: the Device section and the FORMAT AND FUNCTION statement list. The Device section is used to identify which messages are to be formatted and with which formats. The FORMAT AND FUNCTION statement list is used to define formats and functions. A format specifies how a screen is to be built and/or how the message text is to be modified. A function defines a translate table which can be referred to by a format.

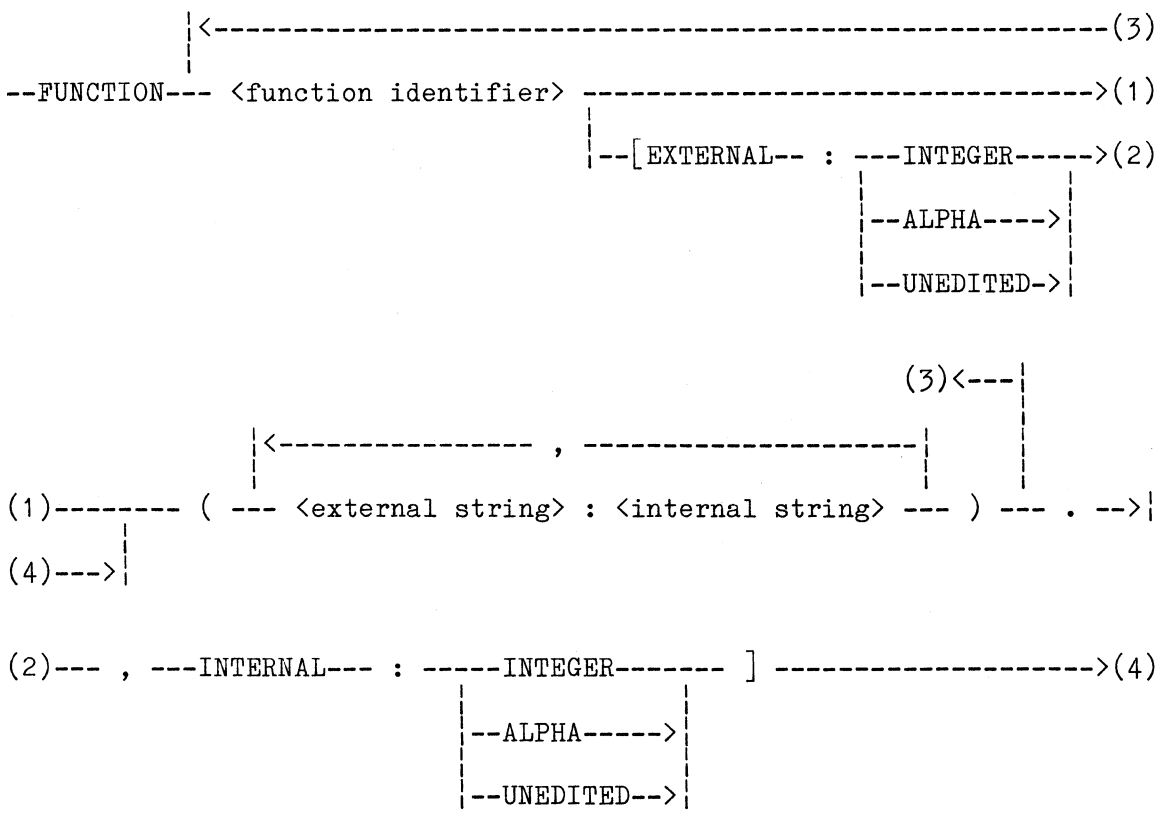
The FORMAT AND FUNCTION statement list is composed of a function declaration list, which can be empty, followed by a format declaration list.

Function Declaration

Syntax:

The following diagram shows the syntax of the Function Declaration.

<Function Declaration>



Semantics:

The function declaration defines functions which can be used in a translate item phrase of a format declaration. A function identifier is required as the first argument of the translate item phrase. The translate item phrase allows a format to translate a string of length n into a string of length m where 0<n<7 and 0<m<7. <String> is therefore limited to a maximum of six characters. Up to 1023 functions may be declared.

A translate pair associates an external string with an internal string. On input, an external string is translated into the associated internal string. On output, an internal string is translated into the associated external string. When an application program deals with the text of a message, it must use an internal string in a translate field. When an operator deals with the text of a message at a station, an external string is used.

Refer to "Format Declaration" for examples of functions used in formats.

The justification and fill part is described in the following example:

Example:

```
FUNCTION GENDER ("MALE":"1", "FEMALE":"2").
FUNCTION DIGITIN ("1":"ONE", "2":"TWO", "3":"THREE"),
DIGITOUT [EXTERNAL:ALPHA, INTERNAL:INTEGER]
    ("ONE":"1", "TWO":"2", "THREE":"3").
```

As the translate module searches for a match between the source text to be translated and an internal string/external string, both the source text and the internal string/external string are placed into character strings of length six for comparison. The justification and fill part enables the user to control the placement of the source text and the internal string/external string into these character strings. If the justification and fill part is empty, it is assumed that both the external string and internal string are unedited. By using the justification and fill part, the user may make either of these strings UNEDITED, INTEGER, or ALPHA.

An unedited string of less than six characters in length is right-justified within a 6-character string with leading nulls (4"00"). A null compared with any character is always considered a true comparison by the translate function.

An integer string of less than six characters is right-justified with leading zeroes. An alpha string of less than six characters is left-justified with trailing blanks.

If, within a given function, the length of each internal string is the same and the length of each external string is the same, it makes little difference whether the strings are unedited, integer, or alpha. However, if strings vary in length, using integer or alpha strings can help to avoid confusion. For example, suppose a function is declared as follows:

```
FUNCTION TEST ("11":"SOME", "1":"ME").
```

Upon input, if the translate function were to search for an external string of 1, it would get a match with 11 because of a NNNNN1. The source text after justification will compare as equal to NNNN11, the external string after justification (where N is a null). A similar phenomenon would occur on output if the translate function was searching for an internal string of ME: NNNNME would match NNSOME. This problem could be avoided by declaring the function as follows:

```
FUNCTION TEST [EXTERNAL:INTEGER,INTERNAL:ALPHA]
              ("11":"SOME", "1":"ME").
```

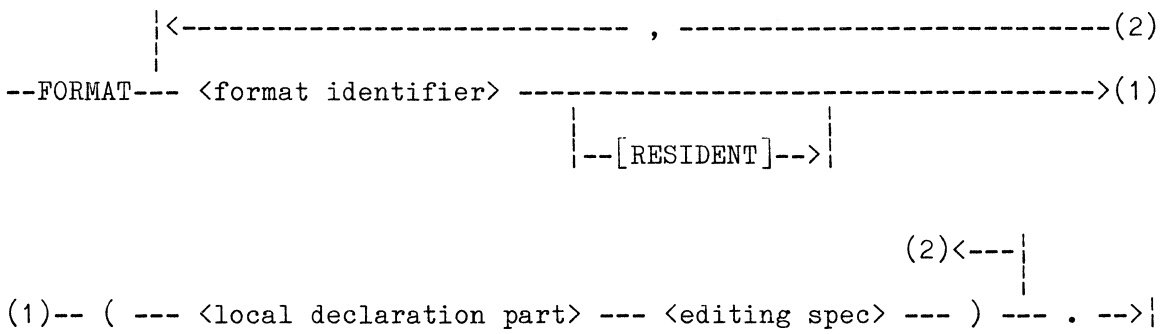
With this declaration, if the source text to be translated on input were "1", it would be converted to 000001. It would not match 000011, but would successfully match 1 justified as an integer. Likewise, source text on output of ME would be converted to MEBBB (where B is a blank); it would not match SOMEBB, but would match ME justified as an alpha string.

Format Declaration

Syntax:

The following diagram shows the syntax of the Format Declaration. The syntax of the Local Declaration Part, the Editing Specification, Editing String, and other components are shown after the format declaration.

<format declaration>



Semantics:

The format declaration is used to define how a screen is to be built and/or how message text is to be modified. When formats are declared in the format declaration, the Device section is used to

indicate which formats are to be applied to which messages. Up to 1023 formats may be declared.

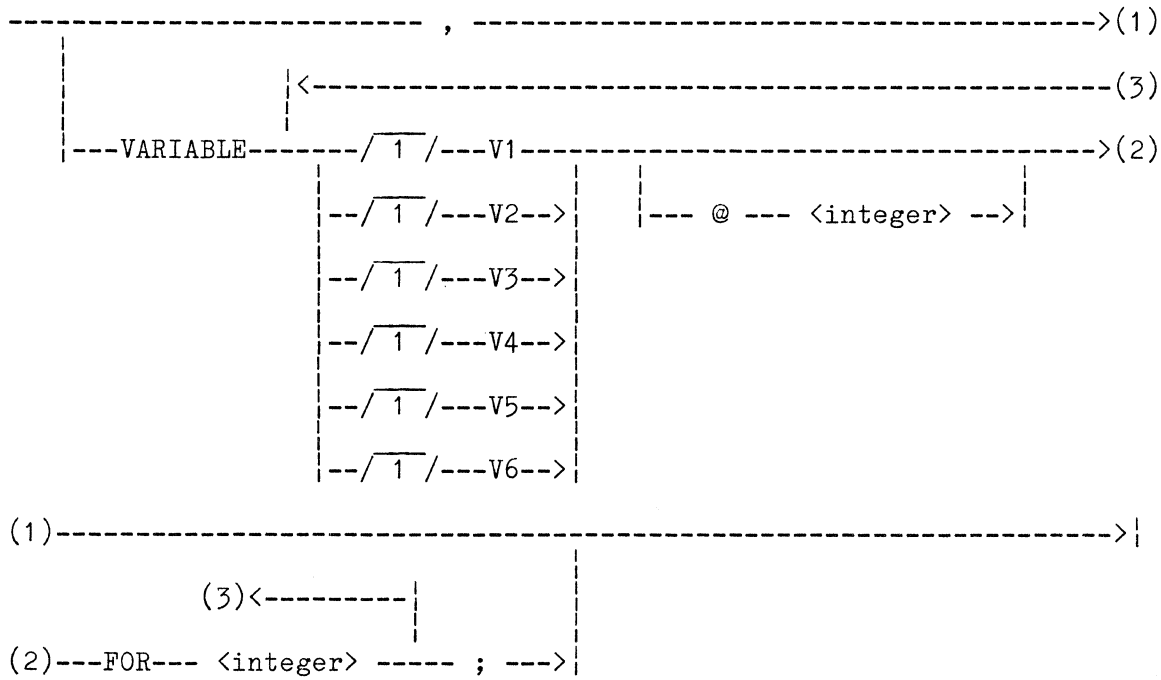
The format part list allows several formats, separated by commas, to be described in a single format declaration. Even though the syntax allows several formats to be described in one format declaration, it is good practice to define one format per format declaration. When a syntax error is encountered in a format part, the TCL scanner skips past any remaining format parts to the next format declaration. Syntax errors in the skipped format parts are not flagged until the format part in error is corrected. If one format is defined per format declaration, more syntax errors can be caught in each run of the TCL compiler.

Each format part associates a format identifier with a particular set of message formatting instructions. The format identifier is referenced in a FORMATSIN statement and/or a FORMATSOUT statement of the Device section.

The special action part, if present, indicates whether the format is a resident format. A resident format is kept in an array in memory instead of on disk. This facility is provided for small, frequently used formats. It is intended to save the input/output overhead that would otherwise be required to retrieve a format from disk before using it. This option should be used with care since its overuse could require significant amounts of memory.

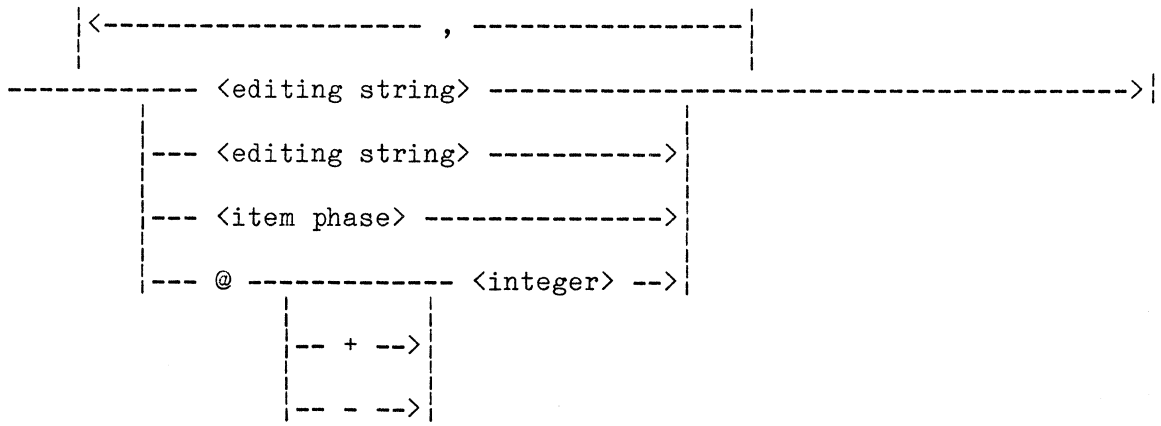
The format description consists of an optional local declaration part and the editing specifications enclosed within parentheses. (Readers unfamiliar with GEMCOS formatting should refer to Section 4 of this manual or to the GEMCOS Formatting Guide.) The following diagram shows the syntax of the local declaration part.

<local declaration part>



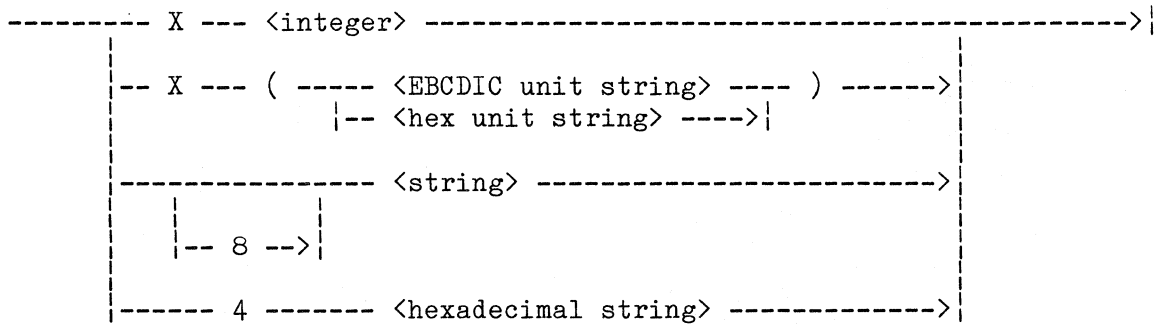
The editing specifications describe the order and length of the fields of a message as well as the manipulation of the message buffer pointers. The editing specifications are a list of editing phrases. An editing phrase can be an editing string, a location specifier, or an item phrase. The syntax of the editing specifications follows.

<editing specifications>



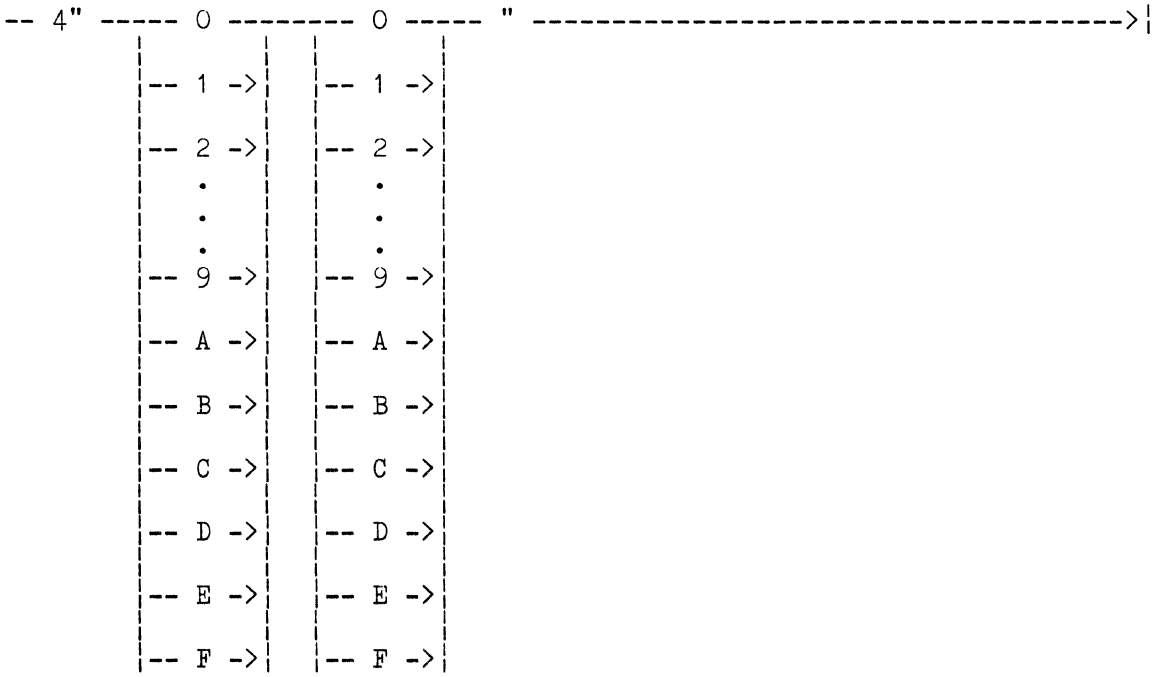
An editing string is either a simple string or a skip field. The simple string is used to place a literal field into a formatted message. A simple string can be an EBCDIC string such as "XYZ" or a hexadecimal string such as 4"OD" (carriage return). The simple string is used extensively when building forms for screen devices. It can be used to create the descriptive text of the protected areas as well as the necessary control characters. A simple string causes the pointer into the formatted message buffer to be updated to the right by the length of the string. The pointer into the source message buffer is unaffected by a simple string editing phrase. The following diagram shows the syntax of the editing string.

<editing string>

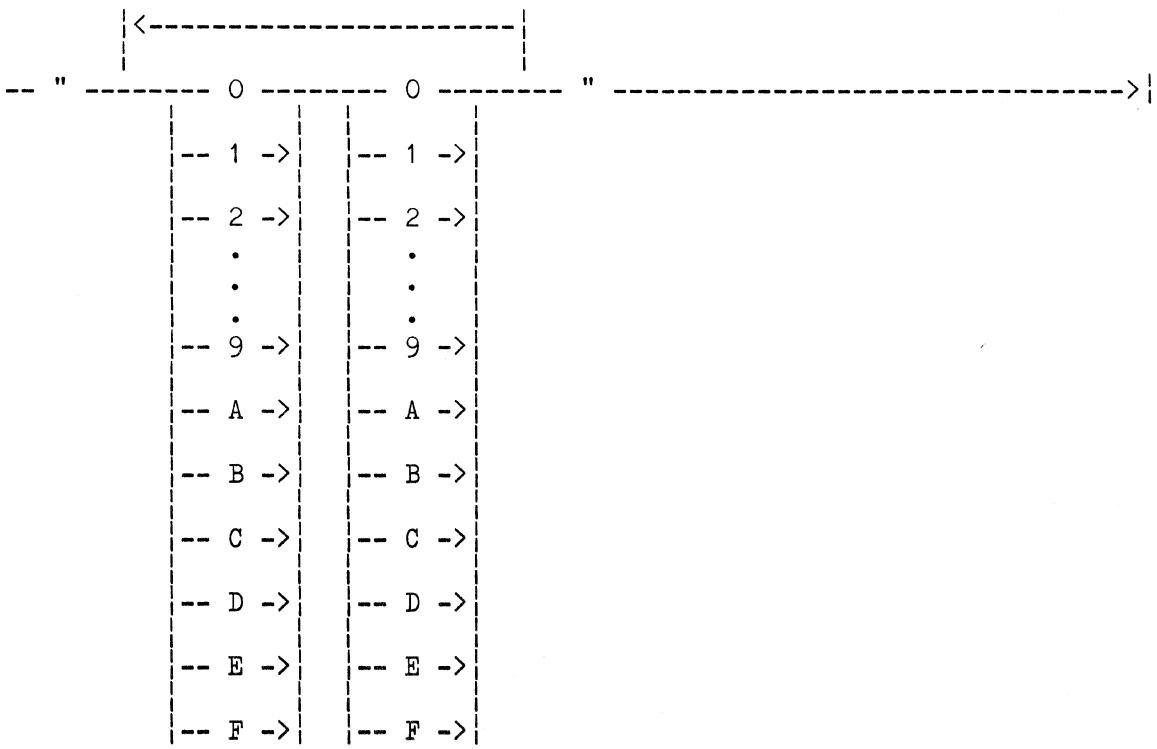


The following diagrams show the syntax of a hex unit string and a hexidecimal string.

<hex unit string>



<hexadecimal string>



Upon input, the skip field causes text in the terminal message buffer to be skipped (by updating the terminal message buffer pointer). The number of characters skipped can be defined by an integer or a delimiter. For example, X3 causes three characters to be skipped while X(",") causes text up to and including the next comma encountered to be skipped.

Upon output, X8 causes eight spaces to be placed into the terminal message buffer while updating the pointer. X(<delimiter>) is undefined for output messages. The program message buffer pointer is unaffected by a skip field.

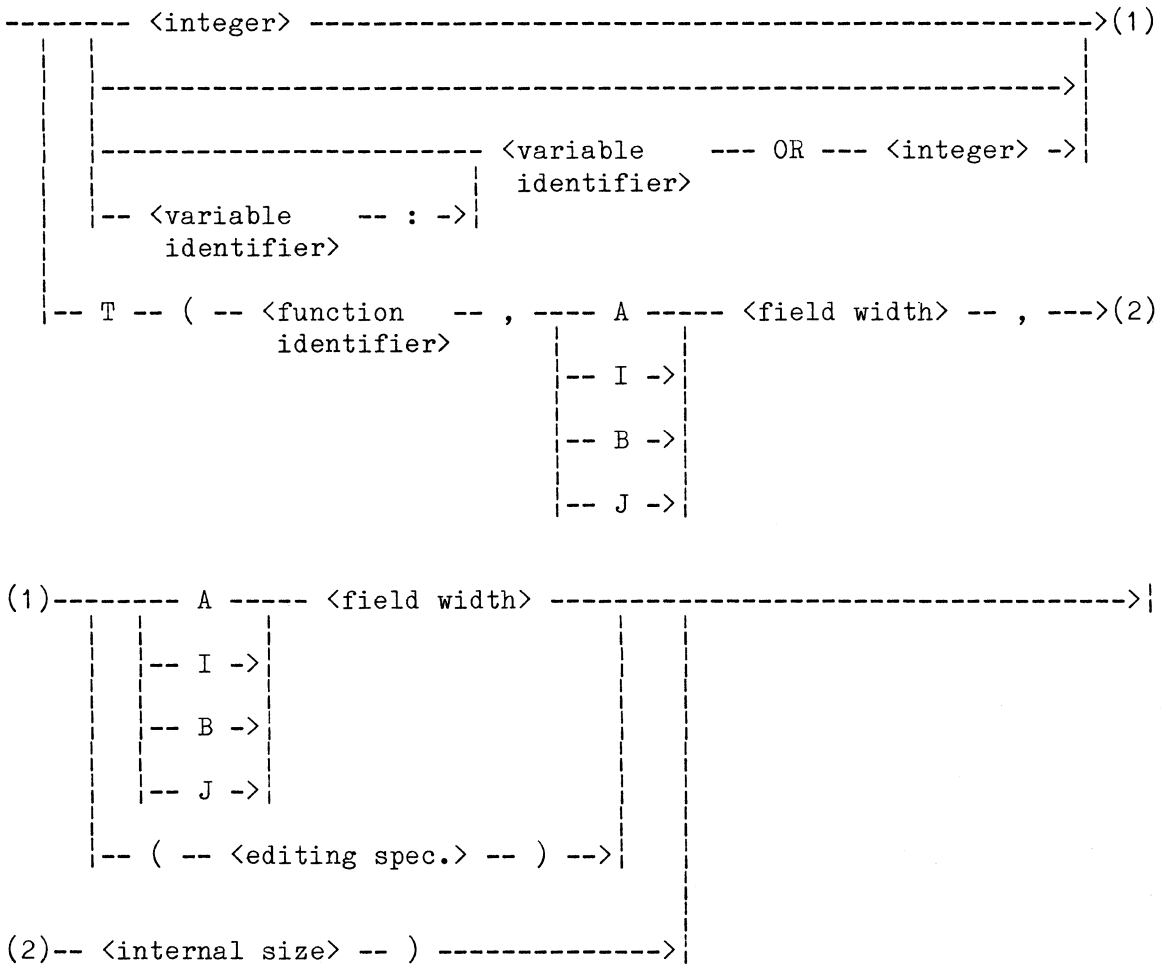
The location specifier is used to manipulate the program message-buffer pointer without affecting the terminal message-buffer pointer. By manipulating the program message-buffer pointer, fields can be skipped, re-ordered and/or re-used.

There are two variations of the location specifier distinguished by the existence of an optional sign. When a sign is present, the program message-buffer pointer is adjusted by <integer> positions to the left (sign is a "-") or to the right (sign is a "+"). When there is no sign, the program message-buffer pointer is set to position <integer>. Care must be taken to keep the pointer within the bounds of the program message buffer. Upon input, the user should also be careful not to overlay good data in the program message buffer.

For more information, refer to "Using Location Specifiers" later in this section.

An item phrase defines a field of a formatted message. A field can be comparatively simple such as six alphanumeric characters, or rather complex, such as a repetition of several variable-length subfields. In order to encompass the wide variety of possible fields, several forms of the item phrase are available. All involve at least one item type, field width pair. The following diagram shows the syntax of the an item phrase.

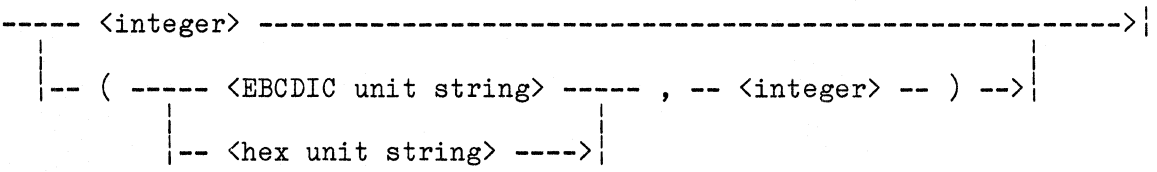
<item phrase>



The item type determines how a field or subfield is to be edited. Four item types are available: A, B, I, and J. A denotes an alphanumeric field, and B specifies a tabbed alphanumeric field. Alphanumeric fields may contain any characters, and leading blanks are considered significant. Truncation or blank filling occurs on the right. I denotes an integer field, while J specifies a tabbed integer field. Integer fields may only contain digits and/or blanks except for imbedded blanks. They are truncated or right-justified with zero filling on the left.

The field width determines the length of a field or subfield. Fields can be fixed or variable in length. The following diagram shows the syntax of the field width.

<field width>



The simplest form of the item phrase is an alphanumeric or integer field with an <integer> <field width> such as A6 or I9. An A6 item phrase results in the move of six characters from the buffer containing the raw message to the formatted message buffer. An item phrase of I9 would move nine characters subject to the editing rules already mentioned. The unprotected areas of formatted screens are usually composed of fixed alphanumeric or integer fields.

A more powerful form of the item phrase employs a <variable field specifier> <field width> such as A("*",6) or I("+",8). The internal size determines the size of the field in the program message buffer.

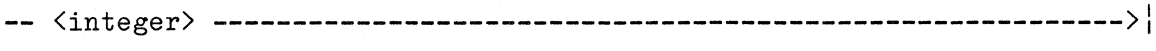
NOTE

> While field lengths of the terminal message buffer may vary, field lengths of the program message buffer are always fixed. The delimiter is used to signify the end of the field in the terminal message buffer. The field begins where the previous field ends.

Upon input, a variable-length field is isolated based on the end of the last field and the delimiter. It is moved into the program message buffer and justified according to the item type. The delimiter is not considered one of the characters of the field and, therefore, is not placed into the program message buffer.

Upon output, a string of characters of length <internal size> is obtained from the program message buffer. It is compressed by truncating trailing blanks or leading zeroes, depending on the item type. The compressed string is placed into the terminal message buffer, and the delimiter is inserted after the compressed string. The following diagram shows the syntax of length <internal size>.

<internal size>



During both input and output, the terminal message buffer is updated to the position following the delimiter, while the program message buffer is moved to the right by <internal size> positions.

Tabbed fields, where the item type is B or J, are similar to variable-length fields on input and the same as fixed fields on output. On input, a tabbed field can end early if the tab character (4"05") is encountered. However, unlike a variable field, where the delimiter must be present, the tab character is not required to end the field. If enough characters are found, the field ends automatically.

For example, a B10 item phrase on input causes characters to be moved from the terminal message buffer to the program message buffer until either 10 characters have been moved, or a tab character is encountered. The program message-buffer pointer is moved 10 characters to the right. The terminal message-buffer pointer is left pointing to the eleventh character, or to the character following the tab, whichever happens first. If the transfer is terminated by a tab character, trailing blanks are placed in the program message buffer to fill out all 10 character positions. The tab character is not placed into the program message buffer.

On output, B5 would achieve exactly the same results as A5, and J7 the same as I7. The tab character is not placed into the terminal message buffer, as is done with the delimiter of a variable-length, nontabbed field.

The default tab character (4"05") can be changed by using a variable field specifier along with the B or J item type. J ("*",5) is the same as J5 except that "*" is the tab character instead of 4"05". B(4"05",10) is identical to B10.

Each item phrase discussed thus far may be repeated by placing a repeat part in front of the item type. A repeat part may be fixed or variable.

A fixed repeat part is designated by an integer. It is a shorthand method of representing an editing phrase list where each editing phrase is identical. For example, 2A6 is the same as A6,A6.

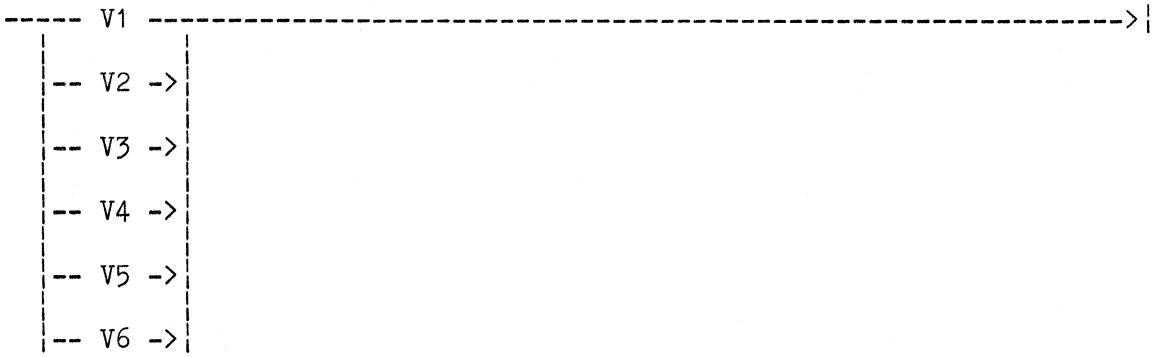
A variable repeat part can only be used on output. It is useful for messages which have a variable number of fields of repeated data, such as tables with columns of values. These messages must have, as one of the data fields, a counter specifying the number of times a particular field will occur.

If a message is to contain a variable repeat part, the format applied to the message must have a local declaration part. The local declaration part specifies where in the message the counters governing the occurrence of the repeated fields are to be found.

Values for variables declared are the first items extracted from the program message buffer. During each variable assignment, the program message-buffer pointer is adjusted by a combination of the optional location specifier and the length of the counter field. The length of the counter field is determined by the integer following the keyword FOR. The value of the counter contained in the program message buffer must be expressed as EBCDIC digits with a value not greater than 255. As many as six variables can be declared per format.

After a local variable has been set to a value extracted from the program message buffer, it can be referred to as a variable identifier in a variable repeat part. A variable repeat part consists of an optional update variable, a variable identifier, the keyword OR and an integer. The object of the repeat part is repeated either the number of times referred to by <variable identifier> or <integer> times, whichever is less. If the update variable is present, its variable identifier is set to <variable identifier> minus the number of times the repeat object was repeated. For example, V2 or 8 would cause its object to be repeated V2 times, but not more than eight times. If V2 had a value of nine, V3:V2 OR 3A5 would cause A5 to be repeated three times and V3 would be set to 6. The original value of V3 is lost. If V2 had been zero, the A5 field would not occur and V3 would be set to zero. The syntax of a variable identifier follows.

<variable identifier>



An editing phrase list enclosed in parentheses is an even more complicated item phrase. This form can be thought of as a field composed of several subfields. An editing phrase list enclosed in parentheses can be the object of a repeat part. Editing phrase lists can be nested to 32 levels of parentheses.

Another complicated form of the item phrase, T(<function identifier>, <item type> <field width>, <internal size>), is a reference to a translate function. The function identifier refers

to a function which must have been defined in a function declaration. The <item type> <field width> describes a field in the terminal message buffer, while internal size describes a field in the program message buffer.

Formatting Errors

When an error is detected while formatting an input message, the MCS sets the Format Error field of the common-area header to a nonzero value as described following. The message is then sent to the application program for which it was bound.

When an error is detected while formatting an output message, the MCS message is still sent to the destination station but, in addition, an error message is sent to the control station specifying what type of error occurred.

Error Type	Description
1	Destination pointer out of bounds
2	Source pointer out of bounds
3	Nondigit in integer field
4	Missing skip delimiter
5	Attempt to use variable repeat on input
6	Missing delimiter or variable field too long
7	Invalid string in translate field

Only the first error encountered is reported; however, the MCS attempts to continue formatting a bad message. When a type-3 formatting error occurs, the nondigit is placed into the erroneous field. For type-6 errors, significant text may be truncated in an attempt to force excessive data into the program message buffer. Type-7 errors result in question marks being placed into the erroneous field. Results are undefined for the other types of errors.

Figures 2-4 through 2-8 list five graded examples (example sets 1 through 5) of three increasingly difficult formats applied to input messages and output messages.

Example set 5 (Figure 2-8) uses the following function declarations:

```
FUNCTION GENDER("  MALE":"1","FEMALE":"2").
FUNCTION NUM1("ONE":"1","TWO":"2","THREE":"3","FOUR":"4",
              "FIVE":"5","SIX":"6","SEVEN":"7","EIGHT":"8",
              "NINE":"9","TEN":"10","ELEVEN":"11",
              "TWELVE":"12").
FUNCTION NUM2 [EXTERNAL:ALPHA,INTERNAL:INTEGER]
             ("ONE":"1","TWO":"2","THREE":"3","FOUR":"4",
              "FIVE":"5","SIX":"6","SEVEN":"7","EIGHT":"8",
              "NINE":"9","TEN":"10","ELEVEN":"11",
              "TWELVE":"12").
FUNCTION DAY ("1":"SUN","2":"MON","3":"TUE","4":"WED",
              "5":"THU","6":"FRI","7":"SAT").
```

Input/ Output	Message as It Appears at the Terminal	<Editing specifications> Applied to Message In Transit	Message as It Appears to the User Program
Input/ Output	ABC1234XY	A3,I4,A2	ABC1234XY
Input	ABC 4XY	A3,I4,A2	ABC0004XY
Input	ABC 4 XY	A3,I4,A2	ABC0004XY
Input/ Output	ABC0004XY	A3,I4,A2	ABC0004XY
Input	AB 5678XY	A3,X4,A2	AB XY
Input	AB GGGGXY	A3,X4,A2	AB XY
Input/ Output	AB XY	A3,X4,A2	AB XY
Input	ABCDE	A2,"*",A3	AB*CDE
Input	AB*CDE	A2,"*",A3	AB**CD
Output	AB*CDE	A2,"*",A3	ABCDE
Input	RIGHT	A5,8"FACE"	RIGHTFACE
Output	RIGHTFACE	A5,8"FACE"	RIGHT
Output	NAME: [HARRY]D C2	"NAME: [",A5,"]",4"12"	HARRY
Output	Name: [ ]D C2	"NAME: [",A5,"]",4"12"	(forms request)

Figure 2-4. Example Set 1 - Formatting Specifications Applied to Input and Output Messages

Input/ Output	Message as It Appears at the Terminal	<Editing specifications> Applied to Message In Transit	Message as It Appears to the User Program
Input	1234XY	I6	1234XY (FMTERR set to 3)
Output	1234XY (control station notified of error)	I6	1234XY
Input/ Output	ABCDXYZ	@4,A4,@1,A3	XYZABCD
Input/ Output	ALPHA	@3,A5	ALPHA
Input/ Output	AB123XY456	A2,@5,I3,@3,A2,@8,I3	ABXY123456
Input/ Output	ABCD	2A2	ABCD
Input/ Output	AB12CD34	2(A2,I2)	AB12CD34
Output	01/28/52	I2,2("/",I2)	012852
Input	01/28/52	I2,2(X1,I2)	012852
Output	* AB CD EF	Variable V1 for 2; "*",V1 or 5(X2,A2)	03ABCDEF
Output	XX 1 2 3YY 4 5	Variable V1 @7 for 2, V2 @5 FOR 2; @1, A2,@9,V2:V1 or 3(X1,I1), @3,A2,@12,V2 or 3(X1,I1)	XXYY000512345

Figure 2-5. Example Set 2 - Formatting Specifications Applied to Input and Output Messages

Input/ Output	Message as It Appears at the Terminal	<Editing specifications> Applied to Message In Transit	Message as It Appears to the User Program
Input/ Output	15P	I("P",5)	00015
Input/ Output	E*	A("*",3)	E
Input/ Output	E*15P	A("*",3),I("P",5)	E 00015
Input	ABCDEFG+	A("+",4)	ABCD (FMTERR set to 6)
Input	12345*AB	I("*",4),A2	1234AB (FMTERR set to 6)
Input	T A1B2AXYZ B	B6,B3	A1B2 XYZ
Input	A1B2C3XYZ	B6,B3	A1B2C3XYZ
Input	T A1B2C3AXYZ B	B6,B3	A1B2C3
Input	T T 12A34A B B	2J5	0001200034
Input	T 123456A B	2J5	1234500006
Input	TT AA BB	2J5	0000000000

Figure 2-6. Example Set 3 - Formatting Specifications Applied to Input and Output Messages

Input/ Output	Message as It Appears at the Terminal	<Editing specifications> Applied to Message In Transit	Message as It Appears to the User Program
Input/ Output	A1B2 XYZ	B6,B3	A1B2 XYZ
Input/ Output	0001200034	2J5	0001200034
Input/ Output	ABCDEF123456	B("*",6),J("+",6)	ABCDEF123456
Input	A*1+	B("*",6),J("+",6)	A 000001
Input/ Output	A 000001	B("*",6),J("+",6)	A 000001
Input/ Output	ABCDEF123456	3(A2,@+2),@3,3(I2,@+2)	AB12CD34EF56
Input/ Output	IJGHEFCDAB	@9,4(A2,@-4),A2	ABCDEFGHIJ
Input	XYZ12345,ABC	A3,X(","),A3	XYZABC
Input	XYZ,ABC	A3,X(","),A3	XYZABC
Input	XYZ ,ABC	A3,X(","),A3	XYZABC
Input	XYZABC	A3,X(","),A3	XYZ (FMTERR set to 4)
Output	ABCDEFGHI	ABC","DEF","GHI	QRST123
Output	D CRESULTS=0053 4	4"0C","RESULTS=",I4	0053
Output	D CRESULTS=0000 4	4"0C","RESULTS=",I4	(forms request)

Figure 2-7. Example Set 4 - Formatting Specifications Applied to Input and Output Messages

Input/ Output	Message As It Appears at the Terminal	<Editing specifications> Applied to Message In Transit	Message As It Appears to the User Program
Input	ONE MALE	T(NUM1,A3,1),T(GENDER, A6,1)	11
Input/ Output	FOURFEMALE	T(NUM1,A4,1),T(GENDER, A6,1)	42
Input/ Output	TWO3	T(NUM2,A3,1),T(DAY,A1,3)	2TUE
Input	SIX#X	T(NUM2,A("#",6),2),A1	06X
Input	ELEVEN#X	T(NUM2,A("#",6),2),A1	11X
Input	TWENTY#X	T(NUM2,A("#",6),2),A1	??X (FMTERR set to 7)
Output	(FOUR)	"(",T(NUM2,A(")",6),2)	04
Input	WED	T(DAY,A3,1)	? (FMTERR set to 7)
Output	??? (control station notified of error)	T(DAY,A3,1)	2
Input/ Output	3	T(DAY,A1,3)	WED
Input/ Output	ONE	XT(NUM2,A6,1),A1,1X	

Figure 2-8. Example Set 5 - Formatting Specifications Applied to Input and Output Messages

Using Location Specifiers

This discussion explains the basic concepts of location specifiers. It is written for the user who has not yet worked with GEMCOS formatting.

The MCS uses two buffers when formatting a message: one buffer contains the message as it appears at the terminal; the other contains the message as it appears to the application program. A message consists of a sequence of one or more fields just as a disk, tape, or card record is composed of a sequence of fields. A format describes the relationship between the fields of a message that are written/read by a program and the fields of the message that are received/transmitted by a terminal.

Input formatting causes a message in the terminal message buffer to be moved, field by field, to the program message buffer. Output formatting moves fields from the program message buffer to the terminal message buffer. When a field is moved, whether by input or output formatting, it is moved under the control of an item phrase, the terminal message-buffer pointer, and the program message-buffer pointer.

An item phrase consists of a field type, a field length, and an optional field delimiter. The field type defines which characters are valid in a field, and controls its justification and fill. The field length determines the number of characters in the field. The field delimiter, if present, designates the character which ends a field. The terminal message-buffer pointer (PT) refers to a particular character position in the terminal message buffer. Likewise, the program message-buffer pointer (PP) refers to a particular character position in the program message buffer.

Pointers PT and PP both begin pointing at the first character (position 1) in their respective messages. As the editing phrases of a format are applied to data fields, the data is moved from one message buffer to the other, and the pointers are updated. Unless specifically instructed to do otherwise, the pointers are updated by moving to the right by the number of characters moved.

Example:

> Assume that the message "ABC 123" was received from a terminal, and it was determined that the format (A3,I4) was to be applied. The situation would initially appear as depicted in Figure 2-9, with the message placed in the terminal message buffer, and the program message buffer cleared and the pointers initialized. The A3 item phrase controls the move of

the first three alphanumeric characters, as depicted in Figure 2-10. As can be seen, "ABC" is placed into the program message buffer, and the pointers are moved three positions to the right.

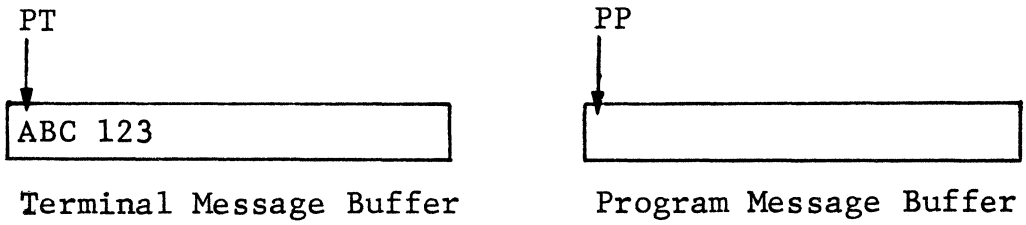


Figure 2-9. Initial Contents of Terminal and Program Message Buffers

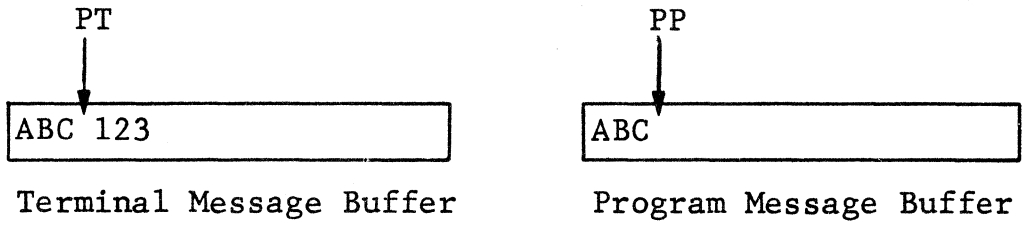


Figure 2-10. Contents of Terminal/Message Buffers After Move Caused by A3 Item Phrase

Then, the I4 item phrase causes " 123" to be moved. During output, integer fields are right justified with zeroes filled and/or blanks converted to zeroes. This "0123" is placed into the program message buffer. Figure 2-11 shows the final situation. At this point, the program message buffer is sent to the appropriate application program.

A higher degree of formatting flexibility may be achieved by moving the pointers without moving text. PT may be advanced without affecting PP by using a skip field (that is, the X editing phrase); but only PT may be advanced. PP may be moved in either direction without affecting PT by using a location specifier.

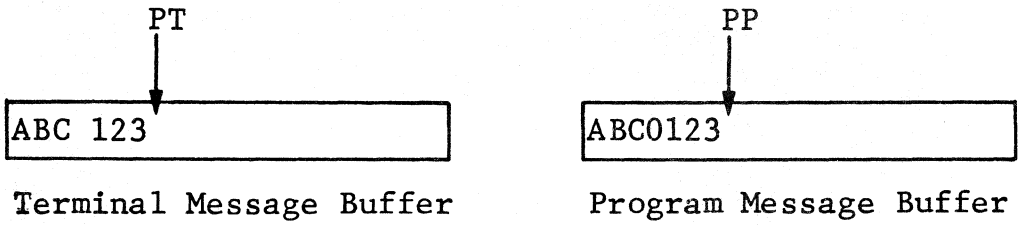


Figure 2-11. Contents of Terminal/Message Buffers After Move Caused by I4 Item Phrase

Figures 2-12 through 2-18 illustrate the effect of applying the formats (A2,@4,A1,X1,@3,A1) to the output message "WXYZ" using the skip field and the location specifier.

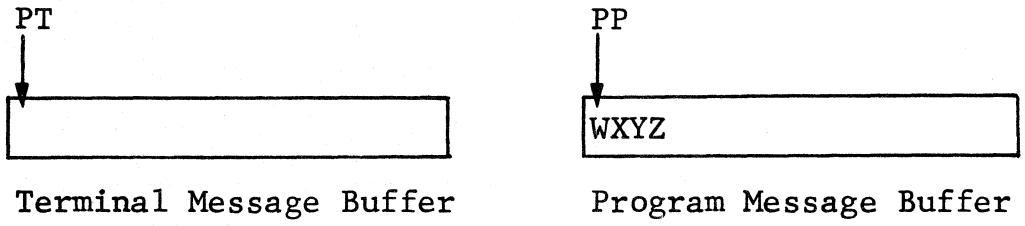


Figure 2-12. Contents of Initialized Buffers

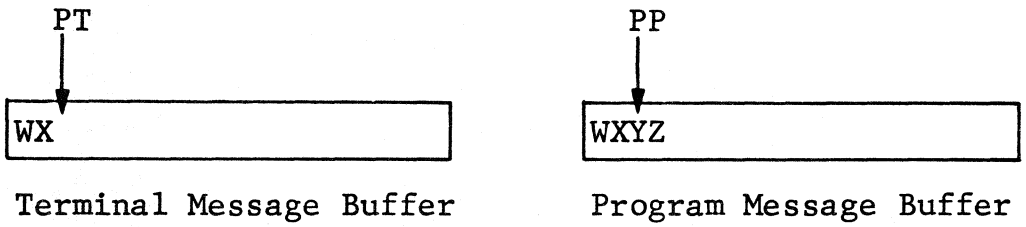


Figure 2-13. Buffer/Pointer Update After Applying Specification A2

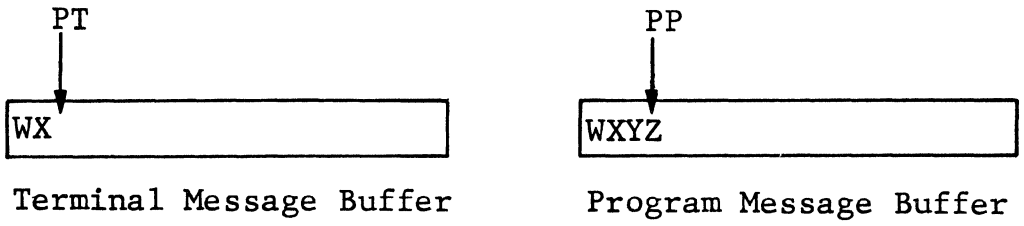


Figure 2-14. Buffer/Pointer Update After Applying Specification @4

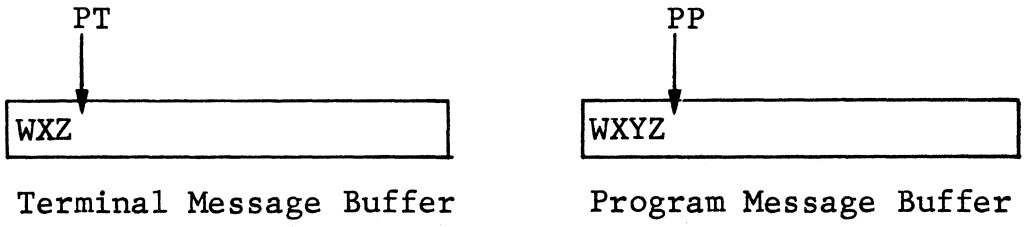


Figure 2-15. Buffer/Pointer Update After Applying Specification A1

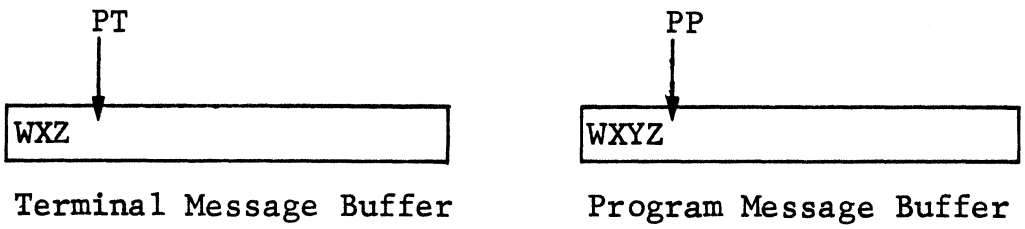


Figure 2-16. Buffer/Pointer Update After Applying Specification X1

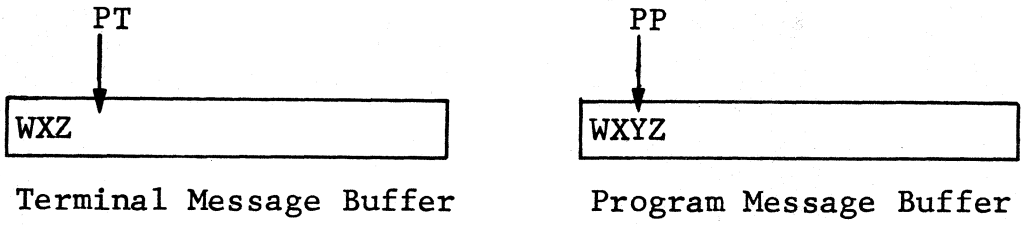


Figure 2-17. Buffer/Pointer Update After Applying Specification @3

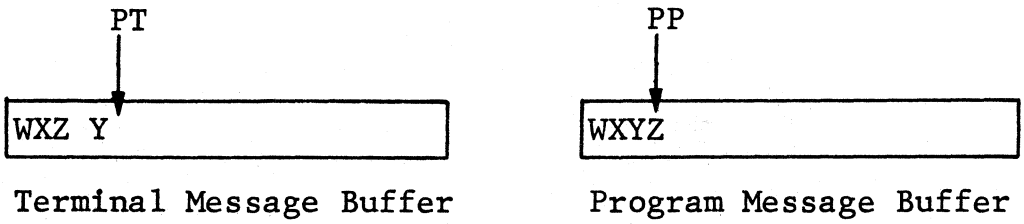
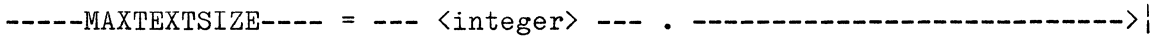


Figure 2-18. Buffer/Pointer Updates After Applying Specification A1 and Sending the Terminal Message Buffer Contents

MAXIMUM TEXT SIZE STATEMENT

Syntax:

<MAXIMUM TEXT SIZE statement>



Semantics:

The MAX TEXT SIZE statement defines the size, in characters, of the longest message that can pass through the MCS. MAXTEXTSIZE has a direct effect upon the memory requirements of a GEMCOS MCS. It is best to keep MAXTEXTSIZE as low as possible. If the MCS has AUDIT specified as TRUE, the user should never attempt to change MAXTEXTSIZE in a REGENERATE MCSTCL run; otherwise, old audit files may have an incompatible record length. Moreover, an increase in MAXTEXTSIZE usually causes a GENERATE and COMPILE to be required so that the MCS can have a larger value stack. If AUDIT is FALSE, MAXTEXTSIZE can be safely lowered on a REGENERATE MCSTCL run. The default value for MAXTEXTSIZE is 125.

When formatting takes place, resultant messages may contain control characters such as tabs or carriage returns. Each control character takes up one or more positions in the formatted message. An allowance for these characters must be reflected by MAXTEXTSIZE.

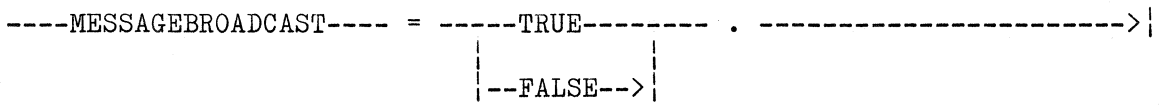
Examples:

```
MAXTEXTSIZE = 1920.
MAXTEXTSIZE = 300.
```

MESSAGE BROADCAST STATEMENT

Syntax:

<MESSAGE BROADCAST statement>



Semantics:

The MESSAGE BROADCAST statement specifies if the code to support the BROADCAST (BRC) Network Control Command is to be generated. By default, MESSAGEBROADCAST equals FALSE.

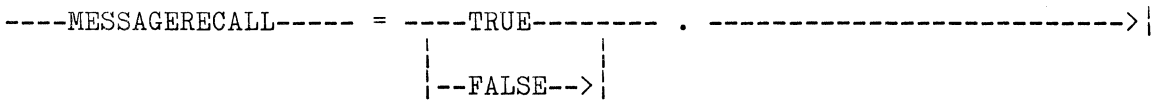
Example:

```
MESSAGEBROADCAST = TRUE.
```

MESSAGE RECALL STATEMENT

Syntax:

<MESSAGE RECALL statement>



Semantics:

The MESSAGE RECALL statement indicates whether the code to support the POP QUEUE (PQ) Network Control Command will be generated. MESSAGERECALL equals FALSE by default.

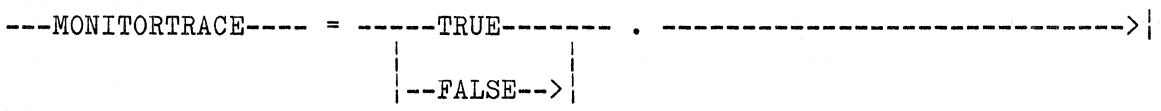
Example:

```
MESSAGERECALL = TRUE.
```

MONITOR TRACE STATEMENT

Syntax:

<MONITOR TRACE statement>



Semantics:

The MONITOR TRACE statement specifies whether to generate logic for the Debug Monitor. When MONITORTRACE is set, CHANGEREQUESTS becomes TRUE by default to include the CMF Network Control Command. The CHANGEREQUESTS code is generated for internal use only. However, users will not be able to use the seven NCC change requests. If the user should want to use the seven NCC change requests, then CHANGEREQUESTS must be set to TRUE in the TCL. If CHANGEREQUESTS equals TRUE and MONITORTRACE equals FALSE, the CMF Network Control Command would not be recognized. By default, MONITORTRACE equals FALSE.

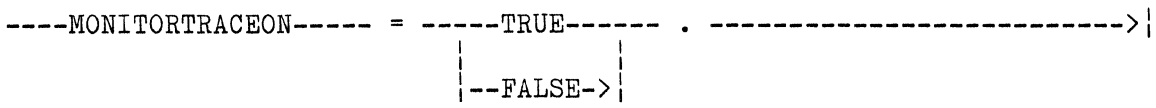
Example:

```
MONITORTRACE = TRUE.
```

MONITOR TRACE ON STATEMENT

Syntax:

<MONITOR TRACE ON statement>



Semantics:

The MONITOR TRACE ON statement allows the user to set or reset the debug monitor flags enabling the initialization procedure to be traced. By default, MONITORTRACEON equals FALSE.

NOTE

The CMF command can be used to set or reset any or all of the monitor flags as soon as initialization is complete.

Example:

```
MONITORTRACEON = FALSE.
```

MY NAME STATEMENT

Syntax:

<MY NAME statement>

Semantics:

Set the MYNAME attribute of the port file GEMCOS uses to 1. Set the YOURNAME attribute of the port subfile to which GEMCOS communicates to the same <identifier>.

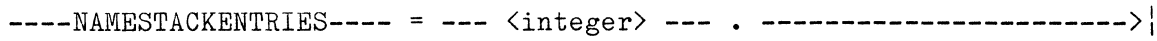
Example:

```
MYNAME = LAI.
```

NAME-STACK ENTRIES STATEMENT

Syntax:

<NAME-STACK ENTRIES statement>



Semantics:

The NAME-STACK ENTRIES statement specifies the maximum number of name-stack entries that need to be reserved for variables declared by user-written code. This parameter is used to ensure that stack sizes are large enough to execute an MCS which contains user-written code. If the value assigned in this statement is not large enough, a name or value-stack overflow error may occur when the MCS is executed.

Name-stack entries are used to store information concerning variables. One name-stack entry is used for each data name that appears in a DECLARE statement. If a data name refers to an array, it would require two name-stack entries. By default, NAMESTACKENTRIES is set to 0.

To achieve optimal memory use, GEMCOS estimates the name-stack space required for its variable declarations and overrides the UPL compiler defaults. If user code is being included, NAMESTACKENTRIES should be set appropriately. The value given to NAMESTACKENTRIES is added to the GEMCOS estimate. If user-written code is not included, the NAME STACK ENTRIES statement may be ignored.

Examples:

```
NAMESTACKENTRIES = 25.
NAMESTACKENTRIES = 100.
```

NCC OK RESPONSE STATEMENT

Syntax:

<NCC OK RESPONSE statement>

```
-----NCCOKRESPONSE---- = ---- <character> ---- . ---------------------><
```

Semantics:

The NCC OK RESPONSE statement defines the message to be returned to a station upon successful completion of a Network Control Command. The string must begin and end with a quote and cannot exceed eight characters in length. By default, NCCRESPONSE is $ (dollar sign).

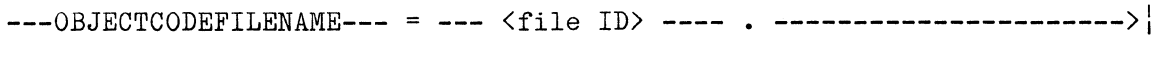
Examples:

```
NCCOKRESPONSE = "NCC OK".
NCCOKRESPONSE = "DONE".
NCCOKRESPONSE = "*OK*".
```

OBJECT CODE FILE NAME STATEMENT

Syntax:

<OBJECT CODE FILE NAME statement>



Semantics:

The OBJECT CODE FILE NAME statement allows for the specification of the MCS object code file name when COMPILE appears in the CONTROL statement. <File-ID> is a B 1000 file identifier. By default, OBJECTCODEFILENAME is MCSSRC/OBJECT.

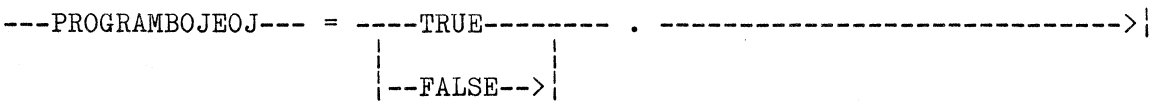
Examples:

```
OBJECTCODEFILENAME = MCS.
OBJECTCODEFILENAME = INVENTORY/MCS.
```

PROGRAM BOJ EOJ STATEMENT

Syntax:

<PROGRAM BOJ EOJ statement>



Semantics:

The PROGRAM BOJ EOJ statement determines if the EXECUTE PROGRAM (EX), HALT APPLICATION PROGRAM (HAP), and FREE (FRE) Network Control Commands are to be supported. By default, PROGRAMBOJEOJ equals FALSE. This statement should be set to TRUE if utility programs are to be generated into the MCS.

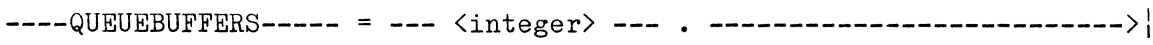
Example:

```
PROGRAMBOJEOJ = FALSE.
```

QUEUE BUFFERS STATEMENT

Syntax:

<QUEUE BUFFERS statement>



Semantics:

The QUEUE BUFFERS statement specifies how many memory buffers are available to the MCS queue before messages begin to overflow to disk. The value assigned to QUEUEBUFFERS directly affects the memory requirements of the on-line system. A value too small or too large can degrade system throughput. It is suggested that the user experiment with this statement to find the most efficient value. QUEUEBUFFERS must not have a value greater than QUEUEDEPTH. <Integer> may range from 1 to 16. By default, QUEUEBUFFERS has the value 1.

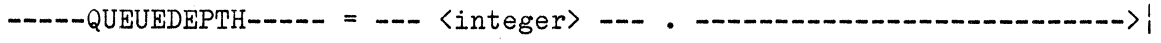
Examples:

```
QUEUEBUFFERS = 5.
QUEUEBUFFERS = 8.
```

QUEUE DEPTH STATEMENT

Syntax:

<QUEUE DEPTH statement>



Semantics:

The QUEUE DEPTH statement specifies the number of messages which may be outstanding in the queue for the MCS. <Integer> may range from 1 to 1023. By default, QUEUEDEPTH equals 20.

Examples:

```
QUEUEDEPTH = 5.
QUEUEDEPTH = 75.
```

QUEUE NAME STATEMENT

Syntax:

<QUEUE NAME statement>

```
----QUEUENAME---- = ---- <remote file-ID> ---- . ---------------------->|
```

Semantics:

The QUEUE NAME statement specifies the external file name of the MCS queue (that is, the remote file opened by the MCS). <Remote file-ID> should appear in a FILE statement in the user's NDL source deck. MCSQUEUE is the default value of QUEUENAME.

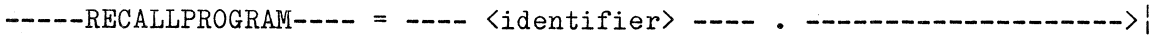
Example:

```
QUEUENAME = MCSRMT.
```

RECALL PROGRAM STATEMENT

Syntax:

<RECALL PROGRAM statement>



Semantics:

The RECALL PROGRAM statement specifies which program is to be designated as the recall program. The recall program is used to recall both audited input and output messages. The identifier must be 10 characters or less in length. By default, there is no recall program.

GEMCOS supplies a recall program called MCSRECALL on the release tape. Further information on MCSRECALL follows.

Examples:

```
RECALLPROGRAM = MCSRECALL.
RECALLPROGRAM = RECALLPROG.
```

Using MCSRECALL to Recall Audited Messages

The following gives further information on MCSRECALL. In the Global section of the TCL, the user is given the option of defining a message recall program to recall any message inserted into an audit file. There are two ways to retrieve messages from the audit file: by simply recalling the last <n> messages, or by recalling messages by time of day. Only one program needs to be declared to the TCL compiler. The trancode(s) used must be declared within the Program section of the recall program. For the syntax of the RECALL PROGRAM statement, see the preceding railroad diagram.

The following rules must be adhered to when defining the attributes of a recall program.

1. The program must be declared as a user program.

2. The COMMONSIZE statement must be absent or set to a value of 60 (default value).

3. The program must not use the auditing capability.

Included on the GEMCOS release tape is a fully functional message recall object program called MCSRECALL. The user need only declare this program to the TCL.

Prior to initially executing the supplied recall program, the user must make the following modifications:

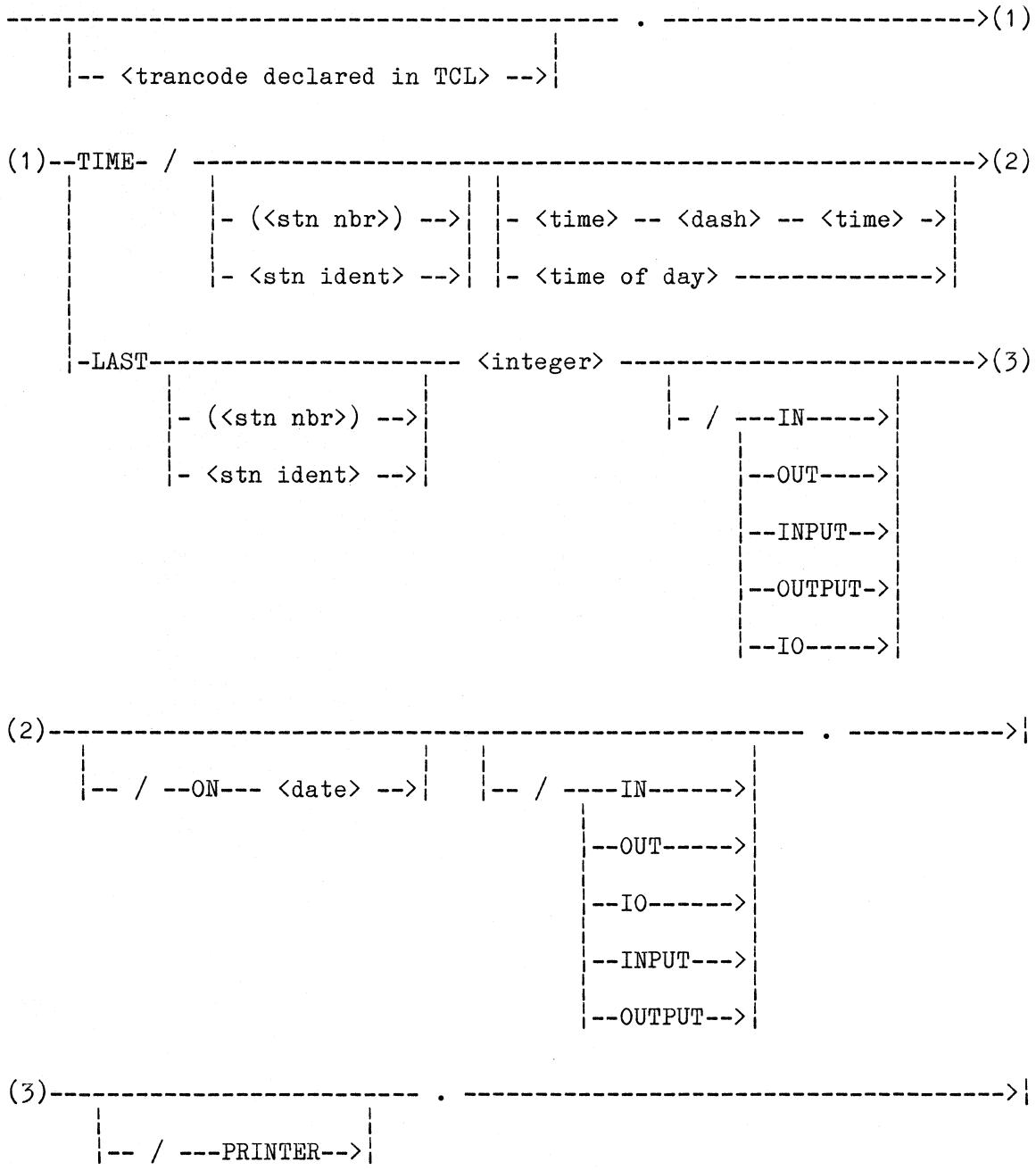
1. The external file name associated with the MCSTIC file must be the same as that associated with the MCSTIC file of the MCS.

2. The external file name of the MCSREM remote file must be a remote file of the Network Controller. Also, the Number of Stations (NST) attribute must be set to the number of stations requested by the remote file.

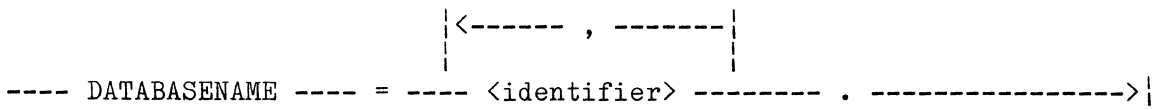
See Appendix B for a summary of the files contained in MCSRECALL.

The syntax of a recall message (as expected by MCSRECALL) is as follows:

<recall messsage>



<DATA-BASE NAME statement>



When the recall source is empty, then the recalled messages are for the station entering the request; otherwise, they are for the station name or number requested.

When the message type identifier is INPUT, only the specified input messages are recalled; when I/O, then both the input and the corresponding output messages are recalled. When empty, only output messages are recalled.

The recall message by time of day option allows the user to specify a time range in which to indicate the messages to be recalled. If a date is also specified, the messages for that date would be recalled if the corresponding audit file or files are on disk.

If LAST is specified, the last <n> messages requested would be recalled. If there are fewer than <n> messages to recall, then the number of messages found would be recalled.

If PRINTER is selected from message destination, then all the recalled messages would be sent to the system printer instead of the requesting station.

Examples (IRC is the user's TCL-defined trancode for the recall program:

```
IRC.TIME/1200.              % Recall the output messages stamped
                            % with time 1200 for this station.

IRC.TIME/1200-1215/IO.      % Recall both the input and output
                            % messages between 1200 and 1215 for
                            % this section.

IRC.TIME/0800-1600/ON       % Recall both the input and output
03/31/79 /IO/PRINTER.       % messages between 8 AM and 4 PM on
                            % March 31, 1979 and send them to
                            % the system printer.

IRC.TIME/(3)/0900-0930      % Recall the input messages initiated
/INPUT.                     % at station 3 between 9 AM and 9:30
```

```
                          % AM and send them to the requesting
                          % station.

IRC.TIME/LSN2/1000-       % Recall both the input and output
1045/ON 04/12/79 /        % messages from station LSN2 between
IO/PRINTER.               % 10 AM and 10:45 AM on April 12,
                          % 1979 and send them to the system
                          % printer.

IRC.LAST/10.              % Recall the last 10 output messages
                          % this station.

IRC.LAST/(2)/5/IO.        % Recall the last 5 input messages
                          % and associated output messages from
                          % station 2 and send them to the
                          % requesting station.

IRC.LAST/50/PRINTER.      % Recall the last 50 output messages
                          % for this station and send them to
                          % the system printer.
```

SIGNAL CHARACTER STATEMENT

Syntax:

<SIGNAL CHARACTER statement>

```
---SIGNALCHARACTER--- = --- <character> --- . ----------------------->|
```

Semantics:

The SIGNAL CHARACTER statement defines the character which, when encountered in the first position of a message, signals to the Network Controller and the MCS that the message is a Network Control Command. The character must be a single character enclosed in quotes. By default, SIGNALCHARACTER is "*".

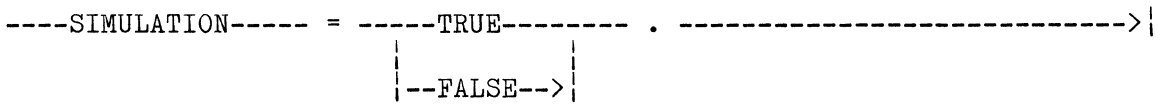
Example:

```
SIGNALCHARACTER = "@".
```

SIMULATION STATEMENT

Syntax:

<SIMULATION statement>



Semantics:

The SIMULATION statement, when set, causes the MCS to open a queue file instead of the usual remote file. The program MCSSIM can be used instead of the Network Controller to simulate input via the card reader. Output is simulated to a line printer using the MCS Monitor Trace code. The source code for MCSSIM is MCSIMS. SIMULATION equals FALSE by default.

Example:

```
SIMULATION = FALSE.
```

SOURCE CODE FILE NAME STATEMENT

Syntax:

<SOURCE CODE FILE NAME statement>

Semantics:

The SOURCE CODE FILE NAME statement allows for the specification of the MCS source code file name when GENERATE appears in the CONTROL statement. <File-ID> is a B 1000 file identifier. By default, SOURCECODEFILENAME is MCSSRC.

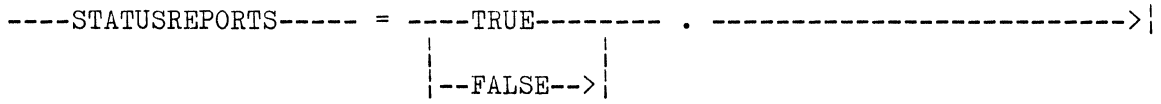
Examples:

```
SOURCECODEFILENAME = MCS/SOURCE.
SOURCECODEFILENAME = SOURCE/FILE.
```

STATUS REPORTS STATEMENT

Syntax:

<STATUS REPORTS statement>



Semantics:

The STATUS REPORTS statement determines whether to include the logic to support the following five Network Control Command status report requests:

1. REPORT FILE STATUS (RFS).
2. REPORT PROGRAM COUNTERS (RPC).
3. REPORT PROGRAM STATUS (RPS).
4. REPORT STATION COUNTERS (RSC).
5. REPORT STATION STATUS (RSS).

STATUSREPORTS equals FALSE by default.

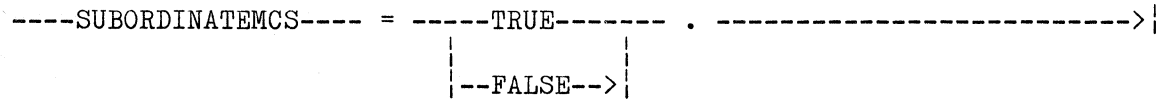
Example:

```
STATUSREPORTS = FALSE.
```

SUBORDINATE MCS STATEMENT

Syntax:

<SUBORDINATE MCS statement>



Semantics:

The SUBORDINATE MCS statement specifies that the GEMCOS MCS is to be executed from and under the control of a supervisory MCS. The supervisory MCS can be any valid MCS, but the primary usage of this option has been designed for execution under SMCS.

The MCS functions depend upon the value of this statement. The following list details the differences between the non-subordinate case (false) and the subordinate case (true).

1. Dummy File Opens

   a. Nonsubordinate: the MCS will attach to itself as many stations as it can using the list of stations in the TCL specifications.

   b. Subordinate: the MCS will attach NO stations to itself.

2. Station Condition at EOJ

   a. Nonsubordinate: the MCS will mark the stations not ready before it goes to EOJ after a *HLT command.

   b. Subordinate: the MCS will leave the stations ready at EOJ after a *HLT command.

3. DFR Command

   a. Nonsubordinate: the DFR NCC will not be allowed.

   b. Subordinate: the DFR NCC will be allowed from any station that GEMCOS controls.

4. Station Condition Report at BOJ

   a. Nonsubordinate: stations not present in the remote file will be so indicated on the ODT. If the remote file is a dummy file, the LSNs of the stations attached will be so indicated on the ODT.

   b. Subordinate: no messages will be displayed on the ODT.

The default value for this option is FALSE.

Special Considerations For Running GEMCOS Under SMCS

1. SUBORDINATEMCS must be set to TRUE.

2. If GEMCOS is set up in the SMCS JOBS file with the AUTO-START option, GEMCOS will be executed with no stations attached. This is correct in recovery mode, as GEMCOS then attempts to attach its previously owned stations (see number 7 following).

3. If GEMCOS is set up in the SMCS JOBS file without the NO-EOF option, GEMCOS will go through its recovery sequence (if recovery is generated into the MCS) when it is reexecuted.

4. Whenever the SMCS command ON is used to gain access to GEMCOS, the GEMCOS MCS will consider this station to be "owned" until the GEMCOS DFR command is used to release it.

   The SMCS command OFF should never be used to return a station to SMCS since GEMCOS will never be informed that the station is no longer under its control.

5. Whenever the SMCS command PASS is used to forward a request to GEMCOS, the GEMCOS MCS will assume ownership of the station. A GEMCOS command DFR will eventually be required to inform GEMCOS to release the station.

6. In order to bring the GEMCOS MCS to EOJ, a *HLT command should be entered from an active control station or, in the case where no stations are active, a *HLT command should be entered from the ODT.

7. Because of the different mechanism of station allocation under SMCS, it may be necessary to run GEMCOS alone if recovery needs to be done on any of its data bases.

   If a system or GEMCOS failure occurs and recovery is needed, the stations which were attached to GEMCOS prior to the failure should not attach to any programs until the recovery is finished. GEMCOS will attempt to reattach the stations it

"owned" prior to the failure. As long as such a station has not attached to another program, SMCS will release the station to GEMCOS.

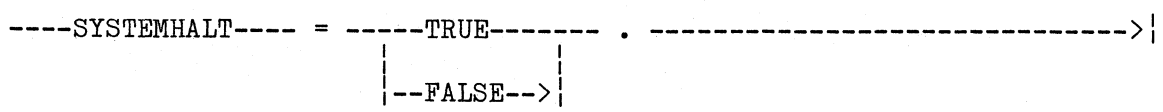
If the station has attached to another program, GEMCOS still attempts recovery, but it may have to use alternate LSNs in the NDL header when it sends messages to programs. GEMCOS may also have to write messages to the print file if it encounters a message for a station it could not attach. (If this happens, the print file can be closed with the CMF command.) In either of these cases, a warning message is written to the monitor stations/ODT.

For additional information on recovery under SMCS, see Section 7.

SYSTEM HALT STATEMENT

Syntax:

<SYSTEM HALT statement>



Semantics:

The SYSTEM HALT statement specifies whether the code for handling the HALT (HLT) Network Control Command is to be generated. When SYSTEMHALT is set to TRUE, CHANGEREQUESTS becomes TRUE (for internal use only). The seven NCC change requests will not be accessible unless CHANGEREQUESTS is set to TRUE in the TCL. SYSTEMHALT equals TRUE by default.

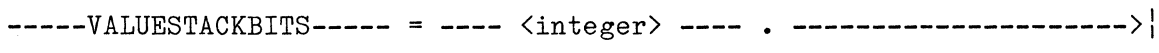
Example:

```
SYSTEMHALT = TRUE.
```

VALUE-STACK BITS STATEMENT

Syntax:

<VALUE-STACK BITS statement>



Semantics:

The VALUE-STACK BITS statement specifies the maximum number of value-stack bits that are needed as a result of user-code data-name declarations. This parameter is used to ensure that stack sizes are large enough to execute an MCS which contains user-written code. If the value assigned in this statement is not large enough, a name or value-stack overflow error may occur when the MCS is executed. The value of a variable which requires 24 or less bits requires no room on the value stack. However, if a variable requires more than 24 bits, or if the variable refers to an array, space would have to be reserved on the value stack for that variable. By default, VALUESTACKBITS equals zero.

In a fashion similar to the NAME-STACK ENTRIES statement, the VALUE-STACK BITS statement enables GEMCOS to achieve optimized memory use. GEMCOS estimates the value-stack space required for its variables and overrides the UPL compiler defaults. If user code is included, VALUESTACKBITS should be set appropriately. The number assigned to VALUESTACKBITS is added to the GEMCOS estimates. If user-written code is not included, the VALUE-STACK BITS statement may be ignored.

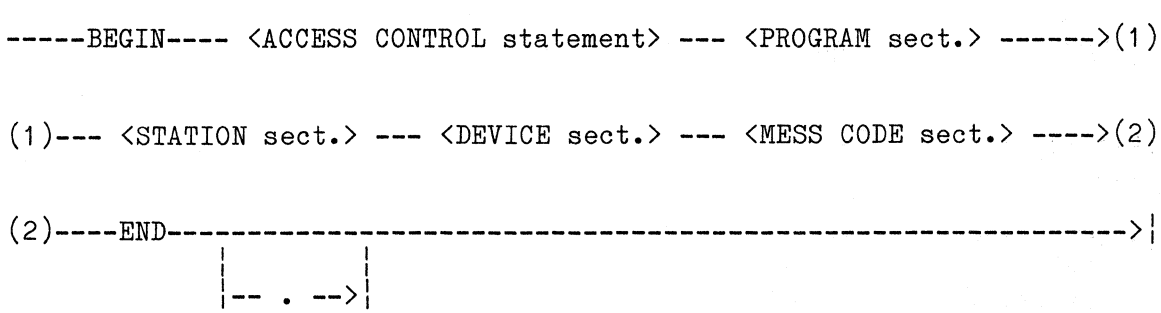
Examples:

```
VALUESTACKBITS = 1000.
VALUESTACKBITS = 256.
```

DEFINITION SECTION

Syntax:

<DEFINITION section>

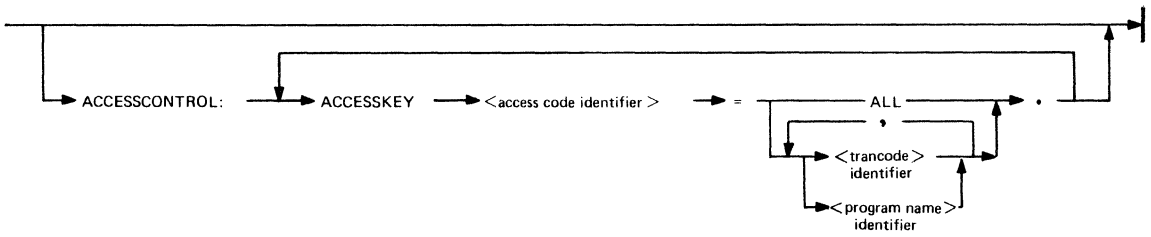


Semantics:

In the Definition section, the user defines access keys (user IDs), programs, and stations, as well as their interrelationships. If the user requires MCS functions not supported by GEMCOS, UPL source code statements can be merged into a GEMCOS MCS by including a MESS code section in the Definition section.

ACCESS CONTROL STATEMENT

The following diagram shows the syntax for the ACCESS CONTROL Statement.



Semantics:

The ACCESS CONTROL statement allows for the specification of access codes. An access code is required as part of the sign-on command syntax (*SGN access code), and identifies the user signing on to the MCS. An access code identifier is an alphanumeric identifier up to six characters in length. Associated with each access code is an item list consisting of transaction codes (trancodes) and/or program names which that particular user is authorized to use.

When a message is received from a station, the MCS searches for a transaction code in the message. If one is present, the MCS would determine if the access code used to sign on at that station is authorized to use that trancode. If the access code is authorized, the message would be routed the appropriate program; otherwise, an error would be returned to the station. If a trancode could not be found in the message, the MCS would verifythat the access code is authorized to use the program currently attached to the station. If so, the message would be routed; if not, an error would be reported.

NOTE

When the value of sign on for a station is FALSE, access control is not in effect at that station. No messages entered at such a station are rejected due to access control restrictions.

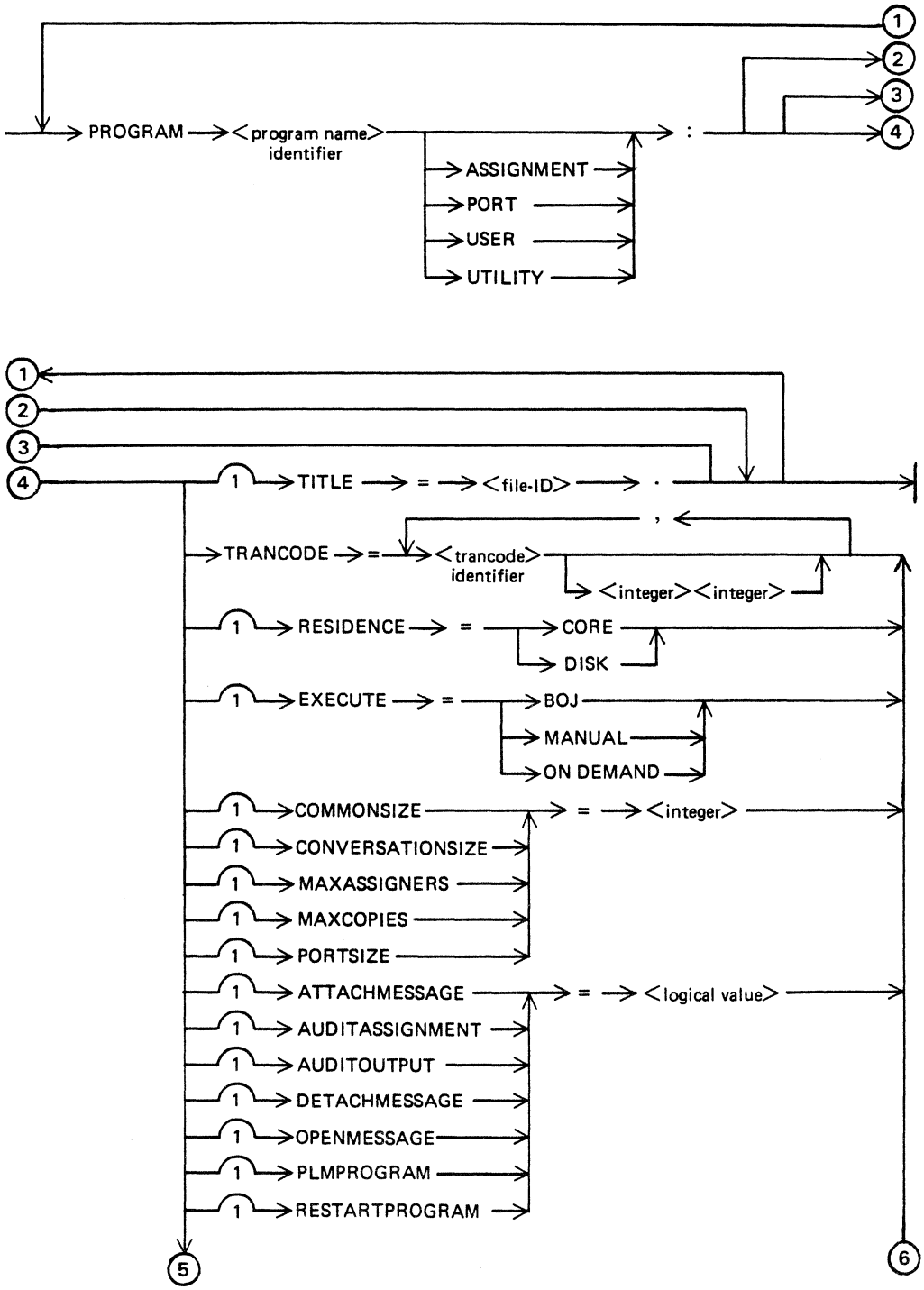
Each trancode encountered in the ACCESS CONTROL statement must appear in a TRANCODE statement of the Program section. Likewise, each program name must appear in a program define of the Program section. If a signed-on user is to have unrestricted use of all the defined transaction codes and programs, the key word ALL may be used. If ALL is used, it must be the only item in the item list.

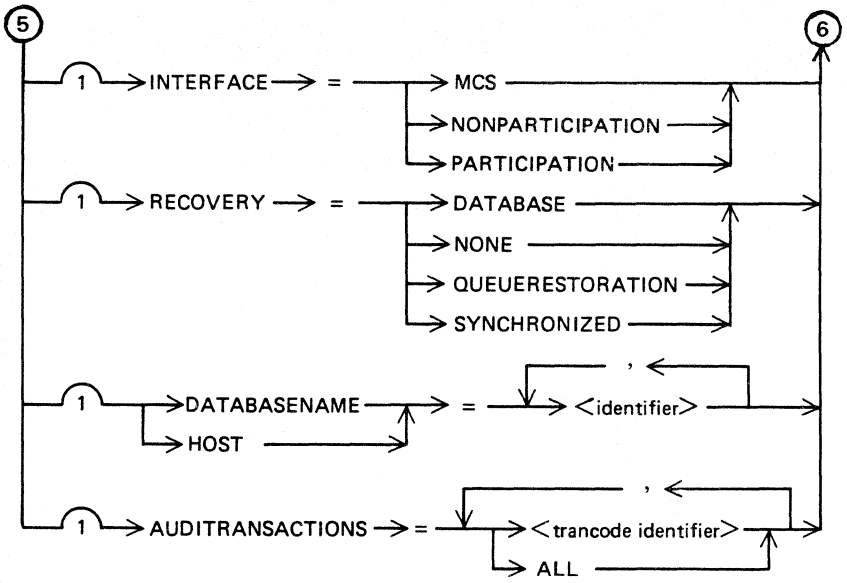
Example:

```
ACCESSCONTROL :
      ACCESSKEY ABCD   = INQ, PAYROLL.
      ACCESSKEY AB1234 = ALL.
      ACCESSKEY AB5678 = INQ, XYZ.
```

PROGRAM SECTION

The following diagram shows the syntax for the Program Section.





Semantics:

The library of on-line programs is defined in the Program section. All programs that open remote files which are to be approved by the GEMCOS MCS must appear in the Program section. If a program attempts to open a remote file consisting of at least one station in the GEMCOS MCS remote file (identified by the QUEUE NAME statement of the Global section), and if the program does not appear in the Program section, the MCS would not allow the file to open.

The Program section is composed of a program define list. Each program define specifies a program name, a program classification, and a program statement list.

The program name is limited to 10 characters and cannot contain slashes. The name can be used optionally in the EX, HAP, RPS, and RPC Network Control Commands instead of PROGRAM TITLE. If there is an ACCESS CONTROL statement, and if the program is not a user program, the program name can appear in its item list to allow certain access codes to use the program.

The program classification specifies to the MCS how this program can be executed, as well as how messages are to be routed to it once it is running. As of the 7.0 GEMCOS release, there are five program classifications: ASSIGNMENT, UTILITY, USER, PASS, and PORT. By default, the program classification is ASSIGNMENT.

Assignment Programs

An assignment program may only be executed from the supervisory console, a card reader, or the Control station. An attempt to execute an assignment program from any other than the Control station by means of the EX Network Control Command results in an operator error.

After being executed, an assignment program eventually opens a remote file in order to gain control of a list of stations in the network. A GEMCOS MCS would grant control of a particular station to an assignment program if the MCS controls the station, and if no other assignment or utility program controls the station. The MCS controls a station when that station appears in the remote file opened by the GEMCOS MCS. When an assignment program opens a remote file, the MCS checks each station defined in the program remote file. If the MCS determines that it cannot grant control of any of these stations, the FILE OPEN would be denied. Otherwise, the MCS approves the FILE OPEN request for the stations in the list for which it is able to grant control. Once control of a station is given to an assignment program, all messages entered from that station that do not contain a trancode of a user program are routed to the assignment program (assuming access control is not violated).

An assignment program retains control of its stations until it resolves to close its remote file. If a HAP Network Control Command is entered from the Control station, the supervisory console, or a card reader, the MCS places an end-of-file character into the queue of the assignment program, which prompts it to close its remote file and go to end-of-job. When an assignment program closes its remote file, the stations are no longer considered busy and can be attached to another assignment or utility program.

Thus, the GEMCOS MCS handles file opening and message routing for an assignment program in much the same way that a Network Controller does when no MCS is present. However, GEMCOS also provides an assignment program with additional functions such as a common-area header, trancode indices, access control, audit, recovery, and formatting.

Utility Programs

A utility program may only be executed from a station in the network. An attempt to use the EX Network Control Command to execute a utility program from the supervisory console or a card reader is denied. A station may not "EX" a utility program when that station is already controlled by an assignment program or another utility program since the station would be considered busy.

Upon receipt of an EX Network Control Command from the station, the MCS determines, in the order listed, the status of the following as they pertain to the utility program:

1. Program is running.

2. Number of stations attached to the program exceeds the limit assigned.

3. Number of program copies exceeds the limit assigned.

When the program is not running, the MCS initiates the program with the ZIP EXECUTE command. Afterward, the initiated program opens a dummy file. Afterward, the MCS attaches the requesting station. (For further information about dummy files, refer to the B 1700 Systems Network Definition Language (NDL) Reference Manual.)

When the program is running, the MCS checks whether the number of stations attached to this program exceeds the maximum assignment limit; if it does not, the MCS would dynamically attach the station to the remote file of the program. However, if the number of stations attached to the program does exceed the limit, the MCS then would proceed to check whether the number of program copies exceeds the limit established. If it does not, the MCS would initiate a copy of the program and attach the station to it. However, if the program copy limit is exceeded, the MCS would display an error message.

Once the attachment occurs, the utility program controls the station. All messages entered from that station which do not contain a trancode or a user program are routed to the utility program.

When the user is finished with a program, the HAP network control command is entered. This prompts the MCS to detach the station from the remote file of the utility program. The station is available and can be attached to another assignment or utility program. When only one station was attached to the program copy, the MCS places an end-of-file character in the utility program queue (for that copy only). The character prompts the program to close the remote file and proceed to end-of-job.

GEMCOS handles a utility program in much the same manner as the B 1700 illustrative MCS handles a program that opens a remote file. However, GEMCOS also provides a utility program with additional functions such as a common-area header, trancode indices, access control, audit, recovery, and formatting.

User Programs

A user program, like an assignment program, may only be executed from the supervisory console, a card reader, or the Control station. An attempt to execute a user program from any station in the network other than the Control station by means of the EX Network Control Command is denied. A user program must use a Participation interface (see INTERFACE Statement below).

After being executed, a user program should open a remote file for stations it can service. The MCS approves the REMOTE FILE OPEN as long as the stations in the remote file are controlled by GEMCOS (those stations not in the remote file of the MCS being deleted from the remote file of the user program).

NOTE

The MCS does not check to see if another on-line program controls the stations, since a user program does not control stations.

User programs can also open a remote file with no stations attached (that is, a file declared in the NDL with FAMILY = DUMMY).

When the GEMCOS MCS receives a dummy file open from a user program, the file open is approved. Any station that is declared in the TCL can communicate with this program, subject to security restrictions.

If GEMCOS is running as a subordinate MCS under the control of SMCS or any other supervisory MCS, any station that is attached to GEMCOS by the supervisory MCS is also able to send trancoded messages to any user program that has previously opened a dummy remote file.

Unlike an assignment program or utility program, a user program receives a message entered from a station in its remote file only if the message has a trancode. At a given point in time, a station may be attached to as many user programs as necessary since the MCS is able to switch messages entered at the station based on a trancode found in the message (a station may only be attached to one assignment or utility program at a time and all messages without a trancode go to that program). A station may be simultaneously attached to an assignment or utility program, even though it may still be attached to user programs.

A user program must have at least one TRANCODE statement in its PROGRAM statement list; otherwise, the program cannot receive any messages.

If several copies of a particular user program are executed, the MCS would distribute the message load evenly among them. This feature can increase system throughput since inputs/outputs (I/Os) can be overlapped.

A user program continues to service the stations in its remote file until it closes its remote file. If a HAP Network Control Command is entered for this program, the MCS would place an end-of-file character in the user program queue, prompting it to go to end-of-job.

Pass Programs

A pass program can be executed at BOJ from the ODT, either manually from the Control stations, or on demand via the PASS command. It can be stopped with a HAP command from any of the Control stations.

After it is executed, a pass program opens a dummy file. But the pass program does not have control over any stations. As long as there are no security restrictions, a station can pass to any pass program at any time.

GEMCOS does not allow a pass program with audit, recovery, or conversation functions. The MAXCOPIES attribute of a pass program is always set to one.

Port Programs

The user can declare programs which use port files rather than remote files. To do this, set PROGRAM TYPE to PORT. The MAXCOPIES attribute is always set to 1.

A port program can be executed manually from a Control station or ON DEMAND. After the port program has been executed, GEMCOS opens a subport file called TPPORT. In order to communicate with GEMCOS, the matching port file in the port program also needs to be opened.

If only one of these port files has been opened, the program with the open port file waits for the matching port to be opened. The status of this program is: WAIT FOR PORT OPEN.

The user can stop any port program by entering a HAP command at any Control station. When this is done, GEMCOS sends a message (Message 27) which tells the program to go to end-of-job and to close its associated subport. The program must close its subport and stop running if it receives this message from GEMCOS.

Examples:

```
PROGRAM A ASSIGNMENT:
  TITLE = PACKA/PAYROLL/.
  TRANCODE = UPDATE.
  COMMONSIZE = 60.

PROGRAM B UTILITY:
  TITLE = EDIT/IT.
  COMMONSIZE = 75.
  RESIDENCE = CORE.

PROGRAM C USER:
  TITLE = FIXIT.
  TRANCODE = OLD(8,1).
  TRANCODE = NEW(9,1).
  RESIDENCE = DISK.

PROGRAM D PASS:
  TITLE = RD.
  INTERFACE = MCS.
  EXECUTE = ONDEMAND.

PROGRAM E PORT:
   INTERFACE
   TITLE      = PORTPROG.
   TRANCODE   = XFER.
   COMMONSIZE = 60.
   PORTSIZE   = 500.
   HOST       = LABASE. % IF HOST STATEMENT IS NOT
                        % DECLARED, THE LOCAL HOST
                        % ON WHICH GEMCOS IS EXECUTING
                        % IS USED.
```

AP300STATUS Statement

Syntax:

<AP300STATUS statement>

```
----AP300STATUS---- = ---- <logical value> ---- . ------------------->|
```

Semantics:

The AP300STATUS statement indicates whether the four-byte AP300STATUS message from the AP300 is forwarded to the attached application program. The status of the AP300 is reported to the Control station or the system SPO when the four-byte status is received. The default value is FALSE.

Example:

```
AP300STATUS = TRUE.
```

ATTACH MESSAGE Statement

Syntax:

<ATTACH MESSAGE statement>

```
----ATTACHMESSAGE---- = ---- <logical value> ---- . ----------------->¦
```

Semantics:

When ATTACHMESSAGE is set TRUE, the program receives a message in its remote file giving the LSN of a station which just attached itself to the program (by means of the *EX Network Control Command). The first station to attach itself does not generate an ATTACHMESSAGE. The station can be obtained from the OPENMESSAGE. The ATTACHMESSAGE consists of a common-area header with the MCSTYPE field set to 2, the LSN field set to the LSN of the attaching station, the SEQNO field set to the next audit sequence number, and the TEXTSIZE field set to 0000. No message text is sent.

When INTERFACE is set to MCS, the common-area header is preceded by a B 1000 MCS Network Controller interface MCS DATA MESSAGE header with the Message Type field set to 80. A program with an interface of Nonparticipation cannot request attach messages. By default, ATTACHMESSAGE is FALSE.

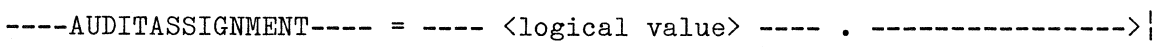
Example:

```
ATTACHMESSAGE = TRUE.
```

AUDIT ASSIGNMENT Statement

Syntax:

<AUDIT ASSIGNMENT statement>



Semantics:

The AUDIT ASSIGNMENT statement directs the MCS whether to audit messages that do not have a trancode. Programs declared as user programs may not use this statement since all messages for that class of program necessarily contain a trancode. User programs that require recovery must use the AUDIT TRANSACTIONS statement. Programs of any other class that require recovery must use this statement or the AUDIT TRANSACTIONS statement or both. By default, the MCS does not audit by assignment.

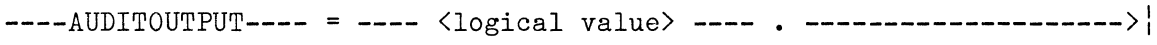
Examples:

```
AUDITASSIGNMENT = TRUE.
AUDITASSIGNMENT = FALSE.
```

AUDIT OUTPUT Statement

Syntax:

<AUDIT OUTPUT statement>



Semantics:

The AUDIT OUTPUT statement directs the MCS to audit all output messages from the program to the station. This statement must be set to TRUE for programs that use synchronized recovery; otherwise, a warning is issued and the statement is automatically set to TRUE. For the MCS to audit output, a program must audit either by assignment or by transaction. Except for synchronized recovery, AUDITOUTPUT defaults to FALSE.

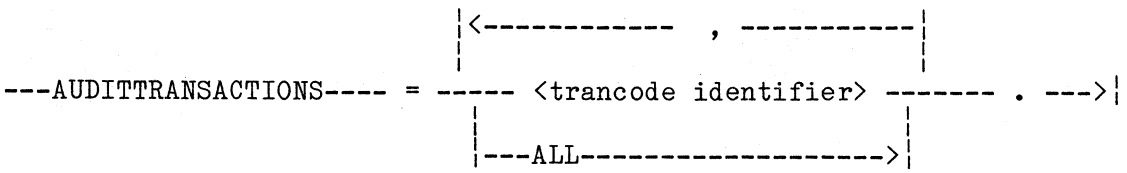
Examples:

```
AUDITOUTPUT = TRUE.
AUDITOUTPUT = FALSE.
```

AUDIT TRANSACTIONS Statement

Syntax:

<AUDIT TRANSACTIONS statement>



Semantics:

The AUDIT TRANSACTIONS statement specifies which previously defined trancodes are to be audited by the MCS. Only transactions that cause the data base to be updated should be audited, since all audited messages are reprocessed during recovery. When ALL is selected, no individual trancodes may be specified and all trancodes for this program are audited. When recovery is required for this program, then either this statement, or the AUDIT ASSIGNMENT statement, or both, must be specified. By default, the MCS does not audit by trancode for any program.

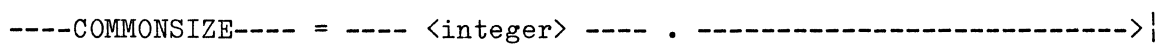
Examples:

```
AUDITTRANSACTIONS = UPD.
AUDITTRANSACTIONS = PAY, OEO1, OEO2, OEO4.
AUDITTRANSACTIONS = ALL.
```

COMMON SIZE Statement

Syntax:

<COMMON SIZE statement>



Semantics:

The COMMON SIZE statement allows the user to specify the length of the header preceding the text of messages exchanged between the MCS and application programs using the Participation interface. <Integer> must be a value from 60 to 200. Bytes 1 through 60 are reserved for GEMCOS-defined fields. Bytes 61 through 200 can be reserved for user-defined fields. User-written code must be merged into the MCS if it is to access, set, or modify bytes 61 through <integer>. The COMMON SIZE statement is optional. By default, COMMONSIZE = 60 (no room reserved for user-defined fields).

Programs using either the Nonparticipation or MCS interface cannot receive a common area, and thus COMMON SIZE cannot be set.

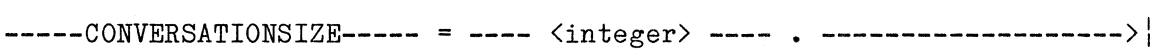
Examples:

```
COMMONSIZE = 60.
COMMONSIZE = 200.
```

CONVERSATION SIZE Statement

Syntax:

<CONVERSATION SIZE statement>



Semantics:

The CONVERSATION SIZE statement is used to establish the size of the conversation area for a program. The size is specified in bytes. The MCS cannot generate conversational capabilities without this statement in the TCL. Anytime this statement is increased to a value greater than any previously declared CONVERSATION SIZE, the TCL must be regenerated and recompiled. The maximum value for this statement is 255.

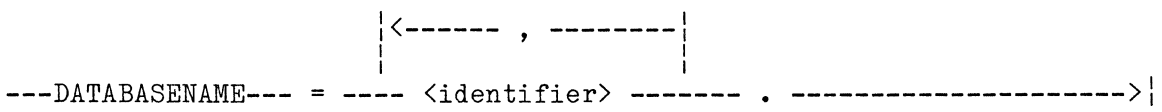
Examples:

```
CONVERSATIONSIZE = 30.
CONVERSATIONSIZE = 45.
```

DATA BASE NAME Statement

Syntax:

<DATA-BASE NAME statement>



Semantics:

The DATA BASE NAME statement associates a program with a data base. When recovery for a program is synchronized or data base, this statement must be present and specify the name of the data base that the program belongs to; otherwise, it is not required. When this statement is required but not given, a syntax error occurs.

When the program is a restart program (RESTART PROGRAM = TRUE), then more than one data base identifier may be specified providing that the restart program services more than one data base. When the program is not a restart program, only one data base identifier may be specified. A data base identifier is an identifier that contains between 1 and 17 characters.

Examples:

```
DATABASENAME = MCSTESTDB.
DATABASENAME = LIVEDB, TESTDB.
```

DETACH MESSAGE Statement

Syntax:

<DETACH MESSAGE statement>

```
---DETACHMESSAGE--- = ---- <logical value> ---- . ---------------------><
```

Semantics:

When DETACHMESSAGE is set to TRUE, the program receives a message in its remote file giving the LSN of the station which has just detached itself from the program (by means of the HAP Network Control Command). The last station to detach itself does not generate a DETACH MESSAGE since the program is informed of the fact (it receives an end-of-file condition on its remote file). The DETACH MESSAGE consists of a common-area header with the MCSTYPE field set to 4, the LSN field set to the LSN of the detaching station, the SEQNO field set to the next audit sequence number, and the TEXTSIZE field set to 0000. No message text is set.

If INTERFACE is set to MCS, the common-area header is preceded by a B 1000 MCS/Network Controller interface, MCS DATA MESSAGE header with the Message Type field set to 80. A program with an interface of Nonparticipation cannot request DETACH MESSAGES. By default, DETACHMESSAGE is FALSE.

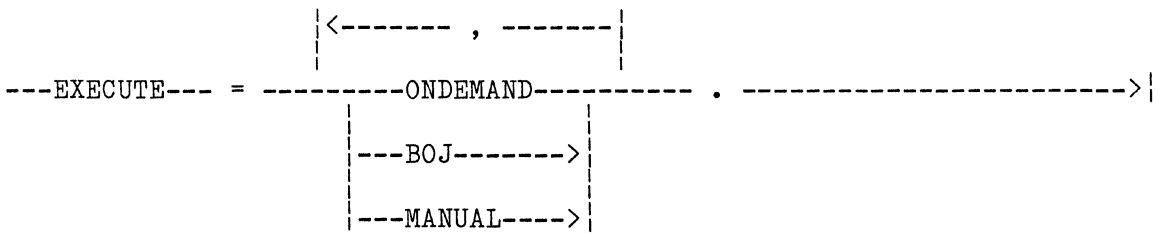
Example:

```
DETACHMESSAGE = TRUE.
```

EXECUTE Statement

Syntax:

<EXECUTE statement>



Semantics:

The EXECUTE statement allows the user to reduce intervention by the console or control station operator during program fire up. Three options are available: ONDEMAND, BOJ, and MANUAL. ONDEMAND and MANUAL may not appear together in the same EXECUTE statement. The default for this statement is MANUAL.

ONDEMAND

This option may only be declared for user and pass programs. Normally, when an operator enters a message containing a trancode for a program that is not running, GEMCOS MCS displays an error message, and the operator must wait until the program is executed through the console or a Control station.

However, when ONDEMAND is selected, GEMCOS MCS ZIP-executes the program when it is not running and a trancode message is received for it. The first message received for the program causes the execution.

The ONDEMAND execution for pass programs is slightly different from the previous process. In this case, the first PASS command to the program causes the execution.

ONDEMAND functions are internal and not visible to the operator. This feature enables the operator to enter messages without interruption. The messages are stored in a "tank file." When GEMCOS MCS receives a FILE OPEN for the program, all "tanked" messages for that program are sent to it in the same order as originally received by the GEMCOS MCS. The tank file is closed when it contains no more messages.

BOJ

The BOJ (beginning-of-job) option can be declared for assignment, user, or pass programs. When the GEMCOS MCS is executed, it automatically executes all BOJ programs unless the MCS needs to perform recovery.

Note that it is advisable to be selective when declaring programs BOJ so the mix is not filled with unnecessary jobs.

MANUAL

MANUAL may be declared for all classifications of programs. When a program declared MANUAL is not running, it must be executed with the EX command. Utility programs can only be declared as MANUAL.

Examples:

```
EXECUTE = MANUAL, BOJ.
EXECUTE = ONDEMAND.
```

HOST Statement

Syntax:

<HOST statement>

```
---HOST--- = ---- <identifier> ----- . ------------------------------>|
```

Semantics:

When initiating a port program, use the HOST Statement to specify the host name attribute. This statement is valid only when the Program Type is PORT. The default value for this statement is NULL.

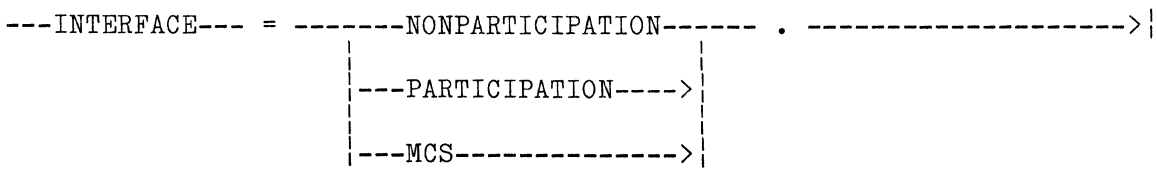
Example:

```
HOST = LABASE
```

INTERFACE Statement

Syntax:

<INTERFACE statement>



Semantics:

The INTERFACE statement determines the path messages follow as they flow between a particular program and the stations in its remote file. It also determines the relationship between the GEMCOS MCS and the program. Three interfaces are available: Nonparticipation, Participation, and MCS. The Nonparticipation and Participation interfaces may only be used by application programs, programs which open a remote file without headers. The MCS interface may only be used by MCS programs, programs which open a remote file with headers. By default, the interface is PARTICIPATION.

Nonparticipation Interface

A Nonparticipation interface is an efficient but static method for a program to communicate with the stations in its remote file. Figure 2-19 depicts the flow of messages in a Nonparticipation interface.

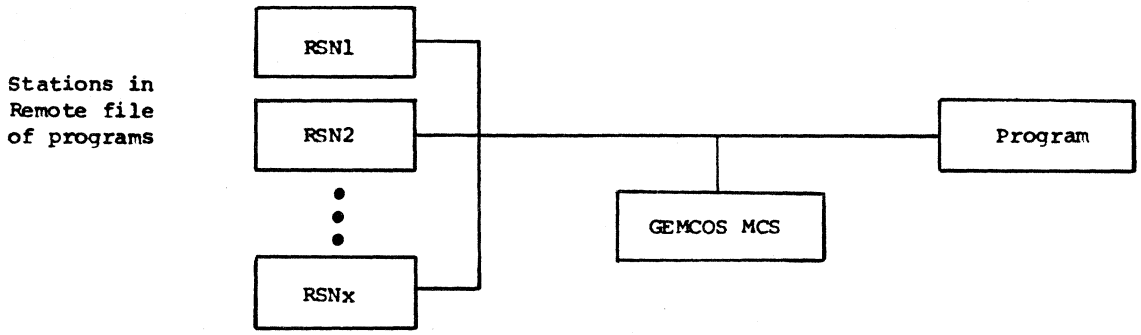


Figure 2-19. Nonparticipation Interface

With a Nonparticipation interface all messages (except those beginning with a signal character) that are entered from all stations in the application program remote file go to the program. The program can write messages to any of its stations. A construct known as a remote key allows the program to determine the source and length of an input and to specify the destination and length of an output.

Messages written by the program or entered from a station beginning with a signal character are sent to the MCS. GEMCOS Network Control Commands reach the MCS by means of this signal character when a Nonparticipation interface is chosen.

Messages beginning with two signal characters that are entered from a station are processed by the MCS in the following manner:

1. When the trancode is found in the message, the transaction is routed to the program specified by the trancode, provided the program is running or declared as ONDEMAND. Output messages from the program are routed back to the station. This allows a user at a station that is attached to a non-participating program, to perform trancode routing to other programs in the network.

2. When the message (starting from the third character position only) contains a message-ID, it is considered to be a forms request, and the blank form is sent back to the station. This feature is only available in the Advanced and Total Versions of GEMCOS.

3. When the message contains neither a valid trancode nor message-ID, it is routed to the program to which the station is attached. The first two bytes (or two signal characters) are not returned with the message.

The Nonparticipation interface is efficient since a typical transaction passes through only one program, the user program (in addition to the Network Controller). This interface is static since a station can only be in one opened (input) remote file at a time, and therefore has access to only one program. In addition, the MCS does not have access to the normal flow of messages and is unable to provide audit, formatting, access control, and its other functions.

When interface is Nonparticipation, the program classification cannot be USER and there cannot be any transaction codes. The common-area header will not be on messages received by the program, and the program must not provide them on output. Thus, COMMONSIZE cannot be set. ATTACHMESSAGE, DETACHMESSAGE and OPENMESSAGE cannot be TRUE. Users at stations in the remote file of a Nonparticipation program can neither use transaction-based routing nor initiate screen requests while the Nonparticipation program is running. Even if a station has been assigned a SCREENSIZE, screen wraparound cannot take place while the station is under control of a Nonparticipation program. Audit, queue restoration, and formatting are not possible, even though these options can be specified in the Global section and can be used by Participation programs attached to other stations in the network while Nonparticipation programs are running.

If stations can be dedicated to a particular program while the program is running, and the program does not require access control, audit, queue restoration, formatting or screen wraparound, it is advantageous to use the Nonparticipation interface.

Participation Interface

When PARTICIPATION is specified as the program interface, all messages entered at stations pass through the MCS before being sent to programs, and all messages written by the program pass through the MCS before being transmitted to stations. The MCS is said to be "participating" in the message traffic flowing between the program and the stations of the

program remote file. Figure 2-20 depicts the flow of messages in a Participation interface.

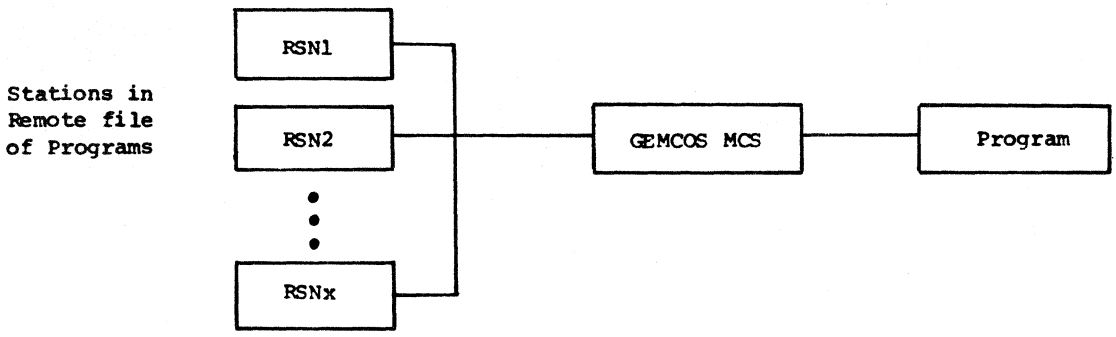


Figure 2-20. Participation Interface

The Participation interface is slightly less efficient in terms of throughput, since a typical transaction passes through three programs: the MCS (during input), the user program, and the MCS again (during output). The slight decrease in efficiency is more than offset by the full complement of centralized functions provided by the MCS message. It can provide a full array of centralized functions (including audit, recovery, formatting, screen wraparound, access control, and various forms of routing).

Common-Area Header with Participation Interface

Programs which use the Participation interface receive and must provide the common-area header. This header, in addition to its other functions, allows the program and MCS to communicate the message text length and the source/destination station to each other.

The common-area header precedes all messages sent to programs using the Participation interface, and it is required in front of all messages written by such programs. The length of the common-area header can vary from 60 to 200 bytes by program as specified in the COMMONSIZE statement. The layout of the common-area header is as follows:

```
01  COMMONAREA.
    05  MSGDESTINATION       PICTURE 9(1).
    05  LSN                    PIC 9(3).
    05  PGMNBR REDEFINES LSN   PIC 9(3).
    05  MTSMSGTYPE             PIC S9(1).
    05  SEQNO                  PIC 9(6).
    05  NDLTIME                PIC 9(7).
    05  TEXTSIZE               PIC 9(4).
    05  TERMTYPE               PIC 9(2).
    05  MSGID                  PIC X(6).
    05  INDEX1                 PIC 9(2).
    05  INDEX2                 PIC 9(2).
    05  ERROR                  PIC 9(1).
    05  FMTERR                 PIC 9(1).
    05  MCSTYPE                PIC 9(2).
    05  INPUTADDR              PIC 9(9).
    05  RETRYCOUNT             PIC 9(1).
    05  RECOVERYSTATUS         PIC 9(1).
    05  OUTPUTADDR             PIC 9(9).
    05  CONVERSATIONSTATUS     PIC 9(1).
    05  CONVERSATIONBOJEOJ     PIC 9(1).
    05  USERAREA               PIC X( ).
```

The following explains each field in the common-area header in detail.

Fields in Common-area Header

MSGDESTINATION

This field can be filled in by the application program to indicate special routing. It is used primarily for program-to-program or program-to station trancode routing. The GEMCOS system fills the field with the default value before sending it to the program. Thus, the application program need not adjust the value unless special routing is required. A list of the values for this field and the default values, set by GEMCOS, follow.

0 Send to indicated station (final destination - no default).

1 Send to indicated program (final destination - no default).

2 Route with trancode (final destination - no default).

3 Route with trancode. Return to station (the GEMCOS system sets the value to zero).

4 Route with trancode, return to program (GEMCOS sets value to 1).

5 This value is set by GEMCOS to indicate that the message originates from a routeheader station. It should not be altered unless an intermediate transaction is required. See Section 10 for explanation of routeheader stations.

LSN

For incoming messages or recovered incoming messages, this field contains the LSN of the originating station. Outgoing messages are sent to the station whose LSN is stored in this field. For attach notifications and detach notifications, this field contains the LSN of the station involved. For open notifications, this field contains the number of stations in the approved FILE OPEN.

PGMNBR

This field contains the program number of the originating program in the event that the message must be routed back to the program. It redefines the LSN field so that no LSN is present if a program number is specified. MSGDESTINATION will be 1.

NOTE

It is the responsibility of the program to keep track of the LSN when doing message routing to another program.

MTSMSGTYPE

Modular Terminal System Message Type. This field is used to identify incoming and outgoing messages when the source or destination is an MTS terminal. Refer to Section 10 for a detailed explanation of this field.

SEQNO

Sequence Number. The MCS assigns a unique number to each message. That number is passed to the application program in this field.

NDLTIME

NDL Time. This is the time that the Network Controller sends the message to the MCS.

TEXTSIZE

Text Size. For incoming messages, this is the length in characters of the message text. It does not include the length of the common-area header. For open notification, it is set to the LSN field multiplied by 3. When the application program writes a message, it must use the ACTUAL KEY of the remote file to specify the text size. In this case, the size must include the size of the common-area header.

TERMTYPE

Terminal Type. The MCS sets this field on incoming messages to a code which identifies the type of the originating device. Terminal-type codes are assigned in the Terminal section of the NDL.

MSGID

Message-ID. The MCS sets this field to the access key signed on to the station from which the message came. If the station does not require sign on, the field is blank.

Users who use both GEMCOS formatting and security should be sure that their programs replace this field either with blanks or with a valid message-ID. If this field does not contain either blanks or a valid message-ID, GEMCOS calls the format-module and searches the list of valid message-IDs, which slows response time.

When the user's program sets this field to a valid message-ID (refer to the OUTPUT FORMATS statement), the MCS formats the message before transmitting it to a station. When the program leaves the field blank, the MCS does not format the message.

INDEX1

Module-Function Index One. When an incoming message contains a valid trancode, and that trancode has module function indices

defined in the TCL, the MCS sets this field to the first index; otherwise, this field is set to zero.

INDEX2

Module-Function Index Two. When an incoming message has module-function indices defined in the TCL, the MCS sets this field to the second index; otherwise, this field is set to zero.

ERROR

When the MCS detects an error while routing a message from a program by trancode, a value is returned. Definitions for these values follow:

0 No error.

1 Missing trancode (trancode routing was specified).

2 Requested program or station not available.

3 Return station ID is invalid.

4 Error in routeheader (processor to processor) message.

FMTERR

Format Error Indicator. When the MCS detects an error while formatting an incoming message, this field is set. Note that errors detected while formatting an outgoing message are reported to the Monitor stations. Refer to the discussion of formatting errors under <format declaration> for an explanation of the values which can be found in this field.

MCSTYPE

Message Type. This field identifies the type of message being exchanged between the user application program and the MCS. The allowed values and their meanings are:

0 On input, this is a message from a station. On output, this is the last (primary) message for the current transaction.

1 Not used on input. On output, this is a secondary message (that is, the program has additional responses to send for this transaction).

2 On input, this is a station attach notification. Not used on output.

4 On input, this is a station detach notification. Not used on output.

6 On input, this is a file open notification. Not used on output.

15 On input, this message instructs the restart program to pass recovery information back to the MCS. Not used on output.

17 Not used on input. On output, this message is sent by the restart program to the MCS. It contains recovery information requested by the MCS.

18 Not used on input.On output, this message is sent by the restart program to inform the MCS that an error was found.

20 Not used on input. On output, this message is sent to the MCS to indicate that the user application program needs recovery.

21 On input, this message instructs the user application program to prepare for recovery. Not used on output.

22 Not used on input. On output, this message is sent by the user application program to inform the MCS that it is ready for recovery (used in response to a type-21 message only).

23 On input, this message is sent by the MCS to the user application program immediately after the remote file is opened. Its purpose is to pass information to the program that must be saved in the restart data set. Not used on output.

24 On input, this message is sent to the user application program instructing it to close its data base and prepare to terminate processing. Not used on output

25 Not used on input. On output, this message is sent by the user application program to inform the MCS that the program has successfully closed its data base and is now ready to terminate processing.

26 Not used on input. On output, the user application program sends this message to inform the MCS that the program would like to go to EOJ.

INPUTADDR

Input Audit Disk Address. This field contains the audit-file disk address of this transaction. When this field is zero, this transaction was not audited.

RETRYCOUNT

Transaction Retry Count. This field contains the number of times this transaction was submitted to the user application program. The value is incremented by one whenever an input transaction causes a user application program to abort.

RECOVERYSTATUS

System Recovery Status. This field indicates the system status at the time this transaction was sent. The allowed values and meanings are:

0 The system is not in recovery mode.

1 The system is in recovery mode caused by a user application program abort.

2 The system is performing an archival recovery.

3 The system is in recovery mode caused by a Clear/Start or an abnormal termination of the MCS.

OUTPUTADDR

Output Audit Disk Address. This field contains the audit file disk address of the output message generated by the user application program. This field is not used by the program.

CONVERSATIONSTATUS

Conversation Status. This field indicates whether the conversation path is clear, a conversation is in progress, or an error occurred by the last message. Descriptions follow for each value that is possible in this field:

0 Path is clear. There is no conversation in progress at the station, or the station is nonconversational.

1 Conversation in progress. Whether the station is comunicating with a program is indicated by the value of the CONVERSATIONBOJEOJ field.

2 Error. Maximum number of conversations was exceeded. The last message is neither audited nor delivered, but returned.

3 Error. Conversation is attempted with a nonconversational station. Message is returned.

4 Error. Conversation attempted with a conversing station. Message is returned.

5 Error. A nonconversational program attempts to initiate a conversation. Message is returned to the program.

CONVERSATIONBOJEOJ

Conversation BOJ EOJ. This field indicates the beginning and the end of a conversation. The field is kept up-to-date through the messages from the user. Descriptions follow for each value that is possible in the field:

For messages to stations:

0 End of conversation, or no conversations in progress. The MCS expects message text immediately after the common-area header.

1 Conversation is beginning or continuing. Conversation text is located between the common-area header and the message text. Conversation text is stored in the conversation area. If GEMCOS is auditing the participating program, the conversation text is audited as well. well.

For messages from stations:

0 Unoccupied. By returning this value, the station indicates that it is open for conversation.

1 Occupied. This value verifies to the program that it is in conversation with the station. Conversation text follows the common-area header.

USERAREA

User Area. If COMMONSIZE is 60, the User-area field does not exist. If COMMONSIZE is greater than 60, the length of the User-area field (n) is COMMONSIZE minus 60. User-written MESS procedures must be written if this field is to contain significant information.

As previously mentioned, the common-area header is placed in front of the text of messages exchanged between the MCS and programs using the Participation interface. The length of the text is determined by the TEXTSIZE field. For incoming messages and recovered incoming messages, the text is the data received from a terminal. For open notifications, the text is a list of 3-character LSNs. No text is associated with attach notifications or detach notifications. For outgoing messages, the application program sets the text to the data to be sent to a terminal.

The attach, detach, and/or open notifications can be requested by a program using the MCS interface. In this case, GEMCOS writes an MCS-to-MCS data message with a message type-80 (refer to Burroughs B 1700 Systems Network Definition Language Reference Manual). The text of this data message is a common-area header. Therefore, a subordinate MCS which expects attach, detach and/or open notifications must be able to handle an MCS-to-MCS data message from GEMCOS in addition to the MCS/Network Controller message types. See Table 2-1 for details. The following legend explains the symbols used in Table 2-1.

Legend for Table 2-1

X	The MCS sets this field, which contains valid information.
U	The MCS sets this field only if the user specified procedures.
Y	The MCS requires the application program to provide valid information in this field.
V	The MCS reads this field only if the user specified procedures.
W	If the program requires queue restoration recovery, the MCS requires the application program to provide information in this field.

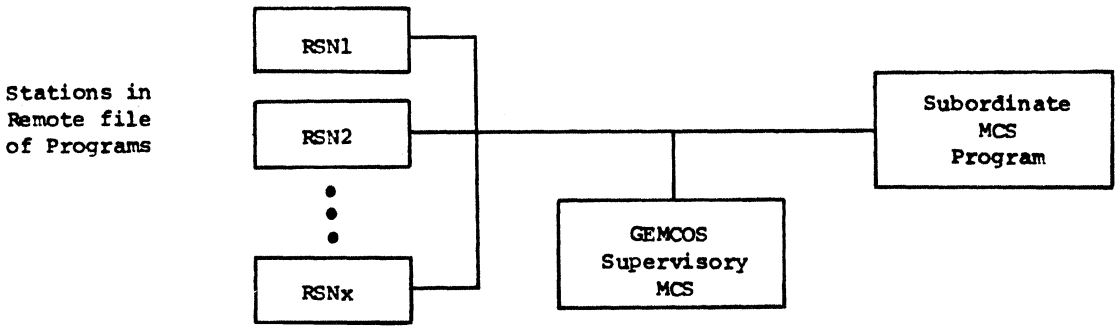
Table 2-1

Common-area Header Fields
Containing Valid Information
by MCSTYPE

Field	Written by MCS						Written by User Program							
	MCSTYPE						MCSTYPE							
	0	2,4 6	15	21	23	24	0	1	17	18	20	22	25	26
MSGDESTINATION	X													
LSN	X	X	X				Y	Y						
PGMNBR	X													
MTSMSGTYPE	X													
SEQNO	X													
NDLTIME	X													
TEXTSIZE	X	X	X	X	X	X								
TERMTYPE	X													
MCSGID	X						Y	Y						
INDEX1	X													
INDEX2	X													
ERROR	X													
FMTERR	X													
MCSTYPE	X	X	X	X	X	X	Y	Y	Y	Y	Y	Y	Y	Y
INPUTADDR	X											W		
RETRYCOUNT	X													
RECOVERYSTATUS	X													
OUTPUTADDR														
USERDATA	U	U					V	V						
TEXT	X		X		X		Y	Y	Y					
CNVERSATIONSTATUS	X													
CONVERSATONBOJEOJ	X					Y	Y							

MCS Interface

When a program is defined as using an MCS interface, the flow of messages is similar to a Nonparticipation interface. All messages (except those beginning with the signal character) entered from each station in the MCS program remote file go to the program. The MCS program can write to any station in its remote file. Figure 2-21 depicts the flow of messages in an MCS interface.



RSN signifies the Relative Station Number.

Figure 2-21. MCS Interface

Two areas in which the MCS interface differs from the Nonparticipation interface are as follows:

1. A program using an MCS interface must open its remote file with headers, thereby identifying itself as an MCS program to GEMCOS and the Network Controller.

2. The program must provide and expect a Network Controller-defined 50-byte header preceeding all data messages. With this header, the program may access tallies and toggles, and may perform functions such as output message switching, communication with the Network Controller, remote file management, system interrogation, and system control. (The Network Controller/Message Control System interface is defined in Burroughs B 1700 Systems Network Definition Language Reference Manual.)

When a program using the MCS interface opens its remote file, GEMCOS assumes the status of a supervisory MCS while the program is considered a subordinate MCS. The supervisory MCS must be entered into the mix before any of the subordinate MCS programs.

A program using the MCS interface can be either a utility program, an assignment program, or a pass program. In brief, stations can dynamically attach to and detach from a utility program via the EX and HAP Network Control Commands, while an assignment program controls a fixed set of stations and can only be initiated from the Control stations, or the ODT. A pass program controls no stations. It can be initiated from the ODT, Control stations, or any station via a PASS command.

Messages entered at stations in a remote file opened by a subordinate MCS (which do not begin with the GEMCOS signal character) go directly to the subordinate MCS and are not seen by GEMCOS. As a result, an MCS program temporarily suspends GEMCOS MCS functions (except certain Network Control Commands) at the stations in its remote file. While stations are in the remote file of an MCS program, they cannot use GEMCOS trancodes, screen wraparound, audit, recovery, or formatting. Messages beginning with the GEMCOS signal character go to the GEMCOS supervisory MCS so that, even while a station is attached to a subordinate MCS, GEMCOS network control commands can be entered.

NOTE

> Network Control Commands affecting the attributes of stations in the remote file of an MCS program cannot be acted upon. The subordinate MCS is responsible for the attributes of the stations it controls.

The B 1000 MCS/Network Controller interface allows subordinate MCS programs to change data communication attributes of associated stations. However, when a station attribute is changed by a subordinate MCS, the change is effective only while the subordinate MCS controls the station. As soon as either the subordinate MCS closes its remote file or the station detaches itself, GEMCOS returns the station to its original status.

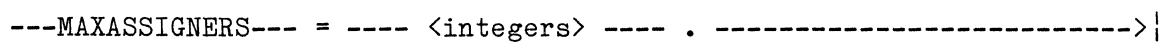
Examples:

```
INTERFACE = NONPARTICIPATION.
INTERFACE = PARTICIPATION.
INTERFACE = MCS.
```

MAXIMUM ASSIGNERS Statement

Syntax:

<MAXIMUM ASSIGNERS statement>



Semantics:

The MAXIMUM ASSIGNERS statement is used for utility programs only. This statement is used to specify the maximum number of stations that can be attached to a program concurrently. The maximum value cannot be greater than the number of stations allowed by GEMCOS. By default, all attachments are applied to one program.

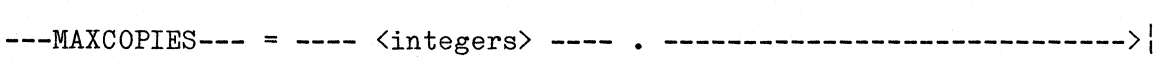
Examples:

```
MAXASSIGNERS = 5.
MAXASSIGNERS = 2.
```

MAXIMUM COPIES Statement

Syntax:

<MAX COPIES statement>



Semantics:

The MAX COPIES statement is used to specify the number of copies of this program that can be running (that is, have a remote file open) at any one time. For assignment and pass programs, the only allowable value is one. For user programs, setting MAXCOPIES greater than one allows multiple copies of the program to be executed manually. (See "User Programs," under "PROGRAM SECTION" earlier in this section.)

The sum of MAXCOPIES for all programs determines how many programs can be running in the MCS concurrently. An increase in the value assigned to MAXCOPIES during regeneration may require generating and compiling so that the MCS has a larger value stack. The value of MAXCOPIES is safely lowered during regeneration. MAXCOPIES is set to 1 by default.

Examples:

```
MAXCOPIES = 3.
MAXCOPIES = 2.
```

OPEN MESSAGE Statement

Syntax:

<OPEN MESSAGE statement>

```
---OPENMESSAGE--- = ---- <logical value> ---- . ---------------------->¦
```

Semantics:

When OPENMESSAGE is set to TRUE, the program receives, as the first message in its remote file, information from GEMCOS listing the LSNs of the stations which comprise the program remote file. The OPENMESSAGE consists of a common-area header with the MCS-TYPE set to 6, the LSN field set to the number of stations in the program remote file, and the TEXTSIZE field set to LSN *3. The text is set to a list of 3-byte LSNs. The OPENMESSAGE is not audited.

When the interface is set to MCS, the common-area header is preceded by a B 1000 MCS/Network Controller interface MCS DATA MESSAGE header with the Message Type Field set to 80. A program with an interface of Nonparticipation cannot request the OPENMESSAGE. By default, OPENMESSAGE is FALSE.

Example:

```
OPENMESSAGE = TRUE.
```

PLM PROGRAM Statement

Syntax:

<PLM PROGRAM statement>

```
---PLMPROGRAM---- = ---- <logical value> ----- . ---------------------->¦
```

Semantics:

The PLM PROGRAM statement is used to determine if the given program is the BNA Port Level Manager program. This program is used to accomplish BNA station transfer. It is a special program which can only be executed at the ODT.

Declare the PLMPROGRAM as follows:

```
PROGRAM <PID> UTILITY:
   TITLE      = BNA/PLM.
   INTERFACE  = MCS.
   PLMPROGRAM = TRUE.
```

The default for PLMPROGRAM is FALSE.

Only one PLMPROGRAM may be declared. Once the program is running, the user must enable station transfer. To do this, enter at the ODT:

```
NW STATIONTRANSFER +
```

The PLMPROGRAM may then be executed by a station using the Execute command.

A station can also be attached to another system. To transfer a station to another system, the user executes the PLM program from a station. Enter the following:

```
*EX BNA/PLM
```

Once the station has been attached to the BNA program, enter:

```
CONNECT TO <host ID>
```

(See the BNA User's Reference Manual for a full explanation of the operating instructions for station transfer.)

Enabling station transfer also allows stations on another system to connect themselves to the MCS on the local system. Such stations must be declared in TCL with VIRTUALSTATION = TRUE. (See the VIRTUALSTATION Statement in this manual for additional information on stations transferring into GEMCOS.)

Examples:

```
PLMPROGRAM = TRUE.
PLMPROGRAM = FALSE.
```

PORT SIZE Statement

Syntax:

<PORT SIZE statement>

```
---PORTSIZE--- = ---- <integer> ----- . ------------------------------>|
```

Semantics:

The PORT SIZE Statement only has meaning for a station with PORTSIZE = TRUE. This statement specifies the maximum number of characters that GEMCOS reads or writes to the port associated with that station. Messages that GEMCOS sends to a port station are sent in pieces of its port size length. Messages that GEMCOS receives from a port station are truncated at the largest value of all port sizes for all port stations. The default value of PORTSIZE is 2000 characters. The maximum value of PORTSIZE is 3000 characters.

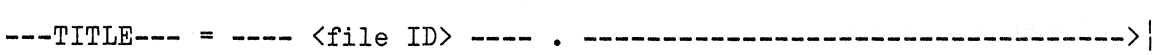
Example:

```
PORTSIZE = 2100.
```

PROGRAM TITLE Statement

Syntax:

<PROGRAM TITLE statement>



Semantics:

The PROGRAM TITLE statement identifies the object-code file name of a program. The file-ID is a B 1000 file identifier, and it is used optionally in the EX, HAP, RPS, and RPC Network Control Commands to refer to the program. When the program name is used in one of these commands, GEMCOS applies the command to the object-code file name specified in the PROGRAM TITLE statement.

The PROGRAM TITLE statement is optional. When it is omitted, the program name is used as the default. Note that the program name is limited to 10 characters and cannot contain slashes.

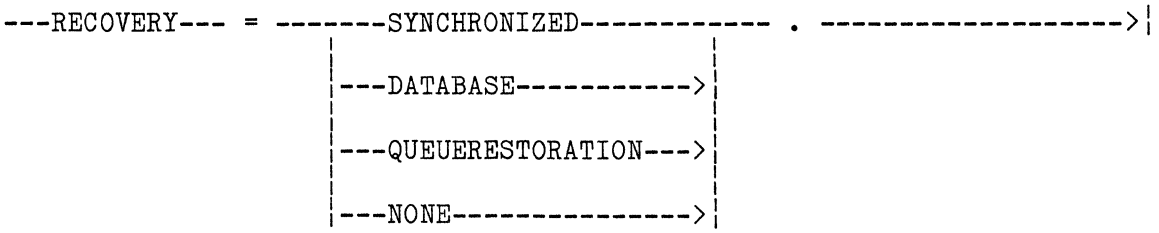
Examples:

```
TITLE = PACK1/X/Y.
TITLE = PGM1.
TITLE = A/B/.
```

RECOVERY Statement

Syntax:

<RECOVERY statement>



Semantics:

The RECOVERY statement declares what type of recovery mechanism (if any) this program undergoes after a system or program failure. Synchronized and data base recovery are for programs that are part of a data base. Queue-restoration recovery is for programs that are not logically associated with any other programs. The default for this statement is NONE. See Section 7 for a detailed explanation of the recovery mechanism.

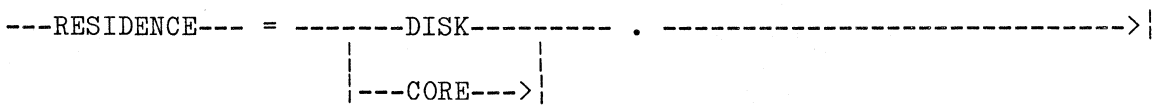
Example:

```
RECOVERY = SYNCHRONIZED.
```

RESIDENCE Statement

Syntax:

<RESIDENCE statement>



Semantics:

The RESIDENCE statement allows the user to specify where the program is to reside when not processing messages. The value of RESIDENCE may be CORE or DISK. A CORE resident program gives faster response but increases the memory requirements of the data communication system. The RESIDENCE statement is optional and defaults to CORE.

Examples:

```
RESIDENCE = CORE.
RESIDENCE = DISK.
```

RESTART PROGRAM Statement

Syntax:

<RESTART PROGRAM statement>

```
---RESTARTPROGRAM--- = ---- <logical value> ---- . ------------------->|
```

Semantics:

The RESTART PROGRAM statement specifies whether or not this program is to be a restart program. If this statement is set to TRUE, then the following must also be done:

1. Recovery must be set to either synchronized or data base.

2. A DATA-BASE NAME statement must be supplied. More than one data-base name is allowed if this restart program services more than one data base, and each data base uses the same type of recovery (synchronized or data base).

Each data base must have exactly one restart program declared, but a restart program can service multiple data bases. By default, RESTARTPROGRAM is set to FALSE.

For information on using the restart program with COBOL74, please see Appendix F.

Examples:

```
RESTARTPROGRAM = TRUE.
RESTARTPROGRAM = FALSE.
```

SUPPRESS GOOD DAY MESSAGE Statement

Syntax:

<SUPPRESS GOOD DAY MESSAGE statement>

---SUPPRESSGOODDAYMESSAGE--- = ---- <logical value> ---- . ---------->

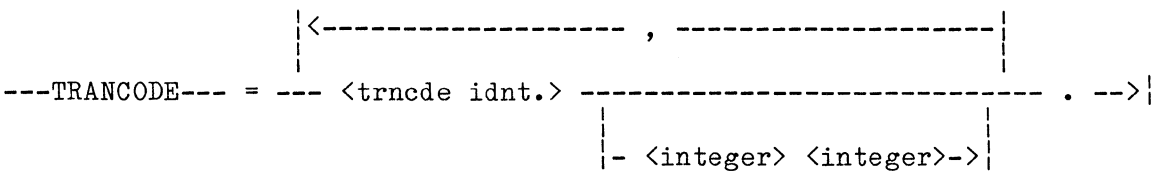
Semantics:

The SUPPRESS GOOD DAY MESSAGE statement is used to prevent a program from receiving the GOOD DAY message at BOJ. A value of TRUE specifies that the program will not receive the "GOOD DAY" message. The default value for this statement is FALSE.

TRANCODE Statement

Syntax:

<TRANCODE statement>



Semantics:

The TRANCODE statement is used to define trancodes and to associate them with programs. A trancode identifier is any string up to ten characters in length. A program of any classification which uses an interface of Participation may have associated trancodes. However, only trancodes associated with user programs cause transaction-based routing to occur. A trancode defined in a TRANCODE statement may occur in the ACCESS CONTROL statement to restrict its use to a specific list of access keys.

The module-function indices may optionally be associated with each trancode. The module-function indices consist of two integer values. Each integer may be a value from 0 to 63. If a trancode has module-function indices, they are placed into the common-area header of messages in which that trancode is present. The receiving program can use the module-function indices in a UPL CASE statement or a COBOL GO TO DEPENDING ON in order to branch to the code which will process that trancode. This eliminates the need for the application program to determine which trancode has just been received. If a trancode has no module-function indices or if there is no trancode in a message, zeroes are placed into the Module-Function Indices field of the common-area header.

A user program must have at least one TRANCODE statement in its PROGRAM statement list. Otherwise, the program never receives any messages. The TRANCODE statement is optional in the PROGRAM statement list of assignment, utility, and pass programs.

NOTE

If input formatting is to take place, a message must have a trancode regardless of the classification of the destination program, so that the MCS is able to determine which format is to be applied. The

trancode is considered as one of the fields of a formatted message.

Examples:

```
TRANCODE = INQ (8, 10), UPDATE (5, 3).
TRANCODE = FIX (18, 1).
TRANCODE = HELP.
```

TRANSACTION CODE POSITION Statement

Syntax:

<TRANCODE POSITION statement>

```
---TRANCODEPOSITION--- = ---- <integer> ----- . ---------------------->|
```

Semantics:

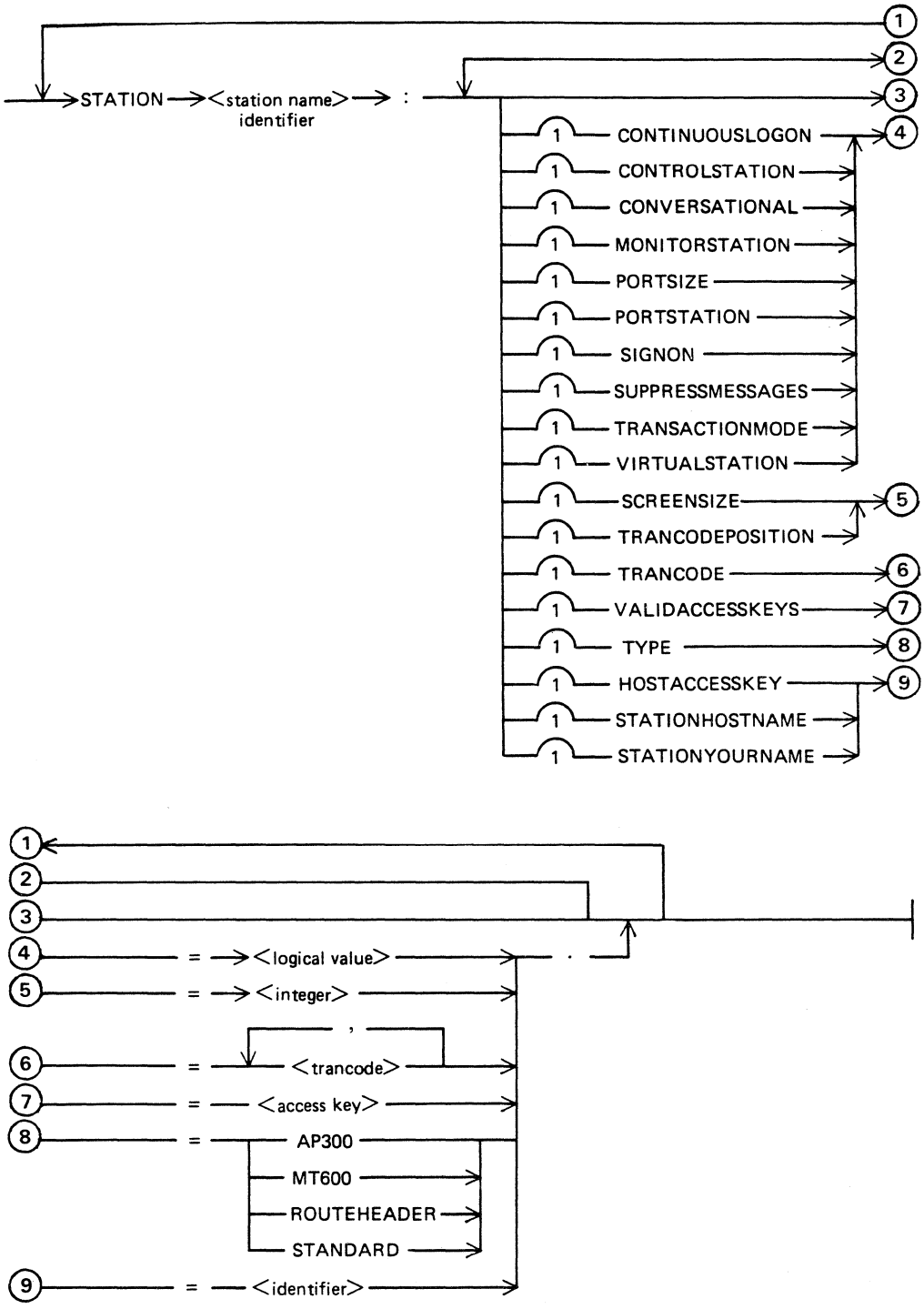
The TRANCODE POSITION statement allows the user to specify where trancodes are found in messages from this program. The position specified in <integer> represents the number of characters after the common-area header.

Examples:

```
TRANCODEPOSITION = 6.
TRANCODEPOSITION = 1.
```

STATION SECTION

The following diagram shows the syntax for the Station section.



Semantics:

The Station section must be present to define various attributes of stations which the MCS is to service. (A GEMCOS MCS opens a remote file whose name is given in the QUEUE NAME statement.) In the FAMILY statement of the File section of the user's NDL source, this remote file was assigned a station identifier list. These are the stations which the MCS services and which must be defined in the TCL Station section.

The Station section is composed of a station define list. Each station define describes one station. The station name must be the station identifier used to refer to that station in the NDL. The STATION statement list is optional. Any of the stations can contain a CONTROL STATION statement and/or a MONITOR STATION statement.

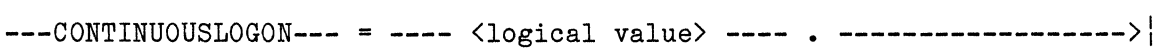
Example:

```
STATION TD800A:
  SIGNON = TRUE.
  SCREENSIZE = 1024.
  TRANCODEPOSITION = 5.
  VALIDACCESSKEYS = ABCD, XXYY, 84080.
STATION TD800B:
  SCREENSIZE = 1920.
```

CONTINUOUS LOG ON Statement

Syntax:

<CONTINUOUS LOG ON statement>



Semantics:

The CONTINUOUS LOG ON statement is used to determine whether the MCS should "remember" who was logged on to the station following a termination of the network (either normally or abnormally). After the network is restarted, if CONTINUOUSLOGON is TRUE for a station, the user would remain logged on. Otherwise, any users who were logged on at the time of failure would be logged off. CONTINUOUSLOGON = TRUE is ignored when auditing is not present in the MCS. By default, CONTINUOUSLOGON is set to FALSE.

Examples:

```
CONTINUOUSLOGON = TRUE.
CONTINUOUSLOGON = FALSE.
```

CONTROL STATION Statement

Syntax:

<CONTROL STATION statement>

```
---CONTROLSTATION--- = ---- <logical value> ---- . ------------------>|
```

Semantics:

The CONTROL STATION statement allows a station to be designated as a control station. Privileged Network Control Commands can be entered from any control station. Any number of stations can be designated as control stations. In the absence of any control stations, only the ODT and the card reader can enter privileged Network Control Commands.

A # 2tion can be changed from a control station to a non-control station (or vice versa) in a GENERATE or REGENERATE run. By default, a station is a non-control station.

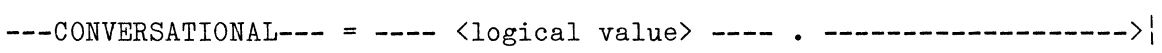
Example:

```
CONTROLSTATION = TRUE.
```

CONVERSATIONAL Statement

Syntax:

<CONVERSATIONAL statement>



Semantics:

The CONVERSATIONAL statement determines whether a station can participate in a conversation. By default, the statement is set to TRUE.

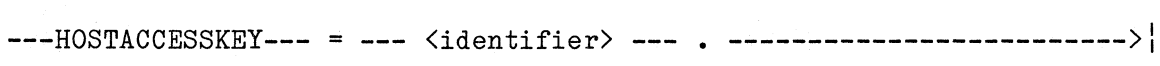
Examples:

```
CONVERSATION = TRUE.
CONVERSATION = FALSE.
```

HOST ACCESS KEY Statement

Syntax:

<HOST ACCESS KEY statement>



Semantics:

The HOST ACCESS KEY statement specifies the name of an access key to be associated with this station. It is only valid if the station's type is ROUTEHEADER. If the corresponding routeheader station on the remote host requires sign on, the specified access key is sent to the remote host as a valid access key of that station. This access key must be included in the VALID ACCESS KEYS statement list in the TCL for the corresponding routeheader station on the remote host.

It is important to note that a host access key received from a corresponding routeheader station on a remote host must be included in the list of valid access keys for this station on the local host.

This access key does not need to be specified in the ACCESS CONTROL statement. The maximum length of the access key is six characters.

Examples:

```
HOSTACCESSKEY = HOST18.
HOSTACCESSKEY = A.
```

MONITOR STATION Statement

Syntax:

<MONITOR STATION statement>

```
---MONITORSTATION--- = ---- <logical value> ---- . ------------------->|
```

Semantics:

The MONITOR STATION statement allows a station to be designated as a monitor station. Errors monitored by the MCS are reported to a monitor station. Any number of stations can be designated as monitor stations. If there are no monitor stations, all system errors are sent to the system ODT.

A station can be changed from a monitor station to a non-monitor station (or vice versa) in a GENERATE or REGENERATE run. By default, a station is a non-monitor station.

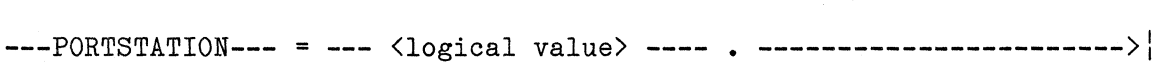
Example:

```
MONITORSTATION = TRUE.
```

PORT STATION Statement

Syntax:

<PORT STATION statement>



Semantics:

When PORTSTATION is set to TRUE, GEMCOS receives input and output through a port file rather than from a data communications station. Use the MY NAME Statement in the Global Section of the TCL to specify the file attributes of the port file GEMCOS uses.

Or use these three statements in the Station Section of the TCL to define the attributes of the port file: the PORT SIZE Statement, the STATION HOST NAME Statement, and the STATION YOUR NAME Statement.

The internal name of the port file GEMCOS uses is HOSTPORT. The default name is GEMPORT. The default value of PORTSTATION is FALSE.

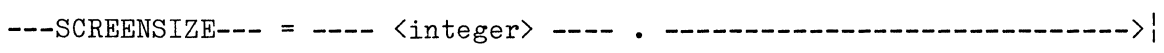
Example:

```
PORTSTATION = TRUE.
```

SCREEN SIZE Statement

Syntax:

<SCREEN SIZE statement>



Semantics:

The SCREEN SIZE statement defines the length of the largest message which may be received by this station. If the MCS determines that a message larger than <integer> characters is bound for the station, the message would be broken into several transmissions until the entire message is sent. The maximum value for SCREEN SIZE is 4096.

CAUTION

When any station define has a SCREEN SIZE statement in its STATION statement list, all station defines must have one. If the SCREEN SIZE statement is not present, no screen wraparound would occur.

The occurrence of a SCREEN SIZE statement causes the screen-wraparound code to be generated into the MCS code file. If no station is defined as having a SCREENSIZE less than or equal to the MAXTEXTSIZE specification, the SCREENSIZE statement should be omitted. The result is a more efficient MCS.

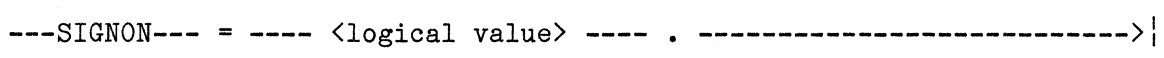
Examples:

```
SCREENSIZE = 1920.
SCREENSIZE = 256.
```

SIGN ON Statement

Syntax:

<SIGN ON statement>



Semantics:

The SIGN ON statement indicates whether a user must sign on at this station prior to entering messages. When SIGNON is TRUE, the operator must sign on with one of the access codes listed in the VALID ACCESS KEYS statement. If VALIDACCESSKEYS is set to ALL, the operator must sign on with one of the access codes listed in the ACCESS CONTROL statement. The SIGN ON statement is optional and, if omitted, defaults to FALSE.

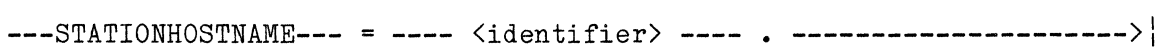
Examples:

```
SIGNON = TRUE.
SIGNON = FALSE.
```

STATION HOST NAME Statement

Syntax:

<STATION HOST NAME statement>



Semantics:

The STATION HOST NAME statement only has meaning for a station with PORTSTATION = TRUE. Set the HOSTNAME attribute of the port subfile GEMCOS uses for this station to <identifier>. The default value of STATIONHOSTNAME is the station name.

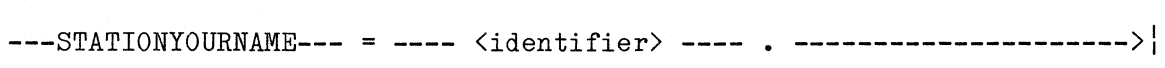
Example:

```
STATIONHOSTNAME = LONDONBASE.
```

STATION YOUR NAME Statement

Syntax:

<STATION YOUR NAME statement>



Semantics:

The STATION YOUR NAME Statement only has meaning for a station with PORTSTATION = TRUE. Set the YOURNAME attribute of the port subfile GEMCOS uses to communicate with the station to <identifier>. The default value of STATIONYOURNAME is NULL.

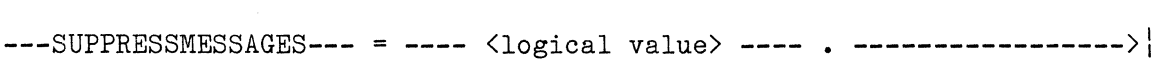
Example:

```
STATIONYOURNAME = LONDON1.
```

SUPPRESS MESSAGES Statement

Syntax:

<SUPPRESS MESSAGES statement>



Semantics:

The SUPPRESS MESSAGES statement is used to prevent a station from receiving certain messages. If this attribute has a value of TRUE, then the station will not receive the following messages:

1. The GEMCOS MCS GOING DOWN message at EOJ.

2. Any message broadcast to all stations (no station list specified).

3. The GOOD DAY message from a program.

4. The THIS STATION RE-ATTACHED TO GEMCOS message sent after reattachment when GEMCOS is running in subordinate mode.

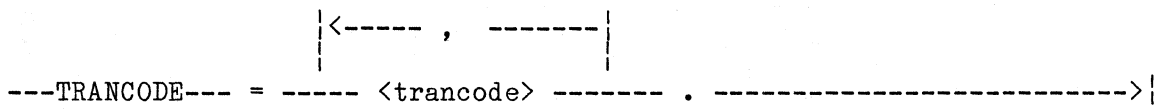
The default value for this statement is FALSE.

Example:

```
SUPPRESSMESSAGES = TRUE.
```

TRANCODE Statement

Syntax: <TRANCODE statement>



Semantics:

The TRANCODE statement is used to define trancodes and to associate them with stations. A trancode identifier is any string up to 10 characters in length. A trancode defined in a TRANCODE statement may occur in the ACCESS CONTROL statement to restrict its use to a specific list of access keys.

By using these trancodes, messages from another station or program can be routed to this station.

NOTE

The module-function indices are not applied to trancodes.

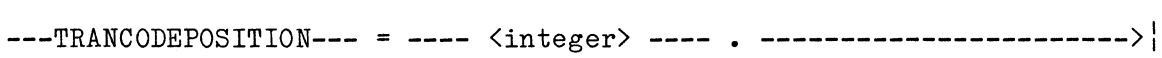
Examples:

```
TRANCODE = STATION, STATION2, HELLO.
TRANCODE = HELP.
```

TRANSACTION CODE POSITION Statement

Syntax:

<TRANCODE POSITION statement>



Semantics:

The TRANCODE POSITION statement allows the user to specify where trancodes are to be found in messages received from this station. By default, TRANCODEPOSITION is 1.

Examples:

```
TRANCODEPOSITION = 5.
TRANCODEPOSITION = 1.
```

TRANSACTION MODE Statement

Syntax:

<TRANSACTION MODE statement>

```
---TRANSACTIONMODE--- = ---- <logical value> ---- . ------------------>|
```

Semantics:

The TRANSACTION MODE statement determines whether a station is allowed to transmit a new input transaction before receiving the response for the previous input transaction. If TRUE, the MCS would return the error response "BUSY" for any input from the station prior to the receipt and transmission by the MCS of the response to the current transaction for the station. Also, this station can only send messages to programs that are declared to use synchronized recovery. If auditing is not present in the MCS, the statement TRANSACTIONMODE = TRUE would be ignored. By default, TRANSACTIONMODE is set to FALSE.

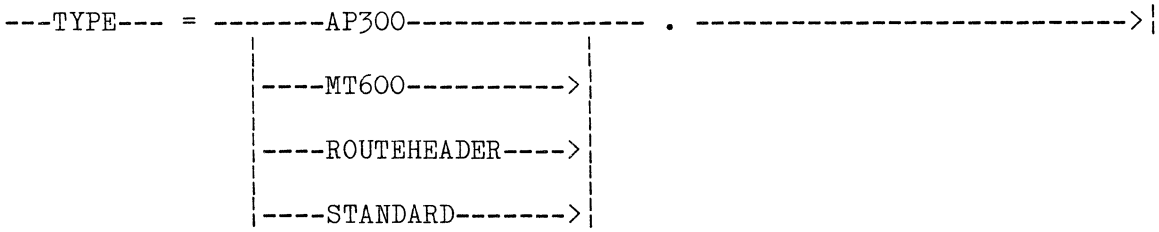
Examples:

```
TRANSACTIONMODE = TRUE.
TRANSACTIONMODE = FALSE.
```

TYPE Statement

Syntax:

<TYPE statement>



Semantics:

The TYPE statement is used to define the physical type of each station. AP300 and MT600 are standard Burroughs terminal devices. ROUTEHEADER indicates that this station is actually a "porthole" to another computer. The GEMCOS MCS on the other computer would contain a corresponding routeheader station. This is the basic component used in computer-to-computer message routing. When ROUTEHEADER is specified, at least one trancode must be defined for this station, and a HOSTACCESSKEY must be specified when the other computer requires sign on on the corresponding station. (Refer to Section 9 for further information about this statement.)

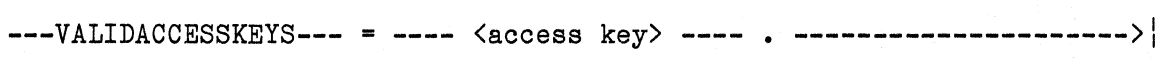
Examples:

```
TYPE = AP300.
TYPE = ROUTEHEADER.
```

VALID ACCESS KEYS Statement

Syntax:

<VALID ACCESS KEYS statement>



Semantics:

A list of valid access keys can be prepared to enforce access key validation at sign-on time. Each access code which appears in a VALID ACCESS KEYS statement must have occurred in the ACCESS CONTROL statement. ALL indicates that any access code can be used to sign on at this station. When the statement is omitted and SIGNON is TRUE, ALL is assumed. This statement has no meaning when SIGNON is FALSE.

Examples:

```
VALIDACCESSKEYS = ALL.
VALIDACCESSKEYS = 84080, 84090, ABCD.
```

VIRTUAL STATION Statement

Syntax:

<VIRTUAL STATION statement>

```
---VIRTUALSTATION--- = ---- <logical value> ---- . ------------------>|
```

Semantics:

The VIRTUAL STATION statement is used to determine whether a station is allowed to transfer to the MCS from another Burroughs system using Burroughs Network Architecture (BNA) Station Transfer.

The stations which are to transfer in to GEMCOS must be declared in the TCL as virtual stations. A virtual station must have a station hostname. An attempt to transfer a station which has not been declared as a virtual station causes Error 156. Set the following attributes (as well as any other attributes needed) for stations transferring in to GEMCOS:

```
STATION <station name>:
    VIRTUALSTATION  = TRUE.
    STATIONHOSTNAME = <host name>.
```

If the statement is set to TRUE, the MCS allows the station to transfer in, provided its hostname matches the hostname defined to the BNA network on the other side of the system. If the statement is set to FALSE, the station is not allowed to transfer in using BNA Station Transfer. The default is FALSE.

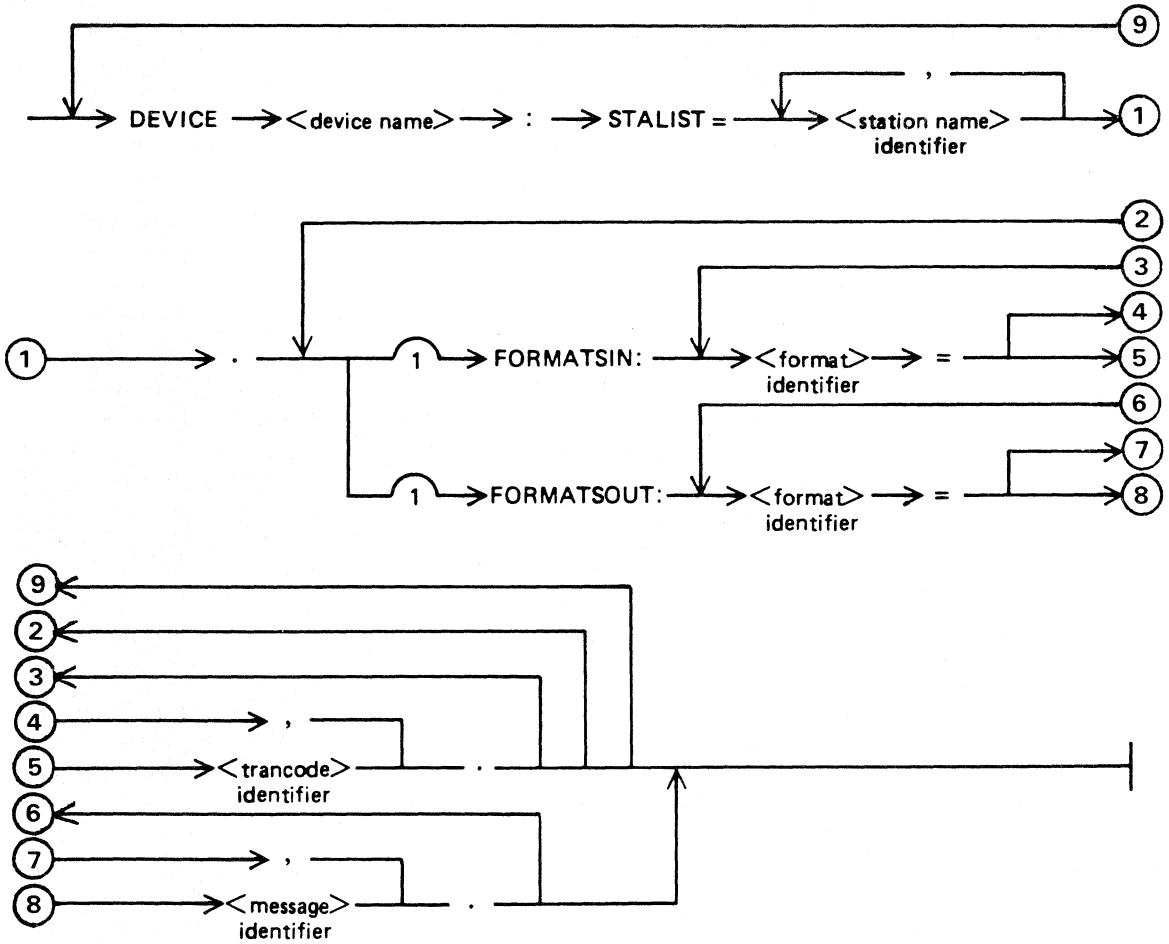
Examples:

```
VIRTUALSTATION = TRUE.
VIRTUALSTATION = FALSE.
```

DEVICE SECTION

Syntax:

The following diagram shows the syntax for the Device section.



Semantics:

The Device section is used to group stations by device class and to indicate which format is to be applied to a message. The Device section should be present only if the FORMAT AND FUNCTION statement list is present in the Global section. The Device section may never occur when the B 1000 GEMCOS release being used is not an Advanced Version.

Each device define consists of a STATION LIST statement, an INPUT FORMATS statement, and an OUTPUT FORMATS statement. The device name may be any identifier and is not referenced elsewhere in the TCL.

In the following example, an input message from TD800A or TD800B with trancode INQ or UPDATE is formatted using format X. A message from TD700A, TD700B or TD700C with trancode INQ or UPDATE has format Y applied.

Similarly, on output, when a program writes a message with the message-ID field of the common-area header set to "PAY" or "SHIP" (left justified with trailing blanks), format X12 is used when the message is bound for TD800A or TD800B. Format Y12 is applied when the message is bound for a station in the device class TD700 (TD700A, TD700B or TD700C). When the message-ID field of the common-area header is set to "RCV", X33 or Y33 is applied, again depending on the destination of the message.

Finally, when an operator transmits "PAY" or "SHIP" (without leading or trailing blanks) from station TD800A or station TD800B, the operator is making a forms request. Using format X12, a message is built up with all blank data fields (A, B, I, J and T). The message is sent to the requesting station. Format Y12 would be used to create the blank-formatted screen if the forms request for "PAY" or "SHIP" came from TD700A, TD700B, or TD700C. If a "RCV" forms request is received, format X33 or Y33 would be applied, depending upon the device classification of the requesting station.

NOTE

Formats X, X12, X33, Y, Y12 and Y33 must have been defined in the FORMAT AND FUNCTION statement list.

Example:

```
DEVICE TD800:
  STALIST = TD800A, TD800B.
  FORMATSIN:
    X = INQ, UPDATE.
  FORMATSOUT:
    X12 = PAY, SHIP.
    X33 = RCV.
DEVICE TD700:
  STALIST = TD700A, TD700B, TD700C.
  FORMATSIN:
    Y = INQ, UPDATE.
  FORMATSOUT:
    Y12 = PAY, SHIP.
    Y33 = RCV.
```

INPUT FORMATS Statement

Syntax:

For the syntax of this statement, see the syntax of the Device Section given previously.

Semantics:

The FORMATS IN statement indicates which format is to be applied to a particular message entered at a station of a particular device class before the message is forwarded to the appropriate program. Only messages with a recognizable trancode are formatted. Each format-ID must be defined in the FORMAT AND FUNCTION statement list, and each trancode must be defined in the Program section. A trancode may be associated with only one format-ID per FORMATS IN statement.

The MCS determines whether an input message is to be formatted by attempting to recognize a trancode in the message text. When a trancode is found, the message is formatted only if the trancode was associated with a format-ID in the device class determined by the station where the message originated.

Example:

```
FORMATSIN:
  MT1 = PAY1, PAY2.
  FMT2 = INV1.
  FMT3 = INV2, INV3.
```

OUTPUT FORMATS Statement

Syntax:

For the syntax for this statement, see the syntax of the Device section given previously.

Semantics:

The FORMATS OUT statement indicates which format is to be applied to a particular message written by a program bound for a station of a particular device class. Only messages with a recognizable message-ID in the Message-ID field of the common-area header are formatted. Each format-ID must be defined in the FORMAT AND FUNCTION statement list. A message-ID can only be associated with one format-ID per FORMATS OUT statement and cannot exceed six characters in length. Station operators can enter a message consisting solely of a message-ID and, in doing so, make a forms request.

When the MCS receives a program message, the Message-ID field in the header is checked. If the field is filled, the contents cause the MCS to format the message according to the format-ID associated with the message-ID in the device class. The device class is determined by the station to which the message is sent.

When the MCS receives a message one to six characters in length without a recognizable trancode from a station, a check is made to determine whether the message consists of a message-ID. If so, the operator entered a forms request. The MCS builds a message with blank data fields using the format-ID with which the message-ID was associated in the device class, determined by the station from which the forms request was received.

Example:

```
FORMATSOUT:
  FOR1 = PAY8, PAY9.
  FMT1 = INV1.
```

STATION LIST Statement

Syntax:

For the syntax of this statement, see the syntax of the Device section given previously.

Semantics:

The STATION LIST statement specifies the stations which are to be considered part of each device class. Each station name must be defined in the Station section, and each station name occurring in the Station section can appear in exactly one STATION LIST statement.

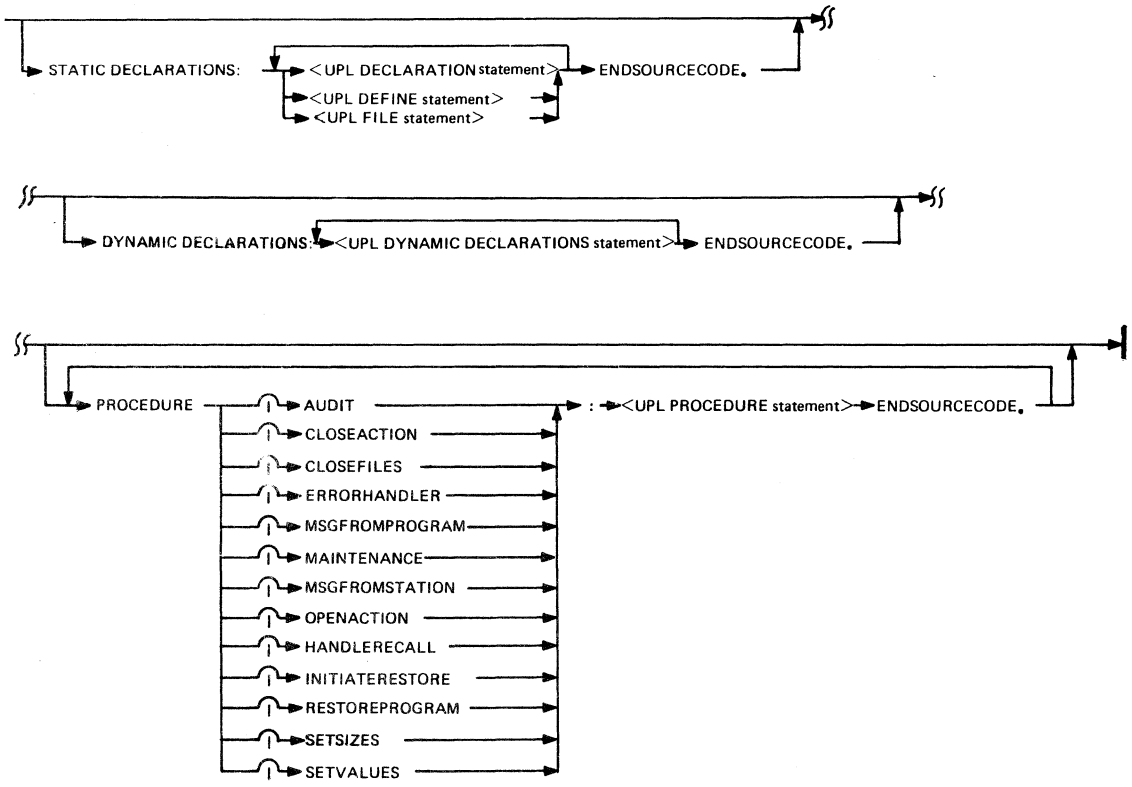
Example:

```
STALIST = TD820A, TD820B, TD820C.
```

MESS CODE SECTION

Syntax:

See the following diagram for the syntax of the MESS CODE section.



Semantics:

The MESS Code section enables the user to include UPL2-mergeable external source statements to supplement or replace GEMCOS MCS functions. Mergeable external source statements are for specialized requirements which demand deviation from the standard GEMCOS logic.

User-written MESS procedures can be merged into key locations in the MCS source code file. The function intended for each procedure is explained, but there is practically no limit to the functions that can be coded. MESS procedures can be inserted in the following code locations:

1. Receiving message from station - for formatting, paging, or routing.

2. Receiving message from program - for formatting, paging, or routing.

3. Processing Network Control Commands - for extending or replacing the capabilities for network control.

4. Auditing - for replacing or supplementing the standard audit feature.

5. Error handling - for extending the standard error-handling logic.

6. Opening - for processing required at system startup.

7. Closing - for processing required at system shutdown, such as closing files used by other MESS procedures.

8. Recalling messages - for disposing of unsent messages when the system is shut down.

9. Initiating recovery - for processing the request of an application program for recovery.

10. Recovery - for replacing the standard recovery logic.

11. Remote file open - for processing that is required when an application program opens a remote file.

12. Remote file close - for processing that is required when an application program closes a remote file.

The source code for MESS procedures is submitted as part of the user's TCL source file. The TCL compiler merges these procedures into the correct places in the MCS logic.

NOTE

> Changes made to the MESS Code section during a REGENERATE MCSTCL run do not affect the MCS source code file.

The MESS Code section consists of static declarations, dynamic declarations, and a procedure define list. The MESS Code section is optional.

Example:

```
STATIC DECLARATIONS:
  DECLARE
       X                          FIXED;
ENDSOURCE.
DYNAMIC DECLARATIONS:
  DECLARE
       SPEC_STRING                CHARACTER(X);
ENDSOURCECODE.
PROCEDURE SETSIZES:
  PROCEDURE MESS_SET_SIZES;
  %
       DECLARE
            ACCEPT_STRING         CHARACTER(5);
  %
       DISPLAY ("ENTER STRING SIZE, XXXXX");
       ACCEPT (ACCEPT_STRING);
       X := BINARY (ACCEPT_STRING);
       END MESS_SET_SIZES;
ENDSOURCECODE.


PROCEDURE SETVALUES:
   PROCEDURE MESS_SET_VALUES;
   %
        SPEC_STRING :=" ";
        END MESS_SET_VALUES;
ENDSOURCECODE.
```

Static Declarations

Syntax:

See the syntax of the MESS Code section for the syntax of the static declarations.

Semantics:

In static declarations the user may make UPL2 global declarations (except dynamic declarations), global defines, and file declarations. The TCL source cards containing UPL2 source statements must be surrounded by a TCL static declarations card and a TCL end card. The static declarations are optional.

If static declarations are present, the NAME-STACK ENTRIES statement and VALUE-STACK BITS statement of the Global section should be set appropriately. GEMCOS combines these values into the UPL2 dollar option, STATIC-MEMORY.

Example:

```
STATIC DECLARATIONS:
  DECLARE
       SECURITY_FILE_OPENED     BIT(1);
  FILE
       SECURITY (TITLE = "MCSSEC",
                 KIND  = DISK,
                 ACCESSMODE = RANDOM,
                 MAXRECSIZE = 80,
                 BUFFERS = 1);
  DEFINE
       F              AS #FIXED#,
       C              AS #CHARACTER#;
ENDSOURCECODE.
```

Dynamic Declarations

Syntax:

For the syntax of the dynamic declarations, see the syntax of the MESS Code Section.

Semantics:

In dynamic declarations, the user may make UPL2 dynamic declarations. The TCL source cards containing UPL2 source statements must be surrounded by a TCL dynamic declarations card and a TCL end card. The dynamic declarations are optional.

If dynamic declarations are present, the NAME-STACK ENTRIES statement and VALUE-STACK BITS statement of the Global section should be set appropriately.

Example:

```
DYNAMIC DECLARATIONS:
  DECLARE DYNAMIC
       WORK_AREA       CHARACTER(AL.NPR_MAX_TEXT_SIZE+250);
ENDSOURCECODE.
```

Procedure Define List

Syntax:

See the syntax of the MESS Code section for the syntax of the procedure define list.

Semantics:

The procedure define list contains user-coded UPL2 procedures, one per procedure define. Each procedure define begins with a TCL procedure introduction card, follows with the user's source statements, and ends with a TCL end card. The MESS procedure-ID of the procedure introduction card specifies to the TCL where the user's code is to be merged. The procedure define list is optional.

If the procedure define list is present, the NAME-STACK ENTRIES statement and VALUE-STACK BITS statement of the Global section should be set appropriately to reflect the space required for variables declared within any of the user-written procedures.

A discussion of each of the MESS procedures is given below.

Example:

```
PROCEDURE MSGFROMSTATION:
  PROCEDURE ZIP_IT BIT(1);
  %
  IF AD.MSG_TEXT_SIZE  >  3
    THEN
      IF SUBSTR(SG.TEXT,0,3) = "ZIP" THEN
      RETURN (1);
  RETURN (0);
  END ZIP_IT;
ENDSOURCECODE.
```

MESS Procedures

This discussion explains when MESS procedures are called, the parameters required to call them, and the values they must return. All MESS procedures are optional.

Care must be taken when writing MESS code to avoid duplicating identifiers already used in the MCS. For this reason, it is useful to know that all MCS identifiers adhere to the following conventions:

1. DEFINEs begin with three characters: "MD_" or "MS _".

2. Files begin with three characters: "MCS".

3. Data names begin with three characters: "XX.", where XX is two alpha characters of the global declarations. Use the 2-character prefix shown at the beginning of the GEMCOS source file.

4. Procedure names begin with four characters: "MCS_".

5. DO-group labels begin with three characters: "ML_".

MESS procedures have access not only to entities declared in static declarations, dynamic declarations, and locally, but also to many data areas, files, and procedures used by the standard MCS modules according to the scope rules of UPL2.

When programming MESS code, segmenting procedures must be followed. Some efficiency in memory allocation may be realized if MESS procedure segments are approximately 800 to 1200 bytes in size, the average size of standard MCS modules.

If global or local variables are declared by the user in the MESS Code section, the NAME-STACK ENTRIES statement and VALUE-STACK BITS statement of the Global section should be set appropriately.

AUDIT

This procedure can either supplement or replace the standard auditing logic. It is called after the standard audit procedure (if generated) is executed. Any files needed by AUDIT must be declared in the Global section of MESS code.

AUDIT must accept one parameter:

An indicator [BIT (1)] which denotes the type of message being sent from the MCS:

1. A value of 0 indicates that the message is bound for a station.
2. A value of 1 indicates that the message is bound for a program.

CLOSE ACTION

This procedure is intended to supplement the MCS remote file close or STATIONDETACH logic. It is called after the MCS performs the necessary steps to verify that a program is no longer on-line. The CLOSEACTION procedure must be a function procedure which returns a value [BIT (1)]. However, the value of the return is of no consequence and is reserved for future use. One useful function of the CLOSEACTION procedure might be to construct and send a notification of the FILE CLOSE to the stations involved.

When the action taken by the CLOSEACTION procedure depends upon whether it is called through a remote FILE CLOSE or through a STATIONDETACH, the source of the call can be determined by checking the data field MS.MSG.HDR.TYPE. When this field contains 16, the CLOSEACTION procedure has been called through a FILE CLOSE. When it contains any other value, it has been called through a STATIONDETACH.

CLOSE FILES

The CLOSEFILES procedure is given control during system shutdown (EOJ). Its primary purpose is to close any files that may have been opened in other MESS routines. This is called only by the MCS.EOJ procedure. It is not called unless the generation parameter SYS-HALT is set.

ERROR HANDLER

The primary function of this procedure is to supplement the standard error handling module of the MCS. As a secondary function, it may also save the contents of MS.MSG.WORK.AREA in case of a data dump.

For certain system errors, a data dump is created. The standard error handling logic induces the dump by synthesizing an RDM Network Control Command in MS.MSG.WORK.AREA. The message that was stored in MS.MSG.WORK.AREA when the error was detected is therefore lost, unless the ERRORHANDLER procedure saves it.

This routine must accept one parameter: the error message number [CHARACTER (3)] which corresponds to the error detected.

ERRORHANDLER must be a function procedure that returns a value [BIT (1)] which tells the MCS what to do with the error condition:

1. A value of 0 (zero) indicates that the error should be processed normally.

2. A value of 1 indicates that the MCS is to exit the error handling module immediately.

The ERRORHANDLER procedure is called by MCS.PRINT.ERROR.

HANDLE RECALL

The HANDLERECALL procedure is given control during system shutdown (EOJ). At this time, the MCS recalls all messages which have been routed to a station, but are still awaiting transmission in the queues of the Network Controller. The HANDLERECALL procedure is invoked each time a message returns to the MCS.

The HANDLERECALL procedure must be a function procedure which returns a value [BIT (1)] specifying disposition of the message:

1. A value of 0 (zero) causes the MCS to print the message on a line printer before discarding it.

2. A value of 1 causes the MCS to discard the message without further processing.

If this procedure is not provided, the MCS prints all recalled messages before discarding them.

This procedure is called only by MCS.EOJ. It is not called unless the generation parameter SYS-HALT is set.

INITIATE RESTORE

This procedure is given control when an application program indicates to the MCS that it needs restoration (by sending a message which has the MCSTYPE field of the common-area header set to 20). It may either supplement or replace the module that performs restoration initialization, depending upon the RESTORATION statement. Its purpose is to perform any initialization that may be necessary to prepare for restoration.

MAINTENANCE

This procedure is given control when a message containing the signal character is received from a station or a program (except for valid SGN messages). The MAINTENANCE procedure is called from the standard maintenance module (MCS_MAINT_CONTROLLER), if generated.

The standard maintenance module is generated if either CHANGEREQUESTS, DATADUMP, MESSAGEBROADCAST, MESSAGERECALL, PROGRAMBOJEOJ, STATUSREPORTS, SYSTEMHALT, RESTORATION, or MONITORTRACE is TRUE, or if an ACCESS CONTROL statement is present. When the standard maintenance module is not generated, the MAINTENANCE procedure is invoked by MCS_MSG_FROM_STATION or MCS_MSG_FROM_USER_PROGRAM.

The MAINTENANCE procedure must be a function procedure which returns a 1-bit value specifying the disposition of the message:

1. A value of 0 (zero) indicates that MAINTENANCE did not process the message. If the standard maintenance module is present, it attempts to scan for a standard command and process it. If the standard maintenance module is not present, the input is ignored.

2. A value of 1 causes the MCS to consider the message completely processed whether the standard maintenance module is present or not.

MAINTENANCE must accept one input parameter: the number of the calling procedures (FIXED).

This procedure can be used to implement new Network Control Commands or change existing commands.

MESSAGE FROM PROGRAM

This procedure is called when the MCS receives a message from an application program (the MCS does not receive messages from programs using the Nonparticipation or MCS interface). The message was not formatted (Advanced Version), nor was it audited. The message may be a request for message restoration or may contain a signal character.

MSGFROMPROGRAM must be a function procedure which returns a 1-bit value specifying the disposition of the message:

1. A value of 0 (zero) informs the MCS to continue processing the message as if there had not been a MSGFROMPROGRAM procedure.

2. A value of 1 means that the message was completely processed. The MCS exits immediately to the main control procedure and does not process this message any more.

This procedure can be used to interpret the USERAREA field (MS_COMMON_USER) of the common-area header. MSGFROMPROGRAM can perform specialized output formatting. Nonstandard routing (e.g., program-to-program message switching) can be performed using this procedure.

MESSAGE FROM STATION

This procedure is called when a message is received from a station. If MSGFROMSTATION is given control, it may assume that there is no data-communication error associated with the message, the MCS is not being shut down, the source station is signed on (if sign-on is required at that station), and the message does not begin with the signal character. The common-area header to be associated with this message has been built in MS_COMMON__AREA <length> (but not yet attached to the message). When a trancode is present, it has been recognized and noted in AD.TRN_INDEX. The message has not yet been formatted and could be a forms request (Advanced version only). The message has not yet been audited.

MSGFROMSTATION must be a function procedure which returns a 1-bit value specifying disposition of the message:

1. A value of 0 (zero) directs the MCS to continue processing the message as if there had been no MSGFROMSTATION procedure.

2. A value of 1 signifies that the MSGFROMSTATION procedure has taken full responsibility of the message. The MCS discontinues processing this message.

If a station is associated with or was attached to the remote file of a program that uses a Nonparticipation or MCS interface, the GEMCOS MCS does not receive any messages from that station. Thus, the MCS cannot pass control to MSGFROMSTATION for such messages.

This procedure can set fields in the USERAREA (AB.COMMON_USER) of the common-area header. It can perform specialized routing, access control, or input formatting, and thus be used for data collection.

OPEN ACTION

This procedure is intended to replace or supplement the action taken by the MCS after a FILE OPEN STATION ATTACH is approved. Normally, the MCS sends a "good day" message to each station in the newly-opened file and, if specified in the TCL, a FILE OPEN notification is sent to the program.

The OPENACTION procedure must be a function procedure which returns a value [BIT (1)]:

1. A value of 0 (zero) causes the MCS to send the "good day" messages and the open notification.

2. A value of 1 causes the MCS to skip the code which sends the "good day" messages and the open notification.

When the action taken by the OPENACTION procedure depends upon whether it is called through a FILE OPEN or through a STATION DETACH, the source of the call can be determined by checking the data field MS.MSG.HDR.TYPE. When the field contains 10, the OPENACTION procedure has been called through a FILE OPEN. When it contains any other value, it has been called through a STATION ATTACH.

RESTORE PROGRAM

This procedure is intended to replace the standard MCS restoration logic. It is called from the main processing loop in MCS.MODULE.MANAGER. It is called once in each iteration of the loop as long as the flag MS.RESTORE.PROGRAM has a value of 1. When it is necessary to handle other network activity during restoration, MESS.RESTORE.PROGRAM must relinquish control (that is, RETURN) occasionally, so that the main processing loop can run through another cycle.

SET SIZES

This procedure is given control during the first phase of initialization logic in the MCS. Its purpose is to specify sizes for any dynamic variables declared in the dynamic declarations.

Example:

Assume that static declarations include:

```
DECLARE   MAX_SIZE                FIXED;
```

and that dynamic declarations include:

```
DECLARE DYNAMIC USER_AREA        CHARACTER(MAX_SIZE);
```

Then the SETSIZES routine must be provided. The procedure define list> might include:

```
PROCEDURE SETSIZES:
  PROCEDURE MY_SETSIZES_ROUTINE;
    MAX_SIZE := 100;
  END MY_SETSIZES_ROUTINE;
ENDSOURCECODE.
```

The SETSIZES MESS procedure is called from MCS_INITIATE_SIZES.

SET VALUES

This procedure is given control during the second phase of initialization logic in the MCS. The purpose is to specify values for variables that were declared in the static declarations.

Suppose the static declarations include the following:

```
DECLARE MAX.VALUES      FIXED,
        USER.DATA       CHARACTER(10);
```

In this case, the SETVALUES routine must be provided. The procedure define list can include the following:

```
PROCEDURE SETVALUES:
    PROCEDURE MY_SETVALUES_ROUTINE;
        MAX_VALUES := 200;
        USER_DATA := "SOMETHING";
    END MY_SETVALUES_ROUTINE;
ENDSOURCECODE.
```

The SETVALUES MESS procedure is called from MCS_INITIALIZE_TABLES.

Once all the parameters for an MCS have been set up in the TCL, the MCS can be executed.

BEGINNING SYSTEM OPERATION

The following gives information on executing an MCS, and on console or card reader input.

EXECUTING AN MCS

Before executing the MCS, the MCSTIC file must be on disk (MCSTIC is created by the TCL compiler). To initiate the MCS, enter:

```
EX MCSSRC/OBJECT
```

If an OBJECT-CODE FILE NAME statement is present in the Global section of the source TCL, enter:

```
EX <file-ID>
```

If the user wants to rename MCSQUEUE by using a FILE statement following the EXECUTE command, the value of SWITCH 7 determines what occurs. If SW7 = 0, the MCS opens the remote file whose name was given in the QUEUE NAME statement.

If SW7 = 1, the MCS opens the remote file whose internal file name is MCSQUEUE.

The external file name of MCSQUEUE can be changed in the EXECUTE statement or with the MODIFY command. This allows the user to override the queue name specified in the TCL. To do this, enter the following:

```
EX <GEMCOS-ID> ; SW7 = 1;
   FILE MCSQUEUE NAME <remote file-ID>
```

Under certain circumstances, a Network Controller may have to be executed.

EXECUTING A NETWORK CONTROLLER

When an MCS is run under system software released prior to 6.1, or when the C entry of the table name is blank, a Network Controller needs to be executed. (This only needs to be done if a Network Controller is not already running.)

When a system software release of 6.1 or later is used, and there is a nonblank C entry in the name table, the system automatically executes the Network Controller. (For a description of the system name table entries, refer to the B 1000 System Software Operational Guide.)

Once the GEMCOS MCS has begun, the user can start application programs and any subordinate MCS programs.

CONSOLE OR CARD READER INPUT TO THE MCS

A Network Control Command may be presented to the MCS at any time by entering an ACCEPT command directly to the MCS. For example, the following message entered at the supervisory console makes STATIONA not ready.

```
<m-n>AX*CSR STATIONA N
```

The <m-n> is the mix number of the MCS, and the asterisk (*) is the signal character. No spaces between AX and the signal character are permitted.

To enter Network Control Commands from a card reader, the operator should ready a card reader with a deck labeled MCSOLICRD and enter:

```
<m-n>AXCARDS
```

The MCSOLICRD deck should contain one Network Control Command per card. The signal character must be in card column one.

The following section, Section 3, contains additional information on using Network Control Commands.

SECTION 3

USING NETWORK CONTROL COMMANDS

This section discusses the functions of Network Control Commands, and provides the syntax for each command. A control station administers the network through Network Control Commands (NCCs). They are used for the following functions:

1. Access Control:

   a. Enable and disable users.
   b. Sign on and off.

2. MCS control.

3. Message Control:

   a. Reroute messages.
   b. Retrieve queued messages.
   c. Send messages to other stations.

4. Program Control:

   a. Execute and terminate application programs.
   b. Report program status.

5. Station status:

   a. Report station status.
   b. Change station status.
   c. Use port stations.

Every Network Control Command generates some kind of response. There are three kinds of responses:

1. Confirmation without data. This response is specified by the user through the NCC OK RESPONSE statement.

2. Confirmation with data. For Network Control Commands which request that data be returned, the data itself serves as confirmation that the command was executed.

3. Rejection. If a command was not successfully executed, a message is returned giving the reason.

A Network Control Command (NCC) consists of a signal character, a short mnemonic command code, and in some cases, one or more parameters. Commands are free in form, with words separated by one or more spaces.

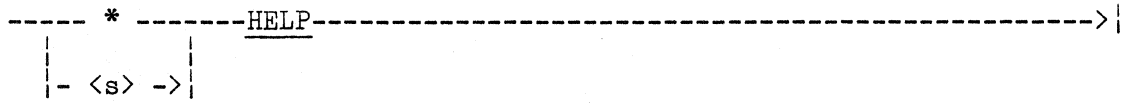
The user may define a signal control character using the Signal Character statement in the TCL. The default for the character is an asterisk (*). In the following diagrams, the signal character option is shown by <s>.

Railroad diagrams are used to show the syntax of the Network Control Commands. Instructions for reading the diagrams are in Appendix G.)

USING THE HELP COMMAND

The HELP command gives information about Network Control Commands. When the user enters this command, a list of Control Commands and their meanings are displayed on the user's terminal.

Syntax:



Semantics:

The HELP command produces a list of the Network Control Commands and a brief explanation of their meanings. Restricted commands are indicated with an asterisk (*). Since a carriage return character (@OD@) terminates each line, nonscreen devices can also receive a formatted HELP screen. Note that the output can consist of several messages, depending on the value of MAXTEXTSIZE.

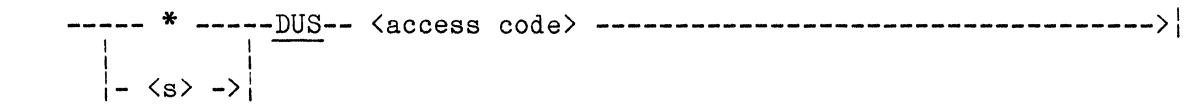
SECURITY CONTROL COMMANDS

These commands can be used only if an ACCESS CONTROL statement appears in the user source TCL.

DISABLE USER (DUS)

The DUS command is used to prevent an access code from being used for logging on.

Syntax:



Example:

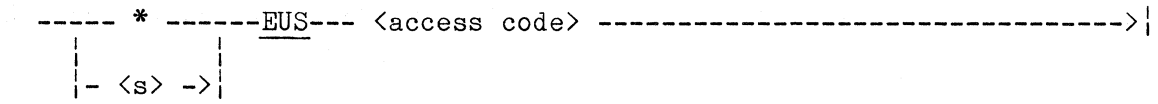
```
@ DUS ABCD
```

In this example, the user ABCD is no longer able to sign on until the access code is enabled again.

ENABLE USER (EUS)

The EUS command is used to mark an access code as enabled. The enabled user-ID, with the correct password, can then be used to sign on.

Syntax:



Example:

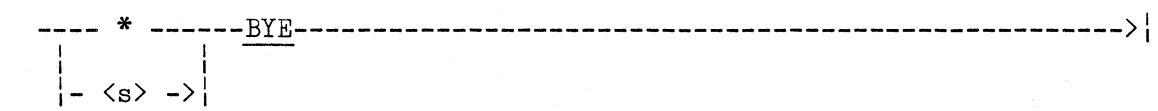
@ EUS ABCD

In this example, the user ABCD may sign on.

SIGN OFF (BYE)

The BYE command disconnects a signed-on user from a station. The user should sign off after completing the transaction to ensure that no unauthorized person is able to gain access to the system. BYE cannot be entered from a station which does not require signing on. If the user is attached to a Utility Program (via the EX command) at the time "*BYE" is entered, an implicit HAP of that utility program is automatically done by the MCS.

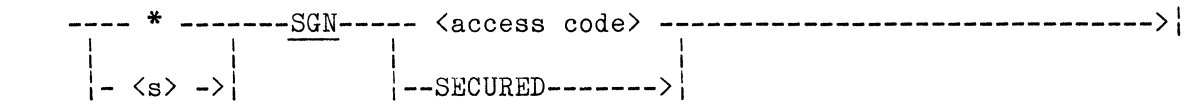
Syntax:



SIGN ON (SGN)

The SGN command is used to gain access to the system at a station which requires signing on.

Syntax:



Examples:

@ SGN ABCD

In this example, the ABCD would be signed on if it is defined in the ACCESS CONTROL statement, and is valid for that station.

@ SGN SECURED

In this example, the user is not actually signing on, but is requesting that GEMCOS return a pre-formatted sign-on screen where the user need only enter a valid access code and transmit. The entered access code will not be visible on the screen. This option is only valid for TD830 or MT983 terminals, or any terminal that supports the TD830 highlight characters.

UPDATE ACCESS KEYS (UPD ACCESSKEY)

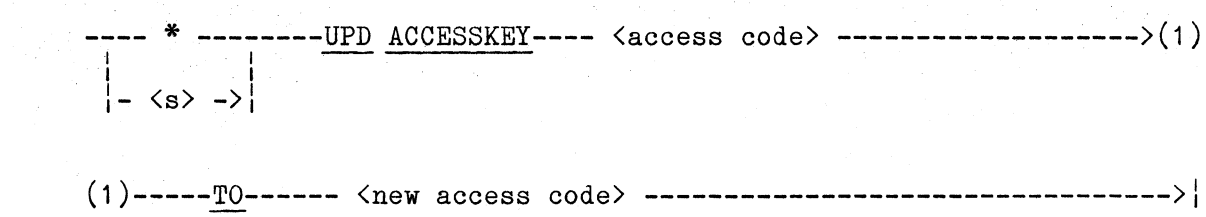
The UPD command also allows the user to update or change existing access codes. The change is permanent, since the MCSTIC file is rewritten with each change. Only access codes which are not signed on may be changed. Attempting to change an access code which is signed on causes an error message to be displayed.

After the entry of a correct UPD command, the GEMCOS MCS displays a message on the ODT. This message states the old access code, the new access code, and the station making the change.

Since the change is permanent, the user's TCL no longer contains the correct access codes. To obtain an up-to-date listing of the access codes, the TCL compiler may be run with CONTROL = REPORT. The UPD command must be entered at a control station.

Note that the UPDATE command can also be used to update or change the names of port stations. Please see Port Stations Commands at the end of this section.

Syntax:



Semantics:

This command changes an existing access code to a new access code.

Examples:

```
$ UPD ACCESSKEY SALES TO SELLIT
@ UPD ACCESSKEY ROBERT TO BOB
```

STATION ATTACHMENT COMMANDS

The following commands attach and detach stations from GEMCOS.

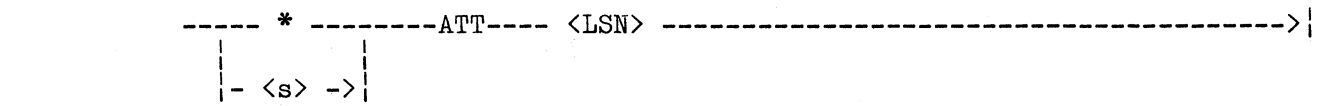
ATTACH LSN (ATT)

When GEMCOS is running subordinate to another MCS, this command allows output-only devices (such as AP1300's) to be attached to GEMCOS. This command is used to attach a station to GEMCOS which has been attached previously to the primary MCS. Please note that the ATT command does not appear in the list of commands on the HELP screen.

First, attach the station to GEMCOS by entering the SMCS ATTACH command. This command is found in the SMCS manual. (See Introduction for the form number of this manual.) Then enter the GEMCOS ATT command. The ATT command causes GEMCOS to recognize the station.

If the station is not defined in the TCL or is attached to another program, GEMCOS returns the message: INVALID LSN <entry>.

Syntax:



Semantics:

Use the logical station number (LSN). The station name is invalid. To detach the station, use the DFR command. Do not use the SMCS DETACH command.

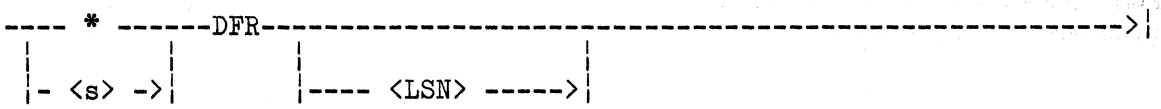
Example:

```
&ATT 9
```

DETACH FROM REMOTE FILE (DFR)

This command is used to detach a station from the MCS remote file. This command is valid only if the station was subsequently attached to the MCS remote file; this means that the station was not given to the MCS during the initial remote file open, but attached later. In order to detach from the remote file, the station cannot be attached to any program. The logical station number (LSN) option should be used to detach statio attached with the ATT command.

Syntax:



Semantics:

If no logical station number (LSN) is entered, the station from which the command is entered will be detached from the MCS remote file.

Example:

```
@ DFR
* DFR 7
```

PROGRAM CONTROL COMMANDS

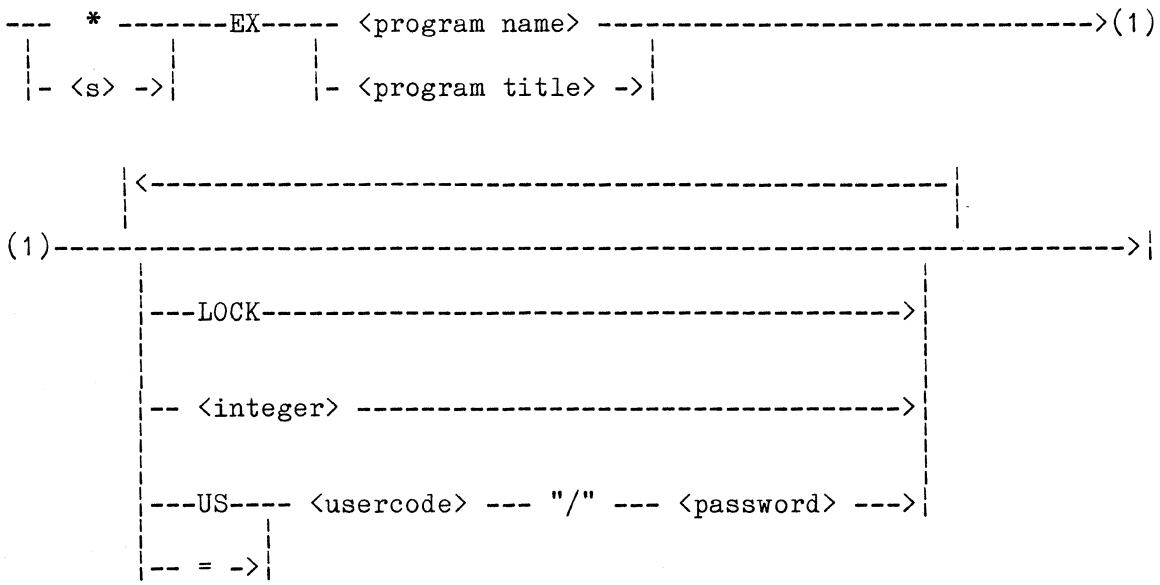
These commands can be used only if PROGRAMBOJEOJ = TRUE.

EXECUTE PROGRAM (EX)

The EX command is used to start assignment, user, utility, or pass programs. It is also used to attach a station to a utility program which is already running. Either the program name or program title may be used in the EX command. In both cases, the MCS will zip-execute the program title. Confirmation of this command merely indicates that the MCS communicated ZIP-execute to the MCP; it does not guarantee that the program actually started.

Syntax:

<EXECUTE command>



Semantics:

The execute options are described in detail in the B 1000 System Software Operational Guide. If the user-password option was entered during a normal run, it will not be present in the ZIP-execute statement used for recovery.

NOTE

GEMCOS does not check for valid usercodes. If the system is unsuccessful performing ZIP-execute with the usercode, the user must initiate the program at the ODT.

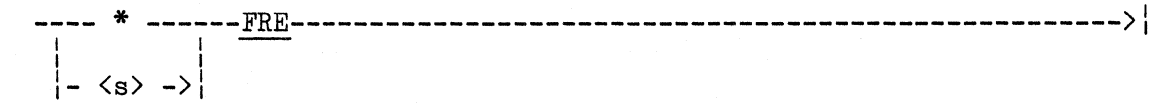
Examples:

```
@ EX A
@ EX PROG/A
@ EX PROG/A 12345
@ EX PROG/A US AB/CD
@ EX PROG/A LOCK US = AB/CD
@ EX PROG/A US AB/CD 12345 LOCK
```

FREE STATION FOR EXECUTION (FRE)

The FRE command is used to free a station that has become locked due to the failure of a utility program.

Syntax:



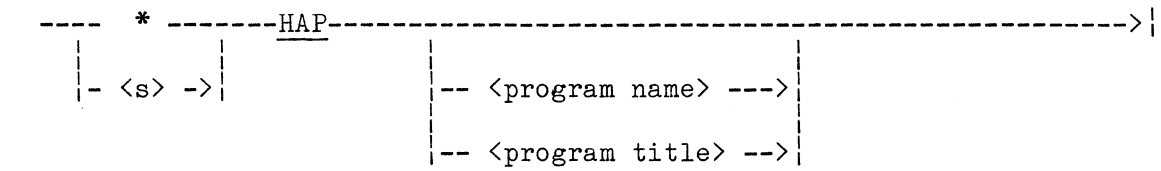
Semantics:

This command can be entered from any station that has become "locked." This condition arises whenever a utility program is executed and fails to open its remote file successfully. When this happens, a further attempt to execute any utility program causes error 151. The FRE command clears the station and allows a utility program to be executed again. This command should be entered only when it is certain that the executed utility program has failed.

HALT APPLICATION PROGRAM (HAP)

The HAP command is used to cause an end-of-file condition on the remote file that an assignment, user, or pass program has opened. In the case of utility programs, the station at which the HAP command was entered is detached from the program. When the last station detaches itself, an end-of-file condition is sent to the utility program. Program-name is optional only for utility programs. If this command is entered during a conversation at a station, the conversation is automatically terminated, and the conversation area is cleared.

Syntax:



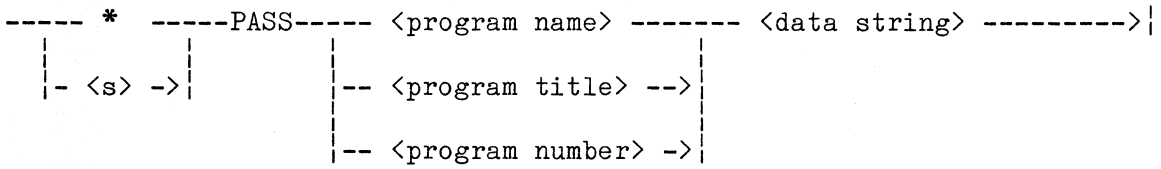
Example:

```
@ HAP PROG/A
@ HAP
@ HAP A
```

PROGRAM PASS (PASS)

The PASS command is used to send messages to a utility, assignment, or pass program from a station not attached to the program.

Syntax:



Semantics:

The program name, title, or number is assigned in the Program section of the TCL.

For a utility program, the purpose of the PASS command is to allow communication with the utility program without having to be attached to the program through the EX command. The program specifier must refer to a utility program that is currently running. The data string is passed to the program as is. To use the PASS command, at least one station in the GEMCOS network must be attached to that utility program which is to receive the data string. All output messages generated by the program are routed back to the station of origin.

For an assignment program, the semantics are the same as those for a utility program. The purpose of allowing a station to pass to an assignment program is to make it easier to use remote-print programs. An assignment program can open a named remote file which contains only remote-printer stations. Input specifications can then be passed to the program from other stations (e.g., TD 830s). These other stations do not have to be attached to the program.

For a pass program, no stations need to be attached to the program. Any user can pass data to any pass program from any station in the GEMCOS network, as long as there are no security restrictions. If a pass program is not currently running, and it also has the EXECUTE option set to ONDEMAND in the TCL, then the first PASS command for that program causes GEMCOS to zip-execute the program automatically.

Examples:

```
@ PASS CANDE ?WHERE ALL
@ PASS RD KB
```

MCS CONTROL COMMANDS

There are two MCS Control commands: AUDIT OK (AOK) and HALT SYSTEM (HLT).

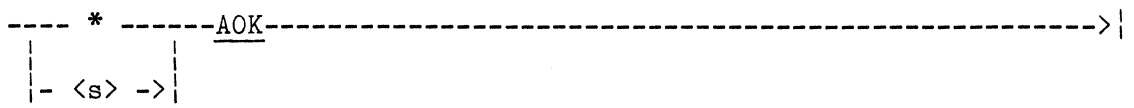
AUDIT OK (AOK)

The AOK command is used in response to a message on the control station or on the console printer of the form:

```
FILE MISSING - MCSAUDIT/AUDITXXX
```

It informs the MCS that the requested audit file is available on disk. This command can be entered only at the console printer through an ACCEPT.

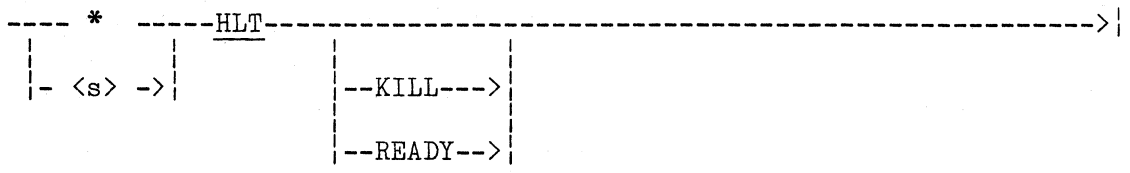
Syntax:



HALT SYSTEM (HLT)

The HLT command brings the data communications system to a stop. This command can be used only if SYSTEMHALT is TRUE.

Syntax:



Semantics:

When KILL is not specified, the system comes to an orderly stop, and untransmitted messages are recalled. When KILL is specified, the system comes to an abrupt stop, and messages may be lost.

When READY is specified, the system comes to an orderly stop, and the stations are made ready. This is helpful when running multiple MCSs.

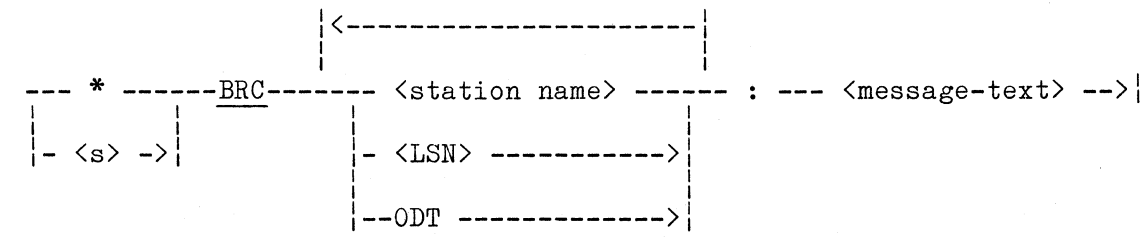
MESSAGE CONTROL COMMANDS

There are two MCS control commands: BROADCAST (BRC) and POPQUEUE (PQ).

BROADCAST (BRC)

The BRC command is used to send a message to other stations in the network. It is available only if MESSAGEBROADCAST is TRUE.

Syntax:



Semantics:

The station name or logical station number (LSN) is assigned in the Station section of the NDL. More than one station name or LSN may be entered, in which case the message is sent to each station. If no station specifier is entered, the message is sent to all stations in the network. If ODT is entered, the message is sent to the console printer.

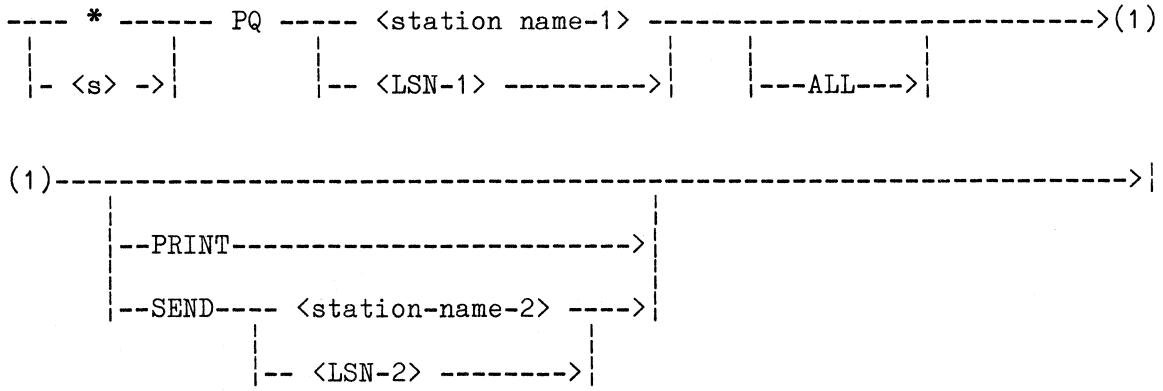
Examples:

```
@ BRC : GOOD MORNING
@ BRC 3 TD4 : WHAT'S HAPPENING?
@ BRC ODT : PLEASE LOAD PROGRAM BLACK/JACK
```

POP QUEUE (PQ)

The PQ command is used to recall messages from the output queue of a station. This command can be used only if MESSAGERECALL is TRUE.

Syntax:



Semantics:

The station name is assigned in the Station section of the NDL. The logical station number (LSN) represents the position of the station definition in the Station section of the NDL.

If ALL is not entered, only the oldest message would be recalled. If ALL is entered, then all messages in the queue would be recalled in order, with the oldest one first. If neither PRINT nor SEND is specified, recalled messages would be discarded. PRINT causes recalled messages to be printed on a system printer. SEND causes recalled messages to be sent to the indicated station.

Examples:

```
@ PQ 5
```

In this example, one message from the queue for station 5 is retrieved and discarded.

```
@ PQ 3 ALL PRINT
```

In this example, all messages for station 3 are recalled and printed.

```
@ PQ TD1 SEND TD2
```

In this example, one message for TD1 is sent to TD2 instead.

REPORT COMMANDS

With the exception of REPORT DATA DUMP, which is controlled by the DATADUMP statement, these commands are controlled by the STATUS REPORTS statement.

When examining error statistics it should be kept in mind that:

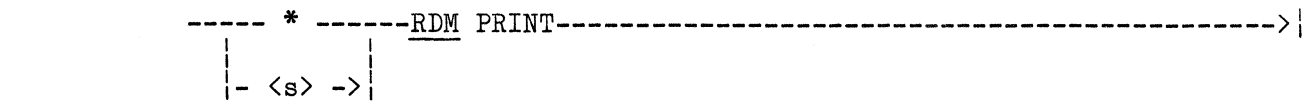
1. Counters start at zero each time the MCS is executed.

2. The MCS increments a counter by the retry limit when the Network Controller reports an error to the MCS, but the Network Controller reports an error only when the retry limit is exceeded. Thus, the number of errors reported in the response to the Network Control Command may be slightly less than the number that actually occurred.

REPORT DATA DUMP (RDM)

The RDM command allows access to the contents of MCS data fields.

Syntax:



Semantics:

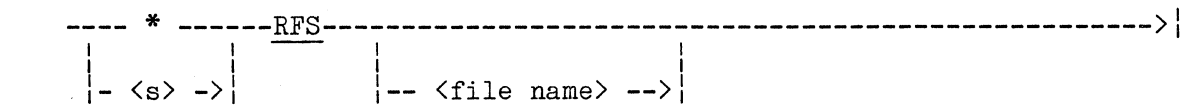
When PRINT is specified, the report is sent to a system printer and the contents of MCS tables are included. When PRINT is not entered, only information that is not in tables is included. Other report commands are available for displaying table information at a remote station.

REPORT FILE STATUS (RFS)

The RFS command returns the following information about a remote file:

1. File name.
2. Queue number of the remote file.
3. Name of the program which opened the remote file.

Syntax:



Semantics:

If a specific file name is not entered, the status of all remote files is returned.

Example:

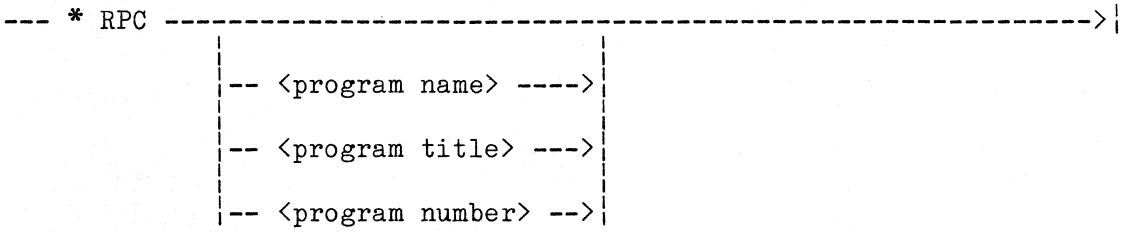
```
@ RFS REMOT1
```

REPORT PROGRAM COUNTERS (RPC)

The RPC command returns the following information about a program:

1. The number of the program.
2. Number of messages sent to the program.
3. Number of messages received from the program.
4. The job number of the program, if it is running.

Syntax:



Semantics:

The program name, program title, or program number is assigned in the Program section of the TCL. If neither a program name, a program title, nor a program number is entered, the status of all programs is reported. If there is more than one copy of a program (MAXCOPIES > 1), statistics are given for all copies of the program, including copy number.

Examples:

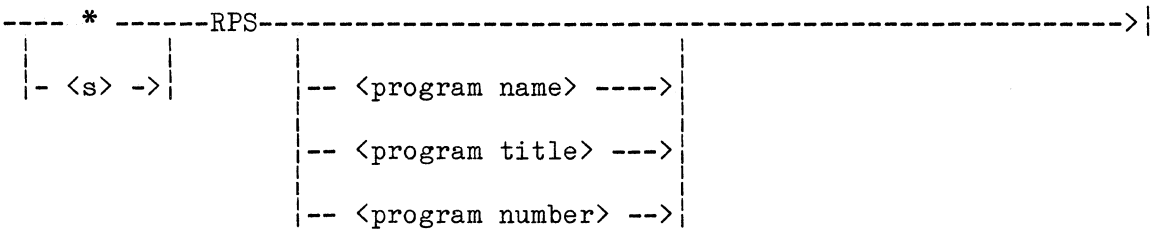
```
@ RPC MY/PROG
@ RPC 2
@ RPC PACK1/USER/PROGRAM1
@ RPC
```

REPORT PROGRAM STATUS (RPS)

The RPS command returns the following information about a program:

1. Name of the program.
2. Title of the program.
3. Whether the program is running.
4. Program classification.

Syntax:



Semantics:

The program name, program title, or program number is defined in the Program section of the TCL. If neither a program name, a program title, nor a program number is entered, the status of all programs is reported.

Examples:

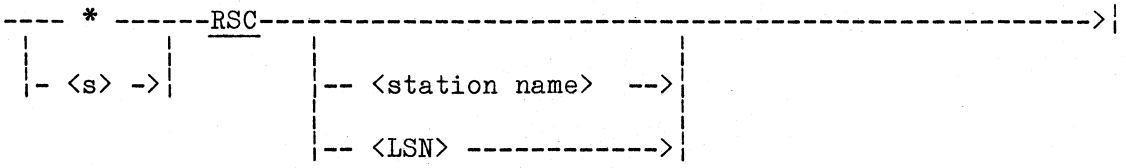
```
@ RPS PROG/A
$ RPS A
$ RPS PACK1/PROG/A
$ RPS
```

REPORT STATION COUNTERS (RSC)

The RSC command returns the following statistics about a station:

1. Number of messages sent.
2. Number of messages received.
3. Number of data communications errors.
4. Number of Network Control Commands affecting the station.
5. Number of changes made.

Syntax:



Semantics:

The station name or logical station number (LSN) is assigned in the Station section of the NDL. If neither station name nor LSN is provided, statistics are reported for all stations.

Example:

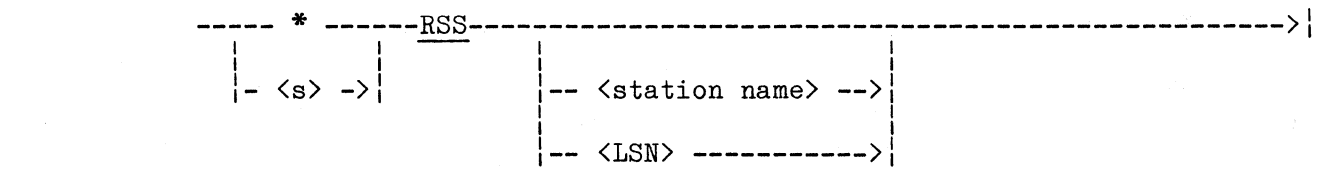
```
@ RSC TD1
```

REPORT STATION STATUS (RSS)

The RSS command reports the following about a station:

1. Whether the station is ready, enabled, signed on, or attached.
2. Usage (input, output, or both).
3. Which program the station is attached to, if any.

Syntax:



Semantics

The station name or logical station number (LSN) is assigned in the Station section of the NDL. If neither option is provided, status is reported for all stations.

Example:

```
@ RSS TD1
```

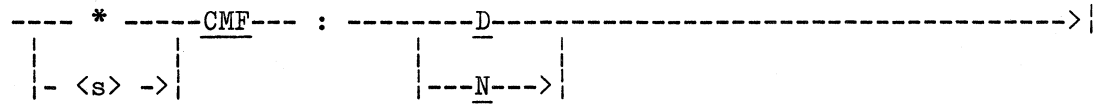
CHANGE COMMANDS

The existence of these commands is controlled by the CHANGE REQUESTS statement and the MONITOR TRACE statement in the TCL statements.

CHANGE MONITOR FLAG (CMF)

The CMF command is used to enable or disable monitoring of procedures in the MCS.

Syntax:



Semantics:

N (normal) sets the monitor flag to 0 (zero); D (diagnostic) sets the monitor flag to 1.

The following command can be used to close the print file if GEMCOS opens the file. (This could happen, for example, with Error 45 -- Invalid Input.) To close the print file, enter:

```
CMF : N
```

Examples:

```
@ CMF : D
```

Procedures will be monitored.

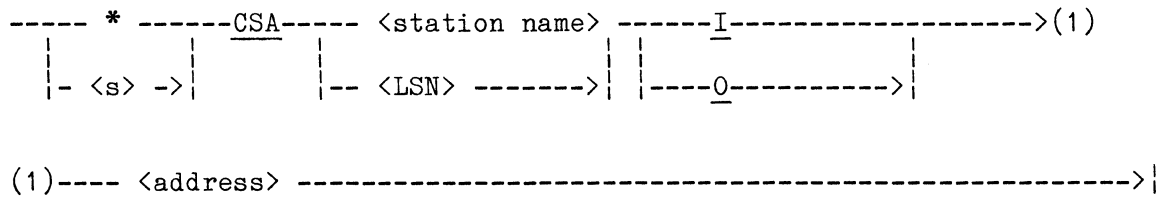
```
@ CMF : N
```

No procedures are monitored.

CHANGE STATION ADDRESS (CSA)

The CSA command is used to give the Network Controller a new logical address for a station.

Syntax:



Semantics:

The station name or logical station number (LSN) is assigned in the Station section of the NDL. Codes I or 0 indicate whether the new address applies on input or output.

Example:

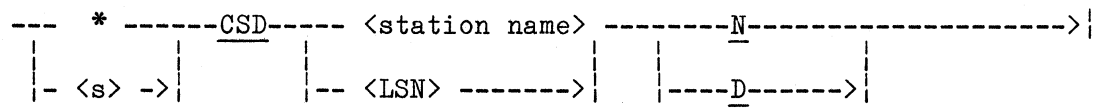
@ CSA 5 0 1A

Station 5 will have an output address of "1A".

CHANGE STATION DIAGNOSTIC (CSD)

The CSD command is used to inform the Network Controller whether to use normal diagnostic request logic (as defined in the NDL) for a station.

Syntax:



Semantics:

The station name or logical station number (LSN) is assigned in the Station section of the NDL. The code N selects normal request logic; D selects diagnostic logic.

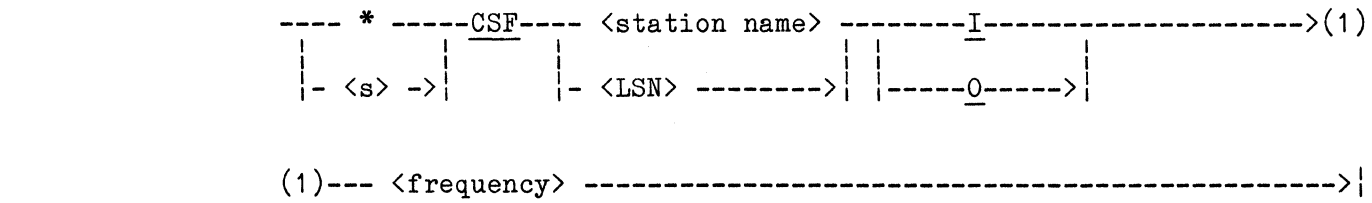
Example:

```
@ CSD 2 D
```

CHANGE STATION FREQUENCY (CSF)

The CSF command is used to assign a new input or output frequency (priority) to a station.

Syntax:



Semantics:

The station name or logical station number (LSN) is assigned in the Station section of the NDL. The codes I and O specify whether the frequency applies on input or output. The variable <frequency> must be an integer less than or equal to 255.

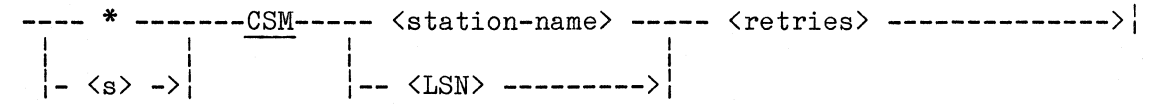
Example:

```
@ CSF TD2 I 250
```

CHANGE STATION MAXIMUM RETRY (CSM)

The CSM command is used to give a new value to the number of times the Network Control tries to retransmit a message when there are errors.

Syntax:



Semantics:

The station name or logical station number (LSN) is assigned in the Station section of the NDL. The variable <retries> must be an integer up to two digits long.

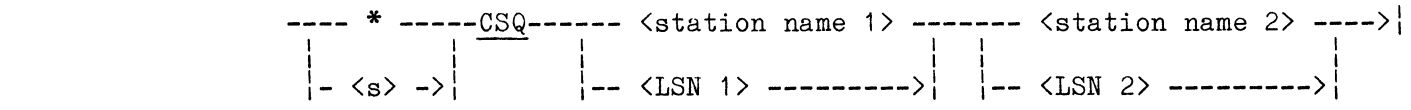
Example:

```
@ CSM 1 16
```

CHANGE STATION QUEUE (CSQ)

The CSQ command is used to re-route messages bound from one station to another station.

Syntax:



Semantics:

Station name 1 or logical station number (LSN) 1 is the original station, and station name 2 or LSN 2 is the new station. If station specifier 1 and station specifier 2 are the same, routing reverts to the original station.

Example:

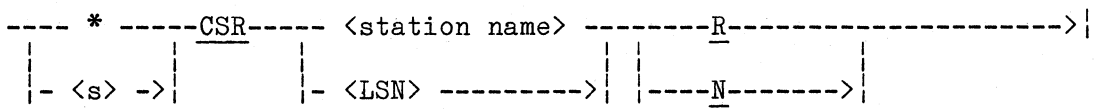
@ CSQ TD1 TD4

Messages which would ordinarily go to station TD1 will now go to station TD4.

CHANGE STATION READY (CSR)

The CSR command is used to mark a station as ready or not ready. The Network Controller does not attempt to do any input/output on a station that is not ready.

Syntax:



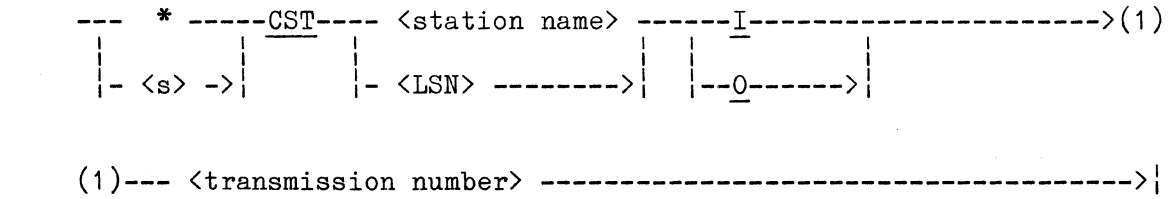
Example:

```
@ CSR 3 R
```

CHANGE STATION TRANSMISSION NUMBER (CST)

The CST command is used to give a new value to the transmission number of a station.

Syntax:



Semantics:

The station name or logical station number (LSN) is assigned in the Station section of the NDL. The codes I and O indicate whether the input or output transmission number is to be changed.

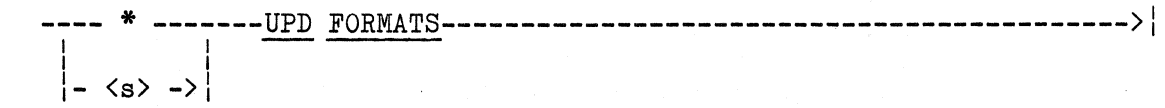
Example:

```
@ CST TD1 I 35
```

FORMAT UPDATE (UPD)

There is one Network Control Command to update formats. This command is only provided in the Advanced and Total Versions of GEMCOS. The UPD command makes updated formats available to all stations in the MCS network. Formats can be updated using the UPDATEFMT option of the Control command in the TCL, or by using the Format Generator or Format Maker. Updated formats are consequently placed in the MCSFORMATS file or the GEMCOS/NONINTERPS file. (For additional information on noninterpretive formatting, see the B 1000 GEMCOS Format Generator User's Guide.) The old version of the format remains available throughout the network until the UPD command is entered. Upon entry of the command, updated formats are available.

Syntax:



Semantics:

This request causes all previously-modified formats to be updated and made available to all stations in the network.

Example:

@ UPD FORMATS

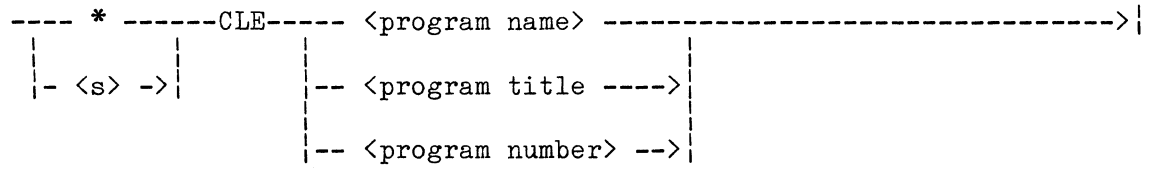
AUDIT & RECOVERY COMMANDS

The existence of these commands is controlled by the AUDITASSIGNMENT, AUDITTRANSACTIONS, and RECOVERY statements of the Program section.

CLEAR DISABLED PROGRAM (CLE)

The CLE command is used to clear disabled programs and to cause recovery to be initiated.

Syntax:



Semantics:

The program name, title, or number is assigned in the Program section of the TCL.

If, after clearing a disabled program, recovery does not begin, at least one other program in the data base is still disabled. When all programs are cleared, recovery of the data base is initiated.

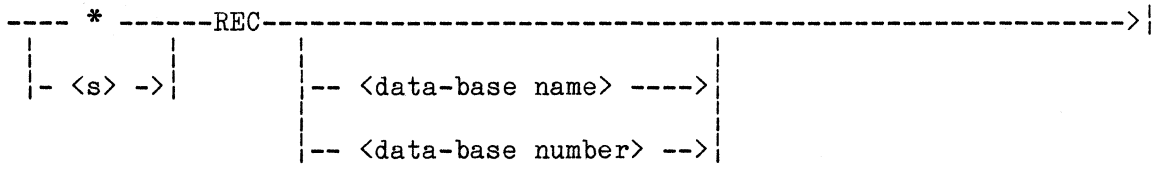
Examples:

```
@ CLE PROG1
@ CLE 2
```

RECOVER DATA BASE (REC)

The REC command is used to initiate recovery of a particular data base or of all data bases declared in the TCL.

Syntax:



Semantics:

The data-base name or number is assigned in the Data Base Name statement of the Program section in the TCL. If neither is specifed, recovery is initiated for all data bases that are declared to this MCS. Otherwise, recovery is initiated only for the selected data base. If any programs in the data base are disabled at the time of the request, an error message is displayed on the control station listing the numbers of the disabled program(s). These programs can then be cleared using the CLE command, and recovery is initiated.

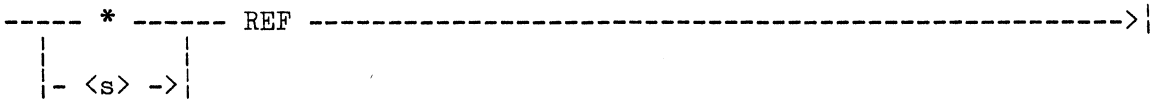
Examples:

```
@ REC
@ REC LIVEDB
@ REC 0
```

REFRESH COMMAND (REF)

The REF command is used to recall the most recent audited output message.

Syntax:



Semantics:

This request causes the MCS to display the last audited output message for the station. An error is returned when there are no output messages for the station.

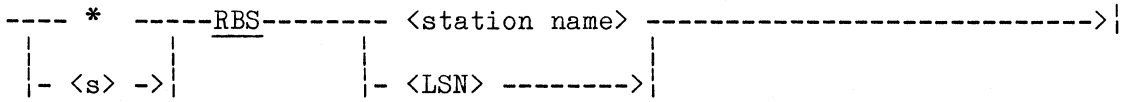
Example:

```
@ REF
```

RESET BUSY STATUS (RBS)

If a station is defined with TRANSACTIONMODE = TRUE, the station can become locked into a busy status. This occurs when a station sends a message to a synchronized recovery program and the program does not respond. Should a station receive error 106, this indicates that the station has a busy status (refer to Appendix D, MCS Error Messages). The RBS Network Control Command must be entered from a control station or from the ODT.

Syntax:



Semantics:

The station name or logical station number (LSN) is assigned in the Station section of the NDL. Entry of this command causes the specified station to be taken out of a busy status. Once the command is processed, the station operator may enter input at the station. If the station is not busy, this command has no effect.

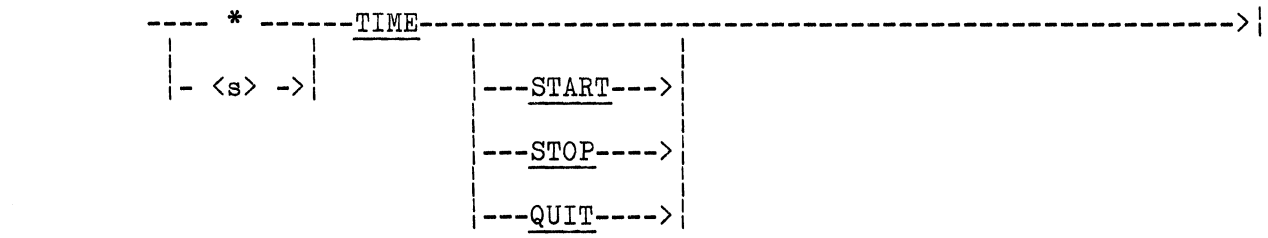
Examples:

```
@ RBS STATION2
@ RBS 3
```

TIME

The TIME command initiates or terminates the GEMCOS timing mechanism. This mechanism gathers statistics about the processing time for various transactions and stores these statistics in a disk file with the internal name of MCSTIME.

Syntax:



Semantics:

The TIME command may contain one of these three options: START, STOP, or QUIT. If none of these options is entered, the current status of the timing mechanism is returned.

TIME START initiates the timing mechanism. The results of each transaction are stored in an array. When the array is filled, the array is written to a disk file.

TIME STOP causes GEMCOS to stop entering data into the array, but does not close the file.

TIME QUIT, on the other hand, causes the contents of the array to be written to disk and the file to be closed with LOCK.

The timing file has a time stamp which is appended to <file-ID>. The time stamp allows multiple timings. The name of the timing file is:

MCSTIME/<time stamp>

When the timing file is closed (by entering *TIME QUIT), and then opened (by entering *TIME START), a new timing file is opened with a different name. Analyze the timing files by compiling the sample source file, GEMCOS/MCSTIMRPTS.

The MCS automatically does a TIME QUIT at end-of-job if the timing function is enabled.

Note that entering the same option more than once has no effect.

Examples:

```
@TIME START
@TIME STOP
```

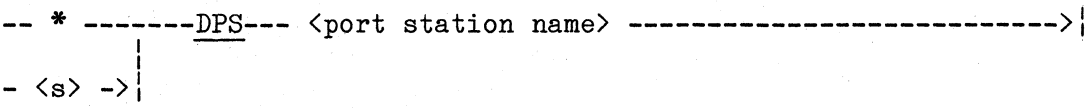
PORT STATION COMMANDS

The following commands are used with port stations.

DISABLE PORT STATION (DPS)

The DPS command is used to disable a port station. When this command is entered, GEMCOS closes the subfile of HOSTPORT for the associated port station and marks the station NOT READY.

Syntax:



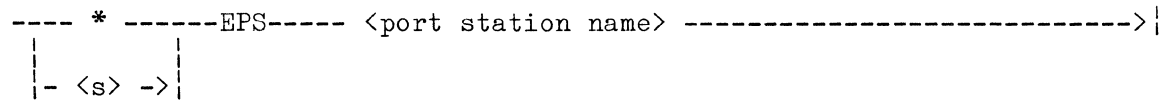
Examples:

```
*DPS PSTN
```

ENABLE PORT STATION (EPS)

The EPS command is used to enable a port station. When EPS is entered, GEMCOS opens the port subfile of HOSTPORT for the associated port station and marks the station READY.

Syntax:



Examples:

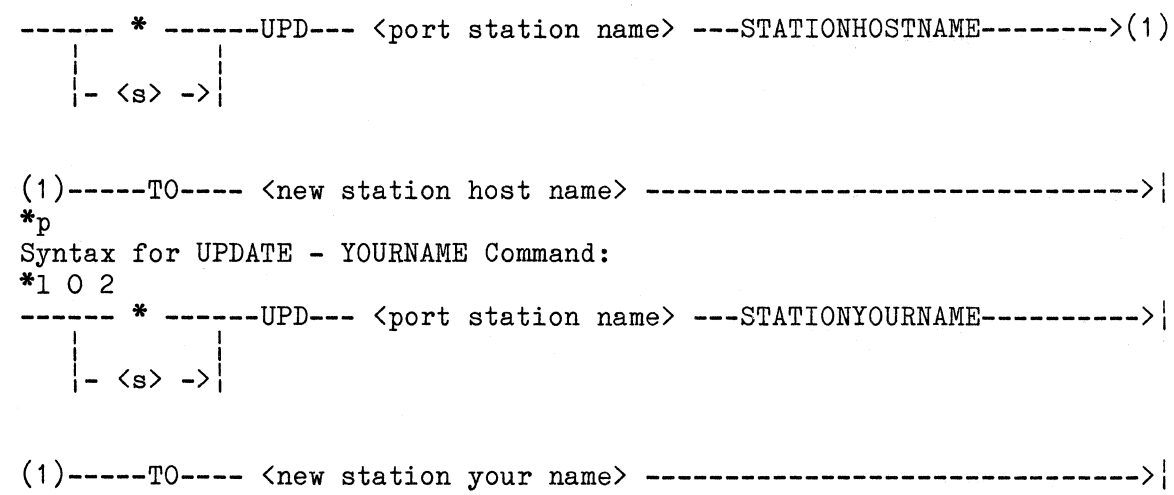
```
*EPS PSTN1
```

UPDATE STATION HOST NAME/STATION YOUR NAME

The UPDATE (UPD) command allows the user to update or change the HOSTNAME and YOURNAME of a port station. The change is permanent, since the MCSTIC file is rewritten with each change. Only those port stations which are fully closed can be changed. The UPD command must be entered at a Control station.

Since the change is permanent, the user's TCL no longer contains the correct STATIONHOSTNAMES and STATIONYOURNAMES. To obtain a current listing of port stations with the correct HOSTNAMES/YOURNAMES, run the TCL compiler with CONTROL = REPORT.

Syntax for UPDATE - HOSTNAME Command:



This command allows the user to update the STATIONHOSTNAME and STATIONYOURNAME of port stations dynamically. The user can always put extra dummy port stations in the TCL. Then when the need arises, the user can update the dummy names to meaningful names without recompiling.

Examples:

```
*UPD PSTN STATIONHOSTNAME TO PARISBASE
*UPD PSTN STATIONYOURNAME TO PARIS1
```

SECTION 4

MESSAGE FORMATTING AND ROUTING

The first part of this section discusses the formatting of messages for application programs. The next part illustrates several practical uses for message formatting. The last part of this section discusses how GEMCOS routes messages.

The material given here on message formatting is not intended to be a complete survey of the topic. For additional information, see the Format and Function List and the Device Section in Section 2. The material in Section 2 describes formatting syntax and semantics.

Also see the B 1000 Generalized Message Control System (GEMCOS) Formatting Guide.

FORMATTING AND APPLICATION PROGRAMS

In an on-line environment, proper message formatting is essential for effective use of the terminal. Properly formatted input and output messages aid readability, and assist both the occasional terminal user and the full-time terminal operator. The formatting of input and output messages is therefore a required function that must be performed by each application program or by some other part of the system.

When an application program formats its own input/output messages, changing or modifying a message format becomes a major problem (e.g., when existing formats are not suitable for a new terminal type). If the formatting is "hard-coded" into the application programs, such changes require program patches and recompilation, and can potentially introduce new problems. In addition, this approach requires more memory to store the formatting code that must be duplicated for each program.

These problems are avoided by using the Generalized Formatting module in the MCS. With this approach, the formatting code is stored in only one location, the MCS. Thus, messages can be changed without modifying the application programs. Since formatting code is generalized in the MCS and is based on disk files MCSFORMATS and GEMCOS/NONINTERPS, format changes can be implemented by merely updating either MCSFORMATS or GEMCOS/NONINTERPS. The MCS does not have to be modified. (For additional information on noninterpretive formatting, see the B 1000 GEMCOS Format Generator User's Guide.)

For non-formatted messages, GEMCOS has another helpful feature called screen wraparound. This feature is discussed in the following material.

SCREEN WRAPAROUND

If a non-formatted message is too long to be accepted by a station, the MCS breaks the message into two or more transmissions. Whenever possible, the message is segmented on word boundaries. The segment is as long as possible, without splitting a word or exceeding the buffer at the station.

Because a formatted mesage could be segmented in the middle of a forms field, the screen wraparound feature is not recommended for formatted messages. The MCS sends a warning message to the Control station if the screen wraparound feature is used to send a formatted message to a station.

The MCS generates code for screen wraparound when the station screensize is declared larger than the MAXTEXTSIZE specification in the Global section of the TCL.

The next material discusses editing phrases used with formatted messages.

GEMCOS EDITING PHRASES

The most common GEMCOS editing phrases are described in the following table.

Editing Phrase	Description
Alpha Item Type	In the form A<integer>, this phrase moves <integer> bytes from the raw message to the edited message.
Integer Item	In the form I<integer>, this phrase moves <integer> bytes from the raw message to the edited message (as does Alpha Item Item Type), and it also verifies that data being entered contains only digits and blanks (no embedded blanks). It also right justifies numbers and

	inserts leading zeroes where required on input.
Skip Field	In the form X<integer>, this phrase inserts <integer> spaces into the message.
EBCDIC String	In the form "<character string>", this phrase places a character string into the formatted message on output.
Hexadecimal String	In the form 4"<hexadecimal character string>", this phrase is composed of digits 0 through 9 and letters A through F. It is used in the same way as EBCDIC string to insert characters in the message; however, it can be used to insert any character, including nonprintable EBCDIC characters. The primary use of this phrase is to insert terminal control characters (e.g., carriage returns, line feeds), many of which are nonprintable.
Function Call	In the form T<function name>, <alpha item type>/ <integer item type>,<integer>, this phrase converts any string of six characters or less to any other string of six characters or less. The function name is used to specify which function is to be used in the REPLACE operation. The alpha item type or the integer item type is used to specify the length and type of the external character string, and the integer is used to specify the length of the internal character string.
Location Specifiers	In the form @<sign><integer>, this phrase overrides the normal mode of sequential operation of the formatter. It may be required to skip around in the message (e.g., if specific fields are to appear in a different order on the terminal than they come from the program). @+<integer> skips <integer> bytes forward; @-<integer> skips <integer> bytes backward.

OUTPUT FORMATTING EXAMPLE

In this example, it is assumed that the purpose of a program is to return information about records in a customer file, and that the program has the following output area and is sending the following data:

```
                    COBOL Description                          Sending Data

  01    SAMPLE-OUTPUT-AREA.
        05  CUSTOMER-NAME          PIC X(30).          THE WILSON FOOD
                                                        STORES
        05  ACCOUNT-NBR            PIC 9(6).           220030
        05  TOTAL-AR-BALANCE       PIC ZZZ,ZZZ.99      15,365.50
        05  TERMS                  PIC X.
*           7 = 7 DAYS
*           1 = 10 DAYS
*           C = COD
*           3 = 30 DAYS
*       05  CURRENT-BALANCE        PIC ZZZ,ZZZ.88      8,420.10
        05  ADDON-PERCENT          PIC ZZ9.            9
        05  OVER-7-DAYS-BALANCE    PIC ZZZ,ZZZ.99      824.10
        05  DATE-LAST-INVOICED     PIC 99/99/99.       08/01/77
        05  OVER-14-DAYS-BALANCE   PIC ZZZ,ZZZ.99      3,281.10
        05  CREDIT-LIMIT           PIC Z(6).           20000
        05  OVER-21-DAYS-BALANCE   PIC ZZZ,ZZZ.99      .00
        05  PRICE-CODE             PIC 9.              2
        05  DELINQUENT-FLAG        PIC 9.              1
*           1 = YES
*           0 = NO
        05  OVER-28-DAYS-BALANCE   PIC ZZZ,ZZZ.99      2,840.20
        05  WAREHOUSE              PIC XX.             A1
        05  ITEM-SUBSTITUTE-FLAG   PIC 9.              1
*           1 = YES
*           0 = NO
        05  CREDIT-BALANCE         PIC ZZZ,ZZZ.99      .00
```

The following TCL could be used to format the output messages for a TD700 which has eight lines of 32 characters each:

```
CONTROL = GENERATE, LIST, COMPILE.
GLOBAL:
     CHANGEREQUESTS = TRUE.
     PROGRAMBOJEOJ = TRUE.
     STATUSREPORTS = TRUE.
     SYSTEMHALT = TRUE.
     MAXTEXTSIZE = 1980.
```

```
     FORMAT
     OUT700 ( 4"0C0000", % CLEAR SCREEN AND HOME CURSOR


     % LINE 1
          X1,A30,X1,
     % LINE 2
          "ACCT-",I6,X2,"TOTAL",X4,A10,
     % LINE 3
          "TERMS-",A1,X6,"CURRENT",X2,A10,
     % LINE 4
          "ADDON-",A3,X4,"OVER  7",X2,A10,
     % LINE 5
          "L/I-",A8,X1,"OVER 14",X2,A10,
     % LINE 6
          "LIMIT-",A6,X1,"OVER 21",X2,A10,
     % LINE 7
          "PRICE-",I1,X1,"DF-",I1,X1,"OVER 28",X2,A10,
     % LINE 8
          "WHSE-",A2,X1,"SUB-",I1,X1,"CREDIT",X2,A10).
BEGIN
     PROGRAM SAMPLE USER:
          TITLE = GEMCOSPACK/GEMCOS/SAMPLE.
          TRANCODE = CUSTIN.
     STATION TD7A:
          TRANCODEPOSITION = 1.
     STATION TD7B:
          TRANCODEPOSITION = 1.
     STATION TD7C:
          TRANCODEPOSITION = 1.
     DEVICE TD700:
          STALIST = TD7A, TD7B, TD7C.
          FORMATSOUT:
               OUT700 = ARINF0.
END.
```

Figure 4-1 depicts the formatted output of the data returned from the application program (with the Format-ID field of the header set to ARINF0) and in turn sent to one of the stations listed under the device "TD700". The MCS uses format OUT700 to format the message.

```
  THE WILSON FOOD STORES
ACCT-220030   TOTAL        15,365.50
TERMS-3       CURRENT       8,420.10
ADDON-   9    OVER   7        824.10
L/I-08/01/77  OVER  14       3,281.10
LIMIT- 20000  OVER  21             .00
PRICE-2 DF-1  OVER  28       2,840.20
WHSE-A1 SUB-1 CREDIT               .00
```

Figure 4-1. Sample Formatted Output
Data Display

It may be desirable to use functions to convert the data coming from the application program to a more readable form. For example, in the preceding format, the TERMS code 3 is not very definitive. It would be better to display something more descriptive like 30 DAY. To do this, the following function is defined:

```
FUNCTION
     TERMS [EXTERNAL:ALPHA,INTERNAL:ALPHA] (
          " 7 DAY":"7",
          "10 DAY":"1",
          "30 DAY":"3",
          "COD   ":"C").
```

This function description converts "7" to "7 DAY", "1" TO "10 DAY", etc. DELINQUENT-FLAG and ITEM-SUBSTITUTE-FLAG displayed as a "1" or a "0" could also be changed to display a Y or N by defining the following function:

```
FUNCTION
     BINARYOUT700 [EXTERNAL:ALPHA,
          INTERNAL:INTEGER](
          "Y":"1",
          "N":"0").
```

To take advantage of these functions, the calls must be inserted in the format being used, necessitating the following change to line 3:

```
% LINE 3
     "TERMS-",T(TERMS,A6,1),X1,"CURRENT",X2,A10,
```

Lines 7 and 8 would then read as follows:

```
% LINE 7
     "PRICE-",I1,X1,"DF-",T(BINARYOUT700,A1,1),X1,"OVER 28",
          X2,A10,
% LINE 8
     "WHSE-",A2,X1,"SUB-",T(BINARYOUT700,A1,1),X1,"CREDIT",
          X2,A10).
```

The TCL deck now appears as follows:

```
CONTROL = REGENERATE, LIST. % IT IS NOT REQUIRED TO GENERATE
          % AND COMPILE THE MCS SINCE WE ARE ONLY CHANGING
          % FUNCTIONS AND FORMATS
GLOBAL:
      CHANGEREQUESTS = TRUE.
      PROGRAMBOJEOJ = TRUE.
      STATUSREPORTS = TRUE.
      SYSTEMHALT = TRUE.
      MAXTEXTSIZE = 1980.


      FUNCTION
          TERMS [EXTERNAL:ALPHA,INTERNAL:ALPHA] (
               " 7 DAY":"7",
               "10 DAY":"1",
               "30 DAY":"3",
               "COD   ":"C"),
          BINARYOUT700
 [EXTERNAL:ALPHA,INTERNAL:INTEGER] (
               "Y":"1",
               "N":"0").
     FORMAT
     OUT700 ( 4"0C0000", % CLEAR SCREEN AND HOME CURSOR
     % LINE 1
         X1,A30,X1,
     % LINE 2
         "ACCT-",I6,X2,"TOTAL",X4,A10,
     % LINE 3
         "TERMS-",T(TERMS,A6,1),X1,"CURRENT",X2,A10,
     % LINE 4
         "ADDON-",A3,X4,"OVER  7",X2,A10,
     % LINE 5
         "L/I-",A8,X1,"OVER 14",X2,A10,
     % LINE 6
         "LIMIT-",A6,X1,"OVER 21",X2,A10,
     % LINE 7
         "PRICE-",I1,X1,"DF-",T(BINARYOUT700,A1,1),X1,
         "OVER 28",X2,A10,
```

```
     % LINE 8
         "WHSE-",A2,X1,"SUB-",T(BINARYOUT700,A1,1),X1,
         "CREDIT",X2,A10).
BEGIN
     PROGRAM SAMPLE USER:
         TITLE = GEMCOSPACK/GEMCOS/SAMPLE.
         TRANCODE = CUSTIN.
     STATION TD7A:
         TRANCODEPOSITION = 1.
     STATION TD7B:
         TRANCODEPOSITION = 1.
     DEVICE TD700:
         STALIST = TD7A, TD7B, TD7C.
         FORMATSOUT:
              OUT700 = ARINFO.
END.
```

Figure 4-2 depicts the formatted output data returned from the application programs and forwarded to the stations as a result of the function description and modified TCL deck.

```
 THE WILSON FOOD STORES
ACCT-220030   TOTAL        15,365.50
TERMS-30 DAY  CURRENT       8,420.10
ADDON-   9    OVER   7        824.10
L/I-08/01/77  OVER 14       3,281.10
LIMIT- 20000  OVER 21             .00
PRICE-2 DF-Y  OVER 28       2,840.20
WHSE-A1 SUB-Y CREDIT              .00
```

Figure 4-2. Example Formatted Output Data Display for Redefined Forms Function

If two TD830s (having 24 lines of 80 characters each) are now added to the network, a new format is required to take advantage of the larger screen. The larger screen permits more descriptive headers and allows for re-arrangement of the data to make it more readable. A function can be used to convert 1 and 0 to YES and NO rather than the more cryptic Y and N. This requires defining the following new function and new format.

New Function:

```
FUNCTION
     BINARYOUT800 [EXTERNAL:ALPHA,
          INTERNAL:INTEGER] (
          "YES":"1",
          "NO":"0").
```

New Format:

```
OUT800 ( 4"0C0000", % CLEAR SCREEN AND HOME CURSOR
% LINE 1
     X17,"CUSTOMER NAME:",X2,A30,4"0D", % 4"0D" IS CARRIAGE
                                        % RETURN
% LINE 2
     4"0D", % LEAVE BLANK LINE
% LINE 3
     "ACCOUNT NUMBER",X5,I6,X15,"TOTAL A/R BALANCE",X2,A10,
     4"0D",
% LINE 4
     "TERMS",X14,T(TERMS,A6,1),X15,"CURRENT",X11,A10,4"0D",
% LINE 5
     "ADDON RATE",X9,A3,"%",X17,"OVER 7 DAYS",X7,A10,4"0D",
% LINE 6
     "DATE LAST INVOICED",X1,A8,X13,"OVER 14 DAYS",X7,A10,
     4"0D",
% LINE 7
     "CREDIT LIMIT",X7,A3,",",A3,X14,"OVER 21 DAYS",X7,A10,
     4"0D",


% LINE 8
     "PRICE CODE",X9,I1,@+1, % SKIP OVER SUBSTITUTE FLAG
     X20,"OVER 28 DAYS",X7,A10,4"0D",
% LINE 9
     "WAREHOUSE",X10,A2,@+1, % SKIP OVER DELINQUENT FLAG
     X19,"CREDIT",X13,A10,4"0D",
% LINE 10
     "SUBSTITUTE FLAG",X4,@-24, % SKIP BACK TO GET SUBS FLAG
          T(BINARYOUT800,A3,1),X18,"DELINQUENT FLAG",X4,
          X4,@+12, % SKIP FORWARD TO GET DELINQUENT FLAG
          T(BINARYOUT800,A3,1)).
```

The new stations and the new device type must be defined, resulting in the following TCL deck:

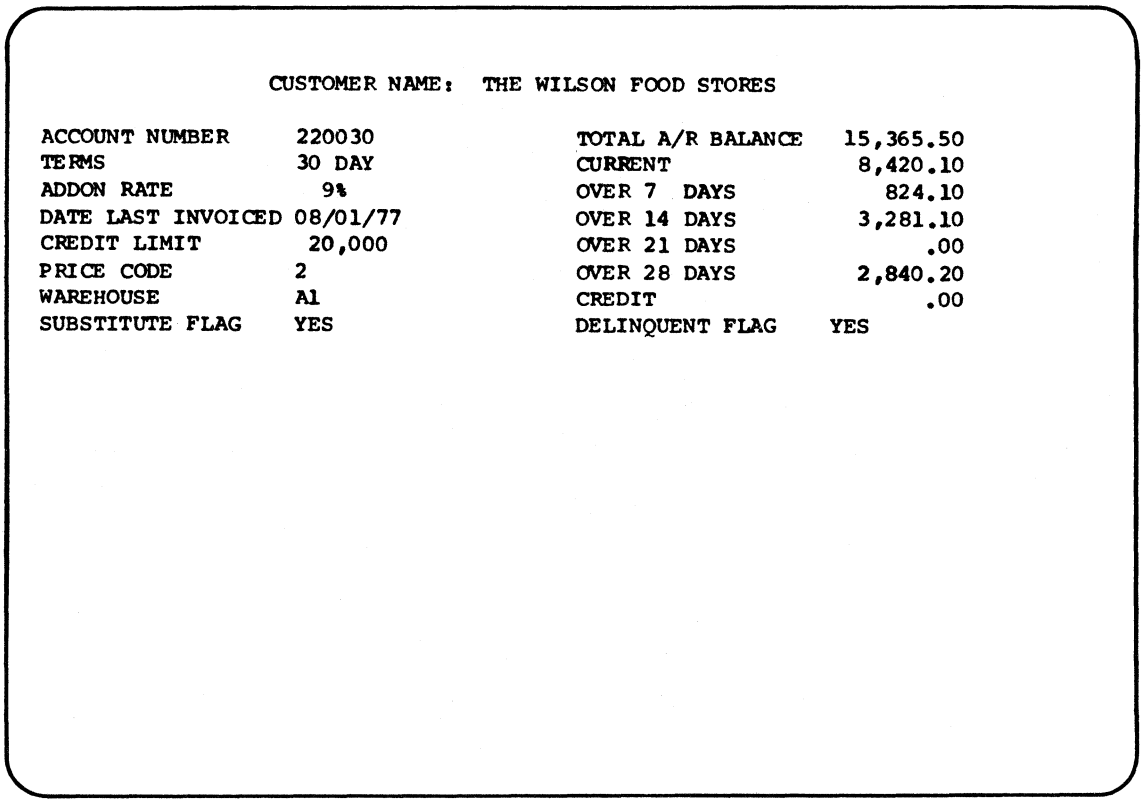
```
CONTROL = REGENERATE, LIST. % IT IS NOT REQUIRED TO GENERATE
     % AND COMPILE THE MCS SINCE WE ARE ONLY CHANGING
     % FUNCTION, FORMAT, STATION AND DEVICE DEFINITIONS.
GLOBAL:
     CHANGEREQUESTS = TRUE.
     PROGRAMBOJEOJ = TRUE.
     STATUSREPORTS = TRUE.
     SYSTEMHALT = TRUE.
     MAXTEXTSIZE = 1980.
     FUNCTION
         TERMS [EXTERNAL:ALPHA,INTERNAL:ALPHA] (
              " 7 DAY":"7",
              "10 DAY":"1",
              "30 DAY":"3",
              "COD   ":"C"),
          BINARYOUT800
 [EXTERNAL:ALPHA,INTERNAL:INTEGER] (
              "Y":"1",
               "N":"0"),
          BINARYOUT800
 [EXTERNAL:ALPHA,INTERNAL:INTEGER] (
               "YES":"1",
               "NO ":"0").
     FORMAT
OUT700 ( 4"0C0000", % CLEAR SCREEN AND HOME CURSOR
     % LINE 1
         X1,A30,X1,
     % LINE 2
         "ACCT-",I6,X2,"TOTAL",X4,A10,
     % LINE 3
         "TERMS-",T(TERMS,A6,1),X1,"CURRENT",X2,A10,
     % LINE 4
         "ADDON-",A3,X4,"OVER  7",X2,A10,
     % LINE 5
         "L/I-",A8,X1,"OVER 14",X2,A10,
     % LINE 6
         "LIMIT-",A6,X1,"OVER 21",X2,A10,


     % LINE 7
         "PRICE-",I1,X1,"DF-",T(BINARYOUT700,A1,1),X1,
         "OVER 28",X2,A10,
     % LINE 8
         "WHSE-",A2,X1,"SUB-",T(BINARYOUT700,A1,1),X1,
         "CREDIT",X2,A10),
         OUT800 ( 4"0C0000", 5 CLEAR SCREEN AND HOME CURSOR
     % LINE 1
         X17, "CUSTOMER NAME:",X2,A30,4"0D", % 4"0D" IS
```

```
          % CARRIAGE RETURN
     % LINE 2
          4"0D", % LEAVE BLANK LINE
     % LINE 3
          "ACCOUNT NUMBER",X5,I6,X15,"TOTAL A/R BALANCE",X2,
          A10,4"0D",
     % LINE 4
          "TERMS",X14,T(TERMS,A6,1),X15,"CURRENT",X11,A10,
          4"0D",
     % LINE 5
          "ADDON RATE",X9,A3,"%",X17,"0VER 7 DAYS",X7,A10,
          4"0D",
     % LINE 6
          "DATE LAST INVOICED",X1,A8,X13,"0VER 14 DAYS",X7,
          A10,4"0D",
     % LINE 7
          "CREDIT LIMIT",X7,A3,",",A3,X14,"0VER 21 DAYS",X7,
          A10,4"0D",
     % LINE 8
          "PRICE CODE",X9,I1,@+1, % SKIP 0VER SUBSTITUTE
                                  % FLAG
          X20,"0VER 28 DAYS",X7,A10,4"0D",
     % LINE 9
          "WAREHOUSE",X10,A2,@+1, % SKIP 0VER DELINQUENT
                                  % FLAG
          X19,"CREDIT",X13,A10,4"0D",
     % LINE 10
          "SUBSTITUTE FLAG",X4,@-24, % SKIP BACK TO GET
                                     % SUBS FLAG
          T(BINARY0UT800,A3,1),X18,"DELINQUENT FLAG",
          X4,X4,@+12, % SKIP FORWARD TO GET DELINQUENT
                      % FLAG
          T(BINARY0UT800,A3,1)).
BEGIN
     PROGRAM SAMPLE USER:
          TITLE = GEMCOSPACK/GEMCOS/SAMPLE.
          TRANCODE = CUSTIN.
     STATION TD7A:
          TRANCODEPOSITION = 1.
     STATION TD7B:
          TRANCODEPOSITION = 1.
     STATION TD7C:
          TRANCODEPOSITION = 1.
     STATION TD8A:
          TRANCODEPOSITION = 1.
     STATION TD8B:
          TRANCODEPOSITION = 1.
     DEVICE TD700:
          STALIST = TD7A, TD7B, TD7C.
          FORMATSOUT:
               0UT700 = ARINF0.
```

```
        DEVICE TD800:
             STALIST = TD8A, TD8B.
             FORMATSOUT:
                  OUT800 = ARINFO.
    END.
```

With the new stations properly defined, the application program does not have to specify the type of terminal its response is to be sent to. When it sends a message to the MCS with the format-ID set to ARINFO, the MCS uses the appropriate format for the specific device type. If the message is in route to station TD7A, TD7B, or TD7C, it appears on the screen exactly as in the previous example. However, if it is going to TD8A or TD8B, the MCS uses format OUT800 and displays the message depicted in Figure 4-3.



CUSTOMER NAME: THE WILSON FOOD STORES

ACCOUNT NUMBER	220030	TOTAL A/R BALANCE	15,365.50
TERMS	30 DAY	CURRENT	8,420.10
ADDON RATE	9%	OVER 7 DAYS	824.10
DATE LAST INVOICED	08/01/77	OVER 14 DAYS	3,281.10
CREDIT LIMIT	20,000	OVER 21 DAYS	.00
PRICE CODE	2	OVER 28 DAYS	2,840.20
WAREHOUSE	A1	CREDIT	.00
SUBSTITUTE FLAG	YES	DELINQUENT FLAG	YES

Figure 4-3. Sample Formatted Output for TD830 Terminal

INPUT FORMATTING EXAMPLE

Expanding the example described above, assume the program needs a message with the following three fields: a 6-character transaction-code field containing CUSTIN, a 6-digit field containing the customer account number, and a 1-digit flag informing the program whether to include aging of the account balance in the response (1 = YES; 0 = NO). Two input formatting requirements are: to build a form on the screen for the operator to fill with in the required information, and to form the data transmitted from the terminal for the application program.

The first objective is met by defining an output format which builds the form on the screen. This format (like all output formats) can be sent to the station by two means. It is sent when an application program sends a message to that terminal with the format-ID of the header set to the appropriate format-ID for that format, or when the terminal operator keys in the format-ID directly. If it comes from a program, it is filled with the data sent from the program. If the request comes from the station, the format is sent to the station as if a program had sent that format with the data area containing all spaces. Thus, the input form for the TD 700 terminals could be built using the following format:

```
EMPTY 700 C 4"0C0000", % CLEAR SCREEN AND HOME CURSOR
% LINE 1
     "[CUSTIN]",4"0D",
% LINE 2
     "CUSTOMER ACCOUNT NUMBER",X2,"[",A6,"]",
% LINE 3
     "RECEIVABLE AGING INFO? [",A3,"]",
     4"1200000005000000").  % PUT TERMINAL IN FORMS
                            % MODE AND TAB ONCE
```

The DEVICE section for the TD700s must also be changed to read:

```
DEVICE TD700:
     STALIST = TD7A, TD7B, TD7C.
     FORMATSOUT:
          OUT700 = ARINFO.
          EMPTY700 = CSTFRM.
```

Figure 4-4 depicts the display that would be sent by the MCS in response to the forms request (when the terminal operator transmits CSTFRM).

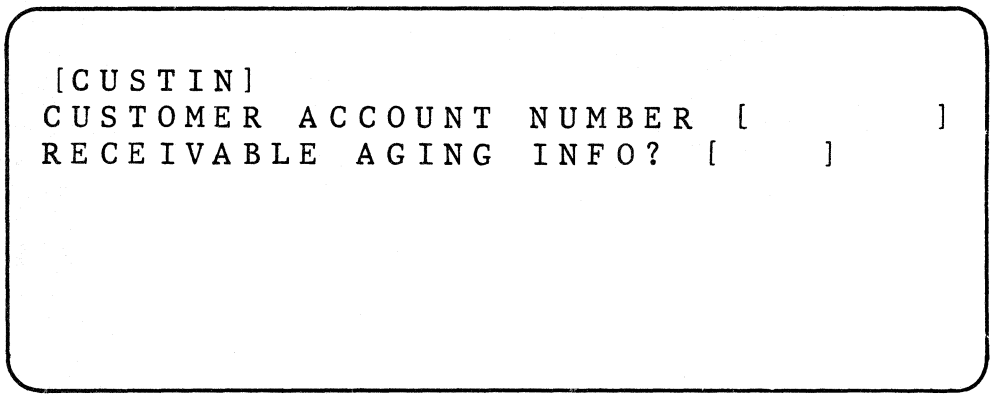


Figure 4-4. Sample Input Form
for TD700 Terminal

The format for the TD830 terminals is the same except for the form left delimiter and right delimiter which are [and], respectively, on the TD700 and 4"1F" and 4"1E" on the TD 830. Thus, the format can be described as follows:

```
EMPTY800 ( 4"0C0000", % CLEAR SCREEN AND HOME CURSOR
% LINE 1
     4"1F","CUSTIN",4"1EOD",
% LINE 2
     "CUSTOMER ACCOUNT NUMBER",X2,4"1F",A6,4"1EOD",
% LINE 3
     "RECEIVABLE AGING INFO? ",4"1F",A3,4"1E",
     4"27E6000005000000"). % PUT TERMINAL IN FORMS
                           % MODE AND TAB ONCE
```

The DEVICE section for the TD830 device would be:

```
DEVICE TD800:
     STALIST = TD8A, TD8B.
     FORMATSOUT:
          OUT800 = ARINFO.
          EMPTY800 = CSTFRM.
```

Thus, when the operator transmits CSTFRM from a TD830, the MCS returns the display depicted in Figure 4-5.

It is also necessary to format the message from the station for the application program. The same format for both the TD700 and the TD830 can be used since both send data in the same format (CUSTIN followed by

six characters of account number, followed by three characters to indicate whether the response is to include aging information).

Since the program expects a 1 or 0 for the aging flag, a function is used to allow the operator to enter YES, Y (or 1 for 1), and to enter NO, N (or 0 for 0). This function is as follows:

```
FUNCTION
     EXTERNAL:ALPHA,INTERNAL:INTEGER [EXTERNAL:ALPHA,
          INTERNAL:INTEGER] (
          "YES":"1",
          "Y":"1",
          "1":"1",
          "NO":"0",
          "N":"0",
          "0":"0").
```

Now the format appears as follows:

```
FORMAT
IN (
     A6,I6,T(BINARYIN,A3,1)).
```

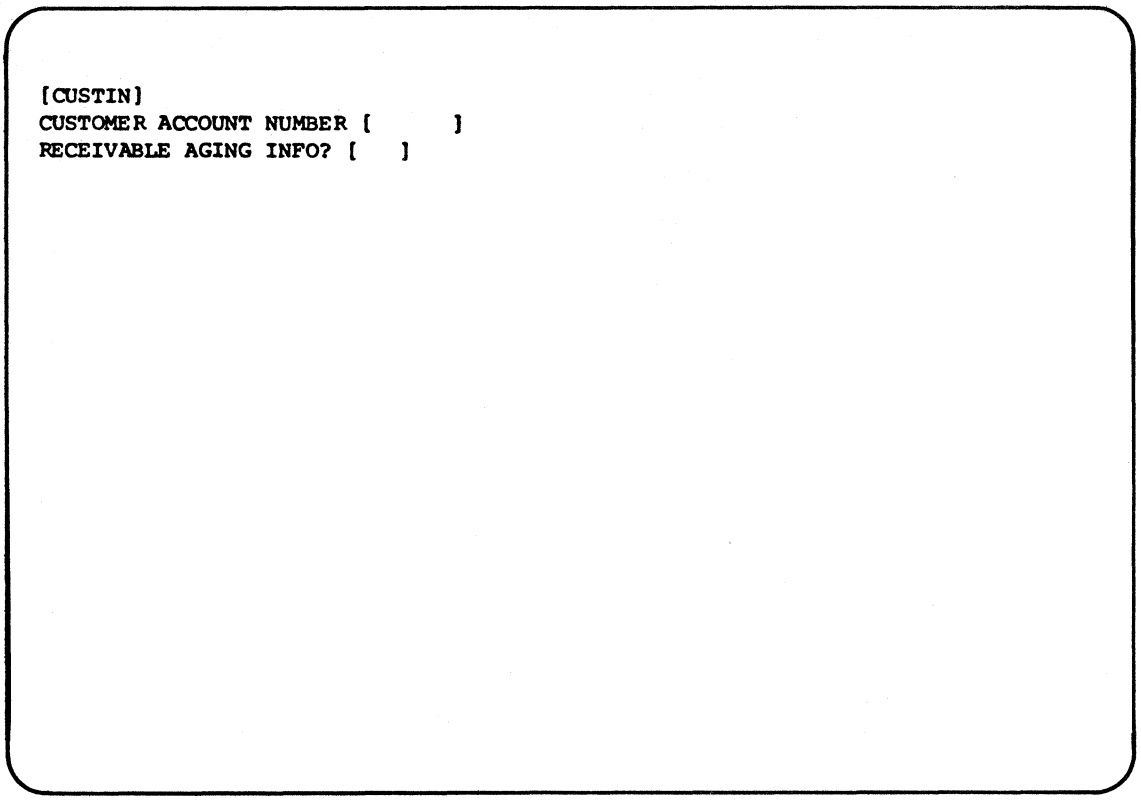


Figure 4-5. Sample Input Form
for TD830 Terminal

Finally, this format must be added to the FORMATSIN portion of the device description of both the TD700 and the TD830. The TCL is as follows:

```
CONTROL = REGENERATE, LIST. % IT IS NOT REQUIRED TO GENERATE
     % AND COMPILE THE MCS SINCE WE ARE ONLY CHANGING
     % FUNCTION, FORMAT, STATION AND DEVICE DEFINITIONS.
GLOBAL:
     CHANGEREQUESTS = TRUE.
     PROGRAMBOJEOJ = TRUE.
     STATUSREPORTS = TRUE.
     SYSTEMHALT = TRUE.
     MAXTEXTSIZE = 1980.
     FUNCTION
          TERMS [EXTERNAL:ALPHA,INTERNAL:ALPHA] (
               " 7 DAY":"7",
               "10 DAY":"1",
               "30 DAY":"3",
               "COD   ":"C"),
```

```
          EXTERNAL:ALPHA,INTERNAL:INTEGER
 [EXTERNAL:ALPHA,INTERNAL:INTEGER] (
                "Y":"1",
                "N":"0"),
          EXTERNAL:ALPHA,INTERNAL:INTEGER
 [EXTERNAL:ALPHA,INTERNAL:INTEGER] (
                "YES":"1",
                "NO":"0"),
          EXTERNAL:ALPHA,INTERNAL:INTEGER
 [EXTERNAL:ALPHA,INTERNAL:INTEGER] (
                "YES":"1",
                "Y":"1",
                "1":"1",
                "NO":"0",
                "N":"0",
                "0":"0").
FORMAT
OUT700 ( 4"0C0000", % CLEAR SCREEN AND HOME CURSOR
% LINE 1
     X1,A30,X1,
% LINE 2
     "ACCT-",I6,X2,"TOTAL",X4,A10,
% LINE 3
     "TERMS-",T(TERMS,A6,1),X1,"CURRENT",X2,A10,
% LINE 4
     "ADDON-",A3,X4,"OVER  7",X2,A10,
% LINE 5
     "L/I-",A8,X1,"OVER 14",X2,A10,
% LINE 6
     "LIMIT-",A6,X1,"OVER 21",X2,A10,
% LINE 7
     "PRICE-",I1,X1,"DF-",T(BINARYOUT700,A1,1),X1,"OVER 28",
     X2,A10,
% LINE 8
     "WHSE-",A2,X1,"SUB-",T(BINARYOUT700,A1,1),X1,"CREDIT",
     X2,A10,0UT800 ( 4"0C0000", % CLEAR SCREEN AND
                                % HOME CURSOR
% LINE 1
     X17,"CUSTOMER NAME:",X2,A30,4"0D", % 4"0D" IS CARRIAGE
                                        % RETURN


% LINE 2
     4"0D", % LEAVE BLANK LINE
% LINE 3
     "ACCOUNT NUMBER",X5,I6,X15,"TOTAL A/R BALANCE",X2,A10,
     4"0D",
% LINE 4
     "TERMS",X14,T(TERMS,A6,1),X15,"CURRENT",X11,A10,4"0D",
% LINE 5
     "ADDON RATE",X9,A3,"%",X17,"OVER 7 DAYS",X7,A10,4"0D",
% LINE 6
```

```
     "DATE LAST INVOICED",X1,A8,X13,"OVER 14 DAYS",X7,A10,
     4"OD",
% LINE 7
     "CREDIT LIMIT",X7,A3,",",A3,X14,"OVER 21 DAYS",X7,A10,
     4"OD",
% LINE 8
     "PRICE CODE",X9,I1,@+1, % SKIP OVER SUBSTITUTE
                             % FLAG
     X20,"OVER 28 DAYS",X7,A10,4"OD",
% LINE 9
     "WAREHOUSE",X10,A2,@+1, % SKIP OVER DELINQUENT
                             % FLAG
     X19,"CREDIT",X13,A10,4"OD",
% LINE 10
     "SUBSTITUTE FLAG",X4,@-24, % SKIP BACK TO GET SUBS
                                % FLAG
     T(BINARYOUT800,A3,1),X18,"DELINQUENT FLAG",
     X4,X4,@+12, % SKIP FORWARD TO GET DELINQUENT FLAG
     T(BINARYOUT800,A3,1)),
EMPTY700 (  4"0C0000", % CLEAR SCREEN AND HOME CURSOR
% LINE 1
     "[CUSTIN]",4"OD",
% LINE 2
     "CUSTOMER ACCOUNT NUMBER",X2,"[",A6,"]",
% LINE 3
     "RECEIVABLE AGING INFO? [",A3,"]",
     4"1200000005000000"),  % PUT TERMINAL IN FORMS
                            % MODE AND TAB ONCE
EMPTY800 (  4"0C0000", % CLEAR SCREEN AND HOME CURSOR
% LINE 1
     4"1F","CUSTIN",4"1EOD",
% LINE 2
     "CUSTOMER ACCOUNT NUMBER",X2,"1F",A6,4"1EOD",
% LINE 3
     "RECEIVABLE AGING INFO? "4"1F",A3,4"1E",
     4"27E6000005000000"), % PUT TERMINAL IN FORMS MODE
                           % AND TAB ONCE
IN (
     A6,I6,T(BINARYIN,A3,1)).
BEGIN
     PROGRAM SAMPLE USER:
          TITLE = GEMCOSPACK/GEMCOS/SAMPLE.
          TRANCODE = CUSTIN.
     STATION TD7A:
          TRANCODEPOSITION = 1.
     STATION TD7B:
          TRANCODEPOSITION = 1.
     STATION TD7C:
          TRANCODEPOSITION = 1.
     STATION TD8A:
          TRANCODEPOSITION = 1.
     STATION TD8B:
```

```
            TRANCODEPOSITION = 1.
        DEVICE TD700:
            STALIST = TD7A, TD7B, TD7C.
            FORMATSIN:
                IN = CUSTIN.
            FORMATSOUT:
                OUT700 = ARINFO.
                EMPTY700 = CSTFRM.
        DEVICE TD800:
            STALIST = TD8A, TD8B.
            FORMATSIN:
                IN = CUSTIN.
            FORMATSOUT:
                OUT800 = ARINFO.
                EMPTY800 = CSTFRM.
END.
```

Figure 4-6 depicts the filled-in input form. When transmitted, the MCS receives CUSTIN12345 YES. The MCS first checks for a trancode and determines that this message is to be sent to the program GEMCOSPACK/GEMCOS/SAMPLE because the trancode is CUSTIN. The MCS then formats the message using the format IN, since the Device section associates the format IN with the format-ID CUSTIN. After formatting with format IN, the message is CUSTINO123451. This message is then sent to the application program.

```
[CUSTIN]
CUSTOMER ACCOUNT NUMBER [12345 ]
RECEIVABLE AGING INFO? [YES]
```

Figure 4-6. Sample Filled-In Input Form

MESSAGE ROUTING

The concept of message routing covers both messages from stations to programs, and messages from programs to stations. The usual case involves a message sent from a station to a program. When the program is finished, it returns a response to the station. GEMCOS uses remote

files, common-area headers, and transaction codes to route messages to application programs.

USING REMOTE FILES

Remote files link application programs and stations. The network designer uses the FAMILY statement in the Network Definition Language (NDL) to associate a group of stations with a remote file. Stations are identified by a number which corresponds to the order in which they were declared in the FAMILY statement.

The application program opens a remote file, and those stations declared in the FAMILY statement are then attached to that remote file. From then on, the Network Controller routes messages from those stations only to that program that opened the remote file.

Messages are exchanged in the record area of that file. In COBOL, an actual key can be used to exchange information about the message (such as message type, message length, source, destination station.) Use the actual key in COBOL only to specify the text size of an outgoing message.

USING COMMON-AREA HEADERS

GEMCOS offers a common-area header option. This option offers parts of the standard remote file interface. The common-area header appears in the record area of a file in front of the message.

When the common-area header is selected, the MCS attaches the common-area header to each message sent to an application program and removes it from each message received from an application program, so that this header does not accompany the message on the Network Controller side of the MCS.

When an application program receives a message from the MCS, the Logical Station Number (LSN) field in the common-area header is set to the originating station. If the LSN has not been changed at the time the application program returns a message to the MCS, the MCS routes the message to the same station.

But if the application program needs the response sent to another station, the LSN field is changed to the correct value for that station. See Section 2 for additional information on using common-area headers.

USING TRANSACTION-BASED ROUTING

Transaction-based routing (TBR) is another optional method of routing messages from the MCS to application programs. In this method of routing, each message has a transaction code, which can occur anywhere in the message text.

A group of transaction codes is associated with each application program. Therefore, a message with a certain transaction code is always routed to the same program, regardless of where the program originated.

Within the limits of security restrictions, a station can send a message to any program by including in the message one of the transaction codes associated with that program.

If a message does not have a transaction code or if the transaction code is invalid, the message is sent to the program attached to the originating station. If no program is attached to that station, the message is rejected.

SELECTING A METHOD OF MESSAGE ROUTING

The method used by the GEMCOS MCS to route messages to application programs is determined by:

1. The program classification (i.e., Assignment, Utility, User, or Pass) specified in the Program section.
2. The type of interface (i.e., Nonparticipation, Participation, or MCS) specified in the INTERFACE statement.

Thus, 10 combinations of program classification and interface are possible: three interfaces for assignment programs, three interfaces for utility programs, one interface (Participation) for user programs, and three interfaces for pass programs. The following example lists all 10 meaningful TCL descriptions.

Example:

```
PROGRAM X1 UTILITY:
  TITLE = X.                              % OPTIONAL
  INTERFACE = NONPARTICIPATION.
  RESIDENCE = DISK.                       % OPTIONAL

PROGRAM X2 UTILITY:
  TITLE = X.                              % OPTIONAL
  INTERFACE = MCS.
  OPENMESSAGE = TRUE.                     % OPTIONAL
  ATTACHMESSAGE = TRUE.                   % OPTIONAL
  RESIDENCE = DISK.                       % OPTIONAL

PROGRAM X3 UTILITY:
  TITLE = X.                              % OPTIONAL
  INTERFACE = PARTICIPATION.              % OPTIONAL
  COMMONSIZE = 100.                       % OPTIONAL
  TRANCODE = X(1,1).                      % OPTIONAL
  OPENMESSAGE = TRUE.                     % OPTIONAL
  ATTACHMESSAGE = TRUE.                   % OPTIONAL
  DETACHMESSAGE = TRUE.                   % OPTIONAL
  RESIDENCE = DISK.                       % OPTIONAL

PROGRAM X4 USER:
  TITLE = X.                              % OPTIONAL
  INTERFACE = PARTICIPATION.              % OPTIONAL
  COMMONSIZE = 100.                       % OPTIONAL
  TRANCODE = X(1,1).                      % OPTIONAL
  OPENMESSAGE = TRUE.                     % OPTIONAL
  RESIDENCE = DISK.                       % OPTIONAL
  EXECUTE = ONDEMAND.                     % OPTIONAL

PROGRAM X5 ASSIGNMENT:
  TITLE = X.                              % OPTIONAL
  INTERFACE = MCS.
  OPENMESSAGE = TRUE.                     % OPTIONAL
  RESIDENCE = DISK.                       % OPTIONAL

PROGRAM X6 ASSIGNMENT:
  TITLE = X.                              % OPTIONAL
  INTERFACE = NONPARTICIPATION.
  RESIDENCE = DISK.                       % OPTIONAL
  EXECUTE   = MANUAL.                     % OPTIONAL
```

```
PROGRAM X7 ASSIGNMENT:
  TITLE = X.                              % OPTIONAL
  INTERFACE = PARTICIPATION.              % OPTIONAL
  COMMONSIZE = 100.                       % OPTIONAL
  TRANCODE = X(1,1).                      % OPTIONAL
  OPENMESSAGE = TRUE.                     % OPTIONAL
  RESIDENCE = DISK.                       % OPTIONAL
  EXECUTE = BOJ.                          % OPTIONAL

PROGRAM X8 PASS:
  TITLE =X.                               % OPTIONAL
  INTERFACE=MCS.
  RESIDENCE = DISK.                       % OPTIONAL

PROGRAM X9 PASS:
  TITLE = X.                              % OPTIONAL
  INTERFACE = NONPARTICIPATION.
  RESIDENCE = DISK.                       % OPTIONAL
  EXECUTE = MANUAL.                       % OPTIONAL

PROGRAM X10 PASS:
  TITLE = X.                              % OPTIONAL
  INTERFACE = PARTICIPATION               % OPTIONAL
  COMMONSIZE = 100.                       % OPTIONAL
  TRANCODE = X(1,1).                      % OPTIONAL
  RESIDENCE = DISK.                       %OPTIONAL
  EXECUTE = BOJ.                          % OPTIONAL
```

Several general comments apply to these TCL descriptions:

1. More than one TRANCODE statement is allowed where a TRANCODE statement is optional or required.

2. In a Program description, no PROGRAM statement may occur more than once.

3. The following defaults are in effect:

   a. The default for OPENMESSAGE, ATTACHMESSAGE, and DETACHMESSAGE is FALSE.

   b. The default for RESIDENCE is CORE.

   c. The default for COMMONSIZE is 60.

   d. The default for INTERFACE is PARTICIPATION.

   e. The default for EXECUTE is MANUAL.

Each program uses one of the 10 possible combinations best. The following discussion explains how to use the TCL to describe the program to GEMCOS in different situations.

For example, a program which was written prior to acquiring GEMCOS might not require any GEMCOS options other than the capability of remote execution by authorized personnel. This program needs to be described in the TCL as X1. Its program name should be associated in the ACCESSCONTROL statement with access keys which are then assigned to authorized persons. Since the program is defined as a utility program, it can be executed from any station. Since it uses the Nonparticipation interface, it does not have to be recompiled to conform to the GEMCOS common-area header or to the B 1700/B 1800 MCS/Network Controller interface.

In another case, the user may be using CANDE and ODESY, which are MCS programs, and station operators would like the ability to switch between these programs. Without GEMCOS, both CANDE and ODESY must be shut down and brought back up to switch a station. This is very inconvenient to the rest of the station operators. However, if both CANDE and ODESY are defined as X2 programs, GEMCOS, as a supervisory MCS, can switch a station between these two MCS programs on demand from the station without interrupting the rest of the stations. Since sessions with CANDE and ODESY are to be initiated remotely, they are described as utility programs. Since both are MCS programs, they must use the MCS interface.

In a third situation, the user might want to design a program which could be initiated remotely and take advantage of the GEMCOS formatting capability. To support remote execution, this program must be a utility program. If formatting is to take place, trancodes must be defined, and a common-area header is required. The X3 TCL description would be used.

Another situation might arise in which there are several stations, each requiring access to several data bases. It would be undesirable to have one large program handle all of the data bases. Conversely, writing several small modular programs for each data base would be more efficient, but would require the user to repeatedly execute and halt these programs. This is an especially awkward and time-consuming process for rarely-used programs that process only a few transactions before the next program must be executed.

The preferable solution is to assign trancodes for each data-base accessing method, write modular programs to handle these trancodes, place these programs in the mix (possibly as disk-resident programs), and describe a series of X4 programs in the TCL. The trancodes would be

used to switch messages from any of the stations to any of the programs (and could be used to take advantage of formatting).

In a fifth situation, the user may have the Remote Job Entry (RJE) package. RJE is an MCS which requires a line to a host system. It is executed from the supervisory console printer or a card reader, not by the host system. If the host system is occasionally used by some program other than RJE which runs under GEMCOS, the host system would have to be included in the remote file opened by GEMCOS. GEMCOS would allow RJE to gain access to the host system, if RJE was described as a X5 program. RJE uses the MCS interface and normally communicates with the same station, making it an assignment program.

Finally, in certain applications, the program attaches itself to stations. It is either undesirable or impossible for stations to initiate the program. Perhaps a station is to be dedicated to a program or the station is an output-only device. Programs which determine the message routing via remote file attachment can be accommodated by being described as either an X6 or X7 program. If no GEMCOS capabilities are required, the X6 description suffices. If GEMCOS is to provide functions (such as formatting, audit, screen wraparound) for the program, the X7 description is necessary.

NON-STANDARD ROUTING

Non-standard routing includes the cases in which messages are routed by trancode from station to station, program to station, or program to program. Note that recovery is not defined for these cases.

Station To Station

The originating station must enter a trancode assigned to the destination station. The message will be routed to the destination, and any response must be explicitly sent by the user of this station.

Routing From Programs

In order for a program to route a message using a trancode, or to any place other than the originator, the MSGDESTINATION field in the common-area header must be set. This field not only specifies the immediate type of routing, but also indicates what to do with the response. Following is a description of the possible values of the MSGDESTINATION field:

Value	Definition
0	Send to station indicated in LSN field (final destination).
1	Send to program indicated in PGMNBR field (final destination).
2	Route by trancode (final destination).
3	Route by trancode (destination should be a program; response to be sent to specified LSN; GEMCOS will set MSGDESTINATION field in common-area header to 0 and set LSN field to LSN supplied by the program in the common-area header before sending the message).
4	Route by trancode (destination should be a program; response to be sent to originating program; GEMCOS will set MSGDESTINATION field in common-area header to 1 and set PGMNBR field to originating program number before sending the message).
5	This value is set by GEMCOS to indicate that this message came from a routeheader station.

If a message is routed to a program via TBR (DEST-TYPE is 3 or 4), GEMCOS would change DEST-TYPE to 0 or 1 before sending the message; i.e., the receiving program does not need to do anything to the common-area header in order to return the message to the originator.

If the request cannot be handled (for example, requested program is not running or requested station is busy), an error message would be returned to the requesting program. See the definition of the ERROR field in Common-Area Header in Section 2 for a list of possible error codes.

NOTE

It is the responsibility of the program to keep track of the LSN when routing a message to another program. The PGMNBR field redefines the LSN field.

Examples:

Assume the following TCL:

```
.
.
.
.
BEGIN
     PROGRAM PROGRAM1 USER:
          TRANCODE = PROG1.
          TRANCODEPOSITION = 1.
     PROGRAM PROGRAM2 USER:
          TRANCODE = PROG2.
          TRANCODEPOSITION = 1.
     STATION STATION1:
          TRANCODE = STA1.
          TRANCODEPOSITION = 1.
     STATION STATION2:
          TRANCODE = STA2.
          TRANCODEPOSITION = 1.
.
.
.
.
```

Example 1 (STATION1 to STATION2):

1. Operator at STATION1 wants to send a message to operator at STATION2. He precedes his message with the trancode "STA2".

2. GEMCOS sends the message to STATION2.

Example 2 (STATION1 to PROGRAM1 to PROGRAM2 to STATION1):

1. Operator at STATION1 sends data with trancode of PROG1.

2. GEMCOS routes the message to PROGRAM1.

3. PROGRAM1 receives the message. It cannot process it, so it places a 3 in MSGDESTINATION (TBR and return to station), places a trancode of PROG2 in front of the message, and sends it.

4. GEMCOS changes MSGDESTINATION to 0 and routes the message to PROGRAM2.

5. PROGRAM2 processes the message, and sends it; it does not need to alter MSGDESTINATION, as it was set to 0 by GEMCOS.

6. GEMCOS routes the message to STATION1.

Example 3 (STATION1 to PROGRAM1 to STATION2 and STATION1):

1. Operator at STATION1 enters data with a trancode of PROG1.

2. GEMCOS routes the message to PROGRAM1.

3. PROGRAM1 receives the message. This transaction indicates that the system will soon need to come down. The other station must be warned, so PROGRAM1 sends a warning by setting the MSGDESTINATION to 2 (TBR - final destination), setting the trancode to STA2, and sending the message. PROGRAM1 then finishes processing, restores the MSGDESTINATION to 0, and sends the final results.

4. GEMCOS routes the warning message to STATION2, and routes the results of the transaction back to STATION1.

Example 4 (STATION1 to PROGRAM1 to PROGRAM2 to PROGRAM1 to STATION1):

1. Operator at STATION1 enters the trancode PROG1 along with his data.

2. GEMCOS routes the message to PROGRAM1.

3. PROGRAM2 receives the message. In processing the message, it is found that certain information is needed which is in a data base available only to PROGRAM2. It places 4 in MSGDESTINATION (TBR and return to program), and sends its data with a trancode of PROG2.

4. GEMCOS changes the MSGDESTINATION to 1 (return to indicated program - final destination), and sends to PROGRAM2.

5. PROGRAM2 receives the message, processes it, and sends the results; note that MSGDESTINATION did not have to be altered because it was set to 1 by GEMCOS.

6. GEMCOS routes the message to PROGRAM1.

7. PROGRAM1 finishes processing, restores the MSGDESTINATION in the common-area header to 0 and sends the final results.

8. GEMCOS sends the message back to STATION1.

SECTION 5

USING PORT FILES

Port file communication is an asynchronous movement of data which proceeds directly from a program to the queues maintained by the Master Control Program (MCP). Port files allow GEMCOS on one computer to commmunicate with GEMCOS on other computers using Burroughs Network Architecture (BNA). They also allow communication between programs on the same computer. This feature is only available with the 12.00 release (or later) of the MCP.

USING STATIONS AS PORTS

A user can declare stations as ports rather than as data communications stations. To do this, the PORT STATION statement in the TCL is set to TRUE.

In this situation, the user also needs a program which emulates a port station and has a port file. The user's program needs to:

1. Specify KIND = PORT.
2. Have an internal name of HOSTPORT.
3. Have a default name of GEMPORT.

USING PORT PROGRAMS

In the Program section of the TCL, users can declare programs which use port files rather than remote files. To do this, the PROGRAM TYPE is set to PORT.

In this case, users also must have a program which emulates a port program. This program must have a port file (specify KIND = PORT) with the name TPPORT.

The MAXCOPIES attribute of a port program is always set to 1.

A port program can be executed in three ways:

1. Manually, from a Control station.
2. At beginning-of-job.
3. By using the ON DEMAND option.

After the port program has been executed, GEMCOS opens a subport of the port file TPPORT. In order to communicate with GEMCOS, the matching port file in the port program also must be opened.

If only one of these port files has been opened, the program with the open port file waits for the matching port to be opened. The status of this program is: WAIT FOR PORT OPEN.

The user can stop any port program by entering a HAP command at any Control station. When this is done, GEMCOS sends a message (see Message 27) which tells the program to go to end-of-job and to close its associated subport. The program must close its subport and stop running if it receives this message from GEMCOS.

The following shows how the port file interface is written in the TCL.

```
PROGRAM E PORT:
   INTERFACE  = PARTICIPATION.
   TITLE      = PORTPROG.
   TRANCODE   = XFER.
   COMMONSIZE = 60.
   PORTSIZE   = 500.
   HOST       = LABASE. % IF HOST STATEMENT IS NOT
                        % DECLARED, THE LOCAL HOST
                        % ON WHICH GEMCOS IS EXECUTING
                        % IS USED.
```

For additional information, see the TCL statements concerning port files, port stations, and port programs. These statements are in Section 2 of this manual.

SUMMARY OF PORT FILE STATEMENTS IN THE TCL

The Global section of the TCL has one statement used with port files, the MYNAME statement.

The Program section contains information on how to specify a port program. The HOST and PORTSIZE statements, which apply to port programs, are also in this section.

Four statements in the Station section are used with the port file interface. They are:

1. The PORTSIZE statement.
2. The PORTSTATION statement.
3. The STATIONHOSTNAME statement.
4. The STATIONYOURNAME statement.

In addition, several Network Control Commands are used with port files. They include:

1. The ENABLE PORT STATIONS (EPS) command.

2. The DISABLE PORT STATIONS (DPS) command.

3. Two commands for updating port programs:

    a. UPDATE STATION HOSTNAME command.
    b. UPDATE STATION YOURNAME command.

Please refer to Section 3 for additional information on these Network Control Commands.

For more information on port files, please refer to:

- Burroughs Network Architecture, Architectural Description Reference Manual.

- Burroughs Network Architecture, Network Control Reference Manual.

- B 1000 Burroughs Network Architecture Installation and Operations Manual.

SECTION 6

SELECTING OPTIONS FOR ACCESS CONTROL (SECURITY)

B 1000 GEMCOS provides for both access security and process security. Access security prevents unauthorized persons from using the system. Process security limits the functions an authorized person is allowed to perform.

ACCESS SECURITY

Access security is implemented by requiring a user to sign on with a valid usercode (access key) before the station accepts any messages. The same user code also can be used to sign on at several stations simultaneously.

Some stations may already be physically secure. Therefore, signing-on is not necessary at all stations. Each installation can specify which stations do not require signing-on.

A list of active usercodes is defined in the TCL and is stored in the MCSTIC file of the TCL compiler. The TCL compiler also updates the list of active usercodes. At any time, a station may be enabled or disabled. If the station is disabled, a user cannot sign on until the station is enabled again.

Usercodes can also be restricted to a specified station or stations. A user is allowed to sign on only if the following conditions are met:

1. The station requires sign-on.
2. The station is not already signed on.
3. The user enters a valid usercode.
4. The access code entered is valid at the station being used.

PROCESS SECURITY

GEMCOS offers two types of process security. Transaction security limits which transaction codes a signed-on user can enter. Program security is used when messages do not have transaction codes.

A usercode can be limited to any combination of stations, trancodes, or programs. As usercodes are defined in the TCL, they are associated with a list of valid transaction codes and program identifiers. When users sign on, they are restricted to those transactions or programs associated with their usercode in the TCL.

DEFINING ACCESS CONTROL IN THE TCL

Access control is defined in three sections of the TCL. The Program section defines each program and the transaction codes for each program. The ACCESS CONTROL statement defines the access codes, the programs, and the transaction codes that each access code can use. The Station section defines the stations in the network and the access codes required to sign on at each station.

A GEMCOS MCS for an on-line system with three programs (a user program with trancodes UPDATE and INQ, a user program with trancode EDIT, and a utility program) could be generated without security using the following TCL deck:

```
CONTROL = GENERATE, LIST, COMPILE.
GLOBAL:
    CHANGEREQUESTS = TRUE.
    PROGRAMBOJEOJ = TRUE.
    SYSTEMHALT = TRUE.
    MAXTEXTSIZE = 1980.
BEGIN
    PROGRAM PAYROLL USER:
       TITLE = PAYROLL.
       TRANCODE = UPDATE.
       TRANCODE = INQ.
    PROGRAM TEXTEDIT USER:
       TITLE = USERPACK/TEXTEDIT/.
       TRANCODE = EDIT.
    PROGRAM GAME UTILITY:
       TITLE = GAME.
    STATION TD8A:
    STATION TD8B:
    STATION TD8C:
END.
```

The following TCL deck can be used to add user codes to this sample system. Each person is required to sign on to GEMCOS with a valid user code before using the system. Once signed on, the user in this example is permitted to use any program or trancode. The user may sign on at any station.

```
CONTROL = GENERATE, LIST, COMPILE.
GLOBAL:
    CHANGEREQUESTS = TRUE.
    PROGRAMBOJEOJ = TRUE.
    SYSTEMHALT = TRUE.
    MAXTEXTSIZE = 1980.
BEGIN
    ACCESSCONTROL:
       ACCESSKEY JOE = ALL.     % ALL MEANS THIS
       ACCESSKEY JIM = ALL.     % ACCESSCODE IS
       ACCESSKEY TOM = ALL.     % ALLOWED TO USE ALL
                                % PROGRAMS AND
                                % TRANCODES.
    PROGRAM PAYROLL USER:
       TITLE = PAYROLL.
       TRANCODE = UPDATE.
       TRANCODE = INQ.
    PROGRAM TEXTEDIT USER:
       TITLE = USERPACK/TEXTEDIT/.
       TRANCODE = EDIT.
    PROGRAM GAME UTILITY:
       TITLE = GAME.
    STATION TD8A:
       SIGNON = TRUE.
       VALIDACCESSKEYS = ALL.   % ALL MEANS ALL
    STATION TD8B:               % USERCODES CAN
       SIGNON = TRUE.           % SIGN ON AT
       VALIDACCESSKEYS = ALL.   % THIS STATION
    STATION TD8C:
       SIGNON = TRUE.
       VALIDACCESSKEYS = ALL.
END.
```

The preceding TCL deck provides three access codes: JOE, JIM, and TOM. All three codes can be used at any station and, once signed on at any station, any program or transaction code can be used. A more restrictive system can be defined. For example, user code TOM could be restricted to trancode EDIT only, and user code JIM to trancode INQ and program GAME, while allowing user code JOE to use all trancodes and programs. The TCL deck with this more restrictive access security scheme would be as follows:

```
CONTROL = GENERATE, LIST, COMPILE.
GLOBAL:
    CHANGEREQUESTS = TRUE.
    PROGRAMBOJEOJ = TRUE.
    SYSTEMHALT = TRUE.
    MAXTEXTSIZE = 1980.
BEGIN
    ACCESSCONTROL:
       ACCESSKEY JOE = ALL.
       ACCESSKEY JIM = INQ, GAME.
       ACCESSKEY TOM = EDIT.
    PROGRAM PAYROLL USER:
       TITLE = PAYROLL.
       TRANCODE = UPDATE.
       TRANCODE = INQ.
    PROGRAM TEXTEDIT USER:
       TITLE = USERPACK/TEXTEDIT/.
       TRANCODE = EDIT.
    PROGRAM GAME UTILITY:
       TITLE = GAME.
       STATION TD8A:
       SIGNON = TRUE.
       VALIDACCESSKEYS = ALL.
    STATION TD8B:
       SIGNON = TRUE.
       VALIDACCESSKEYS = ALL.
    STATION TD8C:
       SIGNON = TRUE.
       VALIDACCESSKEYS = ALL.
END.
```

Finally, certain user codes may be restricted to particular stations. For example, to allow JIM to sign on only at station TD8A, JOE at TD8A and TD8C, and TOM at all three stations, the TCL deck would be as follows:

```
CONTROL = GENERATE, LIST, COMPILE.
GLOBAL:
    CHANGEREQUESTS = TRUE.
    PROGRAMBOJEOJ = TRUE.
    SYSTEMHALT = TRUE.
    MAXTEXTSIZE = 1980.
BEGIN
    ACCESSCONTROL:
       ACCESSKEY JOE = ALL.
       ACCESSKEY JIM = INQ, GAME.
       ACCESSKEY TOM = EDIT.
    PROGRAM PAYROLL USER:
       TITLE = PAYROLL.
       TRANCODE = UPDATE.
       TRANCODE = INQ.
    PROGRAM TEXTEDIT USER:
       TITLE = USERPACK/TEXTEDIT/.
       TRANCODE = EDIT.
    PROGRAM GAME UTILITY:
       TITLE = GAME.
    STATION TD8A:
       SIGNON = TRUE.
       VALIDACCESSKEYS = ALL.
    STATION TD8B:
       SIGNON = TRUE.
       VALIDACCESSKEYS = TOM.
    STATION TD8C:
       SIGNON = TRUE.
       VALIDACCESSKEYS = JOE, TOM.
END.
```

SECTION 8

TESTING, PATCHING, TIMING, AND DEBUGGING

This section discusses programs and procedures to use in testing, patching, timing, and debugging GEMCOS. The first of these programs is MCSSIM, an auxiliary program used to test a GEMCOS MCS.

TESTING

The program MCSSIM makes it possible to test a GEMCOS MCS, including user MESS code, without the presence of a Network Controller or a remote network. Inputs are simulated using a card reader and outputs are listed on a line printer.

The MCS runs in simulation mode when the SIMULATION statement is specified with a value of TRUE. In simulation mode, the MCS changes the device type of MCSQUEUE from a remote file to a queue file. Hence, input message traffic no longer comes from the network but from any program writing into a queue file labeled MCSQUEUE. MCSSIM is such a program. The source of MCSSIM is MCSIMS.

MCSSIM expects a card file labelled MCSSIMCRD. The format of MCSSIMCRD is closely related to the B 1000 MCS/Network Controller interface. The MCS/NC interface consists of a series of message types, each having an explicit function. For a definition of these message types, refer to the B 1000 Systems Network Definition Language (NDL) Reference Manual. The user should be familiar with the MCS/Network Controller interface before attempting to use a GEMCOS MCS in simulation mode.

MCSSIM reads cards in a format defined by the MCS/Network Controller interface and builds a record to be sent to the MCS. The information for each record to be sent to the MCS must be punched beginning in card column 1, extending to column 80, and onto succeeding cards if necessary. Column positions of the cards correspond directly to byte positions of the message types. MCSSIM is able to determine how many cards are required, concatenating the card images into the record to be placed into MCSQUEUE.

For example, if a station-status reply was to be sent to the MCS, one card would be punched with 80 characters of information. A second card would be required for the remaining 35 characters (a station-status reply is always 116 characters long). If a message from a program with a text size of 150 characters were to be simulated, three cards would be

required. The first card would have the appropriate 50-character message header (with the TEXTSIZE field set to 150) and the first 30 characters of text. The second card would contain the next 80 characters of text. The third card would have the remaining 40 characters.

One exception exists to the above MCSSIMCRD card format rule. For data message types, the ERROR field, corresponding to card columns 24 and 25, represents 16 bits of information, not two characters of information. The two card columns cannot be used to express all possible combinations of 16 bits. Therefore, card columns 24 and 25 remain blank. The information to be placed in the ERROR field of the record is expressed in the FILLER field, columns 45 through 50, as a decimal number representing the desired bit configuration.

MCSSIM converts the information in the FILLER field to a bit string, places the results into the ERROR field and blanks out the FILLER field. To simulate a time-out condition, for example, the value 004096 would be punched into the FILLER field.

When MCSSIM is to simulate messages from a program to the MCS, and that program is using a Participation interface, the common-area header must be present. For example, suppose that a 10-character message is simulated from a program whose COMMONSIZE is 60. Two cards are required. The first card would have a MCS/Network Controller header in columns 1 through 50. Columns 51 through 80 would contain the first 30 bytes of the common-area header. Columns 1 through 30 for the second card would hold the remaining 30 bytes of common-area information. The 10 characters of text would follow in columns 31 through 40.

Regardless of whether the MCS is operating in the simulation mode, it must perform its initialization routines. In order to initialize correctly, the MCS must receive information about the network it is to control or simulate. During initialization, the MCS a remote file information reply (message type 29) as the first record in MCSQUEUE.

In normal mode, the MCS receives this reply from the Network Controller by issuing a remote file information request. From the remote file information reply, the MCS identifies the appropriate LSNs. The MCS expects a station status response (message type 21) for each station in its remote file. Here again, in normal mode, the MCS receives these replies from the Logic Network Controller by issuing station status requests. From the station status replies, the MCS is able to associate LSNs with station names.

After the initialization process, the MCS may go through the network restoration procedure. When this is the case, Network Controller change responses (message type 23) may be expected by the MCS. During the initialization process, only change responses and FILE OPEN requests are recognized by the MCS.

Once the MCS ends initialization (or network restoration), it enters its normal message processing mode. Any message can be sent to the MCS for testing purposes. However, care must be taken when the message sent to the MCS causes the MCS to expect a Network Controller response.

For example, a valid Change Station Ready command (*CSR 1n) input to the MCS causes the MCS to issue a change request (message type 22) which must be answered by a Network Controller Change Response message. Somewhere in the MCSSIMCRD deck following the CSR command, there should be a Network Controller change response to the MCS change request. For ease of use, the MCS should not be caused to issue change, recall, or remove requests.

EXAMPLE SIMULATION CARD DECK

In this example (see Figure 8-1), a network with one station, LSNO1, is simulated. The first card of MCSSIMCRD is a remote file information reply specifying that there is one station in the remote file. The second and third cards are a station status reply describing LSNO1 as being ready, enabled, having a retry count of 255, etc. The fourth card simulates a report program status command from LSNO1. Cards five and six simulate a FILE OPEN request by a program named PGMA. A message from PGMA to LSNO1 is simulated with card seven. Note that PGMA has included a common-area header of 60 bytes.

```
?DATA MCSSIMCRD
2900000000000000010000000001       001000
210010001LSN1       11344022500       020207                    07
2550000000001000000000000000000000003
01C001000400100000000000  0000000000000000044C00000:RPS
1030000000000000000000000           PGMA                  0 00002     001CRMT1
     001
0000010068002000C000000  00000000000000000440000000010                0008
                              HI THERE
?END
```

Figure 8-1. Example Simulation Card Deck

PATCHING

Another auxiliary program, MCSFIX, provides the same patching capabilities as the User Programming Language 2 (UPL2) compiler. Thus, a source-code file can be patched, listed, and/or sequenced without having to be compiled. In particular, this program is intended for use with MCSGTS, when it is necessary to release a patch notice.

The file names used by MCSFIX correspond to those of the UPL compiler:

1. Disk input source file is labeled SOURCE.
2. Disk output source file is labeled NEWSOURCE.
3. Card input patch file is labeled CARDS.

MCSFIX always expects the input card file and creates the output disk file (although the disk file may be closed with purge). Records in CARDS fall into one of two categories: control cards and patch cards. Control cards, identified by a dollar sign ($) in column one, specify MCSFIX options. Patch cards and cards without a dollar sign in column 1 are used to create new lines of source code, or to replace existing lines.

Columns 73 through 80 are reserved for optional sequence numbers. When sequence numbers exist in either of the two input files, they should be in ascending order; otherwise, an error is reported. However, when an error is detected, the run is not aborted, even though results are unpredictable (but possibly of value).

The following is a list of available options and their actions:

Option	Action
LIST	This option creates a listing of the output disk file. If a sequence number is specified in columns 73 through 80, the list starts at that number, or if it does not exist, at the next highest number. By default, LIST is set.
MERGE	When this option is set, MCSFIX expects the file labeled SOURCE, merges records from the file CARDS, and creates the output file (refer to NEW option to save the output file). A record from CARDS replaces a record in SOURCE when both have the same sequence number. A record from CARDS is inserted when there is no record in SOURCE with a matching sequence number. (To delete records in SOURCE, refer to VOID option.) When MERGE is not set, an input disk file is not created directly from the input card file. The control card specifying MERGE must precede any patch cards and may not be reset. By default, the MERGE option is off.
NEW	This option causes the output disk file to be saved (closed with lock). The control card with NEW specified must precede any patch cards and may not be reset. By default, the NEW option is off.
NO	This option turns off or reverses the effect of any options which follow it on the same control card. This is different than the UPL compiler (in which NO applies only to the option immediately following it).
SEQ <ssssssss> + IIII	This option causes sequence numbers starting with ssssssss and incrementing by IIII to be applied to the output disk file (and printer listing if LIST is set). IIII may not exceed 9999. NO SEQ terminates any sequencing taking place. Sequence numbers of records in CARDS are always relative to the old sequence numbers in SOURCE regardless of

whether or not the sequencing of the output disk file is specified.

VOID <nnnnnnnn> — The VOID option causes records from SOURCE not to be included in NEWSOURCE (i.e., it deletes them). <nnnnnnnn> must be present in columns 73 through 80. If <mmmmmmmm> is not present, the record in the input disk file with the sequence number <nnnnnnnn> is voided. When <mmmmmmmm> is present in columns 7 through 14, all records with sequence numbers <nnnnnnnn> through <mmmmmmmm>, inclusive, are voided.

To execute MCSFIX, the following control cards are used:

```
? EXECUTE MCSFIX
? FILE SOURCE NAME <input-file-identifier>;
? FILE NEWSOURCE NAME <output-file-identifier>;
? DATA CARDS
(control and patch cards)
? END
```

TIMING

A timing mechanism helps users keep statistics on the amount of time GEMCOS takes to process various transactions. The TIME command activates the timing mechanism. The TIME command is explained in Section 3. A disk file is built containing the timing results. This file is called MCSTIME/<random number>. Turn the timing mechanism on by entering:

```
* TIME START
```

If the mechanism is not turned on, the memory that the disk file requires will not be used. Table 8-2 shows the layout of MCSTIME:

Table 8-2

Layout of MCSTIME

Field Name	Length	Explanation
TYPE	BIT(1)	1 = Input, 0 = Output
SEQUENCE-NBR	BIT(23)	Number in Common Header.
TIME-IN	BIT(24)	Time the message came in.
TIME-OUT	BIT(24)	Time the message went out.
LSN	BIT(8)	Logical Station Number.
TEXT	CHARACTER(6)	If TYPE = 1, first 6 bytes. If TYPE = 2, length of message.

Users can write their own programs to analyze the MCSTIME file, or they can compile the sample report program on the release tape. The source file is called GEMCOS/MCSTIMRPTS. The source is compiled as follows:

```
CO <OBJ FID> UPL2 LI FILE CARD NAME
  GEMCOS/MCSTIMRPTS DISK DEF;
```

To run the report program, simply execute the <object file ID> and file equate the name of the time file. Enter the following:

```
EX <object file ID>; FILE MCSTIME NAME
MCSTIME/<number>
```

The report program also allows users to analyze only selected transactions. To use this option, SWITCH 0 must be greater than 0. The operator is then allowed to select six bytes. Only transactions beginning with these six characters are analyzed.

In addition, users can also create an optional disk file with fields in character format. This file is created if SWITCH 9 is greater than 0. The file is called MCSTIMEOUT, and has the same fields that are printed in the report. Table 8-3 describes these fields and gives their lengths (in bytes).

Table 8-3

Format of Output File

Field	Length	Explanation
ENTRY	5	Sequential number.
LSN	3	Logical Station Number.
DATA	6	First 6 bytes of message.
TIME-IN-1	8	Time the MCS read the input message.
TIME-IN-2	8	Time the MCS wrote the message.
TIME-IN	5	TIME-IN-1 minus TIME-IN-2.
TIME-IN-PERCENT	3	Percentage of total time.

MESSAGE LENGTH	4	Length of this message.
TIME-OUT-1	8	Time the MCS read an output message.
TIME-OUT-2	8	Time the MCS wrote an output message.
TIME-OUT	5	TIME-OUT-2 minus TIME-OUT-1
TIME-OUT-PERCENT	3	Percentage of total time.
NON-MCS-TIME	5	Total non-MCS time.
NON-MCS-PERCENT	3	Percentage of total time.
TOTAL-TIME	6	Total time for this transaction.

Since 8.38 seconds is the maximum time that can be stored as the value returned by the timer function, times in excess of 8.38 seconds will not be accurately recorded.

DEBUGGING

The following material gives information on debugging using the GEMCOS monitor trace and the GEMCOS data dump.

USING THE GEMCOS MONITOR TRACE

The monitor is a procedure within the MCS which produces a listing which can be used to trace the logic flow. Calls on the monitor are made at the entrance to nearly every procedure, and at other key points in the MCS. Since MESS procedures can also call on the monitor, the monitor is a valuable tool for debugging interfaces between MESS code and the standard MCS.

The monitor listing displays:

1. An identification of the procedure now executing.

2. Any information pertinent to the current procedure.

3. The sequence number in the MCS source code file at the point where the monitor was called.

Since the monitor is a generative option, the MCS used for normal operation does not have to carry the overhead of monitor logic. Users can generate a second MCS which is exactly like the production MCS, except that the MONITOR generation parameter is set in the TCL. The second MCS is used only if problems arise that need to be diagnosed.

When an MCS is generated with the MONITOR parameter set, the monitor listing can be turned on and off either dynamically, by using a Network Control Command, or between executions of the MCS, by the TCL compiler. Furthermore, the monitor can be selected for individual procedures, or only for those procedures suspected of being related to a problem. The methods of changing the monitor parameters, calling the monitor trace, and a sample monitor trace are discussed in the following material.

Changing the Monitor Flag

Change the settings of the monitor flag in either of these two ways:

1. When the MCS is running, enter a CMF (change monitor flag) Network Control command.

2. When the MCS is not running, run the TCL compiler with REGENERATE and set MONITORTRACEON TRUE or FALSE.

Calling the Monitor Procedure

If a procedure is to be monitored, it must contain at least one call on the monitor procedure. The monitor is invoked by a UPL2 statement of the form:

```
MD_MONITOR(s1, s2);
```

The MD_MONITOR parameters are as follows:

1. s1 is a string of up to 30 characters specifying the name of the current procedure.

2. s2 is a string of up to 82 characters which contains any information that may be useful for debugging (usually the names and contents of significant data items).

Each time MD_MONITOR is invoked, these parameters are printed in order on one line of the monitor listing, along with the sequence number of the line which called the MONITOR procedure.

Monitor calls are ordinarily the first statement in each routine that can be executed. However, they can be placed anywhere that is useful for debugging.

Sample Monitor Trace

The following is an example of a portion of a GEMCOS trace.

```
        .
        .
        .
        READ BY MCS
<aa>    TYPE=01, VARIANT=0, LSN=004, TEXT SIZE=0005, REMOTE FILE NO=..
        TALLY=000000000, TOGGLE=00000000, TERMINAL TYPE=44
<bb>--  TRAN1

<cc>--  MCS_PROCESS_MSG   23096000 SG.HDR_TYPE=01 SG.HDR_LSN=..
        MCS_MSG_FROM_STATION   20006100 AD.MSG_SOURCE=02 AD.MSG_LSN=..
        MCS_TRANCODE_RECOGNIZER  17200000 AL.NPR_NUM_TRN=0006 MS_ ....

        .
        .
        .
```

Each entry in the trace is preceded by one of two headings: READ BY MCS or WRITTEN BY MCS. The NDL message type is immediately below this heading and indicates the type of message being processed. The remaining fields in the first two lines above (marked <aa>) are fields in the NDL 50-byte header. For documentation on these fields, see the NDL Manual (or DOCUMENT/NDL on the SYSTEM tape.)

The next line of the sample trace is the data text line(s). It is the line marked <bb> above. There are up to 80 characters of data on a line. In this example, the text size is only 5, so the user knows that TRAN1 is the entire message. Unprintable characters are displayed between the characters < and > (for example, <0D25> is a line-feed, carriage return).

The next portion (labeled <cc>) is divided into three columns. The first column contains the names of the called procedures. The second column contains the sequence numbers at which these procedures were invoked. The third column contains the values of those GEMCOS variables which the programmer wanted in the trace. Note that starting with the 6.00 release, the module numbers have been deleted from this section.

USING THE GEMCOS DATA DUMP

A data dump can be created on demand, by entering the Network Control Command *RDM PRINT from any station. The data dump can also be created automatically, when the MCS detects a serious error.

The data dump is divided into sections, some of which are optional, depending parameters which were sent in the TCL.

It provides a "snapshot picture" of the state of the MCS, certain significant data items, and the message work area. It can be used for debugging and for reporting statistical information maintained in the GEMCOS tables.

When a problem occurs, a data dump should be done immediately afterwards. In some cases, it should be done both before and after the problem. Note that the GEMCOS MCS will automatically produce a data dump if a fatal error occurs.

The first section in the data dump is one of the most important. It reflects the status of the Station Table at the time of the dump. The stations are listed in the order that GEMCOS learns of their existence. Index 0 is reserved for the ODT. The actual LSN is shown in the third field from the end of the second line.

To correlate the fields with the source, simply procede down the entries in the Station Table. The order is the same until the field ADDED is encountered in the second line. Although the remaining fields are not in exact order, it should be obvious which headings correspond to which fields. If you have any doubt, simply refer to the routine

MCS-DUMP-STATION-TABLE in the source file. Any fields which have a 1-digit value are Boolean fields. A 1 means true and a 0 means false.

The next table printed is the Station Statistics Table. This section of the report presents a record of data comm activity for each station. Note that stations are shown in the same order as in the previous section.

The following section is a printout of the Active Users Table. There is one entry for each user declared in the TCL. The field USR.PGM.MASK indicates which programs each user is allowed to access. That is, the first digit represents the first program, the second digit the second program, etc. Note that there will be as many digits as there are programs in TCL. If a 1 is listed for a user and program in the table, then that user is allowed to access that program. A 0 indicates that the user cannot access that program.

The field USR.TRN.MASK lists the trancodes which a user is allowed to enter. This mask works in the same manner as the previous one, except that in this case, the digits represent trancodes. Again, there is one digit per trancode declared in the TCL.

The File Table is in the next section. This table shows all the remote files opened under control of GEMCOS. The first entry, index zero, is for the GEMCOS MCS itself. The remaining entries are assigned at file open time. The QUEUE NUMBER field is the remote file number. The field PROGRAM LINK is the Link to Program Table. Finally, the field FILE.STATION.MASK tells which stations are attached to that file. It operates in the same manner as the USER PROGRAM MASK above.

After this is the Program Table. This section contains data for all programs described in the TCL. Again, ENTRY ZERO is reserved for the GEMCOS MCS. The VALID field indicates whether the program is running (1) or not (0). The entry DEFLINK shows the link to the Program Definition Table (PDT) described below. The field FILE LINK is the link to the File Table above.

The following section, the Transaction Table, stores information for each trancode defined in the TCL. The entry PGM.DEF.LINK is a link to the Program Definition Table described below. The entry FORMAT indicates whether a format is to be applied to messages containing this trancode. The entry TBR indicates whether the trancode is for a program (1) or for a station (0). The entries INDEX.1 and INDEX.2 are the optional indices associated with each trancode. Finally, AUDIT indicates whether the trancode is to be audited (1) or not (0).

The Program Definition Table follows. There is is one entry in this table for each program described in TCL. Once again, index zero is reserved for the MCS. The field labelled "TYPE" indicates to which type the program belongs: type assignment program (0), utility program (1), or user program (2).

The field labeled DISTRIB contains the index to the Program Table for the latest copy of the program. The field REC indicates whether the program is in recovery (1) or not (0). The field IFACE indicates the program interface, e.g. MCS. Finally, the field marked PGM.TABLE.MASK indicates whether the program and/or any of its copies is running (1) or not (0). There are as many digits as there are programs possible. (A program with MAXCOPIES = 3 counts as three possible programs.)

Next is the Network Parameter Record Report. These fields contain information from the NPR record in the MCSTIC file. This record contains global information concerning the MCS, as well as pointers to the base records for the different record types.

The subsequent section has global information. This section lists certain data declarations at lexic level zero of the GEMCOS source. This section is printed mainly for debugging purposes.

The next two sections, the Format Table and the Function Table, are both optional. These sections list all the formats and functions by index (in the order declared). The listing of disk or memory with each index depends upon whether the format is resident (1) or not (2).

Following the Format Table and the Function Table is the Audit Table. This report simply lists the audit files that have been created during the current run. These audit files are listed by date-time stamp.

The last section of the report is the Optional Data Base Table. This table shows all data bases declared in the TCL and the status of these data bases. The field labeled RESTRTPDTLINK is the link to the Program Definition Table for the Restart program. The field labeled IN.RECOVERY PDTTBLMASK indicates which programs using that data base are in recovery. The programs in recovery are listed in Program Definition Table order. The first digit represents the first program, and so forth.

SECTION 9

USING THE STATION OPTIONS

B 1000 GEMCOS provides an interface to four different types of stations: AP300, MT600, Routeheader, and Standard. This attribute is specified in the STATION statement within the Station section of the TCL. Depending on the type declared, other parameters may also be required. The following material discusses communication with AP300, MT600, and Routeheader stations. It also discusses copying disk files using routeheaders and Burroughs Network Architecture (BNA) Station Transfer.

AP300

The AP300 is a free-standing matrix printer capable of operating with the data communications interface of a host system. B 1000 GEMCOS provides the interface for on-line operations between the AP300 and application programs.

The AP300 has various options which may be programmatically loaded into the printer from a host program. The AP300 has the capacity to send various status conditions to the computer. B 1000 GEMCOS intercepts the status condition messages and displays them at the Network Controller station. Optionally, as specified by the user, GEMCOS forwards the status message to the program to which the AP300 is attached. (This capability exists only if the AP300 is attached to an assignment program.)

MT600

The MT600 modular terminal (sometimes called the soft terminal) is a sophisticated, forms-processing system. When a form is created on the terminal, a program is also created to direct the processing of the form. After processing, the form is stored either in a file at the terminal or in the host processor. If the form is stored in the host processor, it is loaded down-line at a later time. By user specification, the program directs either some or all data fields of the form to the host processor. Similarly, the form receives either some or all data fields. In addition to the memory allocated for the form and the programs, an additional memory area is available for direct communication with the host processor. This area is referred to as the command message area.

To sort out this complex array of communications possibilities, special headers have been designed to precede messages and indicate the nature of the text being sent. With the exception of command message area (C/M area) messages, all messages from the modular terminal have these headers, and all messages to the terminal are preceded by them. Furthermore, all messages other than continued C/M area and forms messages must be followed by a special trailer.

The GEMCOS interface to the modular terminal does not include formatting the input or output of the terminal. The interface capability serves to do the following:

1. Determine the type of message that was sent from the terminal.
2. Extract the trailer and header from the input message.
3. Determine the type of message to be sent to the terminal.
4. Add the trailer and header (extracted from the input message) to the output message.

A 1-character field in the common-area header serves to identify all message types, directed to or originating from application programs.

PROCESSING INPUT FROM THE MTS

GEMCOS examines all messages from the MTS and determines whether a header is present in each. When no header is present, a zero is placed in the MSG-TYPE field of the common-area header. Zero indicates that the message is from the common area. When the header is present, the message type is taken from the header and placed in the MSG-TYPE field. The header and trailer are then removed from the message text. Auditing, routing, and other types of processing proceed normally.

PROCESSING OUTPUT TO THE MTS

GEMCOS constructs the header for the response to the MTS by employing the message type that was received from the MTS in the common-area header. Before GEMCOS constructs headers for return messages, it validates the message type. Any message having an invalid header identification is discarded, and an error message is sent to the control station. When the message type is zero (0) in the message from the MTS, no header or trailer is sent with the message returned to the MTS. Otherwise, the header and trailer are created and sent with the MTS.

MTS MESSAGE TYPES

The following list provides the valid message types that are in the common-area header of messages passed between programs and GEMCOS.

Message Type	Meaning
0	Send or receive command message area.
2	Send or receive total form definition.
-2	Send or receive first buffer of total form definition.
3	Send or receive condensed form.
-3	Send or receive first buffer of condensed form.
4	Send or receive all data fields.
6	Send or receive selected data fields.
7	Recovery point message.
8	Last continuation buffer (forms only).
-8	Intermediate continuation buffer (forms only).

ROUTEHEADERS (COMPUTER-TO-COMPUTER COMMUNICATION)

GEMCOS provides a single level of computer-to-computer communication through a feature called routeheaders. The term routeheaders refers to a special 48-byte header placed on the message by GEMCOS before the message is routed to a remote host. This header contains routing information used by GEMCOS and is never seen by the user. Either host can be any Burroughs system from the CMS machines through the B 7800, provided both hosts are running a GEMCOS MCS.

A routeheader station (remote) is treated as a normal station by the GEMCOS MCS. It must be declared in the Station section of the TCL. The TYPE statement is required to identify a particular station as a routeheader station. However, this station is imaginary (there is no physical device). Rather, each computer has one or more "portholes" through which it can communicate with another computer. This porthole is the routeheader station.

ROUTING

A message can only be routed from one computer to another computer using trancodes. Messages can ultimately be routed to either a station or a program on the remote host. All trancodes of messages bound for a routeheader station must be declared for that routeheader station. All trancodes of messages bound from a routeheader station must belong tc some program or station. Trancode assignments must be coordinated between the host systems.

No action is required by the user programs or stations regarding the content of a routeheader. The MCS removes the routeheader on input and reattaches it on output. The output message is automatically routed back to the initiating routeheader station. This means that TBR programs do not have to be modified to accommodate routeheader stations.

NOTE

> The MCS uses the LSN field of the common header as an index to the routeheader table. This LSN field therefore should not be used by the program.

PROTOCOL

The MCS originating a message to a routeheader station (from either a station or a program) prefixes the message with the 48-byte routeheader before sending it. The message must contain a trancode of either a program or station in the receiving host system.

The receiving host strips off the routeheader, and stores it in a table when the message is bound for a program. The message is then routed by trancode, if possible. When the message contains no trancode, or the station or program is not available, the message is discarded and an error code is sent back to the initiating routeheader station.

Upon receipt of a message from a program, the MCS will reattach the saved routeheader, and send the message back to the originating routeheader station. If there are no errors, the response would be routed back to the originating station if the message originated from a station or to the originating program (or wherever the program chooses to route the output), if the message originated from a program. If an error exists, an error message would be routed to the originating station or program.

ACCESS CONTROL

The value specified in the HOSTACCESSKEY statement in the Station section is put into the routeheader before sending any message to the other host. When a message is received on the other host, the host access key value is checked against the valid access keys of that routeheader station, when sign on is required for that station. If the host access key is invalid, an error code would be returned to the originating host, and the message would not be processed. This sign-on verification is completely invisible to the user because it is taken care of within the routeheader.

FORMATTING

Input messages from a local station are formatted as a function of the trancode and device defined in the TCL. Messages from a program to a local station are formatted based on the message-ID, or the trancode, if no message-ID is found.

Originating messages from a routeheader station are formatted based on the format-ID in the routeheader, or the trancode, if no format-ID is found.

Originating messages from a program to a routeheader station are formatted as follows:

1. When a message-ID is specified, it is placed in the routeheader. When the return message is received by the MCS, that specified message-ID is returned in the routeheader and will be used as an output format message-ID, if valid. This postponed formatting allows a program to specify a format to be used after a program on the remote host has processed the message and returned it to the local host.

2. In addition, local formatting will occur based on the trancode.

For messages returned from a program to the MCS on the other host, local formatting occurs based only on the trancode. If a format-ID was specified from the initiating host in the routeheader, it is returned in the return routeheader to that host. If no format-ID was supplied from the initiating host, the message-ID supplied by the program is returned in the routeheader if no local formatting was done.

SUSPENSION

When the MCS determines that it cannot store any more routeheaders from a remote host, it temporarily suspends that host from sending any additional messages by issuing a suspension message. When there is sufficient room to store more routeheaders, a resumption message is sent to the suspended host and processing can then continue.

When the MCS receives a suspension message from another host, it will not allow any messages to be sent to that host until it receives a resumption message. During this time, any attempt to send a message to that host will result in the following error message:

PROGRAM OR STATION NOT AVAILABLE

RECOVERY

Recovery of routeheader stations is not implemented in B 1000 GEMCOS.

ERROR HANDLING

When an error is detected during the processing of a routeheader station, an error code is placed in the routeheader before the message is returned to the originating routeheader station. When a return routeheader is received at the originating host, and the error code is non-zero, one of the following messages is sent to the originating station or program:

1. TRANCODE MISSING
2. PROGRAM OR STATION NOT AVAILABLE
3. USERCODE HAS NO ACCESS TO PROGRAM
4. THIS COMPUTER HAS NO ACCESS TO DESTINATION COMPUTER

5. INVALID ROUTEHEADER TYPE

6. ROUTEHEADER TABLE FULL - CAN NOT PROCESS MESSAGE

7. ROUTEHEADER MESSAGE IS LESS THAN 48 BYTES

8. INVALID INPUT FORMAT REQUEST

9. DESTINATION IS NOT A PROGRAM

10. FORMAT ERROR

NDL CONSIDERATIONS

The request sets and controls used by each pair of routeheader stations must be compatible. More than one type of protocol may be applicable for each of the possible inter-system and intra-system connections.

The point-to-point connection requests and control has been tested and works between two B 1000 computers. The poll-select protocol has worked between a B 1000 computer and a B 6000 computer where the B 1000 is the "master" and the B 6000 is the polled "slave."

Example:

In this example it is assumed that there are two B 1000 computers for each TCL description, although in principle this example is applicable to any two computer systems. Each computer will run a GEMCOS MCS. Those TCL statements which deal with routeheaders are presented here for both Computer 1 and Computer 2.

A graphic representation of the following TCL specificiations on each computer is shown in Figure 9-1.

TCL for Computer 1:

```
BEGIN
   ACCESSCONTROL:
      ACCESSKEY HOST2 = ALL.
      ACCESSKEY USER1 = ALL.
%
      PROGRAM PROGRAM1 USER:
         TRANCODE = PGM1.
         EXECUTE  = ONDEMAND.
%
```

```
      STATION STATION1:
         SIGNON           = TRUE.
         VALIDACCESSKEYS  = USER1.
         TRANCODE         = STN1.
%
      STATION HOST2:
         TYPE             = ROUTEHEADER.
         SIGNON           = TRUE.
         VALIDACCESSKEYS  = HOST2.
         HOSTACCESSKEY    = HOST1.
         TRANCODE         = PGM2, STN2.
   END.
```

TCL for Computer 2:

```
   BEGIN.
      ACCESSCONTROL:
         ACCESSKEY HOST1  = ALL.
         ACCESSKEY USER2  = ALL.
%
      PROGRAM PROGRAM2 USER:
         TRANCODE = PGM2.
         EXECUTE  = ONDEMAND.
%
      STATION STATION2:
         SIGNON           = TRUE.
         VALIDACCESSKEYS  = USER2.
         TRANCODE         = STN2.
%
      STATION HOST1:
         TYPE             = ROUTEHEADER.
         SIGNON           = TRUE.
         VALIDACCESSKEYS  = HOST1.
         HOSTACCESSKEY    = HOST2.
         TRANCODE         = PGM1, STN1.
   END.
```

On HOST1:

1. When either STATION1 or PROGRAM1 enters a message with a trancode of STN2, the message is routed to Computer 2 through the routeheader station and delivered to

STATION2.

2. Any message entered that contains the trancode PGM2 is routed, in similar fashion, to PROGRAM2 on Computer 2. If PROGRAM2 is not currently running, it would be executed automatically. The output from PROGRAM2 is automatically routed back to the originator on Computer 1.

On HOST2:

1. When either STATION2 or PROGRAM2 enters a message with a trancode of STN1, the message is routed to Computer 1 through the routeheader station and delivered to STATION1.

2. Any message entered that contains the trancode PGM1 is routed, in similar fashion, to PROGRAM1 on Computer 1. If PROGRAM1 is not currently running, it would be executed automatically. The output from PROGRAM1 is automatically routed back to the originator on Computer 2.

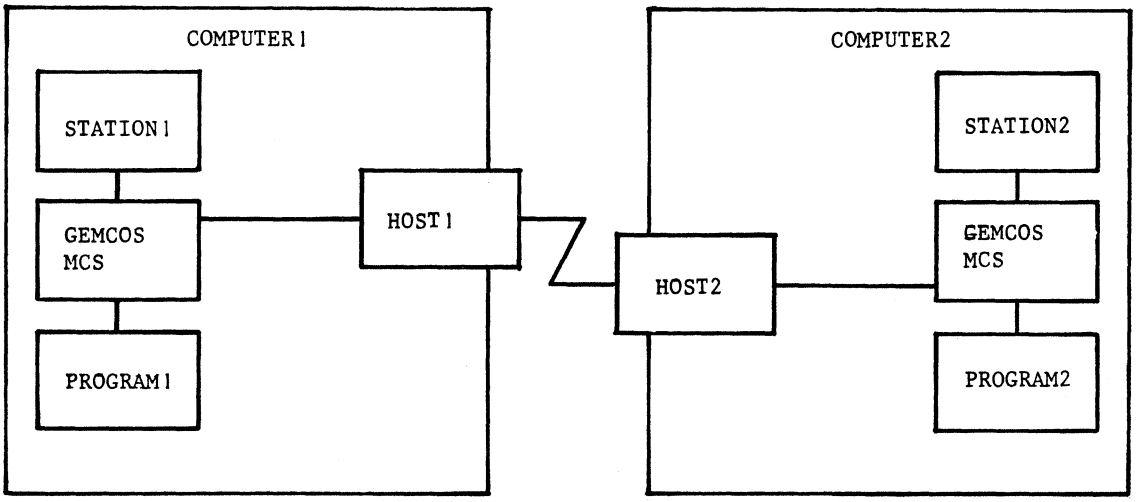


Figure 9-1. Sample Routeheader Configuration

TRANSFERRING DISK FILES

An auxiliary program, MCSFILXFER, allows the user to transfer disk files from one computer to another using the routeheader capability of GEMCOS.

From an MCS-controlled station, a user can enter a command which specifies the name of a file to be transferred to or from another computer. The file transfer program (MCSFILXFER) attempts to transfer the file, and informs the user when the transfer is complete or has failed. This program transfers only one file at a time. Blocking and record size are maintained during a file transfer.

Code files can be transferred across like systems and still be executed. This is possible because all data is transmitted in a manner that is invisible to the user.

The following rules must be adhered to when defining the attributes of the file transfer program in the Program section of the TCL:

1. The program must be declared as a user program.
2. The COMMONSIZE statement must be absent or set to a value of 60 (default size).
3. The program must not use the auditing capability.

Included on the GEMCOS release tape is a fully functional file transfer object program called MCSFILXFER. The user must declare this program to the TCL.

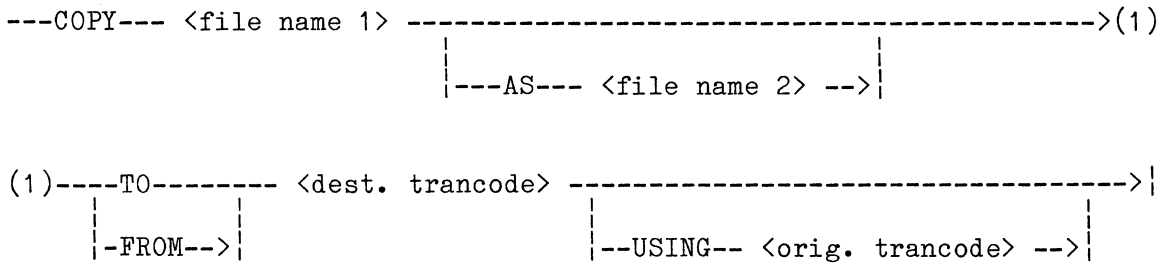
Prior to initially executing the supplied file transfer program, the user must make certain modifications. The external file name of the remote file, MCSREMOTE, must be a remote file known to the Network Controller. Also, the Number Of Stations (NST) attribute must be set to the number of stations requested by the remote file.

See Appendix B for a summary of all files contained in MCSFILXFER.

There are three commands which a user can enter at a given station. The commands are COPY, ABORT and WHAT. The syntax and semantics for these commands follow.

COPY COMMAND

Syntax:



Semantics:

This command is used to initiate the file transfer mechanism. <File name 1> is the name of the file on the system which is sending the file. <File name 2> is the name of the file on the system which is receiving the file. If <file name 2> is not specified, it will default to <file name 1>. Whenever a file name is specified for a B 1000 computer, the above syntax for the file name must be adhered to.

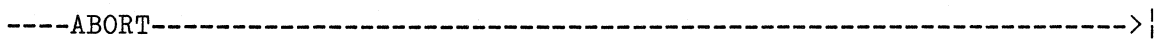
The keyword TO specifies that a disk file is to be sent to the remote computer from the local computer; the keyword FROM specifies that a disk file is to be sent from the remote computer to the local computer.

The destination trancode must be a valid trancode for the remote computer's file transfer program running under a GEMCOS MCS. The user must know this trancode. This trancode and any other possible destination trancodes that the user may wish to use for a particular computer (routeheader station) must be declared for that routeheader station on the local computer.

The origination trancode specifies the trancode which the remote file transfer program is to use to communicate with the local file transfer program. If not specified, it would default to "FTPLS." on a B 6800, FT4800 on a B 4800, FT1800 on a B 1000 and FTCMS on CMS machines. This trancode, and any other possible origination trancodes that the user may wish to use, including the default value of FT1800, must be declared for the file transfer program on the local computer.

ABORT COMMAND

Syntax:

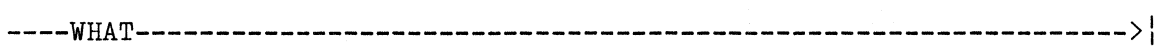


Semantics:

The ABORT command instructs the file transfer program to discontinue the current file transfer, when one is in progress. When the file transfer program receives this command, it instructs the file transfer program on the remote host to discontinue the current transfer, regardless of who initiated the request. This command can be entered at any time.

WHAT COMMAND

Syntax:



Semantics:

This command returns the current status of the file transfer program to the requestor. The possible responses are AVAILABLE and BUSY. If BUSY is returned, the file transfer program will not accept a COPY command.

FILE TRANSFER EXAMPLE

Example:

Assume that a file transfer is to occur between two B 1000 computers. The computer that initiates the file transfer request is called HOSTA, and the other computer is called HOSTB. The following statements exist in the TCL specifications for each GEMCOS MCS on each system.

TCL of GEMCOS on HOSTA:

```
PROGRAM FTS USER:    % FILE TRANSFER PROGRAM ON
                     % HOSTA
   TRANCODE = FT18A, COPY, ABORT, WHAT.
STATION RHA:         % ROUTEHEADER STATION TO HOSTB
   TRANCODE = FT18B.
   TYPE = ROUTEHEADER.
STATION TD830A:      % STATION WHERE COMMANDS WILL
                     % BE ENTERED
```

TCL of GEMCOS on HOSTB:

```
PROGRAM FTB USER:    % FILE TRANSFER PROGRAM ON
                     % HOSTB
   TRANCODE = FT18B, COPY, ABORT, WHAT.
STATION RHB:         % ROUTEHEADER STATION TO HOSTA
   TRANCODE = FT18A.
   TYPE = ROUTEHEADER.
```

A user at station TD830A on HOSTA (under control of a GEMCOS MCS) wishes to transfer the disk file MONDAY/NEWS to HOSTB with a file name of MONDAY/A.NEWS on diskpack BACKUP. The following dialogue accomplishes the requested file transfer:

```
(from TD830A)        : COPY "MONDAY/NEWS" AS
                       "MONDAY/A.NEWS ON BACKUP" TO
                       FT18B USING FT18A

(from FTA)           : COPY INITIATED

(from TD830A)        : WHAT

(from FTA)           : BUSY

(from FTA)           : FILE MONDAY/NEWS TRANSFERRED
                       TO (FT18B)

(from FTB to ODT     : FILE MONDAY/A.NEWS ON BACKUP
 on HOSTB)             TRANSFERRED FROM (FT18A)
```

When file transfer occurs between two B 1000 computers, only one (or neither) file transfer program can use the default origination trancode of FT1800. In this example, the default trancode was not used.

BNA STATION TRANSFER

As of the 7.0 software release, GEMCOS supports Burroughs Network Architecture (BNA) Station Transfer. Users can transfer a station on a remote Burroughs system to their own system and have that station under the control of GEMCOS.

The opposite transfer can also be done. A station under the control of GEMCOS can be transferred to a remote Burroughs system. To do a BNA station transfer while running GEMCOS, the following needs to occur:

1. MCSTCL modifies the GEMCOS remote file MCSQUEUE so that it has a protocol of 98. This allows GEMCOS to be the MCS which actually handles all station transfer activity for the user's system.

2. If GEMCOS is running under SMCS and SMCS is to handle all the station transfers, then GEMCOS needs to be modified. GEMCOS must be modified because only one 98 MCS can be in the mix at one time. Modify GEMCOS as follows:

```
MO <GEMCOS object file> FI MCSQUEUE
    PROTOCOL 0
```

3. Execute BNA programs first at the ODT. (See the BNA Reference Manual for additional information on how to run BNA.) Once the programs have been executed and are in WAIT status, enable the station transfer at the ODT. To do this, enter:

```
NW STATIONTRANSFER +
```

4. The user's TCL must contain an entry for the Port Level Manager's (PLM) program. Declare the PLM program in the TCL as follows:

```
PROGRAM <PID> UTILITY:
   TITLE      = BNA/PLM.
   INTERFACE  = MCS.
   PLMPROGRAM = TRUE.
```

To transfer a station to another system, the user executes the PLM program from his/her station by entering:

```
*EX BNA/PLM
```

Once the station is attached to the BNA program, the user enters:

CONNECT TO <host ID>

(See the BNA Reference Manual for additional operating instructions for station transfer.)

5. The stations which are to transfer in to GEMCOS must be declared as virtual stations in the TCL. A virtual station must have a station hostname. An attempt to transfer in a station not declared in the TCL produces Error 156. For stations transferring in to GEMCOS, set the following attributes (as well as any others required):

```
STATION <station name>:
    VIRTUALSTATION = TRUE.
    STATIONHOSTNAME = <host name>.
```

The station hostname must match the hostname as defined to the BNA network on the other system. If this is not done, the request to transfer in will be denied.

Once a station has been transferred in, it is just like like any other station. However, the RSS Command shows that it is a virtual station. This command also shows its hostname and BNA mode address.

If this station is declared as a control station, then it is able to perform privileged Network Control Commands, such as HLT. If this station is declared as a Monitor station, then it receives error messages.

SECTION 10

USING THE CONVERSATIONAL FEATURE

The conversational feature is a unique method of communication between a participating program and a terminal. Conversations involve three system elements: the station, the program, and the MCS. Participating program messages typically consist of the GEMCOS header and message text. In conversational messages, conversation text is embedded between the header and the message. Conversation text is only updated by the program. When a program sends a message, the MCS removes the conversation text from the message and stores it in the conversation area of memory. As new text is returned from the station, the MCS returns the conversation text to the message before delivering it to the program.

Through this procedure, the program is relieved from declaring tables or allocating memory for the conversation text; instead, these tasks are assumed by the MCS. The MCS allocates a conversation table. Each conversation area is an element of the table. The MCS allocates the conversation table according to the maximum number allowed in the system simultaneously. Areas are only used and brought into memory as required.

Stations may only converse with one program at a time. Conversely, programs may communicate with several stations simultaneously. Normally program messages can still be sent to a station that is conversing with a program. Each conversation is assigned one conversation area. There may be many conversational programs and conversational stations in the TCL. The MCS needs to maintain only one copy of the conversation table. This process eliminates duplicate information and reduces memory usage.

TCL SPECIFICATIONS

To create the conversational capability, three statements are used: CONVERSATIONLIMIT, CONVERSATIONSIZE, and CONVERSATIONAL. For more information on the syntax of these statements, refer to Section 2.

CONVERSATION LIMIT STATEMENT

This statement determines the maximum number of stations that may converse concurrently. The statement is entered in the Global section. Any conversation initiated which exceeds the limit established with this statement causes an error. The initial message of the conversation

which caused the error is not sent to the intended destination; instead, it is returned to the program with an error status in the message header. The program may either periodically attempt to send the initial conversation message until a conversation area is available (i.e., until another conversation has been terminated) or it may send a message notifying the station operator of the temporary shutout.

CONVERSATION SIZE STATEMENT

The conversation size varies among programs. The size for each is established using this statement. When several programs have conversational capabilities, the largest conversation size among them is applied as the conversation size for all conversation areas.

The conversation area allocated by the MCS may be much larger than the conversation sizes specified for certain programs. However, the MCS only provides the number of bytes required by each program. Areas must be allocated at the start of running a program; they cannot be allocated as required. Afterward, the areas may be used and assigned as required.

NOTE

> The message size declared must be large enough for conversation messages. Conversation messages consist of a header, conversation text, and message text. The MAXTEXTSIZE statement is used to establish the message size.

CONVERSATIONAL STATEMENT

This statement is used to assign the conversational capability to stations. The capability is assigned to a station being defined by assigning the value TRUE to the statement; assigning the value FALSE excludes a station from the conversational capability. If a program attempts to communicate with a nonconversational station, an error occurs. (For further information about this statement, refer to "Station Section" in Section 2 of this manual.)

All three statements are declared in the TCL as follows:

1. For the Global section (optional):

   CONVERSATIONLIMIT = <nn>.
   (nn represents a 2-digit number)

2. For the Program section:

   a. CONVERSATIONSIZE = <nnn>.
      (nnn represents a 3-digit number)

   b. INTERFACE = PARTICIPATION.

   c. AUDITOUTPUT = TRUE.
      (This attribute must be declared for recovery of conversations.)

3. For the Station section (optional):

   CONVERSATIONAL = TRUE.
   (TRUE is the default value)

PROCEDURES FOR CONVERSATIONAL PROGRAMS

It is the task of all conversational programs to indicate the beginning and the end of a conversation. They also include the conversation text in the message. Two fields have been added to the message header which enable the program to perform these tasks: the CONVERSATIONBOJEOJ field and the CONVERSATIONSTATUS field.

Once a program receives a message, the value of the CONVERSATIONSTATUS field indicates whether the path is clear for the message, a conversation is in progress at the station, or the message has caused an error. A definition for each value follows.

Value	Definition
0	Path is clear; either no conversation is in progress, or the station is nonconversational.
1	Conversation is in progress at the station. The value of the CONVERSATIONBOJEOJ field indicates whether another program is conversing with the station.
2	Error. The maximum number permitted for simultaneous conversations has been exceeded. The last program message is returned (without the audit) to the program.

3	Error. Program attempted to initiate a conversation with a nonconversational station. The message is returned to the program.
4	Error. Program attempted to converse with a station which is currently conversing with another program. The message is returned to the program in error.
5	Error. A nonconversational program initiated a conversation. The message is returned to the program.

The program designates the appropriate value for the CONVERSATIONBOJEOJ field in the message header. The definition of these values follows.

Value	Definition
0	For messages directed to a station, this value means that no conversation is in progress. The MCS expects message text immediately after the message header. For messages directed to a program, this means that the station is unoccupied, and therefore available for conversation.
1	For messages directed to a station, this value means that a conversation is in progress. Conversation text follows the header and precedes the output text. It is stored in the conversation. However if the program is being audited, both the conversation text and output text would be audited. For messages directed to a program, this means that the station is occupied. The program that is currently conversing with the station receives this value in each conversation message from the station.

A recommended remote file record and working storage definition is presented in Figure 10-1.

NOTE

In Figure 10-1 below, N1 represents the length of the user postion of the common-area header. If there is no user portion, the MCS-USER-AREA should not be given. N2 represents the length of the user

text. N3 represents the length of the user's conversation text. N4 represents N2 minus N3.

```
01  REMOTE-FILE-RECORD.
    05  COMMON-HEADER.
        10                                      PIC 9.
        10  MCS-LSN-NBR                        PIC 9(3).
              .
              .
              .
        10  MCS-OUTPUT-ADDR                    PIC 9(9).
        10  MCS-CONVERSATION-STATUS            PIC 9.
        10  MCS-CONVERSATION-BOJEOJ            PIC 9.
    05  MCS-USER-AREA                          PIC X(N1).
    05  TEXT-AREA                              PIC X(N2).
    05  TEXT-WITH-CONV REDEFINES TEXT-AREA.
        10  CONVERSATION-TEXT                  PIC X(N3).
        10  REGULAR-TEXT                       PIC X(N4).

WORKING-STORAGE SECTION.
01  WS-INPUT                                   PIC X(N4).
      .
      .
      .

01  WS-OUTPUT                                  PIC X(N5).
      .
      .
      .

01  WS-CONVERSATION-AREA                       PIC X(N3).
      .
      .
      .
```

Figure 10-1. Recommended Remote File Record and Working Storage Definition

The proper sequence of events for a user program with conversational capabilities is outlined in the following basic-logic-flow example. (See Figure 10-1 for remote file record and working storage definitions.)

Example:

```
       PROCESS.
          READ MCSQUEUE AT END MOVE 1 TO EOF
              GO TO PROCESS-EXIT.
           IF MCS-TYPE = 24.
               <SEND "TYPE 25" MESSAGE>
               GO TO PROCESS-EXIT.
           IF MCS-TYPE = 21
               <PROCESS & SEND "TYPE 22" MESSAGE>
               GO TO PROCESS-EXIT.
           IF MCS-CONVERSATION-STATUS > 1
               <PROCESS CONVERSATION ERROR>
               GO TO PROCESS-EXIT.
           IF MCS-CONVERSATION-STATUS = 0    OR
               (MCS-CONVERSATION-STATUS = 1  and
               MCS-CONVERSATION-EOJEOJ = 0)
*                NO CONVERSATION TEXT IN MESSAGE
                 MOVE TEXT-AREA TO WS-INPUT
           ELSE
*                CONVERSATION TEXT IN MESSAGE
                 MOVE CONVERSATION-TEXT TO WS-CONVERSATION-AREA
                 MOVE REGULAR-TEXT TO WS-INPUT.
           <START PROCESS>
                 .
                 .
                 .
           IF MCS-CONVERSATION-BOJEOJ = 0  and
              MCS-CONVERSATION-STATUS = 1
              <NO CONVERSATION ALLOWED>
               MOVE WS-OUTPUT TO TEXT-AREA
              <OR BUILD OUTPUT IN TEXT-AREA>
           ELSE
              <MAY INITIATE OR CONTINUE CONVERSATION>
              <TO BEGIN OR CONTINUE CONVERSATION:>
               MOVE WS-CONVERSATION-AREA TO CONVERSATION-TEXT
               MOVE WS-OUTPUT TO REGULAR-TEXT
              <OR BUILD OUTPUT DIRECTLY IN TEXT-AREA>
               MOVE 1 TO MCS-CONVERSATION-BOJEOJ
              <NO CONVERSATION OR TO END A CONVERSATION>
               MOVE WS-OUTPUT TO TEXT-AREA
               MOVE 0 TO MCS-CONVERSATION-BOJEOJ.

           <SEND OUTPUT>

           PROCESS-EXIT.
               EXIT.
```

To avoid reducing recovery efficiency, programs must terminate completed conversations, particularly at stations that continually receive nonconversational messages.

RECOVERY OF CONVERSATIONAL PROGRAMS

The AUDITOUTPUT statement of a conversational program must be declared TRUE in the Program section in order to recover conversation text after a system failure. Conversation text is recovered through audited output messages.

Conversations initiated by a data base or synchronized recovery program which are in progress when a system failure occurs are recovered. New conversations are temporarily excluded from participating stations until the recovery process is complete. However, nonconversational messages are permitted at all stations and programs while the system is being recovered. Stations that are not conversing at the time of the failure may converse with programs that are not required in the recovery process. Once the recovery process is complete, all stations and programs are free to converse.

SUMMARY

The conversational feature provides various benefits. The number of file accesses is reduced by storing data that is repeatedly used by programs. Conversation areas are brought into memory as required. The number of copies of a program does not affect the area size allocated. Also, stations conversing with programs are only assigned one conversation area. These imposed limitations eliminate duplication of information and reduce the space required to store station information for a program. Finally, access to certain transactions and programs can be controlled by restricting the number of stations that are assigned the conversational capability.

SECTION 7

USING AUDIT AND RECOVERY OPTIONS

GEMCOS provides auditing, controlled shutdown, and several options for recovery. This section presents each of these features.

AUDITING

GEMCOS keeps an audit trail of all messages sent to an application program or to a data base. The TCL can be used to create the MCS with just the audit feature, or with both audit and recovery.

When the audit option is used, the MCS makes an audit trail of all messages sent to an application program. For nonsynchronized recovery, messages may be audited. But for synchronized recovery, messages must be audited.

The MCS assigns sequence numbers to messages. These sequence numbers identify the messages. The common-area header communicates the sequence numbers to application programs. In synchronized recovery (available only with the Total version), the MCS also assigns a data-base sequence number when the message is received.

The audit trail is written on a disk file. When the audit file is filled or reaches end-of-job, it is closed and a new one is created. The file is then available for copying to another device.

Each new audit file has its own file identifier. This audit file-ID is MCSAUDIT/AUDITnnn, where <nnn> is a number in the range 0 through 999, and is incremented for each new audit file.

When the MCS needs an audit file for recovery (other than the audit file it is currently creating), and that file is not currently on disk, it displays the following message on the Control station or console printer:

```
FILE MISSING - MCSAUDIT/AUDITnnn
```

While the MCS waits for the backup audit file, it continues to service the network. The operator informs the MCS that the requested backup audit file is available by entering the AOK Network Control Command at the console keyboard.

During recovery, the MCS continues to process messages for programs that are not recovered. When the MCS receives a message which is destined for a program that is being recovered, it sends a message to the originating station indicating that recovery is in progress and that the input message was ignored.

GEMCOS also provides for controlled shutdown when this is needed. The steps in controlled shutdown are discussed in the following material.

CONTROLLED SHUTDOWN

If the data communications system needs to be terminated, GEMCOS provides controlled system shutdown. System shutdown has the following steps:

1. A messsage is sent to all stations, informing them that shutdown has begun.
2. Further input is disabled.
3. The MCS creates an end-of-file condition on all remote files.
4. Any messages remaining in the queues of the Network Controller are recalled and printed on a line printer.
5. The MCS terminates.

Any messages that cannot be delivered to their destinations are accounted for on the printer listing.

The rest of this section discusses GEMCOS recovery options.

SELECTING RECOVERY OPTIONS

B 1000 GEMCOS provides a broad range of recovery options within the TCL. This means that the user can analyze the needs of particular applications and then select the recovery options which best meet those needs.

In this manual, recovery means that a failure has occurred, and that normal transaction processing cannot be continued until some corrective action has been taken. The recovery process is the re-establishment of normal processing.

Throughout this section, assume that the user wants to avoid situations in which file and/or data base reloads are required following a failure. Such failures cause the on-line user network to be non-productive. All transactions that have occurred since the data base was last dumped to backup storage must be reprocessed. The time lost in in this process is costly. The cost is directly related to the following factors:

1. Data base size.
2. Length of time since last data base dump.
3. Transaction volumes since last data base dump.
4. Transaction-throughput activity.

Further, assume that the user desires to free user programmers from recovery concerns as much as possible. This is desirable because recovery solutions are extremely complex for an integrated, batch/on-line system where a data base is being updated in a multiprogramming environment.

The goal of B 1000 GEMCOS is to make the recovery process present, yet virtually invisible to application programs. In return, the user is expected to follow several straightforward programming conventions for normal processing. These conventions vary slightly, depending upon the level of recovery desired.

Although the MCS is "ignorant" of any data management system and contains no data management code embedded within it, the recovery mechanism fully accommodates DMS II (Burroughs Data Management System - Version II). The recovery mechanism works with any data management system as long as the programming conventions discussed in this chapter are followed.

NO RECOVERY

Among the recovery options available to the user is the specification that no recovery be applied to a "failed" program or data base. This option is the default option for every program declared in the TCL.

It may also be specified in the Program section with the following TCL syntax:

```
RECOVERY = NONE.
```

When the system or program fails, no attempt is made to reprocess any messages that may have been lost at the time of failure. This is true regardless of whether the MCS is auditing messages for that program.

RECOVERY UNDER SMCS

The user can recover GEMCOS when it is running under Subordinate MCS (SMCS). To do this, SUBORDINATEMCS must be set to TRUE in the TCL. After a system or GEMCOS failure, GEMCOS tries to reattach all the stations it controlled previously.

GEMCOS must be able to reattach at least one station in order to perform recovery. It needs at least one station to be able to open files. GEMCOS also needs a valid LSN for the Network Definition Language (NDL) header.

NOTE

> Do not change the Network Definition Language (NDL) before starting the MCS, since Logical Station Numbers (LSNs) are used.

It is also possible for GEMCOS to operate without any stations attached if the following conditions have been met:

1. GEMCOS is not in recovery.
2. All stations have entered a DFR command.

In this situation, the only valid command that can be entered at the ODT is a HLT command with no programs running. Any other input causes the following message to be displayed at GEMCOS stations:

```
** NO  STATIONS ACTIVE
```

In most situations, however, GEMCOS must reattach at least one station. Having attached that station, GEMCOS then proceeds with recovery.

Unless the SUPPRESSMESSAGES option has been set for a given station, GEMCOS sends each station a message which tells the operator that the station has been reattached to GEMCOS. GEMCOS also sends this message to the Monitor station(s) or Operator Display Terminal (ODT).

When it has finished attaching stations, GEMCOS sends a message to all of these stations noting the results. The following material discusses situations that happen if GEMCOS cannot reattach a station.

Sometimes GEMCOS cannot reattach a station after a failure because the station attached itself to another MCS before GEMCOS could be reexecuted. If GEMCOS finds a message for such a station in the audit file, recovery is aborted. The user must free the station and retry the recovery.

Note, however, that the common-area header for this message still contains the original (invalid) LSN. GEMCOS displays a message stating that the input LSN is invalid and that the message is being written to the print file.

CAUTION

> If it receives a FILE OPEN message for stations it does not control, GEMCOS aborts. This happens only if, after a failure, GEMCOS cannot reattach stations before they are attached to other programs.

Another special situation occurs with user COBOL74 programs. These programs should not move the Common-Area LSN into the OUTPUT-CD station name. If GEMCOS cannot reattach the station, the COBOL74 program may be assigned an invalid key.

Recovery when GEMCOS is running under SMCS creates several special conditions. Regular recovery is discussed in the following material.

RECOVERY OPTIONS AVAILABLE

The lowest level of recovery, queue restoration, is for programs which do not share a data base with any other program. Queue restoration places the responsibility for determining if recovery is required on the user application program. The user application program must also direct the MCS to begin recovery with a certain specified message. When the MCS receives a request from a program for queue restoration, it re-sends all succeeding audited messages to that program.

The next level of recovery, data base recovery, is for application programs which share a data base with other programs. The only recovery responsibility placed on the user program is to save certain information from the MCS-supplied header in a place where it can be accessed by the restart program (defined later in this section). In DMS II programs, this is normally the restart data set.

The other recovery requirement for such a user program is that the program must notify the MCS when a DMS II abort occurs. When the MCS determines that recovery is needed (at BOJ, because a program has failed, or because a program notified the MCS that a DMS II abort occurred), the MCS executes the restart program. After all messages are received from the restart program, the MCS begins recovery of each program in the data base that requires it. The order of recovery is oldest message first.

The most sophisticated form of recovery, synchronized recovery, is similar to data base recovery. In this type of recovery, the MCS recovers each message in the order that it updated the data base when the message was originally processed. To accomplish this, the MCS assigns a Data Base Sequence Number (DBSN) to each output message from the program. When recovery is required, the order of message recovery is in DBSN order rather than oldest input message first. Also, during synchronized recovery, the MCS performs an output message analysis to minimize duplicate output messages at the station.

QUEUE RESTORATION RECOVERY

When a user application program receives a transaction from the MCS in the course of normal processing, it should check the MCS-INPUT-ADDR field in the common-area header. When this field is nonzero, and this program may need to be recovered later, the data in the MCS-INPUT-ADDR field must be saved.

When recovery is required, the user must return the data in the MCS-INPUT-ADDR field to the MCS. When the MCS-INPUT-ADDR field is zero, the transaction is not audited, and the user application program should ensure that this transaction does not cause any updates unless this program does not use recovery.

The MCS performs queue restoration for programs which have this message in the in the Program section of the TCL:

```
RECOVERY = QUEUERESTORATION.
```

This recovery is initiated when the user application program sends a message to the MCS with the common-area header MCS-TYPE field set to 20. Also, this program should set the common-area header MCS-INPUT-ADDR field to the value it contained in the last completed transaction. This value would have been saved as previously described in the course of normal processing.

The user application program should then ignore all transactions from the MCS until it receives a message with MCS-TYPE set to 21. Next, the user application program should send a message to the MCS with MCS-TYPE set to 22. This "handshaking" process ensures that no extraneous messages are processed by the program.

Finally, the MCS passes all the recovered transactions back to the application program, and normal transactions can again be sent as they are received. The application program determines when recovery is complete by checking the MCS-RECOVERY-STATUS field in the common-area header. A value of zero indicates that this message is a normal message.

When more than one program needs queue restoration concurrently, the MCS passes the messages to the programs in the order they were placed on the audit file in one pass of the audit file. The MCS makes no attempt to reproduce the exact sequence of transactions and/or data base updates which occurred during normal processing.

During queue restoration, the MCS does not perform any output message analysis. The MCS passes all recovered output received from the application programs to the stations.

NONSYNCHRONIZED AND SYNCHRONIZED DATA BASE RECOVERY

The following discussion assumes a basic knowlegde of the B 1000 Data Management System II (DMS II) Reference Manual. Programs that were declared in the Program section of the TCL with a DATA BASE NAME statement fall into one of two categories. The first category consists of programs that were declared with this TCL statement:

```
RECOVERY = DATABASE.
```

This is nonsynchronized data base recovery, called data base recovery in this manual. The second category consists of programs that were declared with the TCL statement:

```
RECOVERY = SYNCHRONIZED.
```

This is synchronized data base recovery, called synchronized recovery in this manual.

Opening a Remote File

When a data base or synchronized recovery program opens its remote file, the first message sent to it has the common-area header MCS-TYPE field set to a value of 23. The first three bytes of the message text area contain the program number, the next three bytes contain the multi-copy number, and the next twelve bytes contain the date/time stamp. In user application programs using DMS II, this information must be saved to place in the restart data set. See Figure 7-1 for a recommended data set definition, and Figure 7-2 for a remote file record definition.

```
RESTARTAREA RESTART DATA SET (
    GEMCOS-LITERAL                  ALPHA (6);
    GEMCOS-PGM-NBR                  NUMBER (3);
    GEMCOS-MULTI-NBR                NUMBER (3);
    GEMCOS-DATE-TIME                NUMBER (12);
    GEMCOS-DATA                     NUMBER (9))
%
POPULATION = 100;
%
RESTARTSET ORDERED SET OF RESTARTAREA
    KEY IS (
    GEMCOS-LITERAL,
    GEMCOS-DATE-TIME,
    GEMCOS-PGM-NBR,
    GEMCOS-MULTI-NBR);
```

Figure 7-1. Recommended DMS II Restart Data Set Definition

In Figure 7-2 below, N1 is the length of the user portion of the common-area header; if there is no user portion, then MCS-USER-AREA should not be specified. N2 is the length of the user's text. N3 is N2 minus 18.

```
01  GEMCOS-REMOTE-FILE-RECORD.
    05  GEMCOS-COMMONAREA-HEADER.
        10  MCS-LSN                        PIC 9(4).
        10  FILLER                         PIC 9.
        10  MCS-SEQ-NO                     PIC 9(6).
        10  MCS-NDL-TIME                   PIC 9(7).
        10  MCS-TEXT-SIZE                  PIC 9(4).
        10  MCS-TERM-TYPE                  PIC 9(2).
        10  MCS-MSG-ID                     PIC X(6).
        10  MCS-INDEX-1                    PIC 9(2).
        10  MCS-INDEX-2                    PIC 9(2).
        10  MCS-FMT-ERR                    PIC 9(2).
        10  MCS-TYPE                       PIC 9(2).
        10  MCS-INPUT-ADDR                 PIC 9(9).
        10  MCS-RETRY-COUNT                PIC 9.
        10  MCS-RECOVERY-STATUS            PIC 9.
        10  MCS-OUTPUT-ADDR                PIC 9(9).
        10  FILLER                         PIC 9(2).
        10  MCS-USER-AREA                  PIC X(N1).
    05  MCS-TEXT                           PIC X(N2).
    05  MCS-TYPE-23-MESSAGE REDEFINES MCS-TEXT.
        10  GEMCOS-HEADER-PGM-NBR          PIC 9(3).
        10  GEMCOS-HEADER-MULTI-NBR        PIC 9(3).
        10  GEMCOS-HEADER-DATE-TIME        PIC 9(12).
        10  FILLER                         PIC X(N3).
```

Figure 7-2. Recommended Remote File Record Definition

Upon receipt of this message, the following COBOL code should be executed in the DMS II user application program:

```
MODIFY RESTARTSET AT
   GEMCOS-LITERAL   = "GEMCOS"                AND
   GEMCOS-DATE-TIME = GEMCOS-HEADER-DATE-TIME AND
   GEMCOS-PGM-NBR   = GEMCOS-HEADER-PGM-NBR   AND
   GEMCOS-MULTI-NBR = GEMCOS-HEADER-MULTI-NBR
ON EXCEPTION
   CREATE RESTARTAREA
      MOVE "GEMCOS"                TO GEMCOS-LITERAL
      MOVE GEMCOS-HEADER-DATE-TIME TO GEMCOS-DATE-TIME
      MOVE GEMCOS-HEADER-PGM-NBR   TO GEMCOS-PGM-NBR
      MOVE GEMCOS-HEADER-MULTI-NBR TO GEMCOS-MULTI-NBR
      MOVE 0                       TO GEMCOS-DATA.
```

Transaction Processing

A transaction is any message in which the MCS-TYPE is set to the value zero. A DMS II application program must do two things in addition to processing each input transaction. First, it checks the MCS-INPUT-ADDR field of the common-area header. If the field is zero, the program should not do any updates to the data base, as this transaction has not been audited and will, therefore, not be recovered.

Second, if the MCS-INPUT-ADDR field is nonzero, it must be saved in the restart data set. To do this, enter:

```
MOVE MCS-INPUT-ADDR TO GEMCOS-DATA.
```

If MCS-TYPE is set to zero, updates to the restart data set would be permitted.

Output messages are then sent to the MCS according to the following:

1. MCS-TYPE should be set to zero on the last message.

2. MCS-TYPE should be set to one on any intermediate transactions.

A minimum of one output message must be sent to the MCS for each audited input transaction.

End-of-Job

The user application programs should be prepared for the receipt of a message from the MCS with MCS-TYPE set to 24. This message instructs the program to close the data base.

If the data base is closed successfully, the program should send the MCS a response message having MCS-TYPE set to 25. The MCS then sends an end-of-file (EOF) indicator, and the program goes to end-of-job (EOJ). If a program abort is detected when the data base is closed, the program should send the MCS a message with MCS-TYPE set to 20.

If an application program closes its remote file and/or goes to EOJ at any other time, the MCS considers the program to be aborted. Application programs may not delete the restart data set record before proceeding to end-of-job.

Program Abort

When an application program associated with a data base aborts (that is, it closes its remote file before receiving an EOF or is abnormally terminated), the MCS marks that program disabled, and stops accepting messages from stations for any program using that data base. When an operator or the network administrator enables the program (using the CLE Network Control Command) from the Control station or the console, the MCS executes the restart program, after taking any corrective action.

After the restart program sends the MCS the necessary recovery information, the aborted program is re-executed by the MCS. All transactions not reflected on the data base are re-sent to the program to which they were originally sent. The MCS then returns to normal processing.

Recovery Processing

When a user application program detects a DMS II abort, it should send the MCS a message with MCS-TYPE set to 20. The program should then ignore all messages from the MCS until it receives a message with MCS-TYPE set to 21. When a user application program receives a message with MCS-TYPE set to 21 (whether or not it has sent a type-20 message), it should execute the following coding sequence:

```
BEGIN-TRANSACTION NO-AUDIT RESTARTAREA
   ON EXCEPTION
      <exception handling code> .
END-TRANSACTION NO-AUDIT RESTARTAREA
   ON EXCEPTION
      <exception handling code> .
```

This code sequence clears the DMS II abort flag prior to the initiation of recovery. An ABORT exception at BEGIN TRANSACTION or END TRANSACTION can be ignored; processing does continue. Finally, the program should send a message to the MCS with MCS-TYPE set to 22. The MCS then sends the program the transactions that were rolled back from the data base. When recovery is finished, the MCS begins sending normal transactions as they are received. The application program can determine when recovery is complete by checking the MCS-RECOVERY-STATUS field in the common-area header. A value of zero indicates that this message is a normal message.

Recovery After System Failure

When an MCS terminates abnormally (i.e., a DS or DP condition arises or a HLT KILL is performed), or the MCP fails and a Clear/Start is necessary, the MCS, when re-executed, executes the restart program and checks the audit file to determine if any transactions are not reflected in the data base. If there are transactions that are not reflected in the data base, the trancode's program or programs are automatically executed by the MCS. All unprocessed transactions are sent to the corresponding programs. Normal processing begins when all recovered messages are sent.

Data Base Recovery (Nonsynchronized)

Data base recovery performs a queue restoration on all application programs in the data base needing recovery. Data base recovery does not attempt to duplicate the order of updates to the data base, or to perform any output analysis. The only difference between data base recovery and queue restoration (after the initialization) is that data base recovery does not allow messages to be entered from stations for any program using the data base until all programs using the data base have finished recovery.

Synchronized Recovery

DMS II rolls back a data base in time to remove logical data base updates that were partially completed at the time of a failure. (A single logical update transaction normally results in multiple data base updates.) During recovery, the MCS performs an audit trail analysis of the state of the rolled-back data base, so that input transactions which were removed from the data base are "recycled" for application processing.

Input transactions which have been successfully audited by the MCS prior to failure, but have not yet reached the point where they are passed to an application program for processing are queued again. "Recycled" transactions can be reprocessed so that the identical update sequence to the data base is maintained.

The MCS does not just send to each program its particular transactions in the same sequence that the program received them prior to failure. Rather, it attacks the more complex problem of re-establishing the exact sequence of events that occurred on the data base. This recovery technique takes into consideration the fact that some variable number of independent program executions could be receiving transactions concurrently, and thus, each transaction could result in multiple concurrent updates to the data base.

This feature becomes extremely important when multiple transactions which result in access to common data are asynchronously flowing through the system at the same time.

With synchronized recovery, application programmers and terminal operators do not need to know about failures. Application programmers and terminal operators can remain unaware of failures, because data bases can be maintained simultaneously which are in total agreement with any messages delivered to the network prior to the time of failure.

Further, to eliminate duplicate transmissions, the MCS analyzes the input transactions which were recycled because of a data base rollback of application-generated output.

To do this, the MCS directs the Network Controller to return a "good results" reply when a primary output message is received at the station.

The MCS ensures that a successful, totally synchronized recovery has occurred. Yet, under certain circumstances, the MCS may lose output messages or send duplicate output messages to stations. Extraneous updates to the data base, the most critical cases, do not occur.

If output messages are lost, the MCS notifies the terminal operator of the extent of that loss. If duplicate output messages are sent, the MCS informs the terminal operator that a list of possible duplicate messages follows. After sending these messages, the MCS informs the terminal operator that the possible duplicate messages are finished. In both cases, the number of messages lost or duplicated is usually quite small.

Recovery-Related Conventions

The next paragrapns explain the conventions which the user needs to follow to ensure a synchronized recovery.

1. A user program must claim transaction-related resources prior to updating any of them. Thus, all necessary data management records should be locked before entering the transaction state. In many situations, this might entail merely locking a particular node within a data hierarchy.

2. A user program must not unlock any transaction resources until the transaction is complete. Thus, the DMS II END-TRANSACTION can be allowed to unlock data management records.

3. A user program must send messages to stations according to the following protocol:

    a. "Secondary outputs" are sent first. These are outputs which are passed to the MCS with the MCS-TYPE field of the common-area header set to 1. The value 1 signifies to the MCS that the user program is not done with the current transaction and wishes to send more output.

    b. The last output generated during normal processing by a user program as a result of receiving a given input transaction is termed the "primary output." The primary output of a transaction is a message sent by the user program with a value of 0 in the MCS-TYPE field of the common-area header. The value 0 signifies to the MCS that the user program is finished with the current transaction. The MCS assigns the Data Base Sequence Number (DBSN) at this time.

The primary output can be delivered to a station other than the originating station, or it can have a text length of zero. In the latter case, no output message is delivered to the station and the MCS audits an output message of zero length.

NOTE

A zero-length message takes a station out of forms mode.

4. A user program must cause the DMSII restart information to be forwarded to the DMS II audit trail when the program leaves the transaction state. The user program should not cause the DMSII restart information to be forwarded when the program is entering the transaction state.

The proper sequence of events for a user program which has synchronized recovery is outlined in the following basic logic flow example.

Example:

```
INITIALIZE.
    OPEN DATABASE.
    READ MCSQUEUE AT END  STOP RUN.
    IF MCS-TYPE NOT = 23  STOP RUN.
    MODIFY RESTARTSET AT
        GEMCOS-LITERAL   = "GEMCOS"                 AND
        GEMCOS-DATE-TIME = GEMCOS-HEADER-DATE-TIME AND
        GEMCOS-PGM-NBR   = GEMCOS-HEADER-PGM-NBR   AND
        GEMCOS-MULTI-NBR = GEMCOS-HEADER-MULTI-NBR
    ON EXCEPTION
        CREATE RESTARTAREA
          MOVE "GEMCOS"                TO GEMCOS-LITERAL
          MOVE GEMCOS-HEADER-DATE-TIME TO GEMCOS-DATE-TIME
          MOVE GEMCOS-HEADER-PGM-NBR   TO GEMCOS-PGM-NBR
          MOVE GEMCOS-HEADER-MULTI-NBR TO GEMCOS-MULTI-NBR
          MOVE 0                       TO GEMCOS-DATA.
    PERFORM PROCESS THRU PROCESS-EXIT UNTIL EOF = 1.
    STOP RUN.


PROCESS.
    READ MCSQUEUE AT END  MOVE 1 TO EOF
                          GO TO PROCESS-EXIT.
    IF MCS-TYPE = 24
        <SEND "TYPE 25" MESSAGE>
        GO TO PROCESS-EXIT.
```

```
    IF MCS-TYPE = 21
        <PROCESS & SEND "TYPE 22" MESSAGE>
        GO TO PROCESS-EXIT.
    <LOCK DATA BASE RECORDS TO BE UPDATED>
    BEGIN-TRANSACTION NO-AUDIT RESTARTAREA.
        <MOVE RESTART INFORMATION TO RESTART AREA>
                *
                *
                *
        <UPDATE DATA BASE>
        <SEND SECONDARY OUTPUTS>
    END-TRANSACTION AUDIT RESTARTAREA.
    <SEND PRIMARY OUTPUT>
PROCESS-EXIT.
    EXIT.
```

HOUSEKEEPING CONSIDERATIONS

In a production environment, it is necessary to do general "housekeeping" at the program, data base, and MCS levels. Programs need to be recompiled, data base and MCS audit files need to be backed up to tape, and most importantly, all of these system modules need to be kept in strict synchronization.

In order to alleviate the confusion of maintaining and preserving the integrity of an on-line GEMCOS, the following "housekeeping" tips are suggested.

1. Do not delete the restart data set record from the restart data set when an application program goes to EOJ.

2. To reset the GEMCOS auditing mechanism, and thus remove all the MCSAUDIT files from disk, the following steps should be taken in the order specified:

    a. Delete all records in the restart data set.

    b. Back up the data base to some off-line medium.

    c. Remove the MCSAUDIT audit files from disk.

    d. Do a REGENERATE of the TCL deck in order to create a new MCSTIC and formats file.

3. It is not recommended that the user copy a saved initial MCSTIC file back to disk. This can cause unpredictable results during normal processing. When it is inconvenient to regenerate the TCL specifications due to their length, an

initial MCSTIC file can be recopied back to disk only if all records are removed from the restart data set concurrently.

RESTART PROGRAM

Every system of programs (data base) that has synchronized recovery must have a restart program defined. Like other TCL programs, the restart program is defined in the Program section of the TCL. The restart program uses the following special syntax:

```
RESTARTPROGRAM = TRUE.
```

Only one restart program may be defined for a system.

When DMS II rolls back a data base, the restart program retrieves all the data base restart data of application programs that will be involved in the recovery mechanism, sorts this information, and passes it to the MCS. The restart program then goes to EOJ. The MCS also executes the restart program if the previous run of the MCS terminated abnormally.

GEMCOS supplies a documented COBOL source file of a sample restart program which is to be used with DMS II. This sample demonstrates the proper passes of control from the restart program to the MCS, as well as the logic flow within the program.

The user can patch this source to replace the naming conventions of the restart data set used by GEMCOS with the user's data names. This sample source also contains information on the values of the common-area header MCS-TYPE field. The only changes the user needs to make to this are to change the data base name and, possibly, the remote file name. Otherwise, this source can be compiled and integrated into the system.

RECOVERY CYCLE

The GEMCOS recovery cycle can be initiated in any of the following ways:

1. By the REC Network Control Command.
2. When a user program passes the abort indicator back to the MCS (a message with MCS-TYPE set to 20).
3. When the MCS is restarted after it has abnormally terminated, or after the operating system has abnormally terminated.
4. When an application program using recovery abnormally terminates (with the CLE Network Command).

In the first three cases, recovery is initiated automatically when the situation is recognized by the MCS. In the fourth case, recovery is indicated after the aborted program is cleared by the operator.

Any transaction which is recycled to a User Program during recovery is flagged with MCS-RECOVERY-STATUS of the common-area header set to one of the following values:

Value	Explanation
0	The system is not in recovery mode.
1	The system is in recovery mode because an application program aborted.
2	The system is performing an archival recovery.
3	The system is in recovery mode because of a Clear/Start or an abnormal termination of the MCS.

In addition, a transaction which causes a user program to abnormally terminate is resubmitted with the common-area header MCS-RETRY-COUNT field incremented by one. The MCS always resubmits an aborted transaction. The user program must decide how to process the transaction.

Figure 7-3 shows the sequence of events at a Control station during recovery after a user program has abnormally terminated. Messages prefixed by (u) are messages that were entered at the station by the user. Messages prefixed by (s) are messages that were sent by the MCS to the station. Any other station that attempted to input transactions during recovery (and was ignored) is informed that recovery is finished. A station that is idle during recovery is not informed that recovery is done.

```
(u) TRN1 <data>

(s) **ERROR 435 126 : UNEXPECTED CLOSE FROM PGM - <nnn>

(s) ?? 127 TO INITIATE RECOVERY, CLEAR PGM - <nnn>

(u) *CLE <nnn>

(s) $OK$

(s) ?? 108 : BEGIN RECOVERY OF DATABASE - <mmm>

(s) ?? 109 : END RECOVERY OF DATABASE - <mmm>
```

Figure 7-3. User Program Abnormal Termination Recovery Cycle

ARCHIVAL RECOVERY

If DMSII reconstruction fails, or if a program bug corrupts the data base, the user may need to reconstruct the sequence of events which the data communications network performed upon the data base. This sequence of events may have occurred over a period of many days.

Archival recovery uses the MCS-created audit files from the original processing of transactions. Before starting archival recovery, the operator must ensure that the data base on disk reflects a known "good" state. If the audit files the MCS needs are not present, the MCS prompts the operator to load the files it needs.

Load the data base and audit files which correspond to the "good" state. If the current MCSTIC file does not correspond to this state (if, for example, a REGENERATE was done later), then also copy in the MCSTIC file.

Use the current MCSTIC file for archival recovery. This file allows recovery back to the last GENERATE or REGENERATE.

If recovery prior to the last GENERATE or REGENERATE is required, then the following files are needed:

1. The previous MCSTIC file.
2. The previous audit files.
3. The previous restart data.

Therefore, always back these files up when doing a REGENERATE.

For both 5.01 and 6.0 GEMCOS, always start archival recovery at the ODT. Make the stations AVAILABLE.

To begin archival recovery, execute the MCS with Switch 1 set to 1 (SW1 = 1). After initialization, the MCS displays the following messages on the display console:

1. ?? 110 : GEMCOS ARCHIVAL RECOVERY
2. ?? 111 : ENTER ARCHIVAL SPECIFICATIONS

At the ODT, supply this information:

1. The name of the data base to be recovered.
2. A list of all programs in the data base that are to be recovered, or, if all programs are to be recovered, the word "ALL".

Supply this information in free format, using the following syntax:

```
<archival recovery request> ::= RECOVER DATABASE
                                <data base specifications>

<data base specifications>  ::= <data base name>
                                <program list>

<program list>              ::= <program name> /
                                <program name> , <program
                                list> / ALL
```

The archival recovery mechanism recovers only one data base per archival run. If more than one data base needs to be recovered, separate archival runs are needed.

GEMCOS does not recover batch programs. The user can only run batch programs after the following steps have been done:

1. GEMCOS has been halted.
2. The audit files have been backed-up.
3. The restart data set has been backed up.

It is possible, however, for a batch program to pass transactions to an on-line program, which then sends the transactions to GEMCOS.

The user can also convert a batch program to an on-line program which sends all of the transactions in one BEGIN-END TRANSACTION statement. In this case, every message but the last one is a secondary output. The last message is the primary output.

Recovery begins when GEMCOS reads the restart data set and ends when the last transaction is processed.

During archival recovery, no messages generated by application programs are released to the stations, and no input is permitted from any station. (Note that NCC commands can be entered through an ODT.)

When archival recovery is finished, the MCS goes to end-of-job. All messages from the MCS during archival recovery are sent to the display console. If one or more programs are disabled at the time of archival recovery, they are automatically enabled before archival recovery begins.

Figure 7-4 shows a typical sequence of events and the console traffic for an MCS archival recovery run. The operator generates messages prefixed by a (u). Either the MCS or the MCP generates messages prefixed by an (s). When archival recovery is complete, the data base has been recovered. The MCS can then be re-executed normally to restart on-line processing.

```
(u) EX GEMCOS/MCS; SW1 = 1

(s) GEMCOS/MCS = 1234 BOJ.

(s) % GEMCOS/MCS : ?? 110 : GEMCOS ARCHIVAL RECOVERY

(s) % GEMCOS/MCS : ?? 111 : ENTER ARCHIVAL SPECIFICATIONS

(u) 1234AX RECOVER DATABASE TESTDB ALL

(s) % GEMCOS/MCS : ?? 115 : BEGIN ARCHIVAL RECOVERY

(s) % GEMCOS/MCS : ?? 108 : BEGIN RECOVERY OF DATABASE -
    <nnn>

(s) % GEMCOS/MCS : ?? 109 : END RECOVERY OF DATABASE - <nnn>

(s) % GEMCOS/MCS : ?? 116 : END ARCHIVAL RECOVERY

(s) GEMCOS/MCS = 1234 EOJ.
```

Figure 7-4. MCS - Archival Recovery Cycle

RECOVERY CONTROL MESSAGES

Many different types of MCS-TYPE messages are passed to and from the MCS during the recovery cycle. Use the following table for quick reference to the Recovery Control Messages.

Table 7-1

Recovery Control Messages

MCS-TYPE Value	Written By	Definition and Usage
15	MCS	Sent by MCS to restart program requesting restart data be retrieved.
17	Restart program	Sent by restart program to MCS. Contains restart data information for all programs in the data base.
18	Restart program	Sent by restart program to MCS whenever a fatal error is encountered.
20	User program	Set to MCS by application program whenever program encounters a DMS II program abort.
21	MCS	Set to application program by MCS informing program recovery is about to be initiated.
22	User program	Sent to MCS by application program in response to type-21 message.
23	MCS	Sent to application program by MCS at FILE OPEN. Contains pertinent restart data information.
24	MCS	Sent to application program by MCS instructing program to close the data base.
25	User program	Sent to MCS by application program in response to type-24 message.

GEMCOS provides effective recovery procedures, while maintaining data base and network integrity. Because GEMCOS has several options for recovery, this procedure is flexible and efficient. In addition, the user can analyze the requirements of application programs and then select the recovery options best suited to the needs of those programs.

APPENDIX A

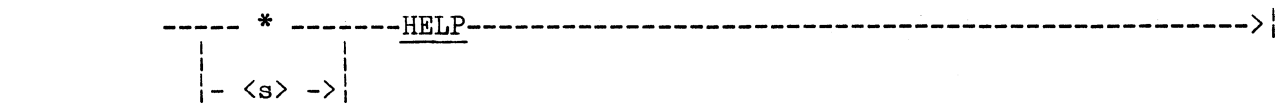
SUMMARY OF NETWORK CONTROL COMMANDS

This appendix summarizes the Network Control Commands. Refer to Section 3 for a complete explanation of these commands. See Appendix G for an explanation of how to read the syntax diagrams.

THE HELP COMMAND

HELP

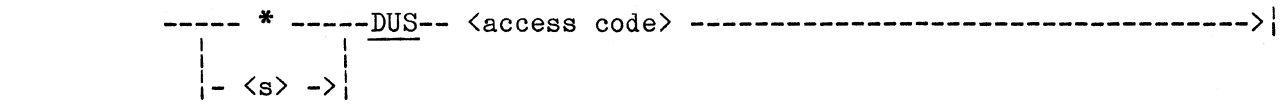
Syntax:



SECURITY CONTROL COMMANDS

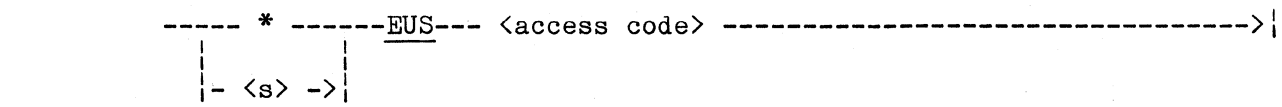
DISABLE USER (DUS)

Syntax:



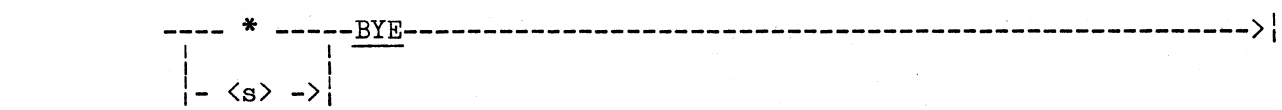
ENABLE USER (EUS)

Syntax:



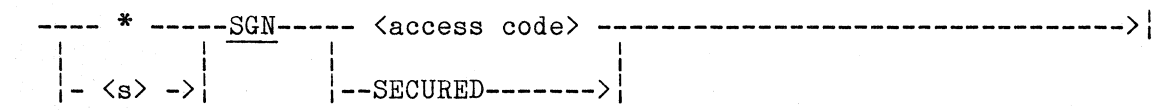
SIGN OFF (BYE)

Syntax:



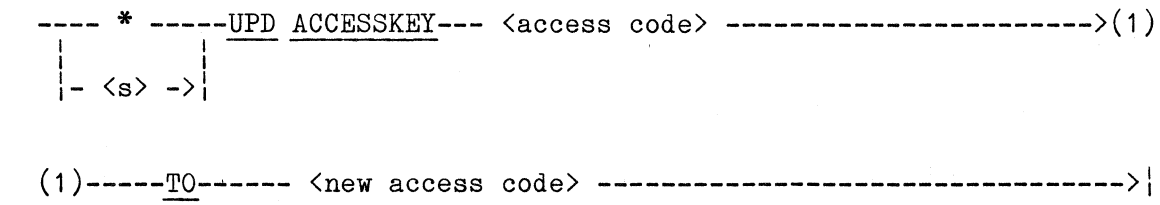
SIGN ON (SGN)

Syntax:



UPDATE ACCESS KEYS (UPD ACCESSKEY)

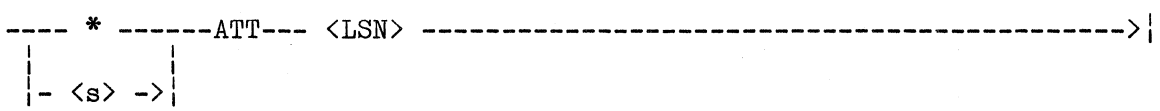
Syntax:



STATION ATTACHMENT COMMANDS

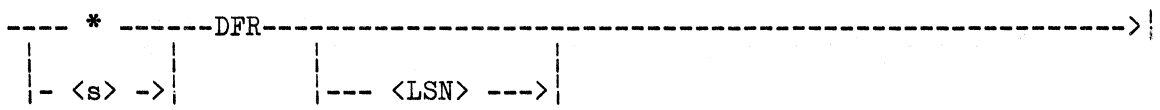
ATTACH LSN (ATT)

Syntax:



DETACH FROM REMOTE FILE (DFR)

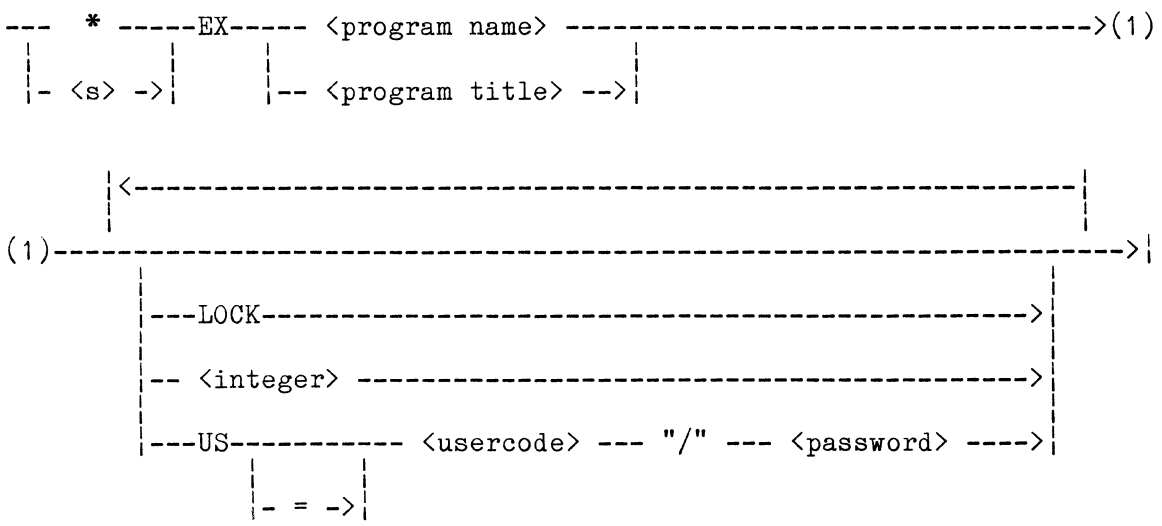
Syntax:



PROGRAM CONTROL COMMANDS

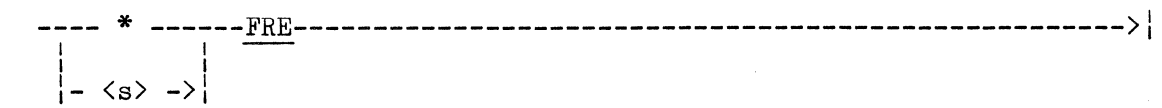
EXECUTE PROGRAM (EX)

Syntax:



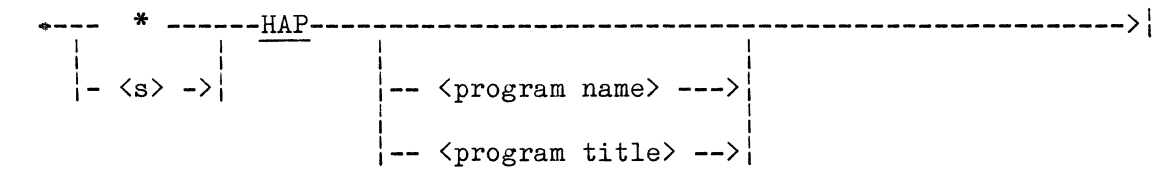
FREE STATION FOR EXECUTION (FRE)

Syntax:



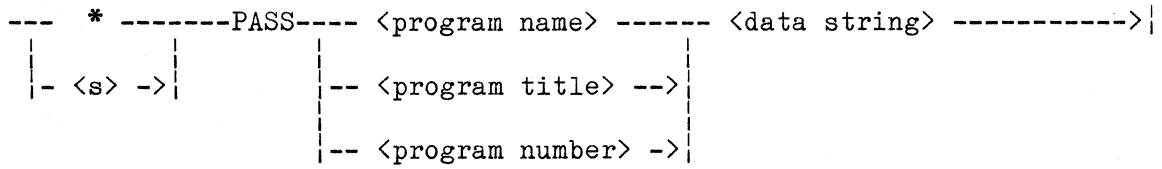
HALT APPLICATION PROGRAM (HAP)

Syntax:



PROGRAM PASS (PASS)

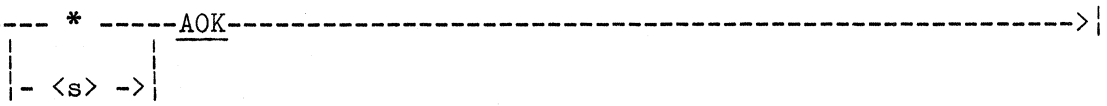
Syntax:



MCS CONTROL COMMANDS

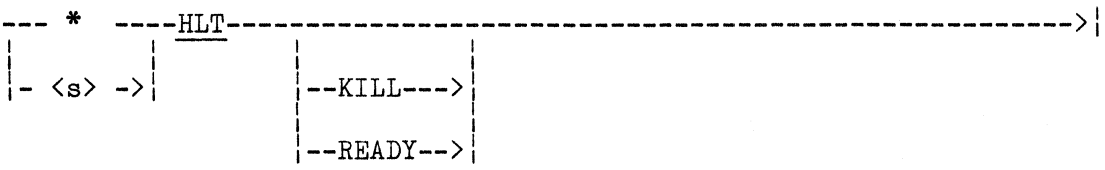
AUDIT OK (AOK)

Syntax:



HALT SYSTEM (HLT)

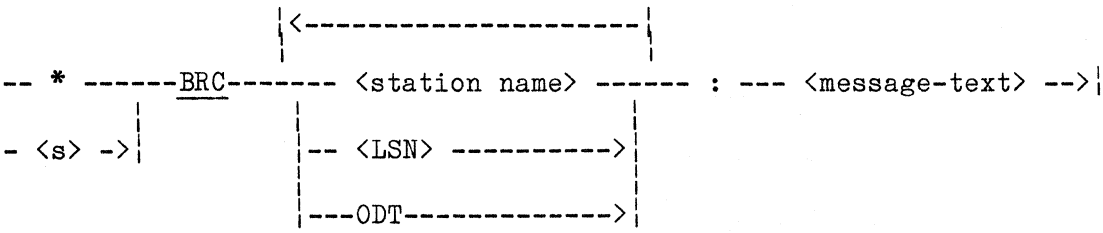
Syntax:



MESSAGE CONTROL COMMANDS

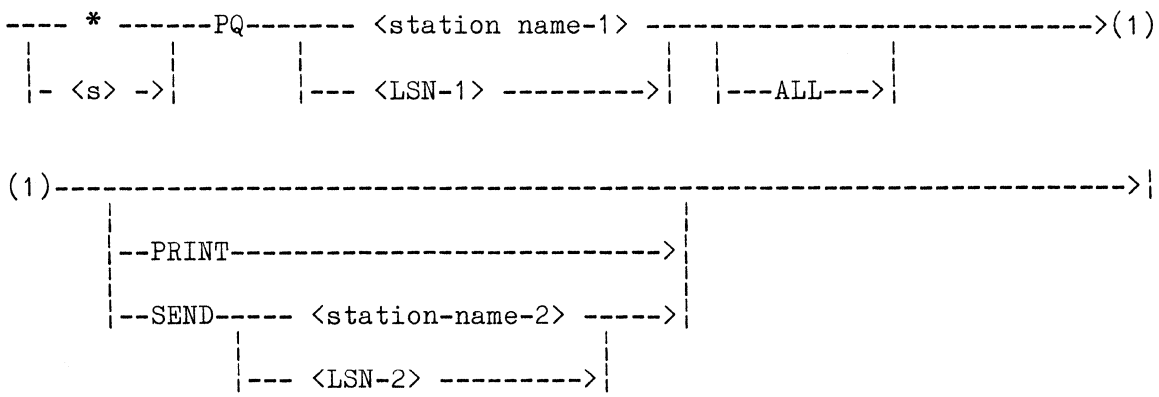
BROADCAST (BRC)

Syntax:



POP QUEUE (PQ)

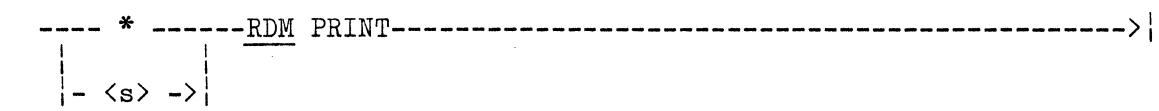
Syntax:



REPORT COMMANDS

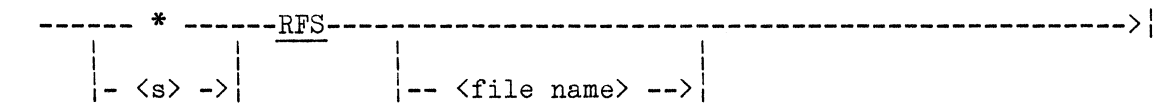
REPORT DATA DUMP (RDM)

Syntax:



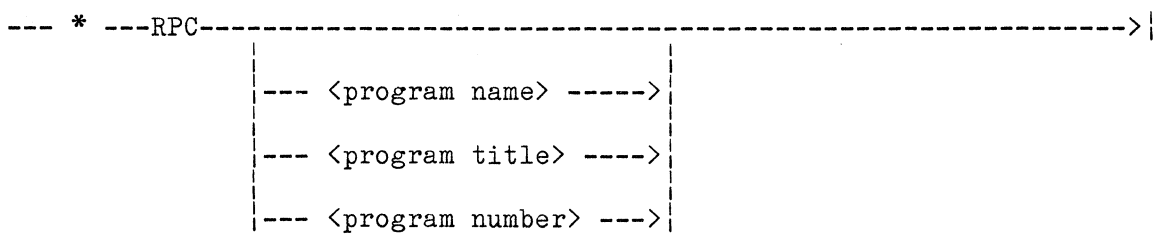
REPORT FILE STATUS (RFS)

Syntax:



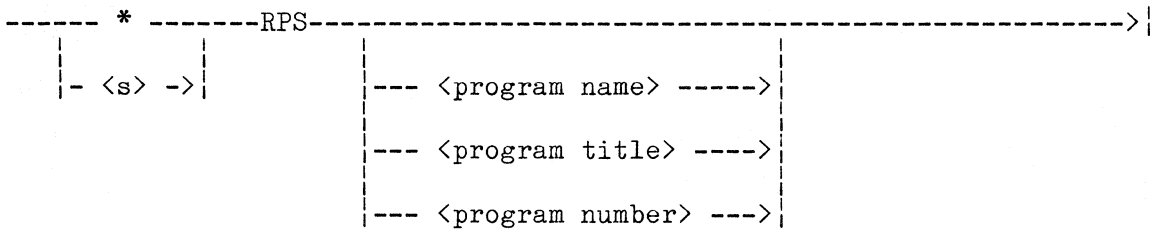
REPORT PROGRAM COUNTERS (RPC)

Syntax:



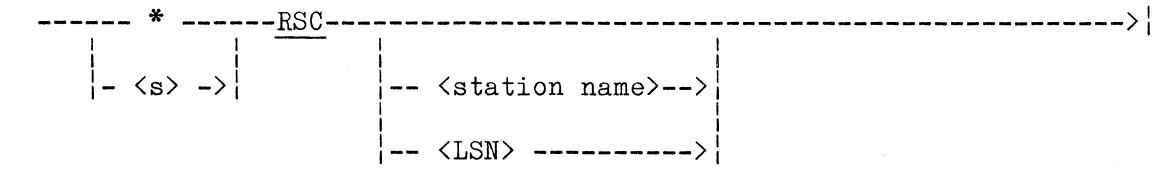
REPORT PROGRAM STATUS (RPS)

Syntax:



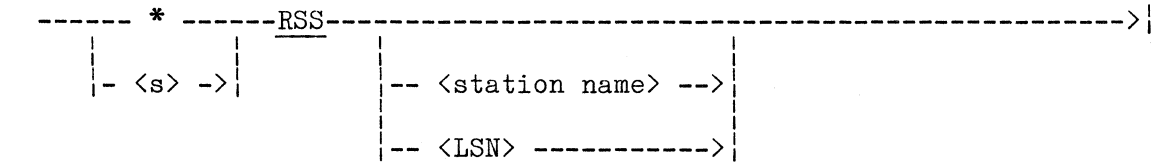
REPORT STATION COUNTERS (RSC)

Syntax:



REPORT STATION STATUS (RSS)

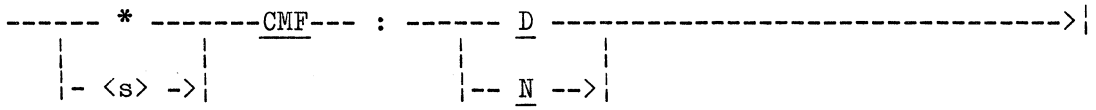
Syntax:



CHANGE COMMANDS

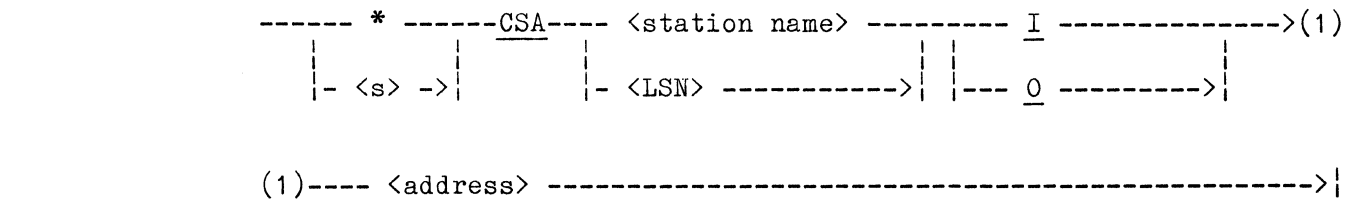
CHANGE MONITOR FLAG (CMF)

Syntax:



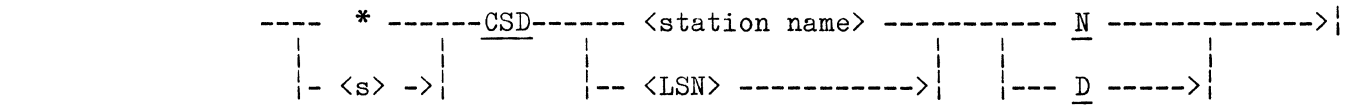
CHANGE STATION ADDRESS (CSA)

Syntax:



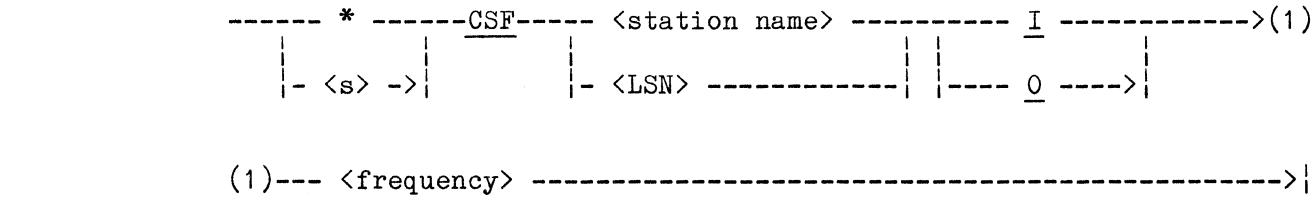
CHANGE STATION DIAGNOSTIC (CSD)

Syntax:



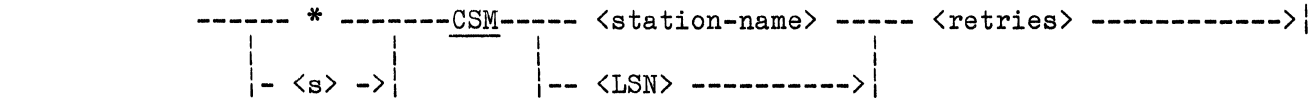
CHANGE STATION FREQUENCY (CSF)

Syntax:



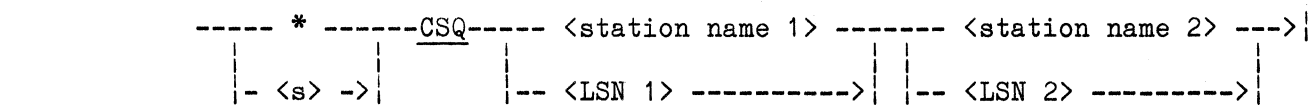
CHANGE STATION MAXIMUM RETRY (CSM)

Syntax:



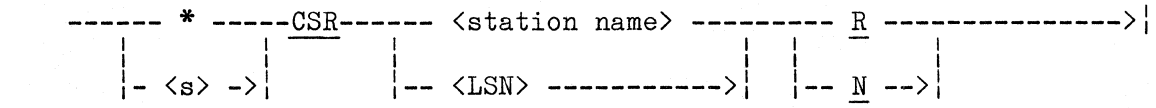
CHANGE STATION QUEUE (CSQ)

Syntax:



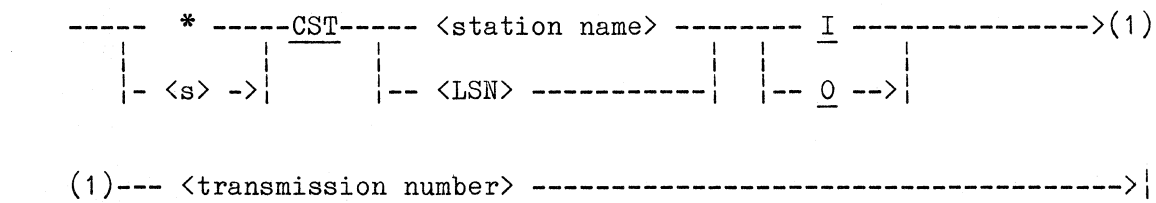
CHANGE STATION READY (CSR)

Syntax:



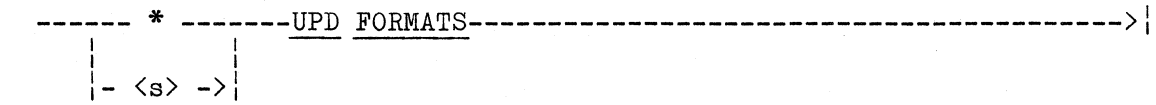
CHANGE STATION TRANSMISSION NUMBER (CST)

Syntax:



FORMAT UPDATE (UPD)

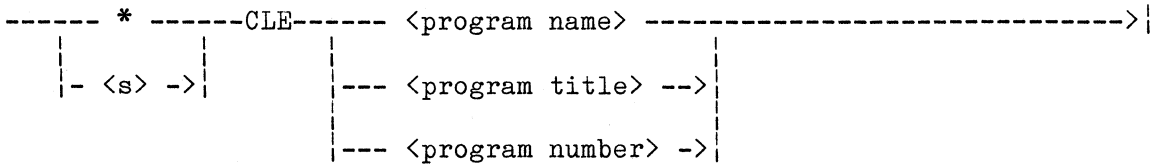
Syntax:



AUDIT & RECOVERY COMMANDS

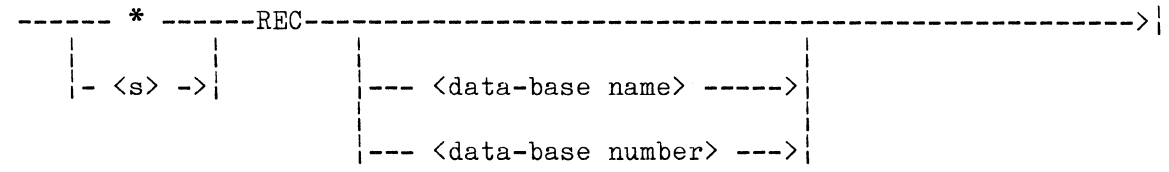
CLEAR DISABLED PROGRAM (CLE)

Syntax:



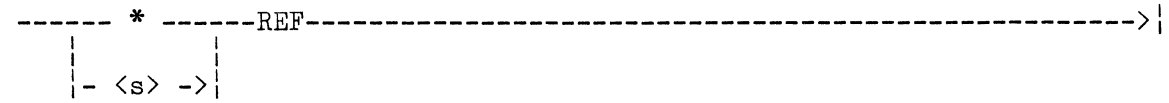
RECOVER DATA BASE (REC)

Syntax:



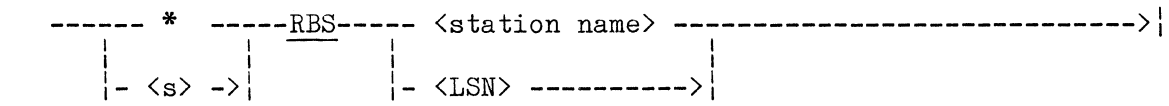
REFRESH COMMAND (REF)

Syntax:



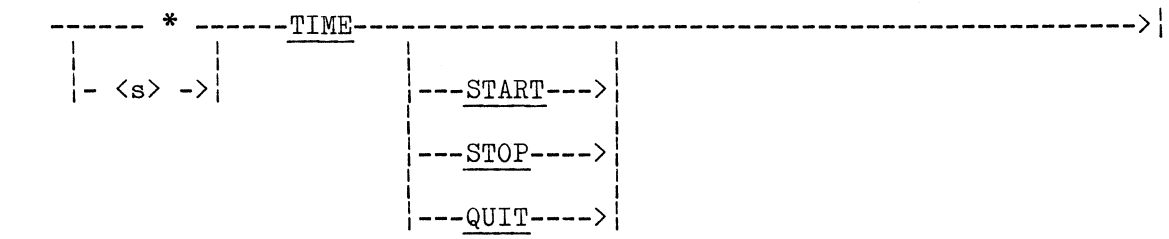
RESET BUSY STATUS (RBS)

Syntax:



TIME

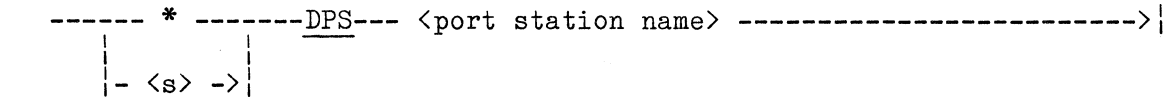
Syntax:



PORT STATION COMMANDS

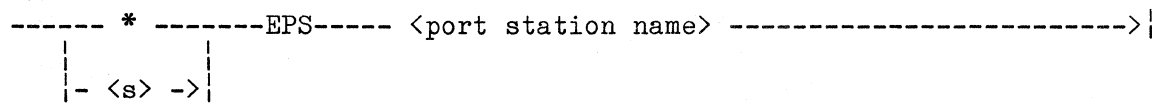
DISABLE PORT STATION (DPS)

Syntax:



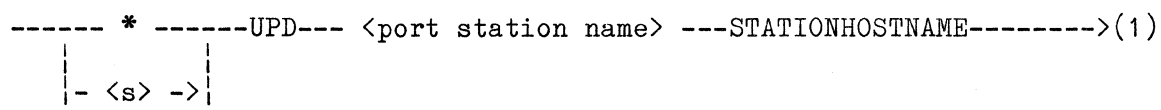
ENABLE PORT STATION (EPS)

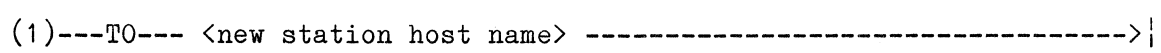
Syntax:



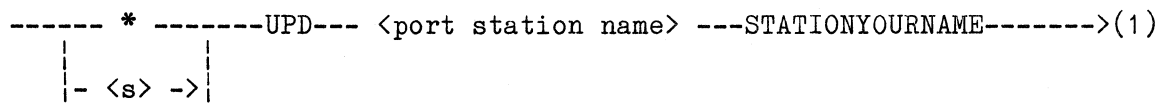
UPDATE STATION HOST NAME/STATION YOUR NAME

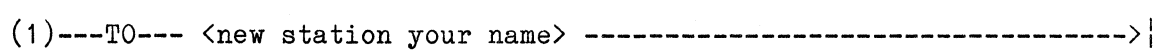
Syntax for UPD - HOSTNAME Command:





Syntax for UPD - YOURNAME Command:





APPENDIX B

SUMMARY OF FILES

The following table summarizes GEMCOS system files.

Program	Internal Name	Input (I) Output (O)	Device	External Name
MCSSRC/OBJECT	MCSTIC	I/O	Disk	
	MCSFORMATS	I/O	Disk	
	MCSQUEUE	I/O	Remote	
	MCSBCKLG	I/O	Queue	
	MCSAUDIT	I/O	Disk	MCSAUDIT/ AUDITnnn
	MCSOLDAUDIT	I/O	Disk	MCSAUDIT/ AUDITnnn
	MCSPRINT	O	Printer	
	MCSTANK	I/O	Disk	MCSTANK/ <random number>
	MCSQFULL	I/O	DISK	MCSQFULL/ <random number>
	MCSTIME	O	DISK	MCSTIME/ <random number>
	MCSZIPPGM	I/O	DISK	
MCSSIM	PRINT.OUT	O	Printer	MCSSIMPRT
	CARDS	I	Card	MCSSIMCRD
	MCSQUEUE	O	Queue	
MCSFIX	SOURCE	I	Disk	
	NEWSOURCE	O	Disk	
	CARDS	I	Card	
	LINE	O	Printer	
MCSGO		I	Disk	MCSGTS
		I	Disk	MCSFTS
		I	Card	MCSCRD
		O	Disk	MCSSRC
		O	Disk	MCSFIT
		I/O	Disk	MCSTMP
		O	Printer	MCSERN

Program	Internal Name	Input (I) Output (O)	Device	External Name
MCSTCL	MCSIN	I	Card	MCSIN
	MCSLST	O	Printer	MCSLST
	MCSPRM	O	Disk	MCSTMP
	MCSTIC	O	Disk	MCSTIC
	MCSFORMATS	I/O	Disk	MCSFORMATS
	MCSERROR	I	DISK	GEMCOS/MCSERROR
	MCSTMP	I/O	Disk	MCSTMP1
	OLDMCSTIC	I	Disk	MCSTIC
	MCSRPT	O	Printer	MCSRPT
	DIRECTORY	I/O	Disk work	MCSDIRECTY
MCSRECALL	MCSTIC	I	Disk	MCSTIC
	MCSREM	I/O	Remote	MCSREM
	MCSAUDIT	I	Disk	MCSAUDIT/ AUDITnnn
	MCSPRT	O	Printer	MCSPRT
MCSFILXFER	MCSREMOTE	I/O	Remote	MCSREMOTE
	MCSDISK	I/O	Disk	

APPENDIX C

LIMITS OF TCL SIZE

The TCL compiler has several size limitations. They restrict the maximum number of various entities such as stations or programs which can be declared in the TCL. These size limitations follow:

The <MAXCOPIES statement> must be less than	256
The number of programs must be less than	1000
The number of trancodes must be less than	1000
The number of message-IDs must be less than	1024
The number of stations must be less than	1000
The number of devices must be less than	1024
The number of functions must be less than	1024
The number of formats must be less than	1024
The number of users must be less than	1000
The number of routeheader stations must be less than.	256
The number of trancodes + the number of stations + the number of programs must be less than	1367
The number of devices * the number of trancodes * 10 must be less than	65535
The number of message-IDs * number of devices * 10 must be less than	65535

APPENDIX D

MCS ERROR HANDLING AND ERROR MESSAGES

This section discusses error handling by the MCS and the format of error messages. The end of this section lists MCS error messages.

ERROR HANDLING BY THE MCS

The GEMCOS error handling subsystem provides the logic needed to handle error conditions not directly related to applications tasks. GEMCOS takes automatic action to keep the system running and communicates the error condition to an operator.

GEMCOS distinguishes three categories of errors:

1. Errors made by a station operator.
2. Persistent data communications errors.
3. System errors.

When GEMCOS detects an error made by a system operator, it sends a message to the operator. If it detects other kinds of errors, GEMCOS sends messages to the Control station or to the console printer, if the Control station is not available. Operators can then use Network Control Commands to diagnose or circumvent the problem.

The Network Controller handles transient data communications errors, but persistent errors are reported to the MCS. The MCS then informs the Control stations.

The MCS also keeps statistics on errors. The station accumulates the statistics, which the operator can retrieve using Network Control Commands.

System errors have several causes. They can result from input/output errors on the peripheral devices the MCS uses, or they can be caused by software problems. The software problems may be in the MCS, the Network Controller, or application programs.

When it detects a system error, the MCS reports the error to the Control station. For serious errors, the MCS also produces a dump of its tables for debugging. The MCS is designed to continue running unless a Control station or the system operator discontinues its operation.

FORMAT OF ERRORS

This appendix describes all standard network messages generated by the MCS to inform users of errors or other conditions. The general format of a system or data communications error message is:

HH:MM ** ERROR NNN : message

The general format of an operator message is:

HH:MM #NNN : message

The number NNN is the message number.

If DATADUMP = TRUE is specified in the TCL, all error messages numbered less than or equal to 30 are accompanied by a table dump. For other messages, a dump can be obtained by using the RDM Network Control Command.

ERROR MESSAGES

0 INVALID HALT PHASE

During system shutdown, the current phase of shutdown is different from the one expected.

SUGGESTED ACTION:

Orderly shutdown may not be possible, and the MCS may have to be discontinued.

1 INVALID STATUS REQUEST

During initialization, the MCS was sending status requests to the Network Controller and the Network Controller responded with an error message stating that some request was not valid. As a

result, the corresponding entry in the MCS tables is not initialized.

SUGGESTED ACTION:

Discontinue the MCS, and execute it again.

2 HARDWARE EXCEPTION ON MCSTIC

While reading from or writing to MCSTIC, the MCS detected a physical disk error (such as a parity error). The MCS suspends itself, awaiting keyboard input from the system operator.

SUGGESTED ACTION:

If the operator responds to the ACCEPT, the MCS attempts to shut down either gracefully (if SYSTEMHALT = TRUE was specified in the TCL) or directly. Otherwise the user must DS or DP the MCS.

3 EOF ON MCSTIC READ OR WRITE

The MCS attempted to read from or write to MCSTIC using an invalid record number. The MCS suspends itself, awaiting a response from the operator to an ACCEPT. If the operator responds, the MCS attempts to terminate normally.

SUGGESTED ACTION:

Analyze the dump for a possible software error. Check the NPR record in MCSTIC for invalid record pointers. A new MCSTIC may have to be created.

4 INVALID RECORD ID IN MCSTIC

The MCS read a MCSTIC record, but the ID field at the beginning of the record was not the one expected. The MCS suspends itself, awaiting a response from the operator to an ACCEPT. If the operator responds, the MCS attempts to terminate normally.

SUGGESTED ACTION:

Analyze the dump for a possible software error. Check the NPR record in MCSTIC for invalid record pointers. A new MCSTIC may have to be created.

6 INVALID MESSAGE SOURCE ON SEND

When attempting to send a response to a Network Control Command, the MCS discovered that the LSN in the header was invalid.

SUGGESTED ACTION:

The message cannot be sent out and is dropped. It can be recovered from the data dump.

8 CONVERSATION MISMATCH IN AUDIT FILE

The MCS attempts to perform a recovery. The conversation information in the MCSTIC file does not match the audit records.

SUGGESTED ACTION:

Review the audit files and rerun the MCS program. If the error recurs, execute a data dump and monitor trace.

10 INVALID MESSAGE VARIANT

The MCS has received a message whose variant field is not zero from the Network Controller.

SUGGESTED ACTION:

The message is discarded, but can be recovered from the dump. The Network Controller is suspect.

11 INVALID FILE OPEN - STATION MYUSE

An application program attempted to open a remote file containing a station whose MYUSE is not consistent with the open type.

SUGGESTED ACTION:

The application program must be discontinued. Either its OPEN must be modified or the MYUSE of the station must be changed in the NDL.

13 PGM NEEDS RECOVERY - STN IS ATTACHED

A utility or assignment program needs to be recovered. GEMCOS cannot reattach one or more stations to the program that were attached at the time of failure because these stations are currently attached to another program.

SUGGESTED ACTION:

This error indicates a situation that GEMCOS cannot resolve. This error only occurs when GEMCOS is running subordinate to SMCS and cannot reattach all of its former stations after a failure. Detach all GEMCOS stations from any programs they may be attached to and then re-execute the MCS.

14 PGM NEEDS RECOVERY - NO STNS ATTACHED

A utility or assignment program needs to be recovered, but GEMCOS cannot reattach to the program any stations that were attached previously at the time of failure.

SUGGESTED ACTION:

This error indicates a situation that GEMCOS cannot resolve. This error only occurs when GEMCOS is running subordinate to SMCS and cannot reattach all of its former stations after a failure. Detach all GEMCOS stations from any programs they may be attached to and then re-execute the MCS.

17 RECOVERY OPENED CURRENT AUDIT FILE

During recovery, the MCS attempted to open the current audit file for message reprocessing.

SUGGESTED ACTION:

Rerun the MCS. If the problem persists, obtain a data dump and monitor trace.

18 BAD AUDIT RECORD DURING RECOVERY

During recovery, the MCS attempted to read or write an audit file with an invalid key.

SUGGESTED ACTION:

Rerun the MCS. If the problem persists, obtain a data dump and monitor trace.

19 FATAL ERROR FROM RESTART PGM - PROGRAM

This error immediately follows the error explaining the nature of the problem.

SUGGESTED ACTION:

Check the logic of the restart program as well as any application programs involved.

20 BAD FILE # IN DETACH MSG

The MCS received a detach message from the Network Controller, but was unable to associate the remote file number with any known program.

SUGGESTED ACTION:

Check the MCS and Network Controller interface logic and analyze the data dump (if present) to obtain the program name.

21 BAD PGM # IN DETACH MSG

The MCS received a detach message from the Network Controller for a program that was not running.

SUGGESTED ACTION:

Check the MCS and Network Controller interface and corresponding data dump.

22 EOF / EXCEPTION ON FORMAT FILE READ

While reading from the format file, the MCS detected either an end-of-file condition or a hardware exception. The MCS terminates when this happens.

SUGGESTED ACTION:

Submit the system dump produced by the GEMCOS MCS to the local Burroughs representative.

23 NO STNS ATTACHED, CANNOT CONTINUE

The MCS received a file-open request from the network controller, but no stations were controlled by the MCS at that time.

SUGGESTED ACTION:

If GEMCOS is in the process of initiating recovery after system failure and is under the control of SMCS, this error indicates that GEMCOS was unable to reattach any of the stations it owned previously, because these stations now are attached to other

programs. When the user attempts to run GEMCOS recovery under SMCS, the stations should not attach to other programs until recovery is complete.

24 RECOVERY ABORTED, STN NOT AVAILABLE - <explanation>

The MCS needs to send a recovered message to the program indicated by the specified station. GEMCOS, however, is unable to send the message because it does not own the station in question.

30 OUTPUT MESSAGE INVALID DUE TO: - <VARIANT>

The send message routine detected an error in the header of a message that was prepared by some other procedure. The meanings of the different possible variants are:

1. HARDWARE TYPE - An attempt was made to send a message to a station which cannot receive messages because of its hardware type (such as a card reader).

2. NDL MSG TYPE - The message type field in the header does not have a valid value.

3. TEXT SIZE - The message text size field in the header is greater then the maximum text size as declared at TCL generation.

4. LSN - The LSN in the header is invalid.

In all cases, except text size errors, the message is dropped. In the case of text size errors, the message is truncated to text-size bytes and sent.

SUGGESTED ACTION:

Obtain a Monitor Trace of the error, if possible, and submit this trace to the local Burroughs representative.

31 POSSIBLE FRACTURED FORMAT AT LSN <LOGICAL STATION NUMBER>

Screen wraparound sent a message containing a format to the indicated station. This is a warning message indicating that the format could be split in the middle of a forms field.

32 OPEN DENIED - MAXCOPIES EXCEEDED

An attempt made to execute more than the maximum number of copies of a given program.

SUGGESTED ACTION:

DS or DP the suspended program. Regenerate the TCL if the program is a user program and more copies are desired.

33 CANNOT HAP UTIL PGM FROM SPO OR CRD

An HAP Network Control Command cannot be entered from the console printer or the card reader for programs which have been described in the TCL as utility programs.

SUGGESTED ACTION:

Enter the command at the station(s) which entered the EX command for this program.

34 OPEN DENIED - OPEN OUTPUT CONFLICT

A program opened a remote file output only but the MCS detected one or more of the following conflicts:

1. Program must be declared NONPARTICIPATION.
2. Program must not be an MCS.
3. Program must not be a user program.

SUGGESTED ACTION:

Re-evaluate the program requirements and modify either the program remote file open or the TCL.

35 STATION NNN DATACOM ERROR: <VARIANT>

The MCS has received an error indication from the network controller about station NNN. The error is reported and logged for statistics reporting (RSC). The following are the valid values for the variant.

1. PARITY ERROR
2. BUFFER OVERFLOW
3. READ MEMORY PARITY
4. TIME OUT
5. BREAK
6. END OF BUFFER
7. LOSS OF DATA SET READY

8. LOSS OF CARRIER
9. ADDRESS ERROR
10. TRANSLATE ERROR
11. FORMAT ERROR
12. READ NOT READY

SUGGESTED ACTION:

Anaylze the data communications network for possible problems with terminals, modems, lines, or other data communications equipment.

36 OPEN DENIED - SYSTEM SHUT DOWN

An application program attempted to open a remote file during system shutdown.

SUGGESTED ACTION: 1.=Analyze the datacommunications network

The program must be discontinued.

37 OPEN DENIED ON REMOTE FILE

An application program attempted to open a remote file to which no stations could be attached.

SUGGESTED ACTION:

Check the FAMILY statement in the File section of the NDL and the Station section of the TCL deck.

38 GEMCOS MCS GOING DOWN

This message is sent to any station which attempts to enter a message during system shutdown. The input message is ignored.

39 INVALID PROGRAM NUMBER

The MCS has detected that a program has gone to EOJ, but is not recognized by the MCS as an active program.

SUGGESTED ACTION:

Use the RDM PRINT Network Control Command and the DM console keyboard commands to obtain diagnostic information. A monitor listing may also be helpful.

40 PROGRAM NOT KNOWN

A program not defined in the TCL was referenced in an EX or HAP Network Control Command.

SUGGESTED ACTION:

Use the RDM PRINT command to obtain a list of valid names.

41 INVALID PROGRAM NAME PROGRAM-NAME

The program named in the Network Control Command either does not exist or does not have a remote file open.

42 PROGRAM NOT IN DISK DIRECTORY PROGRAM-NAME

The program name in the EX Network Control Command is not on disk.

43 INVALID STATION NAME STATION-NAME

The station name in the Network Control Command was not recognized.

44 NO ACTIVE FILE NAMED FILE-NAME

The file named in the Network Control Command either does not exist or has not been opened.

45 INVALID INPUT MESSAGE - VARIANT

When reading from its input queue, the MCS detected an error in the input message. The type of error is identified by the variant:

1. INVALID MESSAGE TYPE FROM PROGRAM NUMBER <program number>

   The MCSTYPE field in the GEMCOS Common Area Header (CAH) is not one of the allowable values described in Section??

2. BAD TEXT SIZE FROM PROGRAM NUMBER <program number>

   The text size is greater than that allowed by the MAXTEXTSIZE specification.

3. INVALID LSN FROM PROGRAM NUMBER <program number>

   The LSN field in the GEMCOS Common Area Header describes an invalid or inactive LSN.

4. MISSING HEADER FROM PROGRAM NUMBER <program number>

   The program did not supply a GEMCOS header.

5. INVALID ROUTEHEADER RETURN FROM LSN FROM PROGRAM NUMBER <program number>

   The Routeheader Table index stored in the LSN field of the CAH pointed to an invalid LSN in the Routeheader Table.

6. INVALID MESSAGE DESTINATION TYPE FROM PROGRAM NUMBER <program number>

   The message destination field of the CAH is not one of the allowable values described in Section 2.

In either case the message is ignored, but is printed for debugging purposes.

SUGGESTED ACTION:

A type error in a message from an application program means that the program has put a value other than 0 (zero) in the type field of the actual key. In case of a SIZE error, longer messages can be allowed by changing MAXTEXTSIZE and regenerating.

46 INVALID OPERATION MNEMONIC

The Network Control Command mnemonic was not recognized.

47 INVALID OPTION ITEM

Some item expected as an option in a Network Control Command is invalid.

48 INVALID CHANGE DATA ITEM

The value of a data item in a Change Network Control Command is invalid.

49 COMMAND MISSING DATA ITEM

An item is missing from a Network Control Command.

50 EX COMMAND IGNORED - STA ATTACHED

An EX Network Control Command was entered from a station when the station was already attached to an assignment or utility program.

SUGGESTED ACTION:

If the station is attached to a utility program, enter a HAP command for that program and retry the EX. If the station is attached to an assignment program, all EX command from this station will return an error until the assignment program closes its remote file.

51 EX COMMAND LOCKOUT - TRY AGAIN

The EX command was temporarily not available; therefore, retry the command.

52 TRANCODE'S PGM NOT ONDEMAND

The station entered a valid trancode, but the program is not currently executing and was not declared in the TCL with the statement "EXECUTE = ONDEMAND".

SUGGESTED ACTION:

Send a message to the Control station or system operator requesting that the necessary program be executed or regenerated with EXECUTE = ONDEMAND.

53 INVALID MONITOR FLAG

The flag specification in a CMF command is invalid.

54 INVALID STATION LIST

A BRC command contains more stations in the destination list than exist in the system.

SUGGESTED ACTION:

Use fewer stations in the list. If the list is null, the message is sent to all stations.

55 FILE NOT OPEN

The MCS received a file close from a program whose remote file is not open.

56 INVALID INPUT - MISSING SIGNAL CHAR

The MCS received a message from the console keyboard and no signal character was present at the beginning of the message.

SUGGESTED ACTION:

Re-enter the command with the signal character (as declared in the TCL) in position 1.

57 TRANCODE'S PGM DIDN'T OPEN THIS STATION

The remote file which was opened by the program which handles this trancode did not contain this station.

SUGGESTED ACTION:

Change the FAMILY statement for this file in the NDL.

58 PROGRAM NOT RUNNING - <PGM NAME>

An attempt was made to pass a message to a utility program which was not running.

SUGGESTED ACTION:

Execute the program from a station before passing a message to it from another station.

59 NO TRANCODE AND STA NOT ATTACHED

The message entered did not have a trancode (or a message-ID if formatting is being used), and the station is not attached to a utility or assignment program.

60 USER ID NOT ALLOWED AT THIS STATION

This user is not allowed to sign on at this station.

61 SECURITY FAILURE - INVALID USER-ID

The user-ID in a SGN command is not recognized by the system.

62 SECURITY FAILURE - USER-ID DISABLED

The user-ID in a SGN command is disabled.

SUGGESTED ACTION:

The user cannot sign on until the user-ID is enabled by the EUS command.

63 SIGN-ON NOT REQUIRED

A SGN command was received from a station which either does not require signing on or is already signed on.

SUGGESTED ACTION:

Proceed with normal transactions as if signed on.

64 SECURITY FAILURE - SIGN-ON REQUIRED

The MCS received a message (other than a SGN command) from a station which requires signing on, but is currently signed off. The message just entered is ignored.

SUGGESTED ACTION:

Sign on with a SGN command.

65 SECURITY FAILURE - RESTRICTED PROGRAM

A message was entered by a user who is not allowed to use the destination program. The input message is ignored.

66 SECURITY FAILURE - RESTRICTED TRANS

A message was entered by a user who is not allowed to use the transaction code in that message. The message is ignored.

67 SECURITY FAILURE - RESTRICTED NCC

A Network Control Command was entered from a station that is not allowed to execute that command. Most commands are allowed only from the Control station.

68 SIGN-ON COMPLETE AT TIME ON DATE

A SGN command was successfully processed.

SUGGESTED ACTION:

The user is now signed on, and interaction with the system may begin.

69 SIGN-OFF COMPLETE AT TIME ON DATE

A BYE command was successfully processed.

SUGGESTED ACTION:

Interaction with the system from this station is prohibited until some user successfully signs on again.

70 CHANGE DENIED

The Network Controller responded to a network-change request with a change-denied message.

SUGGESTED ACTION:

The Network Controller should be checked to determine the reason for denial.

71 INVALID CHANGE TYPE

The Network Controller found the type code in a network-change request to be invalid.

SUGGESTED ACTION:

The maintenance module of the MCS should be checked for possible software error.

72 INVALID CHANGE RESULT

The Network Controller made an invalid response to a change request from the MCS.

SUGGESTED ACTION:

There may be a problem in the logic of the Network Controller.

73 INVALID CHANGE COMMAND

The Network Controller responded to a change request from the MCS, but the response is not what the MCS expected.

SUGGESTED ACTION:

Investigate the maintenance module of the MCS for a possible software error.

74 SEND DENIED ON CHANGE

The MCS found an error in a change request sent from the MCS to the Network Controller.

SUGGESTED ACTION:

The change command cannot be processed. Network activity continues, but further change requests may be ignored. If more changes are critical, the system must be shut down and restarted.

75 STATION NOT ATTACHED TO THIS PROGRAM <PGM NAME>

An attempt was made to HAP a utility program from a station. The station, however, was currently attached to a different program.

76 HAP COMMAND LOCKOUT, TRY AGAIN

The HAP command was temporarily unavailable; therefore, retry the command.

77 OPEN DENIED - PROG ALREADY RUNNING

An assignment program attempted to open a remote file when there already was a copy of that program running. Only user programs can have multiple copies running.

SUGGESTED ACTION:

The assignment program must be DSed.

78 CANNOT EX UTIL PGM FROM CARD OR SPO

An EX Network Control Command cannot be entered from the Control station or card reader for programs which have been described in the TCL as utility programs.

SUGGESTED ACTION:

Either enter the EX command from a station or make the regenerate run of the TCL to describe this program as an assignment program or user program

79 OPEN DENIED - UTIL PGM NOT EXPECTED

A program described in the TCL as a utility program attempted to open a remote file, and the MCS was not expecting the open request (i.e., the program was not executed from a station).

SUGGESTED ACTION:

DS the program and execute it from the station that is to use it.

80 UNEXPECTED ATTACH REPLY RECEIVED

The MCS received an unexpected ATTACH REPLY from the Network Controller. The message is ignored.

SUGGESTED ACTION:

Check the MCS and Network Controller logic.

81 ATTACH REPLY MISMATCH

The MCS received an ATTACH REPLY which attached the wrong station or the wrong file or both.

SUGGESTED ACTION:

Check the MCS and Network Controller logic.

82 ATTACH REQUEST DENIED BY NC, TYPE = DENIAL REASON

The last attach request sent by the MCS was denied by the Network Controller. The station will not be attached. The possible values of denial reason are described in the B 1700 Systems Network Definition Language (NDL) Reference Manual, form 1073715.

SUGGESTED ACTION:

Check the MCS and Network Controller logic.

83 UNEXPECTED DETACH REPLY RECEIVED

The MCS received an unexpected DETACH REPLY from the Network Controller. The message is ignored.

SUGGESTED ACTION:

Check the MCS and Network Controller logic.

84 DETACH REPLY MISMATCH

The MCS received a DETACH REPLY which detached the wrong station or the wrong file or both.

SUGGESTED ACTION:

Check the MCS and Network Controller logic.

85 STATION NOT ATTACHED, INPUT IGNORED

If no participating programs are declared in the TCL, and a station is not attached to a program, any message sent from that station results in this error. Entering *HAP without a program name when not attached also creates this error condition.

86 FMT ERR, DEST PTR OUT OF BOUNDS

The formatted message is larger than MAXTEXTSIZE.

SUGGESTED ACTION:

Check the description of the format in the TCL.

87 FMT ERR, SOURCE PTR OUT OF BOUNDS

The raw (unformatted message) is larger than MAXTEXTSIZE.

SUGGESTED ACTION:

Check the description of the format in the TCL.

88 FMT ERR, NON-DIGIT IN INTEGER FIELD

The formatting code found a non-numeric character or an embedded blank in an integer field.

SUGGESTED ACTION:

Check the format description in the TCL and the application program logic.

89 FMT ERR, MISSING SKIP DELIMITER

The message from the application program was missing a skip delimiter.

SUGGESTED ACTION:

Check the format description in the TCL and the application program logic.

90 FMT ERR, VARIABLE REPEAT ON INPUT

The station message invoked a format containing a variable repeat.

SUGGESTED ACTION:

Change the format description in the TCL to exclude any variable repeats on input.

91 FMT ERR, MISSING DELIMITER

The message from the application program was missing a delimiter.

SUGGESTED ACTION:

Check the format description in the TCL and the application program logic.

92 FMT ERR, INVALID TRANSLATE FIELD

The formatter attempted to translate a portion of the message from the application program using a TRANSLATE function, but the text did not match any of the internal strings of that function.

SUGGESTED ACTION:

Check the format and function description in the TCL and the application program logic.

93 OPEN DENIED - WRONG TCL INTERFACE

Either the application program opened a remote file "with headers" and the TCL INTERFACE statement for that program was not declared to be INTERFACE = MCS, or the INTERFACE was declared to be MCS, but the program did not open its remote file "with headers".

SUGGESTED ACTION:

Correct the TCL INTERFACE parameter to match the program and regenerate MCSTIC.

94 INPUT IGNORED DURING RECOVERY

The MCS received a message which is bound for a program that is undergoing recovery. The input message is ignored.

SUGGESTED ACTION:

When recovery is complete, enter the message again. Messages may be entered for programs not being recovered.

95 HAP NOT ALLOWED - PGM OPENED OUTPUT

An attempt was made by a station to *HAP a program declared to be "output only". This is not accepted since the MCS cannot send an EOF to an output-only remote file.

SUGGESTED ACTION:

The program must be able to finish without aid from the MCS or it must be DSed.

96 CAN'T SEND EOF TO OUTPUT ONLY PGM # <PROGRAM>

During the MCS shutdown phase, programs are sent an EOF indicator which requires them to terminate. However, output-only programs cannot be sent this EOF notice.

SUGGESTED ACTION:

The program must be terminated manually.

97 OPEN DENIED - PROGRAM NOT KNOWN

A program which was not defined in the TCL attempted to open a remote file.

SUGGESTED ACTION:

DS or DP the program. Modify the TCL deck to include this program.

98 OPEN DENIED - PROGRAM DISABLED

A program known to the MCS attempted to open a remote file but the program was marked disabled.

SUGGESTED ACTION:

DS or DP the program. Clear the program (*CLE) from the console printer or Control station. If all programs belonging to the data base are marked OK, recovery is then initiated.

99 OPEN DENIED - PGM OPENED 2 RMTE FILES

A program attempted to open more than one remote file concurrently. The MCS permits a program only one remote open at any given time.

SUGGESTED ACTION:

DS or DP the program. Disallow the program from concurrent remote file opens.

100 ABNORMAL CLOSE DONE ON RESTART PGM - <PROGRAM>

The MCS received a remote file close message from the Network Controller, but the restart program was still active. (See errors 101 and 102.)

SUGGESTED ACTION:

Check the MCS and restart program interface logic.

101 ENTER "OK" TO RESTART RESTRT PGM
102 ENTER "NO" TO KILL THE MCS

These two error messages are displayed on the Control station or console printer when a serious problem is encountered with the restart program. A message immediately preceding these errors explains the nature of the problem.

SUGGESTED ACTION:

Either enter "OK" or "NO" from the console printer by means of the accept mechanism.

103 OPEN DENIED-RESTART PGM NOT EXPECTED

A restart program opened a remote file, but the MCS was not expecting a remote file to be open at that time.

SUGGESTED ACTION:

DS or DP the program. Check the MCS and restart program interface logic.

104 END POSSIBLE DUPLICATE MESSAGES

This error message always occurs in conjunction with error 105. A station receives this message after all possible duplicate messages have been displayed at the station at the finish of a synchronized recovery.

105 BEGIN POSSIBLE DUPLICATE MESSAGES

This error message and error 104 are displayed at the end of a synchronized recovery if there are any messages for a station that the MCS cannot be sure the station has already seen. These messages are displayed between message 105 and message 104.

106 BUSY

An attempt was made to enter a transaction at a station before the output from the previous transaction was received. This only occurs if TRANSACTIONMODE = TRUE for that station. The entered transaction is ignored.

SUGGESTED ACTION:

Wait for output from previous transaction, or enter the RBS Network Control Command to reset the busy status of the station.

107 MSG NOT FOR SYNC DATABASE PGM

An attempt was made to send a message to a program that is not part of a synchronized data base (as declared in the TCL).

SUGGESTED ACTION:

Either restrict messages from this station to synchronized data base programs only, or set the TCL Station statement TRANSACTIONMODE to FALSE.

108 BEGIN RECOVERY OF <DATABASE/PROGRAM> - <number>
109 END RECOVERY OF <DATABASE/PROGRAM> - <number>

These two messages are displayed on the control station or console printer at the start and finish of recovery whenever recovery is initiated. For data base and synchronized recovery, the word DATABASE appears, otherwise the word PROGRAM appears. Additionally, any station that attempts to send a message to a program currently undergoing recovery receives message 109 upon completion of recovery.

110 GEMCOS ARCHIVAL RECOVERY
111 ENTER ARCHIVAL SPECIFICATIONS

These messages are displayed on the console printer whenever the MCS is executed in archival recovery mode.

SUGGESTED ACTION:

Enter the desired specifications as described in Archival Recovery in Section 9.

112 INVALID ARCHIVAL SPECIFICATIONS

The archival specifications entered at the console printer were incomplete or totally incomprehensible.

SUGGESTED ACTION:

Consult Archival Recovery in Section 9 to determine the proper syntax.

113 MISSING OR UNRECOGNIZED WORD - <WORD>

The archival specifications entered at the console printer contained a word (indicated in the error message) that was unrecognized or unexpected.

SUGGESTED ACTION:

Consult Archival Recovery in Section 9 to determine the proper syntax.

114 "ALL" MUST BE ONLY NAME IN PGM LIST

The archival specifications entered at the console printer contained program names in addition to the word "ALL".

SUGGESTED ACTION:

Either enter a list of program names or the word "ALL" at this point in the archival specifications.

115 BEGIN ARCHIVAL RECOVERY
116 END ARCHIVAL RECOVERY

These two messages are displayed on the console printer at the start and finish of archival recovery, respectively. After displaying message 116, the MCS terminates.

117 NO INPUT ALLOWED: ARCHIVAL RECOVERY

An attempt was made by a station to send a message to a program while the MCS was in archival recovery mode. This and any other such request is ignored.

SUGGESTED ACTION:

Do not send any messages to programs during archival recovery.

118 PROGRAM NOT IN DATABASE - <number>

The archival specifications entered at the console printer contain a program name that does not belong to the previously mentioned data base.

SUGGESTED ACTION:

Check the archival specifications against the TCL specifications describing which programs belong to the data base.

119 CANNOT PASS TO USER-TYPE PROGRAM

An attempt was made to pass a message to a User-type program.

120 MCSTIC TIME MISMATCH - <audit file-name>

The MCSTIC date/time stamp in the MCSTIC file for the current audit file does not match the MCSTIC date/time stamp in the audit file.

SUGGESTED ACTION:

Check the validity of the MCSTIC file and the audit file named <audit file-name>.

121 AUDIT FILE TIME MISMATCH - <audit file-name>

The audit file date/time stamp in the MCSTIC file for the current audit file does not match the audit file date/time stamp in the audit file.

SUGGESTED ACTION:

Check the validity of the MCSTIC file and the audit file named <audit file-name>.

122 CURRENT TIME LESS THAN LAST AUDIT

The value of the current time maintained by the system clock is less than the time of the last audit performed by the MCS.

SUGGESTED ACTION:

Check the validity of the system time clock.

124 ROUTEHEADER ERROR - <explanation>

An attempt was made to send a message to a routeheader station or a message was received from a routeheader station that contained an error.

SUGGESTED ACTION:

The <explanation> portion of the error details the problem. It is necessary to examine the TCL on both hosts for possible conflicts.

125 MCS LOST REPLY TO <NN> INPUTS TO PGM - <number>

During synchronized recovery, the MCS detected <NN> input messages from a given station to a program for which no output messages were ever received.

SUGGESTED ACTION:

An inquiry into the data base should be made to see if the transactions in question are on the data base.

126 UNEXPECTED CLOSE FROM PGM - <number>

A program that is part of a data base using audit and recovery was terminated abnormally. The data base is marked as needing recovery. Error message 127 always follows this message.

127 TO INITIATE RECOVERY, CLEAR PGM - <number>

Either this program just terminated abnormally or an attempt was made to initiate recovery (by the MCS or the user). In either case, the program must be cleared before recovery can begin.

SUGGESTED ACTION:

Clear the disabled program using the *CLE command. If recovery does not begin, then at least one other program in the data base is still disabled and must also be cleared.

128 FILE MISSING - <audit file-name>

During recovery, the named audit file was sought on disk by the MCS but was not found.

SUGGESTED ACTION:

The named audit file must be loaded on disk, and an *AOK command must be entered on the console printer.

129 FILE LOCKED - <audit file-name>

During recovery, the MCS tried to open the named audit file but found it was locked by another program.

SUGGESTED ACTION:

The program that locked the named audit file must free it for use by the MCS, and an *AOK command must be entered on the console printer.

130 INCORRECT FILE - <audit file-number>

During recovery, the MCS opened the named audit file and determined that one or both of the date/time stamps in the file did not match the value stored in the MCSTIC file.

SUGGESTED ACTION:

Check the named file to insure that it is the correct file.

131 WRONG AUDIT RECORD REQUESTED - PGM <number>

During recovery, the MCS found an error in the audit file during an attempt to service a program. The program is DSed if it is part of a data base; otherwise, the MCS is terminated.

SUGGESTED ACTION:

Either bring the MCS back up or clear the DSed program.

132 UNEXPECTED TYPE 22 MSG FROM PGM - <number>

This program sent the MCS an unsolicited type-22 message. A program can only send this message upon receipt of a type-21 message. The program is DSed.

SUGGESTED ACTION:

Investigate the logic of this program. Clear the program to initiate recovery.

133 NO AUDITED MESSAGES FOR REF NCC

An *REF command was entered to recall the last audited output message for a station, but the MCS had no audited messages for this station.

134 OLD AUDIT FILE MISSING FOR REF NCC

An *REF command was entered, but the last audited message for that station was in an audit file that is not on disk.

SUGGESTED ACTION:

Load the missing audit file on disk.

135 MISSING PROGRAM NAME

A Network Control Command was entered that required either a program name or number, but none was found.

SUGGESTED ACTION:

Refer to the syntax of the Network Control Command that was in error.

136 INVALID PROGRAM - <token>

The program name or number entered (token) was found to be either invalid or unrecognized.

SUGGESTED ACTION:

Refer to the TCL specifications for a list of all valid program names.

137 PROGRAM NOT DISABLED - <program name or number>

An attempt was made to clear a program (using the *CLE command) that was not disabled.

SUGGESTED ACTION:

If there is any doubt of the status of the program, recovery can be performed on that data base.

138 INVALID DATABASE NAME - <database name or number>

The data base name or number entered was found to be either invalid or unrecognized.

SUGGESTED ACTION:

Refer to the TCL specifications for a list of all valid data base names.

139 PROGRAM DISABLED - <program name or number>

An attempt was made to execute this program either by the *EX command or by entering a trancode for this program, but the program was disabled.

SUGGESTED ACTION:

Clear the program with the *CLE command and, thus, initiate recovery of this data base.

140 EOF ALREADY SENT TO - <program name>

An attempt was made to send a *HAP command to a program more than once.

SUGGESTED ACTION:

If the program does not go to EOJ within a few minutes, investigate the EOF logic of the remote file of the program.

141 INPUT IGNORED; PROGRAM TERMINATING

An attempt was made to send a message to a program that is in the process of halting (a *HAP command was performed on this program).

SUGGESTED ACTION:

Re-execute the program.

142 UNEXPECTED TYPE 25 MSG FROM PGM - <program number>

A program sent the MCS a message with MCS-TYPE field of the common-area header set to 25 without first receiving a type-24 message.

SUGGESTED ACTION:

Investigate the program's MCS interface logic.

143 TYPE 17 OR 18 MSG FROM PROGRAM - <program number>

A program that was not declared in the TCL to be a restart program sent a message to the MCS with the MCS-TYPE field of the common-area header set to 17 or 18.

SUGGESTED ACTION:

Investigate the program's MCS interface logic. Only restart programs can send type-17 or-18 messages.

144 REF NCC CANNOT COME FROM SPO

An attempt was made to enter a *REF command from the console printer. This is not allowed and is ignored.

145 RESTART PGM RETURNED BAD DATABASE - <database-name>

The MCS was expecting the restart program to return the name of the data base to be recovered, but the name returned did not match the MCS's expectations.

SUGGESTED ACTION:

Investigate the restart program's MCS interface logic. The <database-name> is what was received by the MCS.

146 RESTART PGM FOUND ERROR - DATABASE <database-name>

The restart program found an error while accessing the data base. See error messages 101 and 102 for further explanation and suggested action.

147 RESTART PGM RETURNED BAD PDT # - <PDT number>

The restart program returned a bad value for the PDT number. This value was originally stored in the restart data set by the program that created the record.

SUGGESTED ACTION:

Investigate the program logic of all programs in the data base that deals with the type-23 message.

148 RESTART PGM RETURNED BAD PROG # - <PROG number>

The restart program returned a bad value for the PROG number. This value was originally stored in the restart data set by the program that created the record.

SUGGESTED ACTION:

Investigate the program logic that deals with the type-23 message in all programs in the data base.

149 MISSING USERCODE/PASSWORD

An attempt was made to execute a program with the user-password option of the EXECUTE command, but no usercode/password pair was found.

SUGGESTED ACTION:

Refer to the EXECUTE statement in the Program Control Commands Section in Section 3.

150 INVALID USERCODE/PASSWORD

An attempt was made to execute a program with the user-password option of the EXECUTE command, but an invalid usercode/password was entered.

SUGGESTED ACTION:

Refer to the EXECUTE statement in the Program Control Commands Section in Section 3.

151 COMMAND IGNORED, DUPLICATE TRNSMIT - <EX/HAP>

An attempt was made to EXECUTE or halt (HAP) a utility program before a previous EXECUTE or HAP of the program was completed.

SUGGESTED ACTION:

Do not enter consecutive EXECUTEs or HAPs of utility programs.

152 CANNOT DFR FROM NONSUBORDINATED MCS

An attempt was made to detach a station from the GEMCOS remote file. GEMCOS, however, has not been declared a subordinate MCS.

SUGGESTED ACTION:

See the subordinate MCS statement in the Global section.

153 STATION STILL ATTACHED TO A PROGRAM

A DFR Network Control Command was entered from a station which was still attached to a program.

SUGGESTED ACTION:

A HAP command should be entered before attempting the DFR.

154 DUPLICATE OPTION - <lock/charge num/us>

One of the options of the EXECUTE command was entered more than once. Refer to Program Control Commands in Section 3.

SUGGESTED ACTION:

Re-enter command.

155 ENABLE STATION TRANSFER AT ODT

BNA station transfer must be enabled at the ODT before a user can execute the PLM program at a station.

156 VIRTUAL CONNECT DENIED FOR <STN NAME>

A virtual connect request for a virtual station was denied for one or more of the following reasons:

1. The station was not defined in the TCL.

2. The station hostname did not match the station hostname in the TCL.

3. The station was already connected.

157 DISCONNECT FROM VIRTUAL STATION <STN NAME> <HOSTNAME>

A virtual station has disconnected from this MCS.

158 QUEUE FULL CONDITION ENCOUNTERED: <STN NAME>

GEMCOS has detected a full-queue condition at the indicated station. All messages bound for that station are tanked until the station is made ready and put in the receive mode. The messages are then delivered in groups once every minute until the tank is exhausted. The tanked messages can also be flushed from the tank with the PQ command.

159 EOF ON MCSTANK - FILE CLOSED - DUMP

GEMCOS has detected an unexpected EOF condition on its tank file. The file is closed for analysis and a system dump is taken. Processing then continues. The system dump and file should be listed using standard system software. Then submit the results to a Burroughs representative.

160 UNEXPECTED GOOD RESULTS FROM LSN - LSN

The MCS received a good results reply message from the Network Controller for a station that was not expecting this message.

SUGGESTED MESSAGE:

Check the MCS and Network Controller interface logic.

161 INVALID FILE CLOSE - VARIANT

An application program tried to close a remote file which it did not previously open. The close request is ignored. If "variant" is INVALID PROG NUMBER, then the program does not have any remote files open. If "variant" is FILE NOT OPEN, then it has at least one other remote file open.

SUGGESTED ACTION:

Scrutinize the application program for possible logic flaws.

162 INVALID MSGID FOR THIS DEVICE

A forms request was entered at a station and the message-ID was not valid for that station.

163 GOOD RESULTS RETURNED BAD LSN - LSN

The MCS received a good results reply message from the Network Controller, but was unable to recognize the station indicated by the LSN.

SUGGESTED ACTION:

Check the MCS and Network Controller interface logic.

164 INCOMPLETE UPDATE COMMAND

The UPD Network Control Command entered at this station was not complete.

165 INVALID KEY WORD AFTER "UPD"

The second word in the UPD Network Control Command must be either "FORMATS" or "ACCESSKEY".

166 INVALID UPDATE, ACCESSKEY SIGNED ON

An attempt was made to change an active access key with the UPD Network Control Command. Only access keys which are not signed on can be changed.

167 CANNOT UPDATE TO EXISTING ACCESSKEY

An attempt was made to change an access key to an existing access key with the UPD Network Control Command, i.e., the second name in the command was already a valid access key.

168 THIS STATION RE-ATTACHED TO GEMCOS

When GEMCOS is running subordinate to SMCS and a SYSTEM/GEMCOS failure occurs, this message is sent to each station which is successfully reattached. This message can be suppressed by setting SUPPRESSMESSAGES = TRUE in the TCL for that station.

169 PROGRAM BEING ZIPPED - PLEASE WAIT

This message is displayed when a trancode is received for a program which is not running and is declared ONDEMAND in the TCL. GEMCOS continues to respond with this message to entries of the trancode until the program successfully opens its remote file.

170 RECOVERY ERROR, STN NOT AVAILABLE - <explanation>

The MCS received a message from a recovering program and the LSN in the common-area header is invalid. GEMCOS will change the LSN to a valid number and then send the message.

171 COULD NOT RE-ATTACH LSN <N>

When GEMCOS was running subordinate to SMCS, a failure occurred, and GEMCOS could not reattach the indicated LSN. GEMCOS recovers anyway, as long as the station is not needed for the recovery. If that station is needed for the recovery, detach the station from its program and then re-execute GEMCOS.

172 NBR OF STATIONS RE-ATTACHED = <N>

When GEMCOS was running subordinate to SMCS, a failure occurred. This message indicates the number of stations which were re-attached to GEMCOS.

173 DESTINATION STN HAS QFULL CONDITION

The destination station of a PQ command has a QFULL condition and therefore is invalid. Clear the QFULL condition or send the popped messages to a different station.

174 STATION ALREADY ATTACHED TO GEMCOS

An attempt was made to attach a station that is already controlled by GEMCOS. No further action is required.

175 INVALID LSN <entry>

An attempt was made to attach a station with the ATT command. The LSN was invalid because it was non-numeric, not defined in the TCL, or not first attached to GEMCOS with the ATTACH command of SMCS.

176 ONLY VALID FROM SUBORDINATE MCS

The command entered is valid only from an MCS with SUBORDINATEMCS = TRUE in the TCL.

177 EOF/EXCEPTION ON NONINTERP FILE

While reading the noninterpretive format file, the MCS detected either an end-of-file condition or a hardware exception. Although the format requested has not been read, the MCS attempts to continue.

178 NOT ENOUGH DATA FOR FORMAT FROM <program number>

A program requested a noninterpretive format, but it did not send enough data to fill all of the format's output fields.

179 TOO MUCH DATA FOR FORMAT FROM PROGRAM <program number>

A program requested a noninterpretive format, but it sent more data than could fit in the format's output fields.

APPENDIX E

HARDWARE REQUIREMENTS

Peripheral hardware needed to generate and execute a GEMCOS MCS includes:

1. Card reader. (Optional. May be card reader or data communications terminal.)
2. Line printer.
3. Console printer/keyboard.
4. Disk subsystem. (If AUDIT is used, at least 4.6 megabytes are recommended, including MCP and Network Controller requirements, but excluding user data files.)
5. Single-line or multiline control.

A minimum of 38K bytes of main memory (exclusive of MCP requirements) is needed to generate an MCS. The smallest MCS that can be generated requires 7K bytes (exclusive of MCP and Network Controller requirements). Memory requirements increase for each option generated and for each additional entry in MCS tables.

The following chart can be used to estimate memory requirements in bytes for an MCS generated by GEMCOS. The calculation of the run structure as performed by the generator is a relatively complex process, and this chart is a simplification of it. Thus, the number calculated here is only an estimate of the memory requirements. It should be accurate to within 10 percent of the actual value.

The memory requirements for resident formats are difficult to estimate, and it is assumed in the algorithm used in the chart that all formats are to reside on disk. To determine the amount of memory required for resident formats, compile the format descriptions with MCSTCL setting LIST in the CONTROL statement. The printer file labeled MCSRPT will be produced. Consult the pages which refer to FUN and FOR records adding the "lengths" of all functions and formats where "residence" is 1.

Note that the memory requirements of a B 1000 object code file can be obtained with the CODE/ANALYZER program found on a system release tape. However, in the case of a GEMCOS MCS, the calculation of the file space

is not entirely correct since the MCS modifies several attributes of its files at run time.

FILE SPACE

MCSQUEUE

Enter MAXTEXTSIZE on line 1.	1) _____
Enter 50 on line 2.	2) _____
If any program uses the PARTICIPATION interface enter 200 on line 3; otherwise enter 0.	3) _____
Add lines 1, 2 and 3 giving line 4.	4) _____
Enter the number of stations defined in the <STATION section> on line 5.	5) _____
Enter 5 on line 6.	6) _____
Multiply lines 5 and 6 giving line 7.	7) _____
Enter QUEUEBUFFERS-1 on line 8.	8) _____
Enter line 4 on line 9.	9) _____
Multiply lines 8 and 9 giving line 10.	10) _____
MCSTIC	12) 537

MCSPRINT

If MONITORTRACE is TRUE, enter 248 on line 13.	13) _____

MCSAUDIT

Enter AUDITRECORDSIZE on line 14.	14) _____
Enter 357 on line 15.	15) _____
If AUDIT is TRUE, add lines 14 and 15 giving line 16; otherwise, enter 0 on line 16.	16) _____

MCSOLDAUDIT

If recovery is required, enter line 15 on line 17; otherwise enter 0. 17) _____

Add lines 11, 12, 13, 16 and 17 giving line 18. 18) _____

RUN STRUCTURE NUCLEUS

Enter 337 on line 19. 19) _____

F.I.B. dictionary

Enter 80 on line 20. 20) _____

GLOBAL VARIABLES

Enter 368 on line 21. 21) _____

If DATA DUMP is TRUE, enter 3 on line 22. 22) _____

If MESSAGERECALL or CHANGEREQUESTS is TRUE, enter 37 on line 23. 23) _____

If PROGRAMBOJEOJ is TRUE, enter 6 on line 24. 24) _____

If any program uses an interface of Participation, enter 203 on line 25. 25) _____

If SYSTEMHALT is TRUE, enter 5 on line 26. 26) _____

If AUDIT is TRUE, enter 36 on line 27. 27) _____

If recovery is synchronized for any program, enter 234 on line 28; if RECOVERY = DATABASE or QUEUERESTORATION for any program, enter 11 on line 28. 28) _____

If the MCS is to format any message, enter 13 on line 29. 29) _____

If the MCS is to provide access control, enter 35 on line 30. 30) _____

If a CONTROLSTATION is specified, enter 5 on line 31. 31) _____

If any Network Control Command is to be supported, enter 178 on line 32. 32) _____

Add lines 21 through 32 giving line 33. 33) _____

GLOBAL TABLES AND MESSAGE AREAS (Excluding formatting)

If any program uses an interface of Participation, enter 203 on line 34. 34) _____

If any kind of recovery is required, enter AUDITRECORDSIZE + 30 on line 35; if only auditing is required, enter AUDITRECORDSIZE on line 35. 35) _____

Enter MAXTEXTSIZE on line 36. 36) _____

Enter 55 on line 37. 37) _____

Enter the number of stations defined in the station section on line 38. 38) _____

Enter 91 on line 39. 39) _____

Multiply lines 38 and 39 giving line 40. 40) _____

Enter the sum of all copies of all programs on line 41. 41) _____

Enter 51 on line 42. 42) _____

Multiply lines 41 and 42 giving line 43. 43) _____

Enter the number of trancodes defined in the Program section on line 44. 44) _____

Enter 13 on line 45. 45) _____

Multiply lines 44 and 45 giving line 46. 46) _____

Enter the number of programs defined in the Program section on line 47. 47) _____

Enter 9 on line 48. 48) _____

Multiply lines 47 and 48 giving line 49. 49) _____

Enter line 38 on line 50. 50) _____

Enter line 43 on line 51. 51) _____

Enter line 46 on line 52. 52) _____

Enter 49 on line 53. 53) _____

Add lines 50 through 53 giving line 54. 54) _____

Multiply lines 50 and 54 giving line 55. 55) _____

Enter 8 on line 56. 56) _____

If an ACCESS CONTROL statement is present, divide line 55 by line 56 giving line 57. 57) _____

Add lines 34, 35, 36, 37, 40, 43, 46, 49 and 57 giving line 58. 58) _____

GLOBAL FORMATTING TABLES AND MESSAGE AREAS

Enter line 4 on line 59. 59) _____

Enter number of functions defined on line 60. 60) _____

Enter 3 on line 61. 61) _____

Multiply lines 60 and 61 giving line 62. 62) _____

Enter the number of formats defined on line 63. 63) _____

Enter 3 on line 64. 64) _____

Multiply lines 63 and 64 giving line 65. 65) _____

Enter line 60 on line 66. 66) _____

Enter the number of message-IDs defined in the Device section on line 67. 67) _____

Add lines 66 and 67 giving line 68. 68) _____

Enter the number of devices defined in the Device section on line 69. 69) _____

Multiply lines 68 and 69 giving line 70. 70) _____

Enter 10 on line 71. 71) _____

Multiply lines 70 and 71 giving line 72. 72) _____

Enter 8 on line 73. 73) _____

Divide line 72 by line 73 giving line 74. 74) _____

Enter line 67 on line 75. 75) _____

Enter 6 on line 76. 76) _____

Multiply lines 75 and 76 giving 77. 77) _____

Enter 384 on line 78. 78) _____

Add lines 59, 62, 65, 74, 77 and 78 giving line 79. 79) _____

GLOBAL PROCEDURE VARIABLES

If MONITORTRACE is TRUE, enter 45 on line 80. 80) _____

Enter 145 on line 81. 81) _____

Enter 0 on line 82. 82) _____

Enter 18 on line 83. 83) _____

If stations have been assigned SCREENSIZE values in the Station section, add lines 82 and 83 giving line 84. 84) _____

If DATADUMP is TRUE, enter 303 on line 85. 85) _____

Enter 0 on line 86. 86) _____

Enter 0 on line 87. 87) _____

If AUDIT is TRUE, add lines 86 and 87 giving line 88. 88) _____

Add lines 80, 81, 84, 85 and 88 giving line 89. 89) _____

LOCAL VARIABLES

Enter 41 on line 90. 90) _____

If any network control command is to be supported, enter 170 on line 91. 91) _____

If any message is to be formatted, enter 315 on line 92. 92) _____

Enter 123 on line 93. 93) _____

Enter largest value of lines 91, 92 and 133 on line 94.	94)	_____
Add lines 90 and 94 giving line 95.	95)	_____

MESS CODE

Enter the number assigned to VALUESTACKBITS on line 96.	96)	_____
Enter 8 on line 97.	97)	_____
Divide line 96 by line 97 giving line 98.	98)	_____
Enter the number assigned to NAMESTACKENTRIES on line 99.	99)	_____
Enter 3 on line 100.	100)	_____
Multiply lines 99 and 100 giving line 101.	101)	_____
Add lines 98 and 101 giving line 102.	102)	_____
Add lines 19, 20, 33, 58, 79, 89, 95 and 102 giving line 103.	103)	_____
Compute 10% of line 103 and enter on line 104.	104)	_____
Add lines 103 and 104 giving line 105.	105)	_____

OTHER MEMORY CONSIDERATIONS

DICTIONARY CONTAINER	106)	30
MASTER CODE SEGMENT DICTIONARY	107)	30
MEMORY LINKS	109)	100
Add lines 106 through 109 giving line 110.	110)	2150
Add lines 18, 105, and 110 giving the Estimated Memory to Run:		_____

MCSTIC FILE DISK SPACE REQUIREMENTS

A record in the MCSTIC file is 180 bytes long. The file normally contains from 40 to 100 records.

AUDIT FILE DISK SPACE REQUIREMENTS

The disk space used for audit files depends largely on the operational procedures used at a specific installation. The system must be able to hold at least two audit files on disk at one time, since audit files can be stored on some off-line medium such as magnetic tape when closed.

A record in an audit file is 180 bytes by default. If AUDITRECORDSIZE is specified, the length of the record is determined by the user. The file contains AUDITPAGESIZE*40 records.

APPENDIX F

COBOL74 PROGRAMS AND B 1000 GEMCOS

GEMCOS provides interface capabilities between the GEMCOS-generated MCS and COBOL74 programs. Burroughs COBOL74 complies to the coding standards established by the American National Standards Institute (ANSI) of 1974. (For further information about the COBOL74 language, refer to the B 1000 System COBOL74 Reference Manual for the latest release of the B 1000 systems.)

TCL REQUIREMENTS

COBOL74 programs that interface with GEMCOS do not affect or require changes in the specifications established in the TCL. However, the programs must be declared in the Program section of the TCL, and defined as participating programs in order to facilitate COBOL74 interface.

NETWORK DEFINITION LANGUAGE (NDL) REQUIREMENTS

The NDL/LIBRARY file, provided for the 9.0 release of the B 1000 system, includes COBOL74 declarations and the COBOL74SEL request set. Both the declarations and the request set must be copied from the NDL/LIBRARY file and merged into the appropriate area in the user's NDL source file. POLL/SELECT devices, interfacing with COBOL74 programs, require a DIAGNOSTIC statement in the Terminal section of the NDL. This statement must specify the request set, POLLTCTD, in the RECEIVE portion, and COBOL74SEL in the TRANSMIT portion of the statement. An example follows.

Example:

```
TERMINAL TD830:
    ADDRESS = 2.
    TRANSMISSION = 0.
    REQUEST = CANDEPOLTD : RECEIVE, CANDESELTD : TRANSMIT.
    DIAGNOSTIC = POLLTCTD : RECEIVE, COBOL74SEL : TRANSMIT.
    BUFFERSIZE = 2000.
    TYPE = 46.
```

The Network Controller is recompiled upon entering specifications such as those presented in the example above as well as merging declarations and the COBOL74SEL request set. (For additional information, refer to the B 1000 Systems Network Definition Language (NDL) Reference Manual.)

COBOL74 PROGRAM REQUIREMENTS

Before COBOL74 programs send messages to the GEMCOS MCS, they must execute the ENABLE <cd-name> or the RECEIVE INPUT <cd-name> command. Either command generates a network-control FILE OPEN message (TYPE = 18), which is sent to the MCS. The MCS accepts no messages from a COBOL74 program until it receives the FILE OPEN message.

RESTRICTIONS

Multiple COBOL74 programs using communication descriptions (CDs) must not open the same remote file. If this occurs, the results are unpredictable due to the 9.0 system software implementation. This restriction limits the use of the MAXCOPIES statement.

USING THE RESTART PROGRAM

The restart program has been converted to COBOL74. Two source files are on the 7.0 release tape. The first source file is GEMCOS/MCSRSTRT74, which uses a remote file. The second source file is GEMCOS/MCSRSTRTCD, which uses communication descriptions. Both source files should be modified before they are compiled.

APPENDIX G

SYNTAX DIAGRAM CONVENTIONS

This appendix explains the "railroad" syntax diagrams used with the syntax of Transaction Control Language statements and Network Control Commands. The following rules apply to these syntax diagrams:

1. Any path traced along the forward direction of the arrows produces syntactically valid TCL source code.

2. Any "bridge" over a digit may be traversed a maximum number of times specified by the digit. If the digit is followed by an asterisk (*), the path must be crossed at least once.

3. Uppercase letters in the syntax diagrams indicate key words which are literally in the statement.

4. Lowercase letters, words and phrases enclosed within angle brackets (< and >) are either references to an intermediate diagram, or syntactic variables, which represent information to be supplied by the user.

5. A colon must be preceded and followed by a least one space.

6. A file-ID is a B 1000 file identifier. Refer to the B 1000 System Software Operational Guide.

7. A remote file-ID is a 10-character identifier defined in the File section of the NDL.

8. A station name is a 10-character identifier defined in the Station section of the NDL.

9. All other TCL identifiers may contain alphanumeric characters. Identifiers have no intrinsic meaning. They are used to name access codes, programs, trancodes, messages, formats, and functions. The access code identifiers and message identifiers may not exceed 6 characters. Trancode identifiers may not exceed 10 characters. Program name identifiers, format identifiers, and function identifiers should not exceed 30 characters.

10. A logical station number (LSN) is a number that shows the position of the station definition of the Station section in the NDL.

11. A string is any number of characters enclosed within quotes. A string must begin and end on the same TCL source record.

12. An external string or an internal string is a string of 6 characters or less.

13. An EBCDIC unit string is a string with exactly one character.

14. A character is A through Z, 0 through 9, or any valid special character. A character is not enclosed within quotes.

15. An integer is a string of digits (0 through 9).

16. For a definition of a UPL2 DECLARATION statement, UPL2 DEFINE statement, UPL2 FILE statement, UPL2 DYNAMIC DECLARATION statement, or UPL2 PROCEDURE statement, refer to the DOCUMENT/SDL2 file on the release tape.

INDEX