Honeywell

SERIES 600

SOFTWARE

GECOS

TIME-SHARING SYSTEM GENERAL INFORMATION MANUAL

Honeywell

SERIES 600

GECOS

TIME-SHARING SYSTEM GENERAL INFORMATION MANUAL

SUBJECT:

General Description of the Time-Sharing System Including the Command Language, Files, Terminal Usage, and Service Subsystems.

SPECIAL INSTRUCTIONS:

This manual supersedes *GE-600 Line GECOS III Time-Sharing System General Information Manual,* CPB-1643, dated October 1969. Technical changes and additions from the previous edition are indicated by change bars in the margins. This manual includes features implemented in SDL 3.2G.

INCLUDES UPDATE PAGES PUBLISHED AS ADDENDUM NO. 1 (600 SDL 3.3) ON JANUARY 29, 1971.

DATE:

July, 1971

DOCUMENT NUMBER:

CPB-1643A

Preface

This manual is intended to provide information that is common to all users of the Time-Sharing System. As such, it is a guide to other Series 600 Time-Sharing documentation, and contains descriptions of the command language and time-sharing files. This manual includes features implemented in Systems Development Letter 3.2G and 3.3.

The manual is also designed to provide specific information on service/utility subsystems and batch programs, and on terminal usage, some of which will not be pertinent to or identical for all time-sharing terminal installations.

Contents

Page

Page

1. Introduction to the GE-600 Line Time-Sharing System

GENERAL

The GE-600 Line Time-Sharing System operates under the direction of the GE-600 Line Comprehensive Operating Supervisor (GECOS III), and constitutes one dimension of an integrated 3-dimensional information system. Under GECOS III, the three processing dimensions -- batch, remote-batch and time-sharing -- carry on their activities simultaneously, with intercommunication existing between the three processing dimensions. This feature of intercommunication has considerable significance for the user of a time-sharing terminal.

The Time-Sharing System (TSS) consists of a Time-Sharing Executive and a number of independent processing subsystems which operate under the Executive and a common command language. The major subsystems of the Time-Sharing System include the following:

- BASIC -- an algebraic-language compiler/executor, designed for the user with numerical problems involving relatively small amounts of data.

- TSS FORTRAN -- an algebraic-language compiler/loader with extended capabilities for subprogramming, chain overlays, and peripheral I/O, providing full batch-type programming capabilities.

- Text EDITOR and RUNOFF -- a facility for building, maintaining, and reformatting text files of any description.

- ABACUS -- a "desk-calculator" facility featuring sophisticated algebraic capabilities such as functions, summation operator, and remembered variables.

Other subsystems, providing service and utility functions for the Time-Sharing System, are as follows:

- ACCESS -- a file system manipulation subsystem that allows the user to create, delete, and modify file system catalogs, subcatalogs, and named files. The file space, not file content, is manipulated with ACCESS.

- CARDIN -- a facility for submitting of a batch job at a time-sharing terminal for batch-world processing; job status information is available on demand. The SCAN subsystem complements CARDIN with its facilities that scan job output.

- FDUMP -- a remote-terminal, word-oriented file inspection and maintenance facility for permanent files, regardless of their format. The files may have been generated in either batch, remote-batch, or time-sharing environments.

- FORTRAN TRANSLATOR -- permits the user to translate a time-sharing FORTRAN file into batch FORTRAN. The user may design and debug a program in time-sharing FORTRAN and then optimize its processing by converting it to batch FORTRAN.

- HELP -- supplies the user, at his terminal, with a detailed explanation of any system error message.

- JOUT -- provides a means for manipulating output from batch jobs. The batch job could be a CARDIN job with a disposition code of J or JOUT, a remote terminal batch job (GERTS), or a job submitted at the central site.

- Library Editor (LIBED) -- is specifically intended for editing of time-sharing FORTRAN subroutine library files, such files to be subsequently processed by the Library Generator (TSLG) program.

- Library Generator (TSLG) -- permits a user to produce his own library file of time-sharing FORTRAN subroutines, complete with directory, in a form that is acceptable to the time-sharing FORTRAN loader.

- LODX -- a means by which the user can load a subsystem program from a permanent file into the Time-Sharing System for purposes of checkout. Thus, a thorough checkout of user system software can be made before it is integrated into the command structure of the Time-Sharing System.

- Media Conversion Program -- is a batch-world program that may be run either at the central computer site or through a remote/batch terminal. Its purpose is to generate a standard format, time-sharing text file from a suitable card deck, or conversly, to produce a card deck from such a file.

- RBUG -- a conversational debug routine that can be used in conjunction with CARDIN. RBUG has all of the capabilities of the DEBUG routine of the batch world, permitting the user to monitor execution of his program, insert and remove breakpoints, and alter contents of memory locations and registers dynamically -- all in an interactive manner.

- SABT -- retrieves specific locations of the ABRT file for printing at the user's terminal. The file named ABRT must have been created by the user and entered into his Available File Table (AFT). When the system aborts the user's program, the core storage area containing the program is written to the ABRT file.

- SCAN -- provides a means of examining output of a batch job from a time-sharing terminal; the batch job may have been submitted through CARDIN, remote-batch, or as standard-central-site job with its output placed into the file system.

- Terminal Debug Subroutine (TDS) -- permits the user to gain control of a time-sharing subsystem, during checkout, at selected locations within the subsystem. The user may then display and/or patch selected areas, display and/or modify registers, and either return to the subsystem normally or to a specified location within the subsystem.

The primary functions of the time-sharing command language are as follows:

- Initiation of processing within a subsystem (e.g., LIST and RUN commands)
- Storage, retrieval, and purge of permanent files (e.g., SAVE and OLD commands)
- Request for operations on temporary time-sharing files (e.g., NEW and RESEQUENCE commands)
- Request for pertinent operating information (e.g., HELP and STATUS commands)
- Direction of flow of control within the subsystem (e.g., DONE and BYE commands)

A description of the command language is contained in Chapter 2 along with an explanation of time-sharing file usage.

The GE-600 Line Time-Sharing user has not only the "normal" time-sharing facilities at his disposal, such as those described above, but also has access to traditional batch/remote-batch facilities. This capability is provided by a group of functionally-interrelated subsystems called the Terminal/Batch Interface Facility. The time-sharing terminal user has the ability to perform the following operations:

- Access and modify a file of information created in the batch or remote-batch dimension.
- Submit a job, such as a GMAP assembly and execution, to the batch dimension and inspect the output directly from his terminal.
- Establish conversational communication between a batch program and the user's terminal.
- Use an adjacent remote-batch terminal as a high-volume, hard-copy output device, and, indirectly, as a high-volume input device.

The basis for this communication between the several processing dimensions is (1) the GECOS III File System, which provides a common data-base for all users of the system, and (2) the common interface provided by the GECOS III Comprehensive Operating Supervisor. The file system provides automatic storage and retrieval of symbolically-named permanent files on high-capacity storage devices. These files are readily accessible in any processing mode. As a byproduct, the use of physical file volumes, such as card decks and tape reels, actually handled and stored by the user is considerably de-emphasized.

Considerable effort has been made to standardize error messages and comments throughout the Time-Sharing System, and to have error-message explanations immediately available at the terminal. Identical error or exception conditions arising in different GE-supplied subsystems are identified by identical error-message text. Those messages that are not fully self-explanatory are prefixed with a message number enclosed by carets (i.e., <nn>), in almost all cases. This message number relates to a message explanation as given by the HELP subsystem. Upon encountering an error message that he does not fully understand, the user merely need ask for HELP (i.e., select the HELP subsystem) and give the prefixed error-message number when the number is requested. He will then receive further explanation of the error condition and, usually, suggestions as to possible courses of remedial action.

The Time-Sharing System is completely modular and open-ended in that it is explicitly designed to allow user-implemented subsystems, tailored for a specific application, to be added to the GE-supplied subsystems. This implementation of subsystems can be done readily, with no disturbance to the system. Specialized debugging facilities are provided for the checkout of new subsystems simultaneous with normal time-sharing operation.

SCOPE OF MANUAL

This manual is intended to provide the following:

- Information common to all users of the GE-600 Line GECOS III Time-Sharing System, including related documentation.
- Information concerning the general use of types of terminals available to the time-sharing system user.
- Information pertaining to the use of command language and time-sharing files.
- Information pertaining to specific service/utility subsystems and batch programs which are meaningful to the Time-Sharing System user.

Note

The configuration of individual sites may be such that not all the information contained herein is pertinent.

2. Time-Sharing Command Language and File Usage

Operation of time-sharing subsystems is controlled by means of a command language--a set of orders or instructions with which a user requests functions to be performed (e.g., LIST, RUN), manages the flow of control for his session at the terminal (e.g., BYE, DONE), and directs file usage (e.g., OLD, NEW).

The BASIC, TSS FORTRAN, and CARDIN subsystems accept, or recognize, virtually all of time-sharing command language. Other subsystems, Text EDITOR for example, recognize little or no command language (as opposed to subsystem "commands" or conversation). Again, a few commands, (PRINT, for example) are applicable to only one or two subsystems. These variations from the general commonality are summarized in the table "Applicability of Commands By Subsystem," that appears on page 28. When using a particular subsystem, the applicability or non-applicability of a command will usually be self-evident, given an understanding of the nature of the subsystem and the command in question, and of the distinction between "build-mode" and "direct-mode."

Time-sharing commands, strictly speaking, can only be given when a subsystem is in build-mode, a mode in which the subsystem is expecting either file-building input or commands recognized by that subystem. Build-mode is indicated by a system-supplied asterisk at the beginning of each new input line. The alternative mode is direct-mode, in which the subsystem either recognizes specialized subsystem commands (e.g., the EDITOR commands) or conducts a conversational question/answer sequence (as in CARDIN). The RUN command, where applicable, always implies a change to direct-mode. A number of minor subsystems, primarily of the service type, have no build-mode phase and go into direct-mode immediately upon selection. Therefore they recognize no command language.

In the specialized command structures of some subsystems, certain time-sharing commands (e.g., SAVE and DONE) are duplicated, both in syntax and function, at the direct-mode level. These may be loosely considered as time-sharing commands, but are not truely such because they are not recognized in the build-mode of some subsystems; e.g., the Text EDITOR subsystem.

DEFINITIONS

- Line Numbers

  Line numbers are required by the BASIC, TSS FORTRAN and CARDIN subsystems for line sequencing purposes. In the case of BASIC, line numbers are also used as statement numbers. A line number consists of one to eight numeric characters terminated by a nonnumeric character (including a blank).

- Manual Mode

  In manual mode, the user must provide (type) the line numbers for each line.

- Automatic Mode

  In automatic mode, the system provides the line numbers. They are printed as the build-mode request for input (asterisk) is issued. The number is written onto the collector file as a part of the statement.

- New File

  A new file is a temporary file created for the user when he uses the command or response NEW. It is assumed the user will build a file which then may be saved, thus creating an old file. A new file is created by a (destructive) reinitialization of the current file.

- Old File

  An old file is a previously built and saved file which the user selects with the OLD command or response, naming the desired file. The old file is copied onto the current file where it is available to the user for processing or modification.

- Current File

  The current file is a temporary file assigned to the user, on which a new file is built or on which the selected old file is copied. Regardless of the intervening commands or subsystem selections, the current file contains the last NEW or OLD selection, with whatever modifications that may have been entered. The modifications are, therefore, temporary until the file is saved by means of the command SAVE. The original old file, if one existed, will not be altered until a RESAVE command naming the old file is executed.

- Collector File

  The collector file is a temporary file assigned to each user when he logs on. All input which is not a recognizable command is gathered onto this file -- for example, numbered statements. Then, when the file becomes full or a command is typed, depending upon the subsystem, the collector file is merged with the current file and the entire current file is edited and sorted if necessary. For example, when the commands RUN, LIST, or SAVE are encountered in the BASIC subsystem, and data exists in the collector file, it is merged with the current file in sort order. (The collector file is normally transparent to the user.)

- Available File Table

  An available file table (AFT) is provided for each Time-Sharing System user. This table holds a finite number of file names (currently set at 20) which are entered in the AFT when the files are initially accessed (opened). The advantages of the AFT are:

  1. Files requiring passwords or long catalog/file descriptions may be referenced by file name alone, once they have been entered in the table.

  2. Files used repeatedly remain readily available, thus reducing the overhead time and cost of accessing the file each time.

The following commands cause the named permanent files to be placed in the AFT.

RUN	filename(s)
LIST	filename(s)
OLD	filename(s)
SAVE/RESAVE	filename(s)
GET	filename(s)
PRINT	filename(s)
PERM	tempfile, filename

Because the AFT is of finite length, it can become full. If this happens and a command is given which requires a new filename to be placed in the AFT, the command subsystem will print an error message indicating that the AFT is full. At this point, the user must remove any unneeded files from the AFT in order to continue. The STATUS FILES command produces a listing of all of the user's files in the AFT. The REMOVE command can be used to remove specified files from the AFT. The files are not purged or altered in any way; only the name is removed from the AFT and the file is set not-busy.

FILE DESIGNATION

The designation of permanent files in the following discussion of commands is specified in the following formats:

a. filename where the file name only is required.

b. filedescr where the full file description may be used, in any of the following formats:

1. filename

2. filename$password

3. userid/catalog$password...
   /catalog$password/filename$password

If a required password is not given (format 1), the system will explicitly ask for the password.

If a required password is omitted in the string format (format 3), a REQUEST DENIED message will be issued.

If the file was previously opened (e.g., with a GET), only the filename need be given regardless of its full description. If the requested file is not already open, it must emanate directly from the user's master catalog (quick-access type file) in order for formats 1 and 2 to be applicable.

Where desired-permissions and/or alternate-name are applicable, they are specified in the following format:

```
        filedescr,permissions

                  or

        filedescr"altname",permissions
```

where:

permissions may be any one or combination of the following, separated by commas:

READ (or R)

WRITE (or W)

EXECUTE (or E)

APEND (or A)

altname may be a valid file name (one to eight characters), enclosed in double-quote signs.

Where a desired-permissions specification is applicable, a null permissions field implies READ and WRITE permissions; i.e., the default interpretation for desired permissions is R,W.

If a file-segment specification, of the form (i,j) where i and j are line numbers, is given in addition to desired-permissions and/or alternate-name, it must appear last in the specification string; e.g.:

```
        filedescr,permissions(i,j)

                  or

        filedescr"altname",permissions(i,j)
```

Examples:

```
     OLD     FIL1$GOGO,R

     SAVE    /CAT1CAT2$MAYI/FIL0$HERE

     LIST    FILE2$HOHO(1,100)

     PURGE   FIL3$ARIZ;FIL4;FIL5$SUN

     GET     JJONES/DATACAT/BATCHWRLDFIL"INFILE"
```

FILE NAMES, CATALOG NAMES, AND PASSWORDS

File names for time-sharing usage must be eight characters or less in length, and may be composed of alphanumerics, periods, and minus-signs. Catalog names and passwords may be up to 12 characters in length, and composed of the same characters as file names.

If a batch-world file with a name longer than eight characters (12 characters maximum) is to be accessed, it must be given an alternate name (altname) from one to eight characters in length. The renaming is local and temporary. An altname may also be used to temporarily rename one or more of several duplicately-named time-sharing files the user wishes to have accessed concurrently. (Permanent files may be duplicately named so long as they emanate from different catalogs or subcatalogs.)

COMMANDS

Following is a description of the Time-Sharing System commands. Although the command words are spelled out completely in the following descriptions, in general usage those exceeding four characters may be shortened to the first four characters (e.g., RESEQUENCE). Refer to the table Applicability of Commands By Subsystem on Page 28 for applications of the commands to particular subsystems.

- ABC

  Calls the ABACUS subsystem for algebraic-expresswon evaluation.

- ACCESS

  Calls the ACCESS subsystem for time-sharing interface with the file system.

- ASCBCD ascfil;bcdfil

  Under CARDIN, the ASCII time-sharing file specified by ascfil is converted to a standard-system-format BCD file on the permanent file specified by bcdfil, following the question/answer sequence that is initiated by this command if the former file does not contain first-line reformatting information. Both ascfil and bcdfil may be simply a file name or a full file description, as required. The ascfil field may specify also the current file by an asterisk.

- AUTOMATIC

1) AUTOMATIC

Causes the automatic creation of line numbers, by the system, at the point at which the automatic mode is entered (or re-entered), with line numbers initially starting at 010 and incrementing by 10 (or, on re-entry, resuming where the previous automatic numbering left off). These line numbers appear in the terminal copy, and are written in the file, just as though the user had typed them.

2) AUTOMATIC n,m

Causes the automatic creation of line numbers, as above, but starting with line number n and incrementing by m.

3) AUTOMATIC ,m
AUTOMATIC n,

Causes automatic creation of line numbers beginning at 10 and incrementing by m, or beginning at n and incrementing by 10 (on re-entry, the line numbering resumes where it left off).

Normally the line number will be followed by a blank. Any nonblank, nonnumeric character affixed to the end of the command AUTOMATIC will cause the blank to be suppressed. For example: AUTONB or AUTOMATICX.

No commands are recognized while in the automatic mode. The automatic mode is cancelled by giving a carriage return immediately following the issuance of an asterisk and line number by the system.

- BCDASC bcdfil; ascfil

Under CARDIN, the standard-system-format BCD file (permanent) specified by bcdfil is converted to an ASCII time-sharing file on the permanent file specified by ascfil, following the question/answer sequence that is initiated by this command. Both bcdfil and ascfil may be simply a file name or a full file description, as required. The ascfil field may also specify the current file by an asterisk.

- BPUNCH ascfil
- BPRINT ascfil

Under CARDIN, the contents of the ASCII time-sharing file specified by ascfil is converted to BCD and is punched or printed, respectively, at the central computer site, following a question/answer sequence initiated by these commands if the file does not contain first-line reformatting information. These commands allow the user to create hard-copy backup (cards) for his TSS files, and to list long files on a high-speed printer. Ascfil may be simply a file name or a full file description, as required. The ascfil field may also specify the current file by an asterisk.

Since a batch-dimension BMC job is spawned by these commands, the batch $ IDENT-card information is requested by the subsystem.

- BYE

Causes the computation of the user's system-usage charges during the session and disconnection of the terminal.

Depending upon the last-selected subsystem, the AFT may first be scanned for user's temporary files. A message is issued as to the number of temporary files, then the user is queried as to the disposition. Each filename is printed followed by a question mark. The user may respond as follows:

1) carriage return - implies the file is to be released; pass to next file.

2) NONE - implies all of the succeeding files are to be released.

3) SAVE filedescr - specifies that the file is to be saved on the permanent file described by filedescr. (Refer to the PERM command.)

- CATALOG

1) CATALOG

   Lists all catalog and file names which emanate from the user's own master catalog.

2) CATALOG #LIB

   Lists all file names in the library.

3) CATALOG filename

   Prints a list of the attributes of the file specified. The file must emanate from the user's catalog.

4) CATALOG /catalogl/catalog2

   Prints a list of all catalog and file names which emanate from the specified catalog (catalog2 in this case).

5) CATALOG /catalogl/catalog2*

   Prints a detailed list of catalog2's attributes.

Passwords need not be given in these catalog commands. However, CATALOG applies only to strings which originate from the user's (own) master catalog or the library (#LIB).

- DELETE

  1) DELETE a,b,c,d,...

  2) DELETE a-b,c-d

     Lines numbered a through b and c through d are deleted from the current file.

  3) DELETE a,b,c-d,e,f-g,...

     Lines numbered a,b,c through d,e, and f through g are deleted from the current file.

  4) DELETE -n

     Acceptable only as first argument, since it implies deletion of lines from beginning of current file to line n.

  5) DELETE n-

     Acceptable only as last argument, since it implies deletion of lines n through end of current file.

  6) DELETE;*

     Causes deletion of all lines in current file.

- DONE

  Causes exit from the selected subsystem to permit a new subsystem selection.

- EDIT

  Causes the Text EDITOR subsystem to be called into use. Following the READY message, the user may exercise any of the text-editing capabilities available in the Text-EDITOR subsystem. The current file is the recipient of any modification.

- ERASE filedescr 1;filedescr 2;...;filedescr n

  Erases (overwrites) the file space associated with the specified file(s), but does not release the file(s) from the file system. (Refer to PURGE and RELEASE commands.)

- FDUMP

  Calls the FDUMP subsystem for file dumping and correction.

- GET filedescr 1;filedescr 2;...;filedescr n (permissions and altname applicable)

  The permanent file(s) designated by filedescr i will be accessed and the filename(s) placed in the AFT. This is a simple means by which common data files emanating from other user's master catalogs may be opened.

- HELP

  Calls the HELP subsystem, in order to obtain an error message explanation. For example, if the error message

  ```
  009 - SYSTEM UNKNOWN
  ```

  were issued, the user could call HELP and respond to the request

  ```
  PLEASE ENTER MESSAGE NUMBER-
  ```

  with 9, if he desired an error message explanation.

- HOLD

  Prevents any console or master-user issued warning or information message from appearing at the terminal, either in printer or paper-tape output, until a subsequent SEND command is given. The user assumes responsibility for any warnings he may miss while the HOLD is in effect. This command is used primarily during output of listings for display or reproduction purposes. (Refer to the SEND command.)

- JABT snumb (Job Abort)

  Under CARDIN, causes the batch-processing job specified by snumb (and submitted from the same terminal) to be aborted, with an X1 abort-code assigned.

- JDAC name (Job Direct Access)

  Under CARDIN or at the subsystem level, allows a time-sharing terminal user to establish direct access communication (DAC) with a slave program running in the GE-600 system. The DAC is initiated at the subsystem level by

  ```
  SYSTEM? JDAC name
  ```

Under the CARDIN subsystem JDAC is initiated at the command level by

```
SYSTEM? CARDIN
OLD OR NEW-NEW
READY
*JDAC name
```

Name refers to the name of a user supplied DAC slave program (e.g., the GE-600 Time-Sharing System is a DAC slave program). If the program name is not provided in the initial call to JDAC, the system will request a program name. When the direct access program terminates, the return is to the appropriate level (SYSTEM? or build input mode).

- JOUT snumb

  Permits manipulating, from a time-sharing terminal (via a call to JOUT subsystem), the output of certain types of batch jobs.

- JSTS snumb (Job Status)

  Under CARDIN, BASIC, and FORTRAN, causes the current batch-processing status of the job specified by snumb (e.g., 0005T) to be printed at the terminal, in plain text.

- LENGTH

  1) LENGTH

  Generates a report of the content length of the current file, in terms of 320-word blocks.

2) LENGTH filedescr

Generates a report of the type, current size, and content length of the permanent file specified by filedescr. Size and content length are given in units of 320-word blocks.

- LIB filename

File filename from the library becomes the current file.

- LIST

1) LIST

Lists the current file on the terminal.

2) LIST i,j

Lists all lines of the current file whose line numbers are greater than or equal to i and less than or equal to j. In the case of concatenated files where no sort or resequence has been performed, multiple sets of lines numbered between i and j may or may not be listed, if such exist. Either i or j may be omitted. Line numbers 1 or 99999999 respectively will be assumed. If j is omitted, the comma may also be omitted.

3) LIST filedescr (permissions and altname applicable)

Lists the file specified by filedescr on the terminal, without altering the current file. Filedescr must include at least one alpha character if it consists of filename only.

4) LIST filedescr(i,j) 1;...;filedescr(i,j) n
(permissions and altname applicable)

Adjoins and lists the specified files or file-segments on the terminal. The current file is not altered. The current file may be included in the list under the name *. If the list is greater than one line in length, it may be continued on the next line provided the last nonblank character on the first line is a (leading) delimiter.

5) LISTH

Lists the file with a header (date and time) printed at the top of the listing. LIST formats (1), (2), (3), and (4) may all use the LISTH form instead of LIST.

6) LISTEnnn (no intervening blanks allowed)

List the file(s) as specified by the operand; but with all lines to be "broken" or "folded" at the character position (nnn) specified. Listing of the line will be continued on succeeding line(s). If nnn is omitted, the value 72 is assumed. LIST formats (2) through (4) may also use the LISTEnnn form in place of LIST. Files containing overlength lines (records) may be listed in this manner.

7) LISTS n ,n ,n ,...,n

List only the specified line(s) n from the current file.

8) LIST 99999999

If LIST is given with a line number greater than the last line number on the current file, then the last line number of the current file will be printed.

- LUCID

This is used instead of the TAPE command for non-USASCII paper tape input. The input is stored on the time-sharing TAP* file as unaltered eight-bit codes. The TAP* file is left open (unedited in the user's AFT). When a pause greater than one second stops the tape read, the system returns to the subsystem selection (SYSTEM?) level. This command will not function when data communication is via a Low Speed Line Adapter (LSLA) on a DATANET 355. In the EDITOR subsystem, this command takes the form #LUCID.

- NEW

1) A new file (empty current file) is started. (The system will return to the build mode.) The current file is cleared of any prior content.

2) NEWP filedescr (permissions applicable)

The OLD-NEW (OLDN) subsystem will determine if a current file (*SRC) has been defined (opened). If the file is defined, OLDN will deaccess this file. The named file will be created by the NEWP command as a quick-access permanent file. It will have the attributes specified and will be opened with an alternate name of *SRC. If the named file already exists, an error message will be sent to the user. This file will remain the user's current file until another form of the OLD or the NEW command is given.

3) NEWP# filedescr (permission applicable)

Execution is the same as for NEWP except that the created file remains the user's current file until log-off, or until another OLDP, OLDP#, NEWP, or NEWP# command is given. The normal OLD or NEW commands will use this file (i.e. the file specified by OLDP# or NEWP#) as the current file.

- NEWUSER

1) NEWUSER

   Causes the computation of the user's system-usage charges during the session and initiates a new log-on sequence.

2) NEWUSER account number

   Causes the computation of charges for user's previous account number, this account number to be closed, and the new account number specified to replace the old. Accounting data is reinitialized as for a new user but the log-on sequence is bypassed; i.e., the previous user-id and password are assumed.

- OLD

1) OLD filedescr (permissions and altname applicable)

   File filedescr becomes the current file.

2) OLD filedescr(i,j) (permissions and altname applicable)

   Lines i and j of file filedescr become the current file. Filedescr must be a line-numbered file.

3) OLD f(i,j) 1;...;f(i,j) n (permissions and altname applicable)

   The n files or file segments are adjoined in the order listed and become the current file, where f is a filedescr. Adjoining of BASIC files should be done with caution (sequence numbers are also statement numbers). The asterisk designating the contents of the current file (or segment thereof) may appear as a filedescr anywhere in the file list.

   Note that these files or segments are concatenated on the current file and resequencing may be required for satisfactory operation in line-number dependent systems. Sorting or resequencing is not automatic.

4) OLD f(i,j) 1:f(i,j) 2:...:f(i,j) n (permissions and altname applicable)

The n files or file segments are merged together by line numbers, and become the current file, where f is a filedescr (colon-separated). If duplicately numbered statements appear in two or more files, each such statement will appear in the order specified by the file list. The asterisk designating the contents of the current file (or segment thereof) may appear as a filedescr anywhere in the file list.

5) OLD f(i,j) 1;f(i,j) 2:f(i,j) 3;...:f(i,j) n (permissions and altname applicable)

A combination of forms (3) and (4). Concatenation or merging is performed in the order (from left to right) indicated by the file list.

If the file list is too long for one line, the OLD subsystem will request more input if a delimiter is the last non-blank character before the carriage return.

6) OLDP filedescr (permissions applicable)

The OLDN subsystem determines if a current file (*SRC) and/or the file specified in filedescr have been accessed previously. If this file(s) has been accessed, OLDN will deaccess them. The specified permanent file will then be accessed with an alternate name of *SRC and thus becomes the current file. This file will be the user's current file until another form of the OLD or NEW command is given.

7) OLDP# filedescr (permissions applicable)

Execution is the same as for the OLDP command, except that this file remains the user's current file until log-off, or until another OLDP, OLDP#, NEWP, or NEWP# command is given. The normal OLD or NEW commands use this file (i.e. the file specified by OLDP# or NEWP#) as the current file.

NOTE: The OLDN subsystem is called in when the commands OLD, NEW or LIB (normal forms) are given by the user. If a NEWP or OLDP command was issued and then one of the normal forms was typed in, OLDN will deaccess the permanent *SRC file and assign a new temporary *SRC file to the user. The permanent file remains in the user's catalog until he releases it.

If a NEWP# or OLDP# command was issued and then one of the normal forms was typed in, OLDN will retain the permanent file as *SRC. If a NEWP or OLDP was typed in instead of the normal form, the permanent *SRC will be deaccessed, and a new permanent file with the alternate name *SRC will be created and/or accessed.

If a NEWP# or OLDP# command was issued and then followed by another NEWP# or OLDP# command, the OLDN subsystem will deaccess the present *SRC file and then create and/or access the newly specified *SRC file.

- PARITY/NOPARITY

1) PARITY

The data sent from the GE-600 system to a terminal in direct access mode is normally in seven-bit, even parity code. The PARITY command is only used to return to this mode of operation from a NOPARITY mode of operation.

2) When the NOPARITY (NOPA) command is given, all data sent from the GE-600 to a terminal in direct access mode is in eight-bit, parity independent code. This command may be used at the system level or at the command level in BASIC, FORTRAN and CARDIN. The NOPARITY (NOPA) command can only be used with a Type 4 terminal.

- PERM tempfile;filedescr

The temporary file tempfile (created by a user's FORTRAN program) is copied onto the permanent file described by filedescr. If the file does not already exist, it will be created with general read permission. The temporary file name is removed from the AFT and the permanent file accessed (name placed in AFT).

- PRINT

Under CARDIN, print at the terminal all or any part of a source file or concatenation of source files, reformatting the file by use of format-options and/or tab characters, if desired.

1) PRINT

The entire current file will be reformatted and printed.

2) PRINT filedescr(i,j) 1;filedescr(i,j) 2;...;
filedescr(i,j) n

The specified file(s) or file-segment(s) will be adjoined, reformatted, and printed. The current file may be included in the string of files by the name *. The current file, however, will not be affected. If the list is longer than one line in length, it may be continued on the next line provided that the last nonblank character of the line is a leading delimiter.

Following a PRINT command, if the named file does not carry reformatting information, a series of questions are asked of the terminal user. Responses to CARD FORMAT? are:

MOVE - implies line numbers are present and are to be moved to the sequence-number field and printed.

STRIP - implies line numbers are present and are not to be printed.

ASIS - implies line numbers are not present in the file, or that the file is to be printed "as is", except for tab spacing.

NORM - implies MOVE option and the standard tab character and settings:

```
:,8,16,32,73
```

If the response was not NORM, the question TAB CHARACTER AND SETTINGS? is asked. Responses are NORM or a series of tab characters and settings of the form:

$tab_1,setting_{11},setting_{12}...;tab_2,setting_{21},setting_{22}...$

- PURGE filedescr 1;filedescr 2;...;filedescr n

  Releases the specified file(s) from the file system and overwrites the released file space.

  (Refer to RELEASE and ERASE COMMANDS.)

- RECOVER filename $ password (password optional)

  The permanent file designated by filename will be created and/or accessed, and it will become the input collector file emanating from the user's master catalog. The permanent file will be created without general permissions assigned. (The command is #RECOVER when given in the EDITOR subsystem.)

- #RECOVER filename $ password (password optional)

  The permanent file designated by filename will be created and/or accessed, and it becomes the input collector file emanating from the user's master catalog. The permanent file will be created without any general permissions assigned. This command is only applicable to the EDITOR subsystem.

- RELEASE filedescr 1;filedescr 2;...;filedescr n

  Releases the specified file(s) from the file system, but without overwriting the associated file space. (Refer to PURGE and ERASE commands.)

- REMOVE filename 1;filename 2;...;filename n

  Removes the specified file name(s) from the AFT, i.e., deaccesses the named file(s).

- REMOVE CLEARFILES

  Removes all files from the AFT including the current file.

- RESAVE filedescr 1; filedescr 2;...;filedescr n

  The contents of the current file are saved on the previously existing permanent file(s) specified by filedescr i, replacing any prior content thereof. Sorting by line number is or is not done according to subsystem requirements. (Refer to the SAVE command.)

- RESEQUENCE

  1) RESEQUENCE

     The line numbers of the current file are resequenced. The resequencing begins with line number 10 and continues in increments of 10. If BASIC is the selected subsystem, the file is resequenced and statement number references in the program are modified correspondingly (GOTO, GOSUB, IF, ON, Print USING). If FORTRAN or CARDIN was selected, statement number references are not affected.

  2) RESEQUENCE n,m,x-y

     The line numbers of the current file are resequenced and modifications made according to the subsystem selection. The resequencing begins with line number n and continues in increments of m.

     x and y are specified only if partial resequencing is desired. x gives the starting point and y the ending point of resequencing, inclusive. A null x field (i.e., -y) implies "from beginning of file to line y," and a null y field (i.e., x-) implies "from line x to the end of file".

In general, any blanks preceding a line number are stripped off. Unnumbered lines are accepted, except under the BASIC subsystem, and such lines will have line numbers added, as implied or specified in the command. Care should be taken in resequencing concatenated BASIC files as line numbers are also statement numbers, and statement references, after resequencing, may become invalid.

3) RESEX n,m

Line numbers are inserted at the beginning of each and every line in the current file, regardless of whether or not line numbers already exist. The numbering begins with n and increments by m, or optionally, begins with 10 and increments by 10, if n,m are not specified. If the first character of the existing line is a numeric, a blank is inserted following the generated line number. If the first character of the existing line is not numeric, no such blank is inserted.

4) RESE# n,m

Line numbers are inserted at the beginning of each line in the current file, even if line numbers already exist. This numbering begins with n and increments by m, or optionally begins with 10 and increments by 10 if n, m are not specified. If the first character of the existing line is a numeric, a pount sign (#) is inserted following the generated line number. If the first character of the existing line is not numeric, the pound sign is not inserted.

- ROLLBACK filename$password (password optional)

The permanent file designated by filename is accessed with general read and write permissions assigned. This file becomes the input collector file emanating from the user's master catalog. When accessed, the permanent file is read and any data on the file is copied to the current working file, and the last line of good data on that file is printed out at the terminal as follows:

```
LAST LINE OF SAVED DATA IS:
```

followed by the last line of good data (in the EDITOR subsystem, this command is #ROLLBACK).

- #ROLLBACK filename$password (password optional)

  The permanent file designated by filename is accessed with general read and write permissions assigned. This file becomes the input collector file emanating from the user's master catalog. When accessed, the permanent file is read and any data on the file is copied to the current working file, and the last line of good data on that file is printed out at the terminal as follows:

  LAST LINE OF SAVED DATA IS:

  followed by the last line of good data (this command applies only to the EDITOR subsystem).

- RUN

  1) RUN

     Executes the selected subsystem. The source input is the current file. (If BASIC is the subsystem selection and any variation of the RUN command is given, only the current file will be executed; i.e., any information appended after the RUN command is ignored.)

  2) RUN filedescr (permisswons and altname applicable)

     Under FORTRAN, compiles and executes the file specified by filedescr. Under CARDIN, converts and passes the specified file to GEIN.

  3) RUN = filedescr = (option 1,...,option n)
     (permissions and altname applicable if file already exists.)

     Under FORTRAN, compiles and executes the current file using the specified options. Save the object program on the file specified by filedescr. If this file does not already exist, it will be created (with general read permission).

  4) RUN filedescr 1;...;filedescr(i,j) n
     = filedescr x (option 1,...,option n)
     (permissions and altname applicable to already existent files)

     Under FORTRAN, the specified files or file-segments are adjoined and compiled/executed according to the options specified, and the object program saved as file filedescr. The compile options and saving of object file are optional. The designated files may be object or source files. (Object files must be random files.)

The current file may be indicated by an asterisk in the file list. Caution must be exercised to ascertain that the current file contains that which is expected.

If a list is too long to be typed on one line, the subsystem will request more input if a delimiter is the last nonblank character before the carriage return.

5) RUN fs = fh; fc(opt) ulib #fe

Under the Y FORTRAN and JOVIAL subsystems, this command calls in the RUN subsystem to compile and execute, using the parameters and options specified in the command. The format definition is as follows:

fs - set of file descriptors for input to the compiler and/or loader.

fh - single file descriptor pointing to a random file used to save the system loadable file produced by GELOAD if the compilation is successful.

fc - single file descriptor pointing to a sequential file in which the compiler places the binary deck(s) from the compilation(s).

fe - set of file descriptors for files required for execution.

opt - set of options to be used in the compilation/execution.

ulib - sequence of file descriptors pointing to random files containing user libraries.

Refer to Series 600 FORTRAN Y Reference Manual, CPB-1686, for procedures for using this RUN command.

6) RUNH

Executes the selected subsystem and prints a header (date and time) at the top of the program-execution report. RUN formats (1), (2), (3), and (4) may all use the RUNH form in place of RUN.

- SAVE filedescr 1,permissions,size;
  filedescr 2,permissions,size;....filedescr n

The current file is saved on one or more new permanent file(s) defined by filedescr i. Sorting by line number is or is not done, according to subsystem requirements. The file(s) specified will be created with no general permissions or with the permissions specified in the SAVE command. A maximum size can be specified in the command by typing in the word SIZE or the letter S followed by the numeric size value. If no size is specified, the subsystem will determine a maximum size based on the program size.

- SCAN filedescr (permissions and altname applicable)

  Under CARDIN, the SCAN subsystem -- batch-output scanner -- is initiated to scan the file described by filedescr. The desired functions are defined by the question/answer sequence that follows the use of this command.

- SEND

  Cancels the effect of a previous HOLD command, and causes the last message previously withheld to appear at the terminal. (Refer to the HOLD command.)

- STATUS

  1) STATUS

     Lists the user's status as to processor time used, number of file I/O's, and characters output to the terminal; and lists the files that are open.

  2) STATUS FILES

     Lists only the names of the user's open files.

- SYSTEM name

  Exits from the current subsystem and calls the named subsystem, or, if no name is given, returns control to the subsystem-selection level (SYSTEM?). This command permits the user to bypass the normal DONE--SYSTEM? sequence.

- TAPE

  The current file will be built or extended with input from paper tape. Neither line feeds nor rubouts are supplied by the Time-Sharing System. (Yhe command is #TAPE when given in the EDITOR subsystem.)

- #TAPE

  The current file will be built or extended with input from paper tape. Neither line feeds nor rubouts are supplied by the Time-Sharing System. This command is only applicable to the EDITOR subsystem.

Applicability of Commands By Subsystem

Command	BASIC	FORTRAN	CARDIN	EDITOR
ABC	Yes	Yes	Yes	No
ACCESS	Yes	Yes	Yes	Yes*
ASCBCD	No	No	Yes	No
**AUTOMATIC	Yes	Yes	Yes	No
BCDASC	No	No	Yes	No
BPRINT	No	No	Yes	No
BPUNCH	No	No	Yes	No
BYE	Yes	Yes	Yes	Yes*
CATALOG	Yes	Yes	Yes	Yes*
**DELETE	Yes	Yes	Yes	No
DONE	Yes	Yes	Yes	Yes*
EDIT	Yes	Yes	Yes	No
**ERASE	Yes	Yes	Yes	Yes*
FDUMP	No	No	Yes	No
GET	Yes	Yes	Yes	No
HELP	Yes	Yes	Yes	No
HOLD	Yes	Yes	Yes	No
JABT	No	No	Yes	No
JDAC	No	No	Yes	No
JOUT	No	No	Yes	No
JSTS	Yes	Yes	Yes	No
**LENGTH	Yes	Yes	Yes	Yes*
**LIB	Yes	Yes	Yes	No
**LIST	Yes	Yes	Yes	No
**LUCID	Yes	Yes	Yes	No
**#LUCID	No	No	No	Yes
**NEW	Yes	Yes	Yes	No
NEWUSER	Yes	Yes	Yes	No
NOPARITY	Yes	Yes	Yes	No
**OLD	Yes	Yes	Yes	Yes*
PARITY	Yes	Yes	Yes	No
**PERM	No	Yes	No	No
**PRINT	No	No	Yes	No
**PURGE	Yes	Yes	Yes	Yes*
RECOVER	Yes	Yes	Yes	No
**#RECOVER	No	No	No	Yes
**RELEASE	Yes	Yes	Yes	Yes*
REMOVE	Yes	Yes	Yes	Yes*
**RESAVE	Yes	Yes	Yes	Yes*
**RESEQUENCE	Yes	Yes	Yes	No
**ROLLBACK	Yes	Yes	Yes	No
**#ROLLBACK	No	No	No	Yes
**RUN	Yes	Yes	Yes	No
**SAVE	Yes	Yes	Yes	Yes*
SCAN	No	No	Yes	No
SEND	Yes	Yes	Yes	No
STATUS	Yes	Yes	Yes	Yes*
**SYSTEM	Yes	Yes	Yes	No
**TAPE	Yes	Yes	Yes	No
**#TAPE	No	No	No	Yes

* "command" is in direct-mode
** not applicable at subsystem-selection level

3. Time-Sharing Error Messages Explanation

Error messages generated by the various time-sharing subsystems and by the time-sharing Executive fall into two classes, from the viewpoint of explanations:

- Error messages that are considered self-explanatory.
- Error messages that, due to the need for reasonable terseness in conversational messages, may require further explanation for a given user the first few times that the message is encountered.

All messages falling into the second class are prefixed by a message number, usually enclosed by carets (i.e., <nn> , or in some cases <nn<). Further explanation of these messages is immediately available at the terminal through the HELP subsystem. HELP may be called for either at the subsystem-selection level (SYSTEM?) or at the command level under most major subsystems.

HELP message explanations are listed below, indexed under the associated error message(s). These error messages, in turn, fall into two categories from the viewpoint of origin and applicability.

- Error messages originating from the time-sharing Executive, most of which be received only by an implementor of a new, not fully debugged, time-sharing subsystem during its checkout. These messages are numbered between 1 and 49, inclusive.
- Error messages originating from the various time-sharing subsystems, which would be received by a user (as opposed to modifier) of the system, and would indicate faulty usage or, possibly, system malfunction. These user-directed messages are numbered beginning with 50.

Note

On some types of terminals, the carets enclosing the error message number are reproduced as parentheses.

In the description below, generated error messages and their associated HELP subsystem error message explanations are listed in order of their message numbers.

001 - INCORRECT PRIMITIVE

AN ILLEGAL PRIMITIVE HAS OCCURRED IN A COMMAND LIST. CHECK THE COMMAND LIST POINTER IN THE PROGRAM DESCRIPTOR AND THE COMMAND LIST FORMAT AND PRIMITIVES.

002 - BAD FILE I/O COMMAND

IN THE CALLING SEQUENCE OF A DRL FOR FILE I/O, THE COMMAND WORD IS INCORRECT. CHECK THE SUBSYSTEM CODE.

003 - BAD DCW

IN THE CALLING SEQUENCE OF A DRL FOR FILE I/O, A DCW IS INCORRECT. CHECK THE SUBSYSTEM CODE.

004 - location ADDRESS OUT OF RANGE

THE ADDRESS OF A DRL ARGUMENT IS OUTSIDE THE RANGE OF THE PROGRAM. THE NUMBER GIVEN IN THE COMMENT IS THE DERAIL LOCATION. CHECK THE SUBSYSTEM CODE FOR IMPROPER INITIALIZATION.

005 - BAD DRL CODE

THE ADDRESS OF A DRL CODE IS OUT OF THE RANGE OF USABLE CODES OR ILLEGAL FOR THIS SUBSYSTEM. CHECK THE SUBSYSTEM CODE.

006 - LEVEL OF CONTROL TOO DEEP

THE MAXIMUM NUMBER OF CALLS IN THE PROGRAM STACK OR THE CALLSS STACK HAS BEEN EXCEEDED. IN THE CASE OF THE PROGRAM STACK, THIS MEANS THAT THE SELECTED SYSTEMS PRIMITIVE LIST CONTAINED A CALLP, AND IN TURN, THAT SUBSYSTEMS PRIMITIVE LIST CONTAINED A CALLP, ETC. UNTIL THE LENGTH OF THE PROGRAM STACK WAS EXCEEDED. LIKEWISE, IN THE CASE OF THE CALLSS STACK OF SUBSYSTEMS CALLING OTHER SUBSYSTEMS BY MEANS OF THE DRL CALLSS, THE TABLE LIMIT WAS EXCEEDED. REVIEW THE SUBSYSTEM AND DEPTH OF CALLS.

007 - BAD PROG. DESCRIPTION

IN THE PROGRAM DESCRIPTOR, THE POINTER TO THE COMMAND LIST IS ZERO OR POINTS TO NON-COMMAND LANGUAGE DATA. CHECK THE PROGRAM DESCRIPTOR AND COMMAND LANGUAGE LIST.

008 - LOOP IN PRIMITIVES

A NUMBER OF THE PRIMITIVES ARE EXECUTED ENTIRELY WITHIN THE TSS SCAN MODULE. A COUNTER IS INITIALIZED AT THE ENTRY TO SCAN AND A COUNT KEPT OF PRIMITIVES EXECUTED. WHEN THE COUNT EXCEEDS A GIVEN MAXIMUM, IT BECOMES OBVIOUS THERE IS A LOOP. CHECK THE SEQUENCE OF THE PRIMITIVES FOR THE SUBSYSTEM.

009 - SYSTEM UNKNOWN

THE REQUESTED SUBSYSTEM IS UNKNOWN TO TSS OR IS NOT INCLUDED IN THE SYSTEM FOR THIS INSTALLATION. CHECK THE NAME FOR SPELLING TOO.

010 - PROGRAM TOO LARGE TO SWAP

A SUBSYSTEM IS SO LARGE THAT THE NUMBER OF DCW'S REQUIRED TO LOAD OR SWAP THE PROGRAM EXCEED THE MAXIMUM NUMBER OF DCW'S WHICH CAN BE BUILT. CHECK THE SIZE OF THE SUBSYSTEM. PERHAPS THE SUBSYSTEM EXPANDS ITS CORE LIMITS WITH A DRL ADDMEM. CHECK ALL DRL ADDMEM REQUESTS. SEE .LADCW DEFINED IN COMMUNICATION REGION FOR MAXIMUM NUMBER OF DCW'S ALLOWED.

011 - INCORRECT CORE FILE USAGE

A REQUEST TO MOVE CORE FILE SPECIFIES MORE THAN TEN WORDS TO BE MOVED. CHECK ALL DRL CORFIL REQUESTS.

012 - PROGRAM NOT ALLOWED USE OF THIS I/O

PRIVILEGED FILE I/O IS RESERVED FOR SUBSYSTEMS WHICH SPECIFICALLY REQUIRE INFORMATION FROM FILES ALLOCATED TO THE TIME-SHARING SYSTEM. PLEASE REVIEW THE NEED FOR PRIVILEGED FILE I/O AND JUSTIFY IT WITH THE COMPUTING CENTER.

013 - DRL ALLOWED ONLY BY LOGON

THE DRL USER ID CAN BE USED ONLY BY THE LOGON SUBSYSTEM. CHECK THE SUBSYSTEM CODE.

014 - NOT CURRENTLY ASSIGNED

015 - CANNOT RESET ID

THE LOGON SUBSYSTEM IS EXECUTING A DRL USER ID, BUT THE ID OF THE SPECIFIED U.S.T. IS NON-ZERO. A TERMINATE MUST BE EXECUTED FOR THAT USER BEFORE THE U.S.T. CAN BE REUSED. TRY TO DETERMINE WHY THE TERMINATE WAS BYPASSED, OR WHY NEW SYSTEM WAS SELECTED AFTER LOGON.

016 - location OVERFLOW FAULT

THE SUBSYSTEM IN EXECUTION ENCOUNTERED AN OVERFLOW CONDITION AT THE DESIGNATED LOCATION AND THE SUBSYSTEM DID NOT SPECIFY A FAULT VECTOR. THE LOCATION IS RELATIVE TO ZERO (SEE EDIT MAP) UNLESS IT IS A MASTER SUBSYSTEM. THEN THE LOCATION IS RELATIVE TO TSS ZERO, AND ONE MUST DETERMINE THE LOAD ADDRESS OF THE SUBSYSTEM TO DETERMINE THE FAULT LOCATION IN THE MASTER SUBSYSTEM. REVIEW YOUR PROGRAM INPUT FOR INCORRECT DATA BEFORE REQUESTING HELP FROM THE COMPUTING CENTER.

017 - location ILLEGAL OP CODE

THE SUBSYSTEM IN EXECUTION ENCOUNTERED AN ILLEGAL (OR ZERO) OP CODE OR A MME OPERATION AT THE DESIGNATED LOCATION, AND THE SUBSYSTEM DID NOT SPECIFY A FAULT VECTOR.

THE LOCATION IS RELATIVE TO SUBSYSTEM ZERO (SEE EDIT MAP) UNLESS IT IS A MASTER SUBSYSTEM, THEN THE LOCATION IS RELATIVE TO TSS ZERO. ONE MUST DETERMINE THE LOAD ADDRESS OF THE SUBSYSTEM TO DETERMINE THE FAULT LOCATION IN THE MASTER SUBSYSTEM.

REVIEW YOUR PROGRAM CODE AND INPUT FOR INCORRECT DATA BEFORE REQUESTING HELP FROM COMPUTING CENTER.

018 - location MEMORY FAULT

THE SUBSYSTEM IN EXECUTION ENCOUNTERED A MEMORY FAULT AT THE DESIGNATED LOCATION, AND THE SUBSYSTEM DID NOT SPECIFY A FAULT VECTOR.

THE LOCATION IS RELATIVE TO SUBSYSTEM ZERO (SEE EDIT MAP) UNLESS IT IS A MASTER SUBSYSTEM, THEN THE LOCATION IS RELATIVE TO TSS ZERO. ONE MUST DETERMINE THE LOAD ADDRESS OF THE SUBSYSTEM TO DETERMINE THE FAULT LOCATION IN THE MASTER SUBSYSTEM.

REVIEW THE PROGRAM CODE AND INITIALIZATION OF ADDRESS OR INDEX REGISTERS AS WELL AS THE PROGRAM INPUT FOR INCORRECT DATA BEFORE REQUESTING HELP FROM THE COMPUTING CENTER.

019 - location FAULT TAG FAULT

THE SUBSYSTEM IN EXECUTION ENCOUNTERED A FAULT TAG FAULT AT THE DESIGNATED LOCATION, AND THE SUBSYSTEM DID NOT SPECIFY A FAULT VECTOR.

THE LOCATION IS RELATIVE TO SUBSYSTEM ZERO (SEE EDIT MAP) UNLESS IT IS A MASTER SUBSYSTEM, THEN THE LOCATION IS RELATIVE TO TSS ZERO. ONE MUST DETERMINE THE LOAD ADDRESS OF THE SUBSYSTEM TO DETERMINE THE FAULT LOCATION IN THE MASTER SUBSYSTEM.

REVIEW THE PROGRAM CODE AND INITIALIZATION OF ADDRESS OR INDEX REGISTERS AS WELL AS THE PROGRAM INPUT FOR INCORRECT DATA BEFORE REQUESTING HELP FROM THE COMPUTING CENTER.

020 - location DIVIDE CHECK FAULT

THE SUBSYSTEM IN EXECUTION ENCOUNTERED A DIVIDE CHECK FAULT AT THE DESIGNATED LOCATION, AND THE SUBSYSTEM DID NOT SPECIFY A FAULT VECTOR.

THE LOCATION IS RELATIVE TO SUBSYSTEM ZERO (SEE EDIT MAP) UNLESS IT IS A MASTER SUBSYSTEM, THEN THE LOCATION IS RELATIVE TO TSS ZERO. ONE MUST DETERMINE THE LOAD ADDRESS OF THE SUBSYSTEM TO DETERMINE THE FAULT LOCATION IN THE MASTER SUBSYSTEM.

REVIEW YOUR PROGRAM INPUT FOR INCORRECT DATA BEFORE REQUESTING HELP FROM THE COMPUTING CENTER.

021 - BAD STATUS SWAP OUT #S

A BAD I/O STATUS HAS BEEN RECEIVED ON A WRITE DRUM FILE #S, THE SWAP FILE. TRY AGAIN. IF PROBLEM PERSISTS, THE TSS WILL ALERT OPERATIONS. THE PARENTHESIZED NUMBER IS THE STATUS CODE.

022 - BAD STATUS SWAP IN #S

A BAD I/O STATUS HAS BEEN RECEIVED ON A READ DRUM FILE #S, THE SWAP FILE. TRY AGAIN. IF PROBLEM PERSISTS, THE TSS WILL ALERT OPERATIONS. THE PARENTHESIZED NUMBER IS THE STATUS CODE.

023 - BAD STATUS LOAD #P

A BAD I/O STATUS HAS BEEN RECEIVED ON A READ DRUM FILE #P, THE TSS FILE. TRY AGAIN. IF PROBLEM PERSISTS, THE TSS WILL ALERT OPERATIONS. THE PARENTHESIZED NUMBER IS THE STATUS CODE.

024 - BIT POSITION > 35

THE DESIGNATED BIT POSITION IN AN IF TRUE OR IF FALSE PRIMITIVE IS GREATER THAN 35. CHECK THE COMMAND LIST AND PRIMITIVES OF THE SUBSYSTEM.

ERROR-CODE 25 NOT CURRENTLY ASSIGNED.

ERROR-CODE 26 NOT CURRENTLY ASSIGNED.

ERROR-CODE 27 NOT CURRENTLY ASSIGNED.

028 - USER TRIED TO SPACE A RANDOM FILE

A RANDOM FILE CANNOT BE SPACED IN THIS MANNER. USAGE OF THE RANDOM FILE IN THE CORRECT MANNER WILL CLEAR UP THE PROBLEM.

029 - ILLEGAL SYSTEM SELECTION

SOME SYSTEMS, NAMELY THE MASTER SUBSYSTEMS, HAVE RESTRICTED THEIR AVAILABILITY TO CERTAIN USERS. YOU DO NOT HAVE PERMISSION TO USE THE SELECTED SUBSYSTEM. SELECT ANOTHER.

ERROR CODES 30-49 NOT CURRENTLY ASSIGNED.

<50> FILE filename -- reason text

<50< FILE filename -- reason text

(The two messages above refer to permanent files.)

<50> CURRENT FILE -- reason text

<50> COLLECTOR FILE -- reason text

(The two messages above refer to the temporary files *SRC and SY**, respectively.)

<50> WORK FILE -- reason-text

(The message above refers to all other temporary files.)

The possible reason texts for all the above messages are as follows:

```
STATUS 01
I/O ERROR, STATUS 02
NO PERMISSION, STATUS 03
FILE BUSY, STATUS 04
NONEXISTENT FILE, STATUS 05
STATUS 06
DEVICE TYPE UNDEFINED, STATUS 07
STATUS 10
NON-UNIQUE NAME, STATUS 11
MAX. SIZE ERROR, STATUS 12
NO FILE SPACE, STATUS 13
INVALID PASSWORD, STATUS 14
STATUS 15 THRU 33, AND 35
ILLEGAL CHAR., STATUS 34
FILE TABLE FULL, STATUS 36
DUPLICATE NAME, STATUS 37
SYSTEM LOADED, STATUS 40
```

ERROR-MESSAGE 50 EXPLANATION: FILE-SYSTEM ERRORS.

THIS MESSAGE IS ISSUED FOR EITHER ONE OF TWO CASES: (1) THE NAMED PERMANENT FILE COULD NOT BE ACCESSED--<50>, OR COULD NOT BE CREATED--<50<OR (2) A REQUIRED TEMPORARY FILE COULD NOT BE OBTAINED OR EXPANDED. THE REASON GIVEN IN THE MESSAGE IS FURTHER EXPLAINED BELOW:

STATUS 01: THE SPECIFIED USER'S-MASTER-CATALOG DOES NOT EXIST. CHECK USER-ID.

I/O ERROR, STATUS 02: THE FILE SYSTEM HAS ENCOUNTERED AN UNRECOVERABLE INTERNAL I/O ERROR. (THIS DOES NOT IMPLY AN ERROR ON YOUR FILE SPACE.) REPORT THE STATUS TO THE CENTRAL COMPUTER SITE. ALSO RETRY.

NO PERMISSION, STATUS 03: THE NAMED FILE COULD NOT BE ACCESSED BECAUSE YOU HAVE NOT BEEN ALLOWED THE PERMISSION(S) REQUESTED.

FILE BUSY, STATUS 04: ANOTHER USER HAS ALREADY ACCESSED THIS FILE WITH AN ACCESS-MODE PERMISSION THAT LOGICALLY EXCLUDES YOUR REQUESTED PERMISSION; I.E., A GRANTED WRITE PERMISSION EXCLUDES ANY OTHER CONCURRENT ACCESSES AND A GRANTED READ PERMISSION EXCLUDES ANY OTHER ACCESS WITH WRITE PERMISSION. THE FILE, THEREFORE, IS TEMPORARILY BUSY TO SOME OR ALL OTHER USERS. (MULTIPLE CONCURRENT ACCESSES OF A FILE WITH READ PERMISSION, ONLY, IS ALLOWED.)

NONEXISTENT FILE, STATUS 05: EITHER THE NAMED FILE DOES NOT EXIST, AT THE CATALOG LEVEL IMPLIED OR SPECIFIED, OR ONE OR MORE NAMES IN THE CATALOG/FILE DESCRIPTION WAS INCORRECTLY GIVEN. CHECK ALL CATALOG/FILE NAMES. THE COMMAND CATALOG MAY BE USED TO LIST ALL OF YOUR CATALOG AND FILE NAMES.

STATUS 06: THE FILE SYSTEM HAS EXHAUSTED ITS SPACE FOR NEW CATALOGS AND FILE DESCRIPTORS. REPORT THE STATUS TO THE CENTRAL COMPUTER SITE, AND TRY AGAIN LATER.

DEVICE TYPE UNDEFINED, STATUS 07: THE DEVICE TYPE THAT YOU SPECIFIED FOR YOUR FILE IS UNDEFINED TO THE SYSTEM.

STATUS 10: THE SYSTEM HAS TEMPORARILY EXHAUSTED THE AVAILABLE FILE SPACE. TRY AGAIN LATER. (ALSO, PURGE ANY UNNEEDED FILES.)

NON-UNIQUE NAME, STATUS 11: THE NEW NAME THAT YOU HAVE SPECIFIED FOR THE CATALOG OR FILE TO BE MODIFIED IS A DUPLICATE OF A CATALOG OR FILE NAME EXISTING AT THE SAME LEVEL.

MAX. SIZE ERROR, STATUS 12: THE NEW MAXIMUM-SIZE SPECIFIED FOR THE FILE TO BE MODIFIED IS LESS THAN ITS CURRENT SIZE. (MAXIMUM SIZE UNCHANGED.)

NO FILE SPACE, STATUS 13: YOU HAVE USED UP ALL THE PHYSICAL SPACE ALLOTTED TO YOU FOR THE CREATION OF FILES. YOU MUST EITHER PURGE ONE OR MORE UNNEEDED FILES, OR OBTAIN A LARGER FILE-SPACE ALLOCATION.

INVALID PASSWORD, STATUS 14: A REQUIRED PASSWORD EITHER HAS BEEN GIVEN INCORRECTLY OR NOT AT ALL. THE GENERAL FORM FOR SUPPLYING PASSWORDS IN A CATALOG/FILE DESCRIPTION IS: NAME$PASSWORD E.G.: /CAT1$ABC/FIL1$XYZ.

STATUS 15 THRU 33, AND 35: SYSTEM MALFUNCTION. REPORT THE STATUS TO THE CENTRAL COMPUTER SITE, AND RETRY.

ILLEGAL CHAR., STATUS 34: YOU HAVE GIVEN A CATALOG OR FILE NAME, OR A PASSWORD, CONTAINING A CHARACTER OTHER THAN AN ALPHANUMERIC, PERIOD, OR A DASH, WHICH ARE THE ONLY LEGAL CHARS. FOR IDENTIFIERS.

FILE TABLE FULL, STATUS 36: THE NAMED FILE CANNOT BE ACCESSED BECAUSE YOU PRESENTLY HAVE TOO MANY FILES ALREADY ACCESSED (I.E., OPENED). YOU MUST DEACCESS ONE OR MORE OF THESE OPENED FILES. USE THE COMMANDS STATUS FILES, AND REMOVE.

DUPLICATE NAME, STATUS 37: THE FILE NAME SHOWN DUPLICATES A NAME ALREADY IN YOUR AVAILABLE-FILE-TABLE, I.E., AN ALREADY ACCESSED FILE. IF APPROPRIATE, ASSIGN AN ALTERNATE NAME.

SYSTEM LOADED, STATUS 40: THE SYSTEM IS CURRENTLY AT PEAK CAPACITY IN SOME RESPECT, E.G.: CERTAIN INTERNAL TABLE SPACE EXHAUSTED, ETC.

<51> FILE filename -- I/O STATUS yy

<51< FILE filename -- I/O STATUS yy

(The two messages above refer to permanent files.)

<51> CURRENT FILE -- I/O STATUS yy

<51< CURRENT FILE -- I/O STATUS yy

(The two messages above refer to the *SRC file.)

<51> COLLECTOR FILE -- I/O STATUS yy

<51< COLLECTOR FILE -- I/O STATUS yy

(The two messages above refer to the SY** file.)

<51> WORK FILE -- I/O STATUS yy

<51< WORK FILE -- I/O STATUS yy

(The two messages above refer to all other temporary files.)

where yy is the major hardware status returned by IOS.

ERROR-MESSAGE 51 EXPLANATION: INPUT/OUTPUT ERRORS

AN UNRECOVERABLE READ OR WRITE ERROR HAS OCCURRED ON THE SPECIFIED FILE. AN ERROR IN READING IS INDICATED BY THE MESSAGE NUMBER GIVEN AS <51>; AN ERROR IN WRITING AS <51<. REPORT THE I/O STATUS NUMBER AND THE READ OR WRITE INDICATION TO THE CENTRAL COMPUTER SITE. ALSO, IN THE CASE OF "CURRENT FILE" or "WORK FILE", LOG OFF AND TRY AGAIN.

<52> CURRENT FILE NOT DEFINED

ERROR-MESSAGE 52 EXPLANATION

THERE IS NO CURRENT (*SRC) FILE DEFINED IN YOUR FILE TABLE. THIS INDICATES EITHER A SYSTEM MALFUNCTION, OR THAT YOU ARRIVED AT THE PRESENT SUBSYSTEM VIA AN ABNORMAL PATH. SUGGEST YOU RESELECT YOUR DESIRED SUBSYSTEM, OR LOG OFF AND RETRY FROM SCRATCH.

<53> LINES IGNORED BY EDIT
....line(s)......

ERROR-MESSAGE 53 EXPLANATION

THE LINE(S) SHOWN WERE NOT MERGED INTO YOUR CURRENT FILE BECAUSE THEY LACKED LINE NUMBERS.

<54> SYSTEM MALFUNCTION--CURRENT FILE ERROR

ERROR-MESSAGE 54 EXPLANATION

THE FORMAT OF YOUR CURRENT FILE WAS FOUND TO BE IN ERROR. REPORT CIRCUMSTANCES TO THE CENTRAL COMPUTER SITE. SUGGEST THAT YOU LOG OFF AND RETRY.

<55> CURRENT FILE TOO LARGE

ERROR-MESSAGE 55 EXPLANATION

THE COMBINED SIZE OF YOUR SOURCE FILE AND MOST RECENT MODIFICATION- OR ADDITION-INPUT IS TOO LARGE TO BE PROCESSED. SUGGEST THAT YOU SPLIT THE TEXT INTO TWO OR MORE FILES, WHICH CAN LATER BE ADJOINED.

<56> FORTRAN LOADER CODE = nn

ERROR-MESSAGE 56 EXPLANATION

YOUR OBJECT PROGRAM COULD NOT BE PROPERLY LOADED/EXECUTED, FOR THE REASON INDICATED BY THE CODE NUMBER:

0-31 - THESE CODES, GIVEN IN DECIMAL, CORRESPOND TO THE OCTAL STATUS CODES IN HELP MSG. 50
32 - YOUR SAVE-FILE IS NOT LARGE ENOUGH TO CONTAIN THE WHOLE OBJECT PROG.
33 - BINARY PROGRAM BEING SAVED IS NOT IN PROPER FORMAT. ONE OR MORE OBJECT RTNS IN THE PROGRAM WERE PROBABLY CREATED BY SOME MEANS OTHER THAN TSS-FORTRAN (E.G. GMAP). SYMREFS/SYMDEFS ARE NOT COMPATIBLE FOR LINKING WITH TSS-FORTRAN-PRODUCED CODE
34 - FILE BEING LOADED CONTAINS TOO MANY SUBRTNS. (PRESENT LIMIT IS 64)
35 - FILE BEING LOADED IS NOT IN ABSOLUTE FORMAT. WAS PROB. CREATED BY SOME MEANS OTHER THAN TSS-FORTRAN
36 - CKSUM ERROR IN DATA BLOCK(S) OF FILE BEING LOADED
37 - CKSUM ERROR IN CONTROL BLOCK(S) OF FILE BEING LOADED
38 - NOT USED
39 - USER'S ALLOTTED FILE SPACE HAS BEEN EXHAUSTED
40 - BLOCK-COUNT ERROR WHILE LOADING FILE. DATA BLOCKS ARE OUT OF ORDER OR HAVE BEEN DESTROYED
41 - A FILE REQUESTED FOR LOADING CANNOT BE FOUND AS DESCRIBED
42 - ILLEGAL FORMAT FOR USER LIBRARY FILE

057 - RESTRICTED SUBSYSTEM

THE CENTRAL COMPUTER SITE HAS RESTRICTED THE USE OF THIS SYSTEM. THIS MAY BE A TEMPORARY RESTRICTION BECAUSE OF CURRENT LOAD OR A PERMANENT RESTRICTION. PLEASE NOTIFY THE CENTRAL COMPUTER SITE FOR FURTHER DETAILS.

<58> ENTRY LOC < 100

ERROR-MESSAGE 58 EXPLANATION

THE SUBSYSTEM PROGRAM TO BE EXECUTED DOES NOT HAVE THE INITIAL 100-WORD DATA AREA THAT IS REQUIRED OF TSS SUBSYSTEM PROGRAMS.

<59> FILE filename NOT IN TSS FORMAT

ERROR-MESSAGE 59 EXPLANATION

A FORMAT ERROR WAS DETECTED ON THE NAMED FILE. EITHER THE FILE IS NOT A TSS-GENERATED FILE, OR A SYSTEM MALFUNCTION HAS OCCURRED. IN THE LATTER CASE, REPORT THE CIRCUMSTANCES TO THE CENTRAL COMPUTER SITE, AND RETRY THE COMMAND.

<60> NO DATA ON FILE filename

ERROR-MESSAGE 60 EXPLANATION

THE REQUESTED FILE CONTAINS NO USER'S DATA; THE IMPLICATION IS THAT NO DATA HAS BEEN SAVED ON THIS FILE SINCE ITS CREATION.

<61> LAST SAVE/PURGE COMMAND NOT PROCESSED

ERROR-MESSAGE 61 EXPLANATION

YOUR LAST SAVE, RESAVE, PURGE, OR RUN COMMAND HAS BEEN LOST, PROBABLY DUE TO AN INTERVENING EXCEPTION MESSAGE. YOU MAY REISSUE THE COMMAND.

<62> SAVE FILE filename TRUNCATED. NOT BIG ENOUGH

ERROR-MESSAGE 62 EXPLANATION

THE FILE NAMED IN YOUR RESAVE COMMAND IS NOT LARGE ENOUGH (MAX. SIZE) TO CONTAIN ALL OF THE CONTENTS OF THE CURRENT FILE. (SOME OF THE CURRENT FILE HAS BEEN SAVED.) SUGGEST YOU USE THE SAVE COMMAND WITH A 'NEW' FILE NAME, OR MODIFY THE MAXIMUM SIZE OF THE NAMED FILE WITH ACCESS.

<63> TSS FORTRAN CHAIN ERROR, CODE = nn

ERROR-MESSAGE 63 EXPLANATION

YOUR CHAIN LINK COULD NOT BE PROPERLY LOADED. THE REASON CODE GIVEN, IN DECIMAL (1-32), CORRESPONDS TO THE OCTAL STATUS CODES IN HELP MESSAGE 50.

064 - EXECUTE TIME LIMIT EXCEEDED

THE PROGRAM TIME LIMIT SPECIFIED BY THE USER AND/OR THE INSTALLATION HAS BEEN EXCEEDED BY THE OBJECT PROGRAM.

065 - OBJECT PROGRAM SIZE LIMIT EXCEEDED

THE SIZE OF THE OBJECT PROGRAM HAS EXCEEDED THE INSTALLATION SPECIFIED LIMIT.

4. Terminal Usage

GENERAL

This chapter contains general descriptions of and operational procedures for several remote terminals. For complete details pertaining to a particular terminal, the user should refer to the instruction manual accompanying the terminal unit.

TELETYPEWRITER AND TYPEWRITER TERMINAL OPERATION

Remote terminals for communication with the GE-600 Line Time-Sharing System are usually teletypewriters and typewriter-like devices. The latter have standard typewriter mechanisms and can be used "off-line" as such, but are specially adapted for telecommunications. Examples of teletypewriter terminals are the Teletype units, models 33, 35, and 37; the DATEL and IBM 2741 terminal units are examples of typewriter-like terminals.

Operation is essentially similar with either of these kinds of terminals. All of these terminals transmit or receive one complete line of data at a time. This line of data may range in length from "no" characters -- a carriage return only -- to 72 characters. On input (i.e., entry of data by the user) use of the carriage return always terminates the line of data and initiates transmission of that line to the Time-Sharing System.

The teletypewriter terminal may be equipped with a paper-tape reader and punch unit. In this case, the user may employ the paper-tape unit as an auxiliary means of input and/or output. Any information transmitted to or from the terminal is always displayed on the terminal-printer.

Keyboard input is used herein for purposes of illustrating terminal operation, but instructions for the use of paper-tape are also given.

Correction of Typing Errors

Keyboard input is sent to the computer and (effectively) written onto the user's current file in units of complete lines. A line of terminal input is terminated by a carriage return, and no part of the line is transmitted to the system until that carriage return is given. Therefore, corrections to a line-in-progress (i.e., a partial line not yet terminated) can be made.

Correction of a typing error that is perceived by the user before the line is terminated can be done in one of two ways. He may delete one or more characters from the end of the partial line or he may cancel the incomplete line and start over. Character or line deletions are effected by means of two special characters designated as control characters, which differ between terminals.

For teletypewriter terminals

character	control function
(commercial "at" sign)	character deletion
CTRL plus X keys	line deletion

For typewriter-type terminals

character	control function
(degree symbol or])*	character deletion
± (for 2741)	line deletion
± (for DATEL)	line deletion
± plus carriage return	line deletion

* Depends on model of the terminal.

NOTE: Line deletion does not occur until a carriage return is given or ATTN (2741) or INT (DATEL) is pressed.

The usage rules are as follows:

- Use of the character-delete control deletes from the line the character preceding the deletion character; use of n consecutive deletion characters deletes n preceding characters (including blanks).

  For example:

  *ABCDF@E would result in ABCDE being transmitted to the program file.

  *ABCb̸DTF@@@@DEF would result in ABCDEF being transmitted.

  (The characters to be deleted are underlined for illustration.)

- Use of the line-delete control causes all of a line to be deleted. The characters DEL are printed to indicate deletion. For example:

```
*ACDEFG CTRL/X  DEL       (all characters deleted;
                           carriage return automatic)

                         -(ready for new input)

                    or

*ACDEFG ± (carriage return)

DEL                       (all characters deleted)

                         -(ready for new input)
```

Whereas on the teletypewriter the carriage return is automatic, it must be given by the user on the typewriter-like terminal if ATTN or INT is not used.

The control-character pair for each type of terminal cannot be used for other than the deletion function assigned them.

Log-On Procedure

To initiate communication with the Time-Sharing System, the user performs the following steps:

- turns on the terminal
- obtains a dial-tone on the associated phone-set
- dials one of the numbers of his time-sharing center

The user will then receive either a busy signal to indicate that the line is not presently available or a high-pitched tone -- a "beep" -- to indicate that his terminal has been connected to the computer.

Once the user's terminal has been connected to the computer, the Time-Sharing System begins a log-on procedure. Initially, the following message is transmitted:

```
THIS IS THE GE-600 TSS SYSTEM ON date AT time CHANNEL nnnn
```

where time is given in hours and thousandths of hours (hh.hhh), and nnnn is the user's channel number.

Following this message, the system asks for the user's identification:

```
       USER ID --
```

The user responds, on the same line, with the user-id that has been assigned to him by the time-sharing installation management. This user-ID uniquely identifies a particular user already known to the system, for the purposes of locating his programs and files, and accounting for his usage of the time-sharing resources allocated to him. An example request and response might be:

```
       USER ID -- J.P.JONES
```

Note

User's responses are underlined for illustrative purposes.

A carriage return must be given following any complete response, command, or line of information typed by the user. If a charge number is also required for accounting purposes, the user can supply it as follows:

```
       USER ID -- J.P.JONES;1234567E
```

The charge number may consist of from 1 to 12 alphanumeric characters, separated from the user-id by a semicolon. (Refer to NEWUSER command description in Chapter 3.)

After the user responds with his user-ID, the system asks for the sign-on password that was assigned to him along with his user-ID as follows:

```
       PASSWORD
   ZXSEDFREDCFE
```

The user types his password directly on the "strikeover" mask provided below the request PASSWORD. The password is used by the system as a check on the legitimacy of the named user. The "strike-over" mask insures that the password, when typed, cannot be read by another person. (In the event that either the user-ID or password is given incorrectly two consecutive times, the user's terminal is immediately disconnected from the system.)

At this point, if the accumulated charges for the user's past time-sharing usage equals or exceeds one hundred percent of his current resource allocation, he will receive a warning message:

```
        RESOURCES OVERDRAWN n%
```

If his accumulated charges exceeds one hundred and ten percent of his current resources, he receives the message:

```
        RESOURCES EXHAUSTED - CANNOT ACCEPT YOU
```

and his terminal is immediately disconnected. If the user's file space is greater than 88 percent used, he will receive the following information message:

```
       n BLOCKS FILE SPACE AVAILABLE
```

This specifies the number of 320-word blocks of unused file space for this user.

This condition does not affect the log-on procedure; the user is permitted to continue.

Assuming that the user has responded with a legitimate user-ID and password, and has not overextended his resources, the Time-Sharing System then asks the user to select the subsystem that he wants to work with; this is called the subsystem-selection request:

```
        SYSTEM?
```

Major subsystems -- i.e., BASIC, FORTRAN, EDITOR, and CARDIN -- all provide a file-building facility, called "build-mode". In build-mode, the user is free to enter either file-building input or control commands, as he chooses. All other subsystems begin directly with a question/response dialogue between the subsystem and the user that is unique for each subsystem.

For purposes of illustration, the assumption is made that the user decides to work with the BASIC subsystem (the subsequent log-on sequence is representative for any of the major subsystems):

```
        SYSTEM? BASIC
```

A carriage return terminating each separate user response is assumed to be understood. Having selected the BASIC subsystem, the user is asked whether he now wants to start a new file -- i.e., a new program in the case of BASIC -- or wants to retrieve and work with a previously entered and saved file or program. The request message is

```
        OLD OR NEW -
```

If the user wishes to start a new program (i.e., build a new source file), he responds simply with

```
        NEW
```

If, on the other hand, he wants to recall an old source file, he responds with

```
        OLD filename
```

where filename is the name of the file on which the old program was saved during a previous session at the terminal (Refer to the SAVE and RESAVE command descriptions in Chapter 3).

Following either response, the subsystem types the message READY, returns the carriage and prints an asterisk in the first character position of the next line:

```
        READY
        *
```

An example of a complete log-on procedure, up to the point where the BASIC subsystem is ready to accept file-building, correction input, or control commands, might be as follows:

```
THIS IS THE GE-600 TSS SYSTEM ON 07/26/69 AT 14.768 CHANNEL 0012

USER ID -- J.P. JONES
PASSWORD
ABCDEF$GH$KLM
SYSTEM?  BASIC
OLD OR NEW - NEW (NEW is shown arbitrarily for illustration)
READY
*             - (the user begins entering input on this line)
```

Entering Build-Mode Input

After the message READY, the subsystem is in build-mode (as indicated by the initial asterisk) and is ready to accept program statement or text input, depending upon the selected subsystem, and/or command language. All lines of input other than commands are accumulated on the user's current file, which normally will be the file that contains the program or text he wants to work with at this session. If he is building a new file (NEW response to OLD OR NEW--), his current file will initially be empty.

If he has recalled an old file (OLD filename) the content of the named old file will initially be on his current file and any input typed by the user -- excepting control commands -- will, in a line-number dependent subsystem, either be added to, merged into, or will replace lines in the current file, depending upon the relative line-numbering of the lines in the file and the new input. (Refer to "Correcting or Modification of Line-Numbered Files" below.) In a nonline- number dependent subsystem, Text EDITOR for example, any new input will be appended to the end of the file.

Following each line of noncommand-language input and terminating carriage return, the subsystem will supply an initial asterisk, indicating that it is ready to accept more input. In the case of command-language input, the subsystem will normally return to build-mode following execution of the process requested by the command.

In line-number dependent subsystems (BASIC, TSS FORTRAN, CARDIN), a line of file-building input must begin with a line number of from one to eight numeric characters; this number may optionally be preceded by one or more initial blanks. The line number facilitates correction and modification of the source program. The line number is always terminated, (i.e., immediately followed by), a non-numeric character, which may be a blank.

In nonline-number-dependent subsystems, a line of file-building input may begin with any sequence of characters that is not defined as command language for the subsystem being used.

Correction or Modification of Line-Numbered Files

The correction or modification of the current file in line-number dependent subsystems proceeds according to the following rules:

- Replacement: a numbered line will replace any identically numbered line that was previously typed or already contained on the current file; i.e., the last-entered line numbered nnn will be the only line numbered nnn in the file.

- Deletion: a "line" consisting of a line number only, (i.e., nnn), will cause the deletion of any identically numbered line that was previously typed or is already contained on the current file.

- Insertion: a line with a line-number value that falls between the line-number values of two pre-existing lines will be inserted in the file between those two lines.

At any point in the process of entering file-building input in line-numbered subsystems, the LIST command may be given, which results in a "clean," up-to-date copy of the current file being printed. In this way, the results of any previous corrections or modifications can be verified visually. (The several forms of the LIST command are described in detail in Chapter 3.) Following the command OLD filename, the LIST command can be used initially to inspect the contents of the current source file, i.e., the "old" program.

Automatic Terminal Disconnections

Once communication with the Time-Sharing System has been established, any question or request must be answered within ten minutes, except for the initial requests for user identification (user-id) and sign-on password, which must be given within one minute. If these time limits are exceeded, the user's terminal will be disconnected.

Log-Off Procedure

To terminate one's current session with the Time-Sharing System and disconnect the terminal, the BYE command may be given either in build-mode or at the subsystem-selection level:

```
*BYE

 or

SYSTEM?BYE
```

In either case, a report of the user's time-sharing usage charges is given, as illustrated by the following example, and the terminal is disconnected:

```
**RESOURCES USED $ 4.47, USED TO DATE $ 919.02= 92

**TIME SHARING OFF AT 12.655 ON 11/04/69
```

When operating under a subsystem that does not have a build-mode and does not recognize BYE as a response, the response DONE may be used; then BYE may be given at the resulting subsystem-selection level.

To terminate one's current session without disconnecting the terminal, the command NEWUSER may be given in place of BYE; such a procedure will allow another user to log-on immediately following. The current user's log-off report is then printed and a new log-on sequence is initiated.

Terminating an Output Process

A lengthy listing or other output of information at the terminal, initiated for example by a LIST command, may be prematurely terminated by the use of the "interrupt" control peculiar to the type of terminal in use. This interrupt control is as follows:

- For teletypewriter terminals -- the BREAK key
- For typewriter-like terminals -- the ATTN or INT key

This control can also be used for abnormal termination of a program execution. However, the user is cautioned against indiscriminate use of this control since the results of its use are in some cases unpredicatable (in regard to the status of files, for example). The subsystem will normally return to build-mode or to the subsystem-selection level following the use of an interrupt control.

Paper Tape Input in Build-Mode

In order to supply file-building input from paper tape, the user gives the command TAPE (#TAP if the subsystem is Text EDITOR). The subsystem responds with READY. At this point, the user should position his tape in the tape reader and start the device. Input is terminated when either end-of-tape occurs, the user turns off the reader, an XOFF character is read by the paper tape reader, or a jammed tape causes a delay of over one second between the transmission of characters.

The tape may be prepared off-line from the keyboard, or it may be the result of previous output punched by the paper tape unit. If prepared off-line, it should include carriage returns to terminate each line, just as if entering data on-line, plus explicit line feeds to obtain legibility on the terminal-printer during preparation and transmission.

Command language may not be included on the tape. The input should be preceded by several RUBOUT characters and terminated by an XOFF followed by several RUBOUT characters. Neither the XOFF nor the RUBOUT characters will appear in the file.

At present, a maximum of 80 characters are permitted per line of paper tape input. Excessive lines will be truncated at 80 characters, with the remaining data placed in the next line. A maximum of two disc links (7680 words) of paper tape input will be collected during a single input procedure. All data in excess of two disc links will be lost.

Building File from Non-USASCII Paper Tape

In order to supply file-building input from non-USASCII paper tape (unaltered eight-bit codes), the user gives the command LUCID instead of TAPE. The system will read in the tape and store the data on a file without editing or parity modifications. The system does not delete or act on any characters in the data stream, such as DEL, X-OFF, CR, etc. The input will be terminated when a pause of over one second occurs in the data transmission. Termination does not require an X-OFF character, as does normal paper tape input via a DATANET[1] 355 Communications Processor.

> NOTE: LUCID cannot be used if data communication is via a Low Speed Line Adapter (LSLA) on a DATANET 355.

During paper tape input via a DATANET 355, the paper tape input will stop when an error message is to be sent to the terminal.

Automatic Paper Tape Input

At any point during the operation of the Time-Sharing System and at a time when the user must supply keyboard input, a previously prepared paper tape in special format may be used to simulate a sequence of responses, one line at a time. The system need not be in build-mode and direct (i.e., conversational) responses, file-building input, and/or commands may be entered.

This feature allows the preparation of a paper tape for input to the Time-Sharing System and/or subsystem(s) prior to connection with the system and allows terminal operation without supervision during the connection. Such paper tape input may be for a specific subsystem or production-program execution only, or may include anything from log-on through log-off procedures. Obviously such a tape must be error-free.

[1] Trademark

Rev. January 1971

The required format for each input line is as follows:

data string (up to 80 characters)
carriage return
XOFF
RUBOUT (may be multiple, but one is minimum requirement)

Character-delete control characters may be used in the normal fashion. Line-delete controls must be used as follows:

data string (to be deleted)
(line-delete control)
XOFF
RUBOUT (one is minimum)
corrected data string
carriage return
XOFF
RUBOUT

It is suggested that extraneous line feeds not be included in the tape. If, however, the user desires line feeds for terminal-printer legibility, they should be either between the carriage return and XOFF, or one line feed immediately following XOFF. If line feeds occur in any other position, they will be considered as part of the text.

To initiate automatic paper tape input, the user merely need position the tape and start the reader at any time that keyboard input is required.

The terminal will be automatically disconnected if no input is received within 10 minutes of the request for such input, whether via paper tape or keyboard.

DATANET 760 KEYBOARD/DISPLAY TERMINAL OPERATION

General Characteristics

The DATANET* 760 Keyboard/Display Terminal Unit (DTU760) is a cathode ray tube display plus keyboard device on which a complete "page" of input may be composed before transmission to the Time-Sharing System, and on which a complete "page" of output from the system may be received. The page may consist of up to 26 lines of data, each line containing a maximum of 46 characters. (The 46-character line is equivalent to the 70- to -80 character line allowed by the teletypewriter and typewriter-like terminals, insofar as a "line" is defined by the Time-Sharing System.)

The DTU760 Terminal consists of two elements, the keyboard module and the CRT display module. The keyboard module is a physically separate unit from the display module. The standard cable length provided between keyboard and display modules is 4 feet; however, for special applications the cable distance may be up to 100 feet.

The AUTOMATIC command cannot be used with DATANET 760 terminals.

*DATANET, Registered Trademark

Operation of the DTU760 differs essentially from keyboard/printer types of terminals as follows:

- More than one line of data may be transmitted to or received from the system at one time.
- Special character-delete and line-delete control characters are not applicable, as all typing errors may be corrected by means of controls inherent in the DTU760.

The user enters data by typing on the keyboard as on a typewriter. The characters and symbols are instantaneously displayed as they are typed. A special entry marker appears on the display to indicate the location of the next character to be entered. The marker automatically indexes with each character entry or may be manually spaced forward or backward, and up or down. It may also be reset to the first character position of the page or line. In addition to providing repetitive character entry capability, a repeat key allows a continuous scanning movement of the marker. Changes or corrections are made by relocating the marker to the erroneous character and typing the correct one. Erasure of the entire display is accomplished by a Form Feed key.

A TAB key allows the user to quickly and efficiently enter information into an user-composed or computer-stored format. Depressing the TAB key causes the entry marker to scan the display until it finds a vertical line, where it stops. These vertical lines, which serve as tab-stop markers, can be positioned anywhere on the display surface by the user or the computer.

The user completes the composing, verifying and correction of the entry with the terminal "off-line". When satisfied that the information is correct, the user positions the "end of text" symbol (C) opposite the last line of characters to be transmitted; then returns the entry marker to the first character to be sent; and depresses the transmit key. Successive characters are transmitted up to the "end of text" symbol.

Keyboard Module

The keyboard module contains a standard Alpha-Numeric Key group and a Command Key group. Additional key groups are available on an optional basis.

Alphanumeric Key Group

Alphanumeric Keys -- The alphanumeric character-set contains upper case alphabetic characters, numerics 0-9, and special characters. The commonly-used ASCII special characters are all present except the ↑ (up-arrow) character, generally used to denote exponentiation in algebraic subsystems. The up-arrow is replaced by the BLK (blink) character for such subsystems.

The following keys cause the entry and display of a corresponding character with a single key action.

Key Label and Character Displayed

A	H	O	V	2	9
B	I	P	W	3	:
C	J	Q	X	4	;
D	K	R	Y	5	,
E	L	S	Z	6	-
F	M	T	0	7	/
G	N	U	1	8	.

The space bar erases any character under the entry marker, displays a blank space, and moves the entry marker one space forward.

The following characters are generated by using the SHIFT key in conjunction with the designated key.

Key Label and Character Displayed

!	&	+	@
"	'	<	—
#	(	=	┌
$	)	>	▕
%	*	?	▏

The BLK (blink) key initiates the blink code (displayed as a space), causing the blinking of all following symbols up to the first space or up to and including character position 46.

The RPT key is used to repeat an alpha-numeric key operation as long as the RPT key is depressed.

Control Keys -- Control keys for the manipulation of the entry marker are as follows:

Key Label	Operation
BS	Backspace entry marker one space. If entry marker is in character position one of a line, depression of the BS key will not affect the entry marker position.
FS	Forward space entry marker one space.
LF or LINE FEED	Line Feed - Moves entry marker down one line at the same character position. If the entry marker is initially located on the last line of the display, it will automatically return to the top line.
ETX	End of Text - Positions ETX symbol (C) at end of the line on which the entry marker is located.
RLF	Reverse Line Feed - Moves entry marker up one line at the same character position. If entry marker is on first line, RLF key will not affect the entry marker position.
LR or RETURN	Line Return - Moves entry marker to first character position of the same line.
FF or FORM	Form Feed (clear display) - Erases the entire display except for the mode character, ETX symbol (C) and the optional function characters. This command also automatically page-returns the entry marker.
TAB	Moves entry marker from its initial position to the character position following the next vertical line symbol. If the entry marker reaches the end of a page before a vertical line is found, the entry marker will return to the beginning of the page.

NEW LINE Moves entry marker to first character position on the next line. If the entry marker reaches the last line, it will return to the first character position of the top line.

Command Key Group

Command keys for the manipulation of data are as follows:

Key Label Operation

PRT Print - Causes the information displayed to be transmitted to the computer with a print request character in the message header. Symbol "P" appears in mode display position when the pRT key is depressed, and keyboard entry is prevented until the message transfer is complete. The entry marker is moved to the first character position of the line following the ETX symbol (C) when the information transfer is complete. If the ETX symbol is on the last line of the page, the entry marker is moved to the first character position of the first line.

LOC Local - Allows entry of data from the keyboard only. Symbol "L" appears in the mode display position when local operation is selected.

REC Receive - Allows the computer to enter or update the display at any time. Entry of data from the keyboard is also allowed. The symbol "R" is displayed in the mode display position when REC operation is selected.

TX Transmit - On input, requests transmission of the message to the computer. During output, requests another page of data to overlay that already on the display. When transmission is requested, keyboard entry is prevented until the transmission is completed. The symbol "T" appears in the mode display position when the TX key is depressed, and the entry marker is positioned on the line following the ETX symbol (C) when transmission is complete. If the ETX symbol is on the last line of the page, the entry marker is moved to the first character position of the first line.

← Backspaces entry marker one space. Repeated if held down. If entry marker is in character position one of a line, depression of← will not affect the entry marker position.

→ Forward spaces entry marker one space. Repeated if held down.

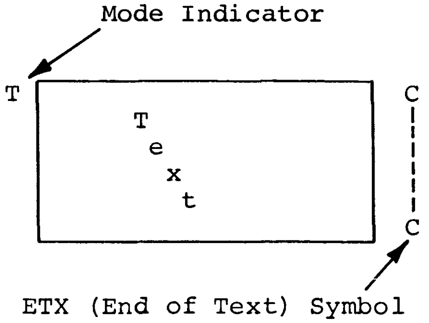
↓ Line Feed - Moves entry marker down one line at the same character position. If the entry marker is initially located on the last line of the format, it will automatically return to the top line. (Repeated if held down.)

↑ Reverse Line Feed - Moves entry marker up one line at the same character position. When the entry marker reaches the top line of the display, the Reverse Line Feed operation has no effect. (Repeated if held down.)

PR Page Return - Returns entry marker to the first character position on the first line.

ETX End of Text - Positions ETX symbol (C) on display at end of line on which the entry marker appears.

Mode Indicator and End-of-Text Symbol Positions



Mode Indicator	Meaning
T	Indicates that the transmit (TX) key has been depressed and that the computer-generated acknowledgment message has not yet been received by DTU760.
blinking mode indicator	Indicates that an error was recorded during the last message transmitted from computer to DTU760.
P	Indicates that the print (PRT) key has been depressed and that the computer-generated acknowledgment message has not yet been received by DTU760.
L	Indicates that the local (LOC) key has been depressed, placing DTU760 in off-line compose mode.
R	Indicates that the receive (REC) key has been depressed, placing DTU760 in receive mode.

The position of the ETX symbol (C) is varied vertically in the right margin of the display by means of the ETX key. The symbol is positioned by the user to the last line of text to be transmitted, this last line being indicated by the entry marker positioned over some character within the line.

Display Module

The display module is a viewing device for use during composition and editing of information prior to transmission and for the presentation of response data received from the computer.

The entry marker, a blinking horizontal line at the top of a character position, always indicates where the next typed character will appear. The user may position this marker as follows:

1. FF - Form Feed clears the display and moves the entry marker to the upper left corner (first character position).

2. Use of alphanumeric keys moves the entry marker to the next position on the display. LR key moves the marker to the left side of the display, LF key moves the entry marker down one line.

3. → (right arrow) key moves the display entry marker to the right.
   ← (left arrow) key moves the marker to the left.
   ↑ (up arrow) key moves the marker towards the top of the display and the ↓ (down arrow) key moves the marker towards the bottom of the display.

4. The PR key moves the entry marker to the upper left corner of the display.

The display may be cleared by striking the FF key. Characters may be changed by positioning the entry marker and entering a replacement -- character key or the space bar (blank).

DATANET 760 has three modes: local, receive, or transmit, indicated by an L, R, or T in the upper left corner of the display. The user may compose input in local or receive mode (when the keyboard is unlocked). When ready to transmit, the user must execute the following three steps.

1. Position the ETX symbol (C) -- on the right margin of the display -- to the last line of input to be transmitted by positioning the entry marker on the last line (any character position) and striking the ETX key.

2. Position the entry marker on the first character to be transmitted. Full lines (46 characters + CR + LF) are always transmitted except for the first line if the user positions the marker in other than the first position.

3. Strike the TX key. The mode indicator will change to T then to R when transmission is successful. If the mode indicator blinks, it is an indication that an error was recorded during the last message transmitted from the computer.

To respond to the full-page-blink, (blinking asterisk below last line of text), when ready for the next page, strike the ETX key and PRT or TX key. The use of the PRT key results in a clearing of the display before the appearance of new text. The use of the TX key results in the overlaying of old text (already on the display) by the new text.

When striking the PRT or TX keys, the user should observe the mode character at the upper left corner of the display to be sure the function was performed.

Log-On, Log-Off, Break, and Disconnection Procedures

Log-On and Log-Off

The log-on procedure for the DTU760 is as follows:

- Turn the unit on by means of the ON/OFF switch and wait approximately 30 seconds for warm-up. The mode indicator, ETX symbol, and the entry marker will appear on the display.

- Enter and transmit the following log-on request message:

```
$*$ab PASSWD,nn,DAC,GE-TSS
```

where:

ab is any two-alpha character station code

nn specifies the desired page size in number-of-lines per page; nn may be 04, 08, 16 or 26.

The Time-Sharing System normally responds with the same log-on sequence as is produced on teletypewriter/ typewriter-like terminals. The user's log-on responses are also standard at this point.

The log-off procedure is also exactly as for other terminals. Note, however, that unlike teletypewriter terminals, the DTU760 must be turned off manually following disconnection.

Break and Disconnection

A "BREAK" (interrupt) signal is transmitted to the system via the DTU760 by means of the following control message:

```
$*$BRK
```

This message could be used to interrupt some lengthy output process, such as the unwanted remainder of a long listing, or to interrupt execution of a user's program.

Under circumstances where it might be necessary to force disconnection of the DTU760 from the system, the following message could be composed and transmitted:

```
$*$DIS
```

Following disconnection, the log-on request message may again be transmitted.

EXCEPTIONS TO STANDARD SUBSYSTEM USAGE

Exceptions to standard subsystem usage as a result of operating with a DTU760 terminal are as follows:

- In using the Text EDITOR subsystem, the first-column carriage-return "escape command" effecting the change from build-mode to editing-mode, is replaced by a double pound-sign (##).

- With the use of algebraic subsystems BASIC and ABACUS, the up-arrow (↑) symbol used as the exponentiation operator is replaced by a BLK (blink) character preceding the exponent. The blink character itself is displayed as a blank, and causes the exponent character(s) following the blank, in turn, to blink.

- Output on the screen will be by page; that is, when a subsystem fills a page (4, 8, 16, or 26 lines), the output will halt until the user presses the PRT button. A blinking asterisk is the signal for a response. To receive the next page, the response should be:

  1. ETX-TX to transmit the data delineated by the entry marker and EXT.

  2. ETX-PRT to clear the screen first.

The ground rules for the counting of lines are:

1. Lines > 46 characters are continued on the next line. No attempt is made to break at word boundaries.

2. A last line on the page > 46 characters will be reserved so that the entire line will appear at the beginning of the next page.

3. Counting of input lines (carriage return and line feed following a request counts one line) is ignored under the assumption that the current systems can use only one line (already counted) and specific DATANET 760 applications will count, FF, ETC, internally.

- The automatic line numbering mode (command AUTO) is inoperable from a DATANET 760.

- Files built by a DATANET-760 will have lines limited to 46 characters. A line may not be made longer by "wrapping" to the next line as is done when outputting files build on another terminal with longer lines.

5. Service Subsystems and Programs

ABACUS SUBSYSTEM

The ABACUS (ABC) subsystem is an algebraic-expression evaluator that may be called either at the subsystem-selection level or at the command-language level by the name ABC. The function of ABACUS is that of a powerful desk calculator, with the ability to calculate and "remember" the value of symbolic variables. ABACUS features a summation operation and employs commonly used mathematical constants and functions.

Use of ABACUS

```
SYSTEM?ABC          (if at subsystem-selection level)

          or

*ABC                (if at command level)
```

The initial call (SYSTEM? or command level) may contain, on the same line, the expression to be evaluated; e.g., *ABC 1.379 ↑ 2. Otherwise, ABACUS issues a question mark (?) as a request for input. The possible forms of input are:

```
? expression

? x = expression                 where x is a variable

? FOR x = a,b,c;                 where a,b,c specify a
    expression in x                range of values for x

? FOR x = a,b,c;                 where y is also a
    y = expression in x            variable
```

From one to three FOR specifications may be employed before the expression, separated by semicolons; i.e., FOR x = ...; FOR y = ...; FOR x = ...;.

The results of each expression evaluation are printed immediately. ABACUS then issues another request for input (?). A null response (.i.e., carriage return only) or DONE causes an exit from the subsystem.

An expression is composed of operators, numbers and/or variables, and/or constants and functions, conforming to ordinary arithmetic and algebraic rules. The permissible operators are:

+ (addition)
- (subtraction)
* (multiplication)
/ (division)
↑ (exponentiation)
& (summation)

Parentheses may be used to indicate grouping of operations, according to standard usage. (Refer to "Order of Evaluation and Use of Parentheses" below.)

Numbers

Numbers may be written as:

Integers: e.g., 1, -25, 7063
Fractions: e.g., .1, -.0005, .3681400
Mixed numbers: e.g., 1.5, 812.764
Scientific notation: e.g., 1E10, 2.41E-3, -3215E7

Numeric operands may contain up to 18 significant digits. Printed result values are, however, limited to a maximum of seven places in the fractional part. Precision is kept internally to 18 places, however.

Variables

Variables (names to which numeric values can be assigned) are composed of 1 to 8 alphanumeric characters, the first of which must be alphabetic; e.g., A, B5, SUMSQUAR. There are two types of variables, according to usage: (1) FOR variables, i.e., those defined in a FOR specification, and (2) label variables; these appear on the left of an equal sign but not preceded by FOR. In input of the form "X = expression", X is a label variable. The distinction to be noted between the two types is that label-variable values are "remembered" between expression evaluations. The values assigned to a FOR variable hold only for the expression associated with the FOR specification(s); they are not remembered for a subsequent expression. For example:

```
? X = SIN (30/RADIAN)             (X is a label variable)
X = +0.5                        - (answer)
? X ↑ 2*27.9
+6.975                          - (answer)
```

Constants and Functions

Constants and functions available in ABACUS are:

Constant Name	Predefined Value	
PI	3.14159...	
RADIAN	57.295.....	internal precision to
E	2.71828...	18 significant digits

Function Name	Meaning
ABS (x)	Absolute value of x
ATN (x)	Arctangent of x
COS (x)	Cosine of x
EXP (x)	e to the power of x
LOG (x)	Natural logarithm of x
SIN (x)	Sine of x
SQR (x)	
or	Square root of x
SQT (x)	
TAN (x)	Tangent of x

For trigonometric functions, x denotes an angle measured in radians.

Function names are reserved words; i.e., they cannot be used as variable names. Constant names are not reserved; actually they are "remembered" variables with preset values. Their value may be changed by using the name as a label variable.

Summation Operator and FOR Variables

The summation operator, &, may only appear at the beginning of an expression, and the entire portion of the expression following it is assumed to be the argument to be summed (regardless of the use of parentheses). From one to three variables may be given a range of values for the summation by means of FOR statements.

The FOR statement(s) must precede the associated expression in the same line of input, separated by semicolons. The form of the FOR statement is:

```
FOR x = a, b, c;
```

where:

a is the initial value of x,
b is the limiting value of x, and
c is the step-value or increment (optional)

If the step-value, c, is not specified, 1 is assumed. Substitutions for a, b, and c may be positive or negative integers, expressions, or predefined variables.

For example:

```
? FOR X = 1, 5;  FOR Y = 7, 50,9;  Z = &(X+Y)*PI
Z = 2199.1149
```

In summations, all FOR variables are treated as summation indices and in the case of summations over two or three FOR variables, the indicated summations are nested. Each summation variable takes on the values a, a+c, a+2c,...up to but not exceeding the value b. Thus the expression above would expand as follows:

$$Z = \sum_{X=1,2,...}^{5} \quad \sum_{Y=7,16,...}^{43} \qquad (X+Y)$$

$$Z = ((1+7)\pi + (1+16)\pi + (1+25)\pi +...+(5+34)\pi + (5+43)\pi)$$

Although an expression containing a summation operator must be preceded by one or more FOR specifications (in order to be meaningful), FOR variables may also be used in expressions that do not contain the & operator. For example:

```
? FOR A = 3,11,2;  FOR B = 1,3;  X = A ↑B
```

In these cases, the expression will be evaluated separately for each possible combination of FOR values (as is done in FORTRAN). The output from the example expression just above would appear as:

```
A        B        X

3        1        3
3        2        9
3        3       27
5        1        5
5        2       25
5        3      125
7        1        7
7        2       49
7        3      343
9        1        9
9        2       81
9        3      729
11       1       11
11       2      121
11       3     1331
```

If a label variable is used, as in the above example (X), the last-determined value is "remembered" for the variable.

Continuation Lines

ABACUS accepts only one expression at a time, with or without associated FOR specifications. Normally, this is represented on one line of terminal input. If, however, the expression (with any preceding FOR specifications) requires more than one line, the first line can be terminated with a $ plus carriage-return combination--rather than a carriage return only. This denotes the next line to be continuation of the first, and ABACUS responds with another input request (?).

Order of Evaluation and Use of Parentheses

Expressions are evaluated from left to right, with operations performed in the following order:

1) functions
2) ↑
3) * and /
4) + and -
5) &

Care must be exercised, especially with regard to successive exponentiation, to ensure that the order of evaluation implied by the above rules is the order intended. If the intention is otherwise, parentheses must be used to force the desired order of evaluation.

All operations within a subexpression enclosed in parentheses will be performed before any operations to the right of that subexpression. Grouping of operations to any depth may be indicated by means of nested sets of parentheses. An exception is the summation operator, &, which may not be enclosed in parentheses.

Implicit multiplication is allowed preceding a parenthesized subexpression, but not between two such subexpressions. For example: 3(x) is equivalent to 3*(x); but (x) (y) is illegal, and must be written as (x) * (y).

The argument of a function, which may be any expression, must be enclosed in parentheses.

Mode and Precision of Calculation

All calculations are performed in double precision floating-point, with consequent precision (but not accuracy, necessarily) to 18 places. Displayed results are limited to a maximum of seven places in the fractional part (rounded), but 18 significant digits are carried internally. This may result in a small discrepancy between displayed intermediate and final results, in a sequence of related evaluations.

GECOS III FILE SYSTEM AND ACCESS SUBSYSTEM

GECOS III File System

The GE-600 Time-Sharing System utilizes the capabilities of the GECOS III file system, which is a logical mechanism for storing and retrieving permanent files and is common to all system programs operating under the GE-600 Comprehensive Operating Supervisor. A file system can store many files on many unspecified external, "background" storage devices, and the user normally need not be concerned with the device his file is on nor with the characteristics of the device.

Structure of File System

The file system is described in detail in the manual GE-600 Line GECOS III File System, CPB-1513. However, the main features of interest to the time-sharing user are repeated here.

The file system is a tree structure whose origin is the system master catalog. The primary nodes of the tree are user's master catalogs; the lower-level nodes are subcatalogs created by the user. The terminal points of the structure are the files themselves. The file system has a limit of seven levels for either building or accessing files. Figure 1 shows the file system's hierarchical structure.

Catalogs and Files

A catalog consists of a definition containing a catalog name, password, and permissions. Since it contains no user data, a catalog can be neither read nor written except by the file system itself. An ACCESS function is provided, however, to direct the file system in the creation and modification of subcatalogs.

A file, as known to the GECOS III file system, consists of a definition containing file name, file size, password, permissions, and a description of the physical file space. The file definition is distinct from the physical file space which may contain user data and can be read or written.

Passwords

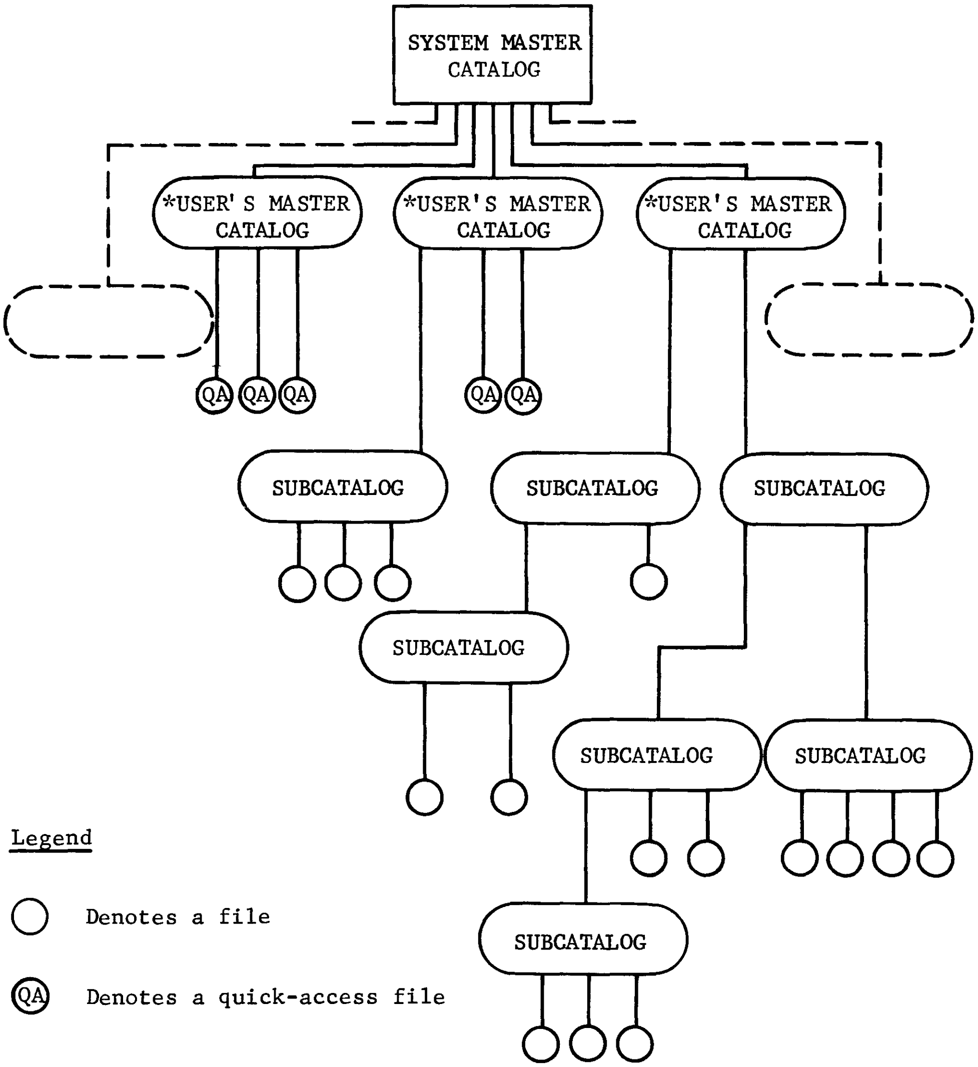
Passwords may be attached to any catalog or file. A password simply allows a user to traverse a catalog/file string. A user can get to a given catalog or file only if he can give the passwords for all higher level catalogs in the string. (When traversing a string, a password must not be given if none has been attached.) The originator of a given string is required to give the necessary passwords when traversing a string.

Permissions

Users permissions, both general and specific, can be attached to any catalog or file. When permissions are attached at the catalog level, they apply to all subordinate catalogs and files. The originator of a catalog/file string implicitly has all permissions for that string but must give all applicable passwords.

Users permissions are as follows:

Read - allows a file to be read
Write - allows a file to be written
Append - (presently treated as WRITE)
Execute - allows a file to be executed only (no list of file, etc.)
Purge - allows catalogs and/or files to be purged from the system (specific permission only)
Modify - allows catalog and/or file definitions to be changed (specific permission only)



All user-ID's must be unique within the system; all subcatalog and file names are automatically qualified by the user's master catalog name and the names of any intermediate subcatalogs. The system master catalog cannot be accessed by the normal user.

*Identified by the user-ID.

Figure 1. Logical Structure of the File System

When operating under the Time-Sharing System, each file requested by an NEW or OLD command is a temporary working file, called the current file, that will "disappear" at the end of a user's session at the terminal unless he saves it. In the case of OLD, the current file is effectively a "scratch copy" of the named permanent file, allowing modifications to be made and tested without changing the original OLD file, unless the user explicitly "resaves" on that file.

File content is written (i.e., stored) on a permanent file only by means of the SAVE or RESAVE commands. In addition, the SAVE command implicitly causes the file to be created. Provided that the file named by the SAVE command does not already exist, the system automatically creates a permanent, external file and writes the contents of the current working file onto it. If the SAVE operand consists simply of a file name, the file created will be of the quick-access type. This means that the file emanates directly from a user's master catalog without intervening subcatalogs (see Figure 1). If a catalog-string precedes the file name in the SAVE operand, the subcatalog(s) named was previously created via the ACCESS subsystem.

If the file to be written on already exists (whether created by a prior SAVE, by ACCESS, or by a batch activity), a RESAVE command causes the contents of the current file to be written to the named permanent file, replacing whatever contents existed in the latter.

If the characteristics of the quick-access type of file are suitable to the user's requirements, then the command-language facilities offered by the system are sufficient; ACCESS need not be used. The OLD and GET commands, for example, offer full retrieval capabilites for files of any type or level.

ACCESS Subsystem

Capabilities

For users who wish to utilize some or all of the capabilities of the file system, the ACCESS subsystem provides an interface, thus allowing the user to perform the following:

- Create hierarchical structures of subcatalogs and files
- Attach passwords to his subcatalogs and files
- Give general permission to all other users to access his files in specified ways

- Give specific permissions, by user-ID
- Protect a given file or set of files against any mode of access
- Gain the permitted types of access to another user's files
- Gain the permitted types of access to files created in the batch-processing environment
- Modify catalog name, password, and/or permissions on an existing catalog
- Modify file name, size, password, and/or permissions on an existing file
- Purge or release an existing file or catalog/file string
- List all catalogs and files which emanate from a given catalog
- Rename files temporarily for a given job
- Create a random file with a logical-record-size attribute, if such a requirement exists.

Use of ACCESS

ACCESS consists of a number of functions which, together, provide a conversational facility for the following:

- Creating and purging/releasing catalogs and files
- Modifying catalog and file attributes (name, size, password, permissions)
- Accessing and deaccessing files
- Listing catalogs and files

The operation of ACCESS consists of responses, via the terminal, to a sequence of English-language questions. All of the "standard vocabulary" associated with the user's responses may be abbreviated for convenience in keying-in. A non-conversational short form of input is also provided for more experienced users and users with batch FILSYS experience.

ACCESS is not a means of reading or writing permanent file content. OLD and SAVE/RESAVE perform these functions. ACCESS is selected before proceeding to a desired processing subsystem; it is used for example to "create" a file (i.e., the file definition and the file space), before the substantive file is built under another subsystem. The OLD/NEW sequence and the SAVE/RESAVE commands of the succeeding subsystem(s) are still applicable.

Using the ACCESS subsystem, a file may be created at the system level by specifying the ACCESS subsystem name, the create command short form (CF), and the file description in one string in response to SYSTEM?. For example:

```
SYSTEM?ACCESb̸CF,/CAT1$PASSWD1/CAT2$PASSWD2/FIL1$PASSWD3,B/1,3/,R,W,E
```

The creation of the file is performed as in the conventional question-response sequence in the ACCESS subsystem. After this function is performed by access, the user is returned to the function selection level.

NOTE: Any ACCESS function may be specified in a single string response to SYSTEM?, similar to this example.

Some general rules can be cited for the use of ACCESS.

1. The ability of a user to access files and otherwise manipulate catalog/file structures (e.g. modifying and purging) depends upon his knowing the necessary file definitions. Beyond this, the file system has two file and catalog protection features--passwords and permissions.

   Permissions provide the file creator with a positive protection feature; if permissions are not explicity granted, his catalogs and files are completely protected by default. The user must assign to others any degree of access he wishes them to have. But, since specific permissions for a given user do not add to, but replace, any general permission that may have been given, specific permissions may be used to exclude a given set of users from one or more types of access.

   Passwords provide an additional level of protection. If passwords are assigned by the creator of a catalog/file string, they must be supplied in order to pass through the string.

   The creator of a catalog/file string is exempt from any access-mode restrictions he imposes (i.e., he implicitly has all permissions for his own catalogs and files), but he must give all passwords.

   The MODIFY permission, which allows another user to change file names, catalog names, file size, passwords, and/or permissions, also implies the ability of this other user to create catalogs and/or files emanating from the master catalog.

2. The definition of a particular catalog or file must include the names of all higher-level catalogs that must be traversed to arrive at that point. The catalog string would include at least the user's master catalog. A file definition, then, is the complete catalog string plus the file name.

3. Each user's master catalog must be created for him before he can use the system. It has no password or permissions associated with it, and is unalterable. The installation usually controls the generation of this catalog.

Identifiers and Delimiters in User Responses

User responses are composed of the following:

- Identifiers
- Keywords
- Word delimiters
- Line delimiters

Identifiers consist of file names, catalog names, user-IDs and passwords. They can consist of alphabetics, numerics, periods, and minus signs. Each identifier can be up to 12 characters in length except for file names, which are limited in length to 8 characters.

In response to the question FILE NAME$PASSWORD?, issued by the Access File function, a file name of up to 12 characters may bc specified (i.e., the name of a batch-environment file), if followed by an alternate name of 8 characters or less, enclosed in parentheses. In response to FILE TO BE PURGED?, a file name of up to 12 characters could be specified, if the file to be purged were not created in the Time-Sharing System environment.

Keywords consist of function names, access types (permissions), and several file-type parameters, of limited interest, that are described under "Special Features." Keywords are used in responses to questions. All keywords, except EXCLUDE, DELETE and GEN'L, can always be abbreviated to the initial character, or a two-character acronym in the case of function name (e.g., R for READ permission or CC for Create Catalog function).

The file-size specification in the response to FILE NAME, SIZE (BLKS), MAX SIZE? (Create File function), is a decimal number denoting the number of blocks required. This may be considered a special case of a keyword.

Word delimiters are the slant or virgule (/), the dollar-sign, and the comma. Blanks may be used freely in responses except within function names; they are in no sense delimiters and are ignored.

The use of the three word delimiters is as follows:

The / delimiter has two functions:

1. In catalog-strings, / indicates that a subcatalog name follows and is concatenated to the preceding catalog in the string. An initial / indicates that the following subcatalog-string (if any) is concatenated to the user's master catalog. A response to CATALOG STRUCTURE TO WORKING LEVEL? of / and carriage return is equivalent to the user's own user-ID; i.e., it positions the user to his own master catalog.

2. In specific permissions, a / indicates that a user-ID follows.

The $ delimiter is used only to concatenate a password to a catalog or file name.

The , delimiter is used as a general separator for keywords; i.e., for separating access-types and sizes, and separating file names from following keywords or sizes.

The line delimiters are a carriage return, an asterisk plus carriage return, or a double asterisk plus a carriage return. Each of these serves to terminate a response, but with a different effect.

1. Carriage Return: A carriage return following a response generally signifies that the user wishes to remain at the same catalog position (if relevant), and proceed to the next question in logical sequence. This may be the next question in a set, or the initial question again.

   When only a carriage return is given, (i.e., a "null" response) however, it has several possible meanings:

   - In response to the question CATALOG STRUCTURE TO WORKING LEVEL? a carriage return only is equivalent to the user's own user-ID or a / and carriage return. Any of these responses requests that the user be positioned to his own master catalog.

   - A carriage-return only following a question that logically requires a response (e.g., NEW CATALOG?), causes an immediate return to the question FUNCTION?.

- The question SPECIFIC PERMISSIONS? recurs each time a response is given (delimited by a carriage return), since only one set of specific permissions can be given in each. If only a carriage return is given, the information received so far is processed, and the first question below CATALOG STRUCTURE TO WORKING LEVEL? is reissued (i.e., NEW CATALOG? or FILE NAME, SIZE (BLKS), MAX SIZE?), allowing a new catalog or file to be created at the same catalog level.

- A carriage-return only response to FUNCTION? causes a return to the question SYSTEM?.

2. Single Asterisk Plus Carriage Return: If a single asterisk plus a carriage return is given in reply to a question, either with or without a substantive response, ACCESS processes the information it has and returns to the first question at the same catalog level (e.g., to skip any further questions in the set). ACCESS, of course, must have sufficient information to process.

3. Double Asterisk Plus Carriage Return: If a double asterisk plus a carriage return is given, either with or without a substantive response, ACCESS processes the information it has and returns to the question FUNCTION?. It implies that the user is finished with the current function. ACCESS, of course, must have sufficient information to process.

In addition to the changes in level of operation produced by the several line delimiters, a response of DONE to any question causes an exit from ACCESS. No processing is performed and the question SYSTEM? results.

ACCESS Functions

The initial communication from ACCESS, following subsytem selection, is a request for a choice of function; i.e., FUNCTION?.

The functions that may be requested follow; the response may be spelled out, or abbreviated as underlined:

- CREATE CATALOG
- CREATE FILE
- ACESS FILE
- DEACCESS FILE
- MODIFY CATALOG

- MODIFY FILE
- PURGE CATALOG
- PURGE FILE
- RELEASE CATALOG
- RELEASE FILE
- LIST CATALOG
- LIST SPECIFIC

The effect produced by each function is as follows:

CREATE CATALOG - creates a subcatalog.

CREATE FILE - defines file space and attributes for a given file name. It does not bring a file into the available file table.

ACCESS FILE - brings a file into the available file table.

DEACCESS FILE - takes a file out of the available file table.

MODIFY CATALOG - modifies the name, password, and/or permissions associated with a given catalog.

MODIFY FILE - modifies the name, maximum size, password, and/or permissions associated with a given file.

PURGE CATALOG - deletes a catalog from the system along with any subcatalogs and files which are subordinate to it. All released file space is overwritten.

PURGE FILE - deletes a file from the system, overwriting the released file space.

RELEASE CATALOG - deletes a catalog from the system, along with any subcatalogs and files which are subordinate to it. Any released file space is not overwritten.

RELEASE FILE - deletes a file from the system, but without overwriting the released file space.

LIST CATALOG - lists the names of the catalogs and files which emanate from this catalog.

LIST SPECIFIC - lists in detail the description of the catalog or file specified.

Following the response to FUNCTION?, ACCESS asks the user to describe the catalog-string, catalog, or file. Each function has a fixed set of questions, with several of the questions common to each set. Some of the questions do not logically require a response; e.g., PASSWORD? (there may be none). If no response is applicable, only a carriage return is given.

All the functions, except DEACCESS FILE, first request a definition of the existing catalog-string. Then the name of the catalog or file to be processed is next, along with size attributes in the case of a file. Passwords and permissions are then requested, as appropriate.

Short-Form Usage of ACCESS Functions

Once the user has become familiar with the conversational, or question/response sequence, form of ACCESS, he may use a short form of function specification which effectively eliminates questions normally asked. In this short form, the function name (e.g., CREATE FILE) is followed directly by all the user-entered information needed to complete the function specification, usually all on one line. Each item of information is separated from other items by commas.

The information entered in this short form is much the same as that given as responses in the conversational mode, but with additional keywords. The format is identical to that of the batch-world file system (FILSYS) input.

The general format, in response to the initial question FUNCTION?, is:

```
function-name, catalog/file string, option, ..., option
```

where:

catalog/file string is the same as in conversational responses, and *options* are:

Form	Function
PASSWORD/password/	password assignment
access-type	general permissions
access-type/ user-ID,...,user-ID/	specific permissions
$\left\{\begin{matrix}\text{BLOCKS}\\\text{LINKS}\end{matrix}\right\}$ /x,y/	size assignment
MODE/mode/	mode assignment

Access type and mode are defined under each applicable function description. Options may appear, comma-separated, in any order. The keywords BLOCKS and LINKS may be abbreviated to the first letter, as may the access type and mode words. Options unique to the Modify Catalog and Modify File functions are described along with those functions.

The short-form reply may be extended to two or more typing lines by terminating a line with a word delimiter (slant, comma, or dollar-sign) (plus carriage return), at a convenient point, implying that the input is not complete but is to be carried over to the next line or lines.

Questions and Responses

Sets of questions associated with each function follow, along with the general form of the response to each question. The minimum required user response is underlined for illustrative purposes. Each set is followed by illustrative examples.

1) FUNCTION? CC

CATALOG STRUCTURE TO WORKING LEVEL?

user-ID/cat-name$password/.../cat-name$password

NEW CATALOG? cat-name

PASSWORD? password

GENERAL PERMISSIONS? access-type,...,access-type

The access types follow; all may be spelled out, or abbreviated as underlined; except for EXCLUDE, which must be spelled out:

READ

WRITE

APPEND

EXECUTE

MODIFY (specific permission only)

PURGE (specific permission only)

EXCLUDE

SPECIFIC PERMISSION?

access type,...,access type/user-ID/.../user-ID

If no response to the question SPECIFIC PERMISSION? is given, (i.e., only a carriage return), the catalog is created and the question NEW CATALOG? is reissued.

Example Replies (user responses are underlined):

FUNCTION? CC

CATALOG STRUCTURE TO WORKING LEVEL?

JDOE/CAT1$ABC

This response states that there is a subcatalog named CAT1 that is concatenated directly to the user's master catalog identified by the user-ID JDOE, and that it is desired to create a new catalog from this level. The password ABC was attached to catalog CAT1 when it was created.

NEW CATALOG? CAT2

This response indicates the name of the catalog, CAT2, created at this point.

PASSWORD? AOK

This response associates the password AOK with this catalog. A carriage return alone would indicate that no password is to be assigned.

GENERAL PERMISSIONS?

The lack of a response here indicates that general permission is not granted at this level for any type of access to subsummed files. A response of READ, EXECUTE would indicate that any unspecified user has permission to read and execute (if meaningful) any file that emanates from this catalog.

```
SPECIFIC PERMISSION?  READ/BJONES/ASMITH
```

```
SPECIFIC PERMISSION?  READ,WRITE,PURGE/ALLONG
```

This combination of responses states that the users who have logged onto the system under the names BJONES and ASMITH can pass through this level with read permission for any files below, and that the user ALLONG can pass through with read, write, and purge permissions.

```
SPECIFIC PERMISSION?
```

The carriage return alone means that no further specific permissions are to be given; the catalog is now created and the question

```
NEW CATALOG?
```

is reissued, allowing the user to create another catalog at the same level (i.e., also emanating from CAT1).

Alternative forms of the response to CATALOG STRUCTURE TO WORKING LEVEL? are as follows:

```
/CAT1$ABC
```

Assuming the user to be JDOE, this response is equivalent to the one given above, JDOE/CAT1$ABC. The initial slant indicates the user's own master catalog.

A response of

```
/
```

indicates that the user desires to create directly from his master catalog. This response is equivalent to his user-ID alone.

Example short-form reply:

```
FUNCTION? CC,/CAT1$ABC/CAT2,PASSWORD/AOK/,READ/BJONES,
? ASMITH,ALLONG/,WRITE/ALLONG/,PURGE/ALLONG/
```

2) FUNCTION? CF

CATALOG STRUCTURE TO WORKING LEVEL?

user-ID/cat-name$password/.../cat-name$password

FILE NAME, SIZE (IN BLCKS), MAX SIZE?

file name, initial size (blocks), maximum size (blocks)

PASSWORD? password

GENERAL PERMISSIONS? access type,...,access type

The access types follow; they may be spelled out, or abbreviated as underlined:

READ

WRITE

APPEND

EXECUTE

MODIFY (specific permission only)

PURGE (specific permission only)

SPECIFIC PERMISSION?

access type,...,access type/user-ID.../user-ID

Random-File Specification: If required, a file can be created with a random-access-treatment indication, by responding to the FILE NAME, SIZE (IN BLCKS), MAX SIZE? question as follows:

file name, initial size, max. size, R

If random (R) is specified, a further question will be asked:

LOGICAL RECORD SIZE? record size in words

Random-I/O files for TSS FORTRAN are required to have a logical-record-size attribute; if use of random files does not require this attribute, a response with a carriage-return only is required.

If the file is created as random, it cannot be accessed as linked (i.e., sequential), but the converse is true. Refer to Access File function description below. Further details for random specifications are given under "Special Features" below.

Example Replies (user responses are underlined):

FUNCTION? CF

CATALOG STRUCTURE TO WORKING LEVEL?

/CAT1$ABC/CAT2$AOK

This response defines user-ID/CAT1/CAT2 as the catalog-string from which the file is to emanate. The initial slant indicates that the succeeding string is concatenated to the user's own master catalog.

FILE NAME, SIZE (IN BLCKS), MAX SIZE? FIL1,4,12

This response asks for a file space of 4 blocks initially, with a maximum eventual size limit of 12 blocks, named FIL1.

PASSWORD?

No password is assigned to this individual file.

GENERAL PERMISSIONS? READ

SPECIFIC PERMISSION?

None are granted at this level, but those granted at the level of CAT2 (CREATE CATALOG in the previous example) apply to this file.

The lack of a response means the end of the information relevant to the creation of this file. The file is created, and the question

FILE NAME, SIZE (IN BLCKS), MAX SIZE?

is reissued. This permits creation of other files at the same level.

Example short-form reply:

FUNCTION? CF,/CAT1$ABC/CAT2$AOK/FIL1,B/4,12/,R

Note. File mode by default is linked (sequential); i.e., MODE/LINKED/.

3) FUNCTION? AF

CATALOG STRUCTURE TO WORKING LEVEL?

user-ID/cat-name$password/.../cat-name$password

FILE NAME$PASSWORD? file name(alternate name)$password

PERMISSIONS DESIRED?

access type,...,access type

The access types follow: they may be spelled out, or abbreviated as underlined:

READ

WRITE

APPEND

EXECUTE

Random File Specification: A file can be accessed for random-treatment, whether created as random or linked, by responding to the FILE NAME$PASSWORD? question with:

filename,R$password

or

filename(alternate name),R$password

If the file was created as linked, the random-treatment indication is temporary; i.e., for the current access only. If the file was created as random, the ,R specification is superfluous.

Example Replies (user responses are underlined):

FUNCTION? AF

CATALOG STRUCTURE TO WORKING LEVEL?

JDOE/CAT1$ABC/CAT2$AOK

The user in this case is not the creator of the file to be accessed, so he must define the user's master catalog (e.g., JDOE) from which the file emanates, along with any required subcatalogs and passwords.

FILE NAME$PASSWORD? FIL1

If a password were required, it would be concatenated to the name with a dollar-sign; i.e., FIL1$ABC.

PERMISSIONS DESIRED? READ

General read permission was granted for this file. (Several specific read permissions were also granted at the level immediately above CAT2.) Termination of this response with only a carriage return causes the file to be accessed and the request

FILENAME$PASSWORD?

to be reissued.

Example Short-Form Reply:

FUNCTION? AF,JDOE/CAT1$ABC/CAT2$AOK/FIL1,R

4) FUNCTION? DF

FILE NAME? file name (or CLEARFILES)

The response for this function is the name of the file to be deaccessed. The name supplied is always the name under which the file was accessed, whether this was the actual name or a temporary alternate name. If CLEARFILES is used, all of the user's available files are deaccessed, including his temporary files.

Example Short-Form Reply:

FUNCTION? DF,FIL1

5) FUNCTION? PC

CATALOG STRUCTURE TO WORKING LEVEL?

user-ID/cat-name$password/.../cat-name$password

CAT. TO BE PURGED? cat-name

PASSWORD? password

Example Replies (user responses are underlined):

FUNCTION? PC

CATALOG STRUCTURE TO WORKING LEVEL?

/CAT1$ABC

This response defines the subcatalog CAT1 concatenated to the user's own master catalog.

CAT. TO BE PURGED? CAT2

PASSWORD? AOK

The dollar-sign is used only when the password is concatenated directly to a file or catalog name. The request

CAT. TO BE PURGED?

is reissued.

Example Short-Form Reply:

FUNCTION? PC,/CAT1$ABC/CAT2$AOK

6) FUNCTION? PF

CATALOG STRUCTURE TO WORKING LEVEL?

user-ID/cat-name$password/.../cat-name$password

FILE TO BE PURGED? file name

PASSWORD? password

Example Replies (user responses are underlined):

FUNCTION? PF

CATALOG STRUCTURE TO WORKING LEVEL?

JDOE/CAT1$ABC/CAT2$AOK

The user in this case is ALLONG, not the file creator.

FILE TO BE PURGED? FIL1

PASSWORD?

The user (ALLONG) was given specific purge permission at the level of CAT2.

The request

FILE TO BE PURGED?

is reissued.

Example Short-Form Reply:

FUNCTION? PF,JDOE/CAT1$ABC/CAT2$AOK/FIL1

7) FUNCTION? RC

The question/response sequence and the short-form reply for this function are completely analogous to those for the Purge Catalog function. The Release Catalog function would normally be used in preference to Purge Catalog -- as it is more economical -- unless the user has a very stringent file-security requirement.

8) FUNCTION? RF

The question/response sequence and the short-form reply for this function are completely analogous to those for the Purge File function. The Release File function would normally be used in preference to Purge File -- as it is more economical -- unless the user has a very stringent file-security requirement.

9) FUNCTION? MC

CATALOG STRUCTURE INCLUDING CATALOG TO BE MODIFIED?

user-ID/cat-name$password,...,cat-name$password

NEW NAME? new cat-name

$$\text{PASSWORD?} \left\{ \begin{matrix} \text{new password} \\ \text{DELETE} \end{matrix} \right\}$$

$$\text{GENERAL PERMISSIONS?} \left\{ \begin{matrix} \text{access type,...,access type} \\ \text{DELETE} \end{matrix} \right\}$$

$$\text{SPECIFIC PERMISSION?} \left\{ \begin{matrix} \text{access type,...,access type/} \\ \text{user-ID.../user-ID} \\ \text{DELETE/user-ID/.../user-ID} \end{matrix} \right\}$$

Example Replies (user responses are underlined):

FUNCTION? MC

CATALOG STRUCTURE INCLUDING CATALOG TO BE MODIFIED?

/CAT1$ABC/CAT2$AOK

NEW NAME?

A carriage return only response means that the catalog name is to remain unchanged.

PASSWORD? XYZ

The original password AOK is replaced by XYZ.

GENERAL PERMISSIONS? READ

As originally created, general permissions were not assigned at this level. This response replaces this null set with READ permission.

SPECIFIC PERMISSION? R,W/BJONES

This response replaces the original specific READ permission for BJONES with READ and WRITE permission.

SPECIFIC PERMISSION? DELETE/ASMITH

This response cancels any permissions for ASMITH that previously existed.

SPECIFIC PERMISSION? R,W,P,M/ALLONG

This response replaces the original set of READ, WRITE and PURGE permissions for ALLONG with READ, WRITE, PURGE, and MODIFY.

SPECIFIC PERMISSION?

The carriage return implies that no further modifications are to be made; the changes are now processed and the question

CATALOG STRUCTURE INCLUDING CATALOG TO BE MODIFIED?

is reissued.

Special Short-Form Option Formats

To rename a catalog:

```
NEWNAME/catalog/
```

To exclude, by user-ID, from any general permissions:

```
EXCLUDE/user-ID,..., user-ID/
```

To delete specific permissions, by user-ID:

```
DELETE/user-ID,...,user-ID/
```

To delete all general permissions:

```
DELETE/GEN'L (or simply DELETE)
```

Note. EXCLUDE and DELETE may not be abbreviated.

Example Short-Form Reply

```
FUNCTION?  MC,/CAT1$ABC/CAT2$AOK,PASSWORD/
?XYZ/,R,R/BJONES,ALLONG/,W/BJONES,ALLONG/,
?P/ALLONG/,M/ALLONG/,DELETE/ASMITH/,EXCLUDE/
?ASMITH/
```

10) FUNCTION? MF

CATALOG STRUCTURE INCLUDING FILE TO BE MODIFIED?

user-ID/cat-name$password/.../ cat-name$password/file name$password

NEW NAME? new file name

NEW MAX SIZE? new maximum size (in blocks)

PASSWORD? $\left\{\begin{array}{l}\text{new password}\\ \text{DELETE}\end{array}\right\}$

GENERAL PERMISSIONS? $\left\{\begin{array}{l}\text{access type,...,access type}\\ \text{DELETE}\end{array}\right\}$

SPECIFIC PERMISSION? $\left\{\begin{array}{l}\text{access type/user-ID/.../user-ID}\\ \text{DELETE/user-ID/.../user-ID}\end{array}\right\}$

Example Replies (user responses are underlined):

FUNCTION? MF

CATALOG STRUCTURE INCLUDING FILE TO BE MODIFIED?

/CAT1$ABC/CAT2$XYZ/FIL1

NEW NAME? MASTER1

NEW MAX SIZE? 20

This response increases the maximum file size to 20 blocks (originally 12).

PASSWORD? DEPT37

This response attaches the password DEPT37 to this file (none originally assigned).

GENERAL PERMISSION? DELETE

The original general READ permission is deleted.

SPECIFIC PERMISSION? P/BJONES

PURGE permission for user BJONES is added at this level. This permission applies to this file only, but he also has READ and WRITE from the CAT2 level.

Special Short-Form Option Formats

To rename a file:

NEWNAME/file name

To exclude, by user-ID, from any general permissions:

```
EXCLUDE/user-ID,...,user-ID/
```

To delete, by user-ID, specific permissions:

```
DELETE/user-ID,...,user-ID/
```

To delete all general permissions:

```
DELETE/GEN'L/

    or

DELETE
```

Note. EXCLUDE and DELETE may not be abbreviated.

Example Short-Form Reply:

```
FUNCTION? MF,/CAT1$ABC/CAT2$XYZ/FIL1,N/

? MASTER1/,B/20/,P/DEPT37/,DELETE,P/BJONES/
```

11) FUNCTION? LC

CATALOG STRUCTURE INCLUDING CATALOG TO BE LISTED?

user-ID/cat-name,...,cat-name

Example Replies (user responses are underlined):

```
FUNCTION?  LC

CATALOG STRUCTURE INCLUDING CATALOG TO BE LISTED?

/CAT1
```

Passwords need not be given in the catalog structure. A user is permitted to list only his own catalogs on the LIBRARY catalog.

A list of the catalogs and files emanating from CAT1 would now be output.

12) FUNCTION? LS

CATALOG STRUCTURE INCLUDING CATALOG OR FILE TO BE LISTED?

user-ID/cat-name,...,cat-name(or)file name

Example Replies (user responses are underlined):

```
FUNCTION? LS

CATALOG STRUCTURE INCLUDING CATALOG OR FILE TO BE LISTED?

/CAT1
```

Passwords need not be given in the catalog structure and will not be included in the catalog or file description which is output. A user can list only his own catalogs or files.

The description of CAT1 would now be output.

The system will provide the following information about the catalog or file:

```
FILE NAME-
ORIGINATOR-
DATE CREATED-
DATE CHANGED-MONTH/DAY/YEAR (T.O.D.)
LAST DATE ACCESSED-
MAX FILE SIZE-
CURRENT FILE SIZE-
FILE TYPE-RANDOM(xxxxxx)
  where: xxxxxx  is file type if ASCII or IDS
DEVICE-
GENERAL PERMISSIONS-
SPECIFIC PERMISSIONS-
FILE IN ABORT STATUS (if file is in abort status)
```

Examples of Line Delimiter Use

The line delimiters can be used in several ways to either shorten the question/response sequence, or terminate a function at any given point.

Examples of the effect of different response terminations are as follows:

FUNCTION? CC

The carriage return alone implies a master catalog.

NEW CATALOG? 001*

Passwords or permissions are not wanted for this catalog and no further questions are wanted. Return is to NEW CATALOG? level.

NEW CATALOG? 002

PASSWORD? PASS2**

No permissions are to be assigned to this catalog, and creation of catalogs at this position is finished. Return is to function level.

```
FUNCTION?  CF

CATALOG STRUCTURE TO WORKING LEVEL?

/002$PASS2

FILE NAME, SIZE (IN BLCKS), MAX SIZE?  02.1,1,3

GENERAL PERMISSIONS?  READ

SPECIFIC PERMISSION?  W/RJJONES**
```

Creation of files at this level has been completed.

```
FUNCTIONS? carriage return  (or DONE)
```

Finished with ACCESS.

```
SYSTEM?
```

Return to the subsystem-selection level.

Special Features

Files created by means of the Create File function are not necessarily contiguous; i.e., successive links of a multi-link file are not necessarily in physical sequence on the storage device. Furthermore, both the Create File and Access File functions assume that the file will be treated as a linked file. For the standard subsystems provided with the Time-Sharing System, these file characteristics are suitable because linked files are required.

If, however, in the use of a given subsystem, it would be advantageous to have contiguous files, this characteristic can be specified in response to FILE NAME, SIZE (IN BLCKS), MAX SIZE?. The form of this response is:

file name, initial size C

The parameter C indicates, in Create File only, that a contiguous file is desired. No maximum size may be specified.

Similarly, if random treatment of files is required in a given user-written subsystem, a file can either be created as a random file or accessed as a random file. If created as such, it is always treated by the GECOS I/O Supervisor as a random file. If it is created as a linked file, it can be accessed as a random file, but in that case, the random treatment indication is temporary; i.e., it applies to that access only.

The forms of the random specification are as follows:

For Create File, the response to FILE NAME, SIZE (IN BLCKS), MAX SIZE? is:

file name,initial size,maximum size,R

or

file name,initial size C,R

For Access File, the response to FILE NAME$PASSWORD? is:

file name,R$password

In both responses, the parameter R (always preceded by a comma) indicates that the named file is to be treated as a random file.

In the case of Create File only, the additional question LOGICAL RECORD SIZE? is asked, allowing the user to specify a fixed logical-record-size attribute as required of random files by TSS FORTRAN. If this attribute is not needed, the user may respond with simply a carriage return.

In the short-form response, random files can be specified by:

```
MODE/RAND/    or MODE/R/
```

Linked files can be specified explicitly, either by:

```
MODE/LINKED/  or MODE/L/

        or

MODE/SEQ      or MODE/S/
```

or, more simply, by default.

Contiguity cannot be specified in the short-form response.

Request Denied Messages

The following messages are printed following a complete function request, and indicate that the request could not be satisfied. The reason for denial is given in each case.

REQUEST DENIED-NEW NAME SAME AS AN EXISTING NAME

A new catalog or file name has been given that is the same as an existing catalog or file name at the same level.

REQUEST DENIED-FILE SPACE REQUESTED EXCEEDS ALLOWED

The user has requested file space exceeding the amount that has been allotted to him in his System Master Catalog entry.

REQUEST DENIED-NEW SIZE LESS THAN CURRENT SIZE

In MODIFY FILE, a new file size has been specified which is less than that currently used by the file.

REQUEST DENIED-SYSTEM MALFUNCTION

An unrecoverable I/O error has occurred.

REQUEST DENIED-PERMISSION NOT GRANTED

The user does not possess the requested permission(s).

REQUEST DENIED-FILE BUSY

The requested file is currently busy to the type of permission(s) requested.

REQUEST DENIED-INCORRECT CAT/FILE DESCRIPTION

This denial is given whenever required passwords are not included or the catalog/file description is not logically correct.

REQUEST DENIED-SYSTEM LOADED

The requested file function cannot be completed because there is temporarily no file space available.

REQUEST DENIED-YOUR AVAILABLE FILE TABLE IS FULL

The user has too many files accessed (open) at the same time. This situation can be eliminated by deaccessing some of the accessed files.

REQUEST DENIED-FILE NAME A DUPLICATE, MUST GIVE ALTERNATE NAME

An ACCESS FILE has been done where the file name is a duplicate of a file which the user currently has open. The alternate name capability can be used to avoid this situation.

REQUEST DENIED-YOU CAN LIST ONLY "YOUR" OWN CATALOGS OR FILES

The user has requested a listing of catalogs and/or file descriptions (List Catalog or List Specific) for catalogs and/or files that do not emanate from his own user's-master-catalog.

REQUEST DENIED-UNDEFINED STATUS

A system malfunction (other than an internal I/O error) occurred during the attempt to satisfy the requested function. Contact the resident system-maintenance authority or the central computer site.

Input Error Messages

The following messages are printed immediately following the input in error and the original question is repeated.

ERR-ILLEGAL CHARACTER

A character other than an alphabetic, numeric, period, or dash has been included in an identifier. An upward arrow (↑) points at the character in error.

ERR-INVALID DELIMITER

An otherwise valid delimiter has been given out of place. An upward arrow (↑) points at the delimiter in error.

ERR-XXXXXXXXXXXX-MUST BE LESS THAN 13 CHARACTERS

The designated identifier is limited to 12 characters.

ERR-XXXX-IS NOT A LEGITIMATE PERMISSION

Legitimate permissions are READ, WRITE, APPEND, and EXECUTE, plus PURGE and MODIFY as specific permissions only.

ERR-XXXXXXXXXXXX-MUST BE LESS THAN 9 CHARACTERS

The designated identifier is limited to 8 characters.

ERR-XXXX-MUST BE AN INTEGER

A non-numeric character has been included in field XXXX.

ERR-XXXXX-MUST BE LESS THAN 10000

The field is limited to four digits.

ERR-INPUT REQUIRED

A null response was given to a question which requires input.

ERR-INITIAL SIZE GREATER THAN MAX SIZE

In defining the file size, an initial size greater than the maximum size was given.

RECOVERY SUBSYSTEM

The RECOVERY subsystem gives the terminal user the ability to make his collector file (see page 13 for definition of collector file) permanent and to catalog it under his System Master Catalog (SMC). Thus a user has the ability to recover his last input lines in any situation where an accidental or unexpected disconnect occurs.

The collector file will contain the lines of data entered via the terminal which have not yet been edited into the current file (see page 13 for definition of current file). The number of lines in the collector file may vary because of line length and the amount of data entered since the last edit of the collector file to the current file. In general, it will contain up to the last 70 lines. The RECOVERY subsystem, when used with the OLDP and NEWP functions (see pages 18 and 20 for definition), provides the terminal user with the ability to recover the entire file when a disconnect occurs.

The RECOVERY subsystem is initiated through the common command langauge of the following systems:

BASIC

Time-Sharing FORTRAN

Text EDITOR

CARDIN

The terminal user can request the RECOVER command at any level; that is, at the subsystem selection level or at the command level.

The RECOVERY subsystem also permits the use of the ROLLBACK command to recover the collector file at the user's next terminal session. The ROLLBACK command can be issued only at the command level.

RECOVERY Subsystem Operation

In its basic operation, the RECOVERY subsystem dumps data currently on the temporary input collector file to the working file and creates and/or accesses a permanent file specified in the command (by filename) with an alternate name. If this permanent file already exists in the user's System Master Catalog, the file is checked to assure that it conforms to the minimum requirements for an input collector file. The major requirements for an input collector file are that the file must be a random file and it must be at least 640 words (two blocks) long. A longer file will be accepted, but only 640 words will be used.

If the filename is not in the user's SMC, a file will be created and given predefined attributes. It will then be accessed by an alternate name. Accessing the file by the alternate name puts the alternate name in the Peripheral Allocation Table (PAT). The RECOVERY subsystem then switches the two PAT's; one PAT representing the temporary input collector file and the other PAT representing the permanent recovery file. Thus, all reference to the input collector file now points to the PAT describing the permanent recovery file.

The procedure for termination, user log-off, or disconnect is the reverse of the procedure described above. The Available File Table (AFT) will contain at least two PAT's (assuming that recovery was requested). The names associated with these PAT's will be switched and the files will be deallocated by the TERM module of the Time-Sharing System.

When the terminal user issues the ROLLBACK command, the RECOVERY subsystem will again copy any data currently on the temporary input collector file to the current working file. It will then access the file specified in the command. When accessed, the permanent recovery file is read and any data in this file is also copied on the current working file. The last line of good data on this file, preceded by an identifying message, is printed out on the terminal. Thus, when the user receives a SUCCESSFUL message following a ROLLBACK command, he is ready to type into an empty 640-word collector file.

A terminal user may issue any number of RECOVERY and/or ROLLBACK commands during his session at the terminal. When subsequent commands are issued, the previous RECOVERY file is deaccessed, and a new RECOVERY file is created and/or accessed. The perm file remains in the user's catalog until he specifically releases it.

If an error occurs during the creation and/or accessing of the new RECOVERY file, the terminal user will be working with a temporary input collector file and not his RECOVERY file. The data on the RECOVERY file may be unrecoverable (because of a missing end-of-file) if the terminal user tries to access this file through any other subsystem.

Questions and Responses

The following paragraphs describe sets of questions and general responses associated with the RECOVER and ROLLBACK commands. In these descriptions, the general response to each question is underlined to set it apart.

```
SYSTEM? RECO FIL1$ABC
SUCCESSFUL
```

The RECOVERY subsystem is called to create and/or access FIL1. Control is then returned to the subsystem selection level SYSTEM?. #RECO cannot be given at this level.

```
SYSTEM? BASIC
OLD OR NEW - OLD FIL2
READY
* RECO FIL3
SUCCESSFUL
```

The user has specified that FIL2 be written on his current working file. RECOVERY subsystem is then called to create and/or access FIL3. Control is returned to the previous calling level.

```
SYSTEM? EDIT
OLD OR NEW - NEW
READY
*#REC FIL4
SUCCESSFUL
```

The user requests the EDIT subsystem and a current working file. At the command level, the user calls for RECOVERY to create and/or access FIL4. Control is returned to the previous calling level.

```
SYSTEM? BASIC
OLD OR NEW - NEW
READY
*10 PRINT
*20 PRINT
*RECO FIL5$BCA
SUCCESSFUL
```

The BASIC Editor is called to sort and merge lines 10 and 20 onto the current working file. RECOVERY subsystem is then called to create and/or access FIL5. Control is returned to the previous calling level.

```
SYSTEM? BASIC
OLD OR NEW - NEW
READY
*RECO FIL6$CAB
*10 PRINT
*20 PRINT
*30 PRINT
```

Assume that at this point there is a GE-600 disconnect. The user will do the following to recover his last input lines.

```
SYSTEM? BASIC
OLD OR NEW - NEW
READY
*ROLL FIL6$CAB
SUCCESSFUL
```

Prior to the disconnect, this operation was normal. When the system is restarted after the disconnect, the user calls in the RECOVERY subsystem by issuing the ROLLBACK command. The RECOVERY subsystem will access FIL6 and sort and merge the data onto the current working file. When the SUCCESSFUL message is issued, the user is ready to type into an empty 640-word collector file. Return is to the previous calling level.

Error Messages

The following error messages are printed on the terminal following a complete request.

FILE IS NOT 2-BLOCKS LONG

> The requested file size is less than the minimum size required. (Create by other subsystems.)

THIS FILE IS NOT A RANDOM FILE

> The requested file is not a random file. (Created by other subsystem.)

THIS FILE IS NOT A PERM FILE

> The requested file is not a permanent file in the user's SMC. Files cannot be subcataloged.

FILE <u>filename</u>, STATUS XX

> When the requested file was created and/or accessed, an illegal status occurred.

RECOVERY FILE--I/O STATUS XX

> The RECOVERY file is read and there is a bad return status.

IMPOSSIBLE TO RECOVER DATA FROM RECOVERY FILE

> Two 640-word blocks were read in and the last good line of data could not be determined.

Error Messages with Response

INVALID INPUT RETYPE

> This error message indicates that the filename and/or the password is too long or the filename contains an illegal character. The response may be:
>
> Filename$password (the corrected format)
>
> or
>
> Carriage Return (Return to previous calling level)

After two unsuccessful attempts the following message is given and return is made to the previous calling level:

```
INVALID INPUT
```

TIME-SHARING MEDIA CONVERSION PROGRAM

The Time-Sharing Media Conversion Program is a batch-world program that may be run either at the central computer site or through a remote/batch (GERTS) terminal. Its purpose is to generate a standard format, time-sharing text file from a suitable card deck, or conversely, to produce a card deck from such a file, however generated, thus saving the file in card form.

Operational Description

The media conversion program will perform the following functions:

- INPUT - create a standard format, time-sharing text file from cards. If the INSERT or MOVE option is used, # signs will be inserted between the line number and the first character of numeric data.
- OUTPUT - create a card deck from a standard format, time-sharing text file. # signs between the line number and the text will be deleted.

INPUT identifies the control card requesting the file-creating function and takes the following mutually exclusive options:

Option	Result
ASIS,i,j	The text file will be generated from the input cards, from the columns specified by i to j. Standard columns (default option) for i to j are 1 to 80.
MOVE,i,j,m,n	The text file will be generated from the input cards, from the columns specified by i to j. Line numbers will be taken from columns specified by m to n. Standard columns for i to j are 1 to 72, and for m to n are 73 to 80.

INSERT,i,j,m,n — The text file will be generated from the input cards and from the columns specified by i to j. Lines will be sequence-numbered, starting with m and incremented by n. Standard columns for i to j are 1 to 72. Standard values for both m and n are 10.

ASCII — The text file will be generated from input cards, using a binary deck previously punched from this program.

COMDK,option — The text file will be generated from input cards consisting of a COMDECK (compressed source deck). This option is used in conjunction with the ASIS, MOVE, or INSERT options. If ALTER's are to be made at the time the file is generated, a $ DATA I*,,COPY and a $ ENDCOPY card must be employed.

Sample INPUT Control Cards

```
INPUT,MOVE,1,60,73,80
```

Text file data is to be taken from columns 1 to 60 of the punched cards and line numbers are to be taken from columns 73 to 80.

```
INPUT,COMDK,ASIS,1,80
```

Text file data is to be taken from columns 1 to 80 of the input cards (a COMDECK).

INPUT can start in any column of the control card but no imbedded blanks are allowed.

OUTPUT identifies the control card requesting the card-deck producing function, and takes the following mutually exclusive options:

Option	Result
ASIS,i,j	The text file is read and a BCD card deck is punched in the columns specified by i to j. Standard columns (default option) for i to j are 1 to 80.

MOVE,i,j,m,n,l

The text file is read and a BCD card deck is punched, moving data to columns specified by *i* to *j*. Line numbers are moved to columns specified by *m* to *n*, right-justified. L specifies the label to be punched starting in column 73, left-justified. Standard columns for *i* to *j* are 1 to 72 and for *m* to *n*, 73 to 80.

STRIP,i,j

The text file is read and a card deck is punched, stripping off line numbers, with data moved to the columns specified by *i* to *j*. Standard columns for *i* to *j* are 1 to 80.

Note

With the above output options, data is converted from ASCII to BCD before punching.

ASCII

The text file is read and a binary deck containing the file text is punched. (See "Binary Card Format" below.)

Sample OUTPUT Control Card

```
OUTPUT,ASIS,1,56
```

The text file is punched into columns 1 to 56 of the card deck.

OUTPUT can start in any column of the control card but no imbedded blanks are allowed.

Definitions

- Each line will be punched on a separate card, starting in the column specified (OUTPUT function).
- A line number is an initial string of numeric characters which terminate with a nonnumeric character. Blank is such a nonnumeric character.
- In the case of the MOVE option, the line numbers are stored right-justified in the columns specified.

- The format of a line in a text file is as follows:

  (nnn) dddd....d (carriage return character)

  where *nnn* is the line number (optional; must be all numeric and must terminate with a nonnumeric character) and *dd...d* is the data.

  The carriage return character terminates the data line.

Errors

SE ABORT - A binary card is out of sequence.
Card number is printed out.

CK ABORT - Checksum of card does not agree with the computed checksum.

NB ABORT - First data card is not binary, but ASCII was specified on control card.

CP ABORT - No control card found (keyword may be misspelled).

DATA LINE TOO LONG FOR I,J FIELD

...portion of the line specified by *i* to *j*...

- (Occurs on OUTPUT only.) If a line of the file is too long for the specified *i* to *j* field (i.e., nonblank characters are being discarded), this warning message will be issued along with the portion of the line specified by *i*,*j*. A maximum of 20 such messages will be given. The complete file will be punched, as specified by the *i* to *j* field options.

Binary Card Format

Word 1	7/9 punch and number of data words (maximum=21)
Word 2	Checksum
Word 3	Card number, starting at 0
Words 4-24	Text

Sample Deck Setups

A sample deck setup to accomplish media conversion is as follows:

```
     $  SNUMB   XXXXX
     $  IDENT   account number,name
     $  USERID  name$password
     $  PROGRAM TSCONV
     $  PRMFL   OT,R/W,L,userid/filename
     INPUT,ASIS
          .
     (Data deck)
          .
     $  ENDJOB
     ***EOF
```

A sample deck setup to accomplish media conversion in the case of a COMDECK plus ALTER cards is as follows.

```
     $  SNUMB   XXXXX
     $  IDENT   account number
     $  USERID  name$password
     $  PROGRAM TSCONV
     $  PRMFL   OT,R/W,L, userid/filename
     $  DATA    I*,,COPY
     INPUT,COMDK,ASIS,1,80
          .
     (Data cards -- COMDECK)
          .

     $  ENDCOPY
     $  UPDATE
          .
     (ALTER deck)
          .

     $  ENDJOB
     ***EOF
```

NOTE: As indicated in these deck setups, OT is the only file code used. IN is not used for the input file.

FORTRAN TRANSLATOR SUBSYSTEM

The FORTRAN Translator (TRAN) is a time-sharing subsystem that allows the user to translate a GE-600 Time-Sharing FORTRAN source program into a batch FORTRAN source program, with little or no "hand recoding". The translator substitutes, wherever possible, one or more batch-acceptable statements for each noncompatible TSS FORTRAN statement.

Translation occurs in the input/output and data-specification statement categories. Several statement forms, primarily in the I/O area, are not automatically translatable. In these cases, the translator issues a message noting the untranslatable statement and pauses to allow the terminal user to enter a replacement statement. Thus, all noncompatible statements are detected by the subsystem and a large majority are automatically translated.

The user has the option of saving the translation either in BCD (batch S*) form, in ASCII (time-sharing) form, or in both. A BCD file of the final translated program is provided for batch processing and is available to the batch dimension via the file system. Batch-compiler input can be called directly from the BCD permanent file with a $ SELECT or a $ PRMFL S* card in the batch-job control deck. For example, if the control deck was submitted by means of the CARDIN subsystem, it would appear as follows:

```
0010$:IDENT:JDOE
0020$:USERID:JDOE$PASSWORD
0030$:FORTRAN
0040$:SELECT:JDOE/BCD
0050$:ENDJOB
```

where BCD is the name of a BCD-translation file from the FORTRAN Translator.

The ASCII form is useful for obtaining a listing at the terminal; the listing may then be further updated or modified and passed directly to GEIN via CARDIN:

```
**0001$:IDENT:JDOE
**0002$:FORTRAN
  0008 (Comment inserted by Translator)
  0009  LOGICAL KK001  (Inserted by Translator)
    .  )
    .  } ASCII translation data
    .  )
  0990
**1000$:ENDJOB
```

where lines marked ** are inserted by the user in CARDIN build-mode and lines 0008 through 0990 represent an ASCII-translation file from the FORTRAN Translator, named, for example, ASCII.

For both examples above, the CARDIN sequence would be as follows (user responses underlined):

```
SYSTEM?CARDIN
OLD OR NEW-NEW      (for BCD example)
      (or) OLD ASCII (for ASCII example)
READY
*
.   (enter control cards as per examples above)
.
*RUN
CARD FORMAT AND DISPOSITION?  NORM
```

The NORM response implies MOVE and standard tab character and settings:

```
:,8,16,32,73
```

General Usage

The translator takes its program input from the user's current file (as do most other time-sharing subsystems).

The user selects the FORTRAN Translator with the name TRANslator (only the first four characters are needed) at the subsystem-selection level (SYSTEM?). He then normally responds OLD filename to the OLD OR NEW - question, naming the time-sharing (ASCII) file that contains the program to be translated. If, however, the current file already contains the desired program, he may simply respond SAME to OLD OR NEW -.

Since the translator may replace a single TSS FORTRAN statement with one or more batch statements, the user should insure that his input file is regularly line-ordered and that his line numbering has an origin of ten or greater, with increments of at least 10.

The translator uses FORTRAN statement numbers, or "statement-label" numbers (not line numbers), of 32000 and subsequent for created format statements that replace quoted data in PRINT statements. Therefore, the user must replace any statement numbers in this range to avoid duplicate reference.

Following the response to OLD OR NEW -, the translator issues a series of questions (described in detail below) which permit the user a number of options in the operation of the translator. The options allow the user to:

- Choose a BCD or ASCII save file, or both, as explained above.
- Choose to have line numbers either stripped, moved to the label field (cols. 73-80) of the batch-statement card, or moved with a constant prefix. (Applies to the BCD save file only.)
- Choose to have a listing of the translation in progress.
- Control the assignment of file codes, including a save, recall, and modification of a file-code table.

Detailed Usage

The conversation between the system and the user, beginning with the selection of the translator, is as follows:

SYSTEM? TRANS

OLD OR NEW - OLD $\underline{\text{filename}}$

BCD SAVE FILE NAME? $\left\{\begin{array}{l}\underline{\text{filename}}\\ \underline{\text{carriage}}\text{ return}\end{array}\right\}$

If the user wishes a translation file in BCD form, for batch input, he specifies the name of a file, previously defined or not, upon which the translated program is to be saved. If he does not want a BCD save file, he simply responds with a carriage return.

The following question is asked only if the user has responded $\underline{\text{filename}}$ to the question above:

LABELS? $\left\{\begin{array}{l}\text{MOVE or carriage return only}\\ \text{STRIP}\\ \underline{\text{abcde}}\ (i_1, j_1);\ \underline{\text{fghij}}\ (i_2, j_2)\ldots\end{array}\right\}$

This question requests information about what is to be placed in the label field (columns 73-80) of the source statement on the BCD file. The meaning of the responses shown is as follows:

MOVE or carriage return only -- move the line numbers found in the input file into the label field of the BCD file.

STRIP -- ignore the line numbers and leave the BCD label field blank.

abcde (i_1 ,$_1$j)... -- move the line numbers to the label field prefixed by the specified alphanumeric characters, where:

abcde represents the alphanumeric label prefix,

i represents the starting line number, and

j the final line number to which this prefix is to be added.

Multiple sequences of line numbers, with different prefixes can be specified. If an interval of line numbers is found that has not been specified by the user, only the line number will be placed in the label field.

Note that: abcde (i_1, j_1) = abcde (j_1, i_1)

abcde (,j_1) = abcde (0, j_1)

abcde (i_1,) = abcde (i_1, 99...9)

ASCII SAVE FILE NAME? $\left\{\begin{array}{l}\text{filename}\\ \text{carriage return}\end{array}\right\}$

If the user wishes to save a time-sharing version of his translation file so that it may be updated or modified from a terminal, he may designate a new or previously defined file upon which to save the translated data. A carriage-return only indicates that an ASCII file is not desired.

Note

The user must designate either a BCD or an ASCII translation file, or both. If he fails to do so, the following message is issued:

NO TRANSLATION FILE REQUESTED

and the user is returned to the SYSTEM? level.

LIST? $\left\{\begin{array}{l}\text{YES or Y}\\ \text{carriage return}\end{array}\right\}$

The user is asked if he wants an on-line listing of the translation while it is in progress. A carriage-return only indicates no listing is desired and in this case, only the fatal errors will be printed at the terminal. If a listing is desired, the original time-sharing FORTRAN statement will be printed, immediately followed by its translation. The translation may consist of a reproduction of the original statement, if no change is necessary, or of one or more substituted statement(s).

COMMENT? $\left\{\begin{array}{l}\underline{\text{any data}}\\ \text{carriage return}\end{array}\right\}$

In order to allow the user to distinguish one translation from another, he is asked for a comment card (on a BCD file, it will appear as a label preceding each page of his listing). A carriage-return only indicates that no comment is desired; otherwise, the user may type any data that he desires and it will be inserted as the first record in the BCD file or line number 8 of his ASCII translation file.

FILE TABLE FILE? $\left\{\begin{array}{l}\underline{\text{filename}}\\ \text{carriage return}\end{array}\right\}$

A major portion of the time-sharing to batch translation is the replacement of permanent time-sharing file names by numeric file codes. The translator builds a table of these associations, which the user may save on a permanent file from one execution of the translator to the next. In response to this query, the user may type the name of a file-table file previously generated by this subsystem. Refer to the SAVE FILE TABLE? question below. A carriage-return only indicates no previous file-table file for this program.

MODIFY FILE TABLE? $\left\{\begin{array}{l}\text{YES or Y}\\ \text{carriage return}\end{array}\right\}$

The user is allowed to specify which file names in his time-sharing program are to be associated with particular file codes for this translation (whether or not he indicated a prior file-table file), as follows.

The following pair of questions are conditional upon a YES or Y response to the question MODIFY FILE TABLE?.

FILE CODE? $\left\{\begin{array}{l}\text{1 through 43}\\ \text{carriage return}\end{array}\right\}$

FILE NAME? $\left\{\begin{array}{l}\underline{\text{filename}}\\ \text{carriage return}\end{array}\right\}$

If a positive response was given to the modification query, the user is asked to type the numeric file code and the file name referenced in the time-sharing program to be associated with it. The file-code/file-name questions will be repeated until the user responds with a carriage return only to either question. If the user types an illegal file code, the subsystem will issue the following message:

```
INVALID FILE CODE (1 TO 43 ONLY)
```

and repeat the file-code question. When modifying a previously existing file-table entry gotten from a prior file-table file, the old file-code/file-name association must be either explicitly replaced or blanked out; otherwise duplication occurs. For example, if the prior file table contains the association:

FILE CODE	FILE NAME
07	XYZ

and the user wishes to switch the name XYZ to file code 10, he must respond as follows (user-responses underlined):

```
FILE CODE? 07
NAME? carriage return
FILE CODE? 10
NAME? XYZ
```

Otherwise, the file table will appear as:

FILE CODE	FILE NAME
07	XYZ
10	XYZ

LIST FILE TABLE? $\left\{\begin{array}{l}\text{YES or Y}\\ \text{carriage return}\end{array}\right\}$

The user may choose to list the content of his file table which resulted from this execution, after the input file has been translated. An example file table list is as follows:

FILE CODE	FILE NAME
01	AFILNAM
07	XFILE
.	.
.	.
41	LASTFIL

SAVE FILE TABLE? $\left\{\begin{array}{l}\underline{\text{filename}}\\ \text{carriage return}\end{array}\right\}$

The user may save the content of the file table from this execution for a later execution of the translator. The named file may or may not be a previously defined permanent file.

Fatal Errors During Translation

Each TSS FORTRAN statement is examined to see if it is an acceptable batch statement. There are four possible results of this examination:

- No data change -- statement is batch compatible.
- A single translated statement.
- Multiple translation statements.
- A fatal error--no translation possible. In this case, the subsystem will print the fatal statement and issue the following message:

  ```
  FATAL ERROR, REPLACE BY:
  ```

  The user may type in a statement to replace the original. He should not reenter either the line number or formula number of the statement, but only the statement itself. The line number and formula number, if they existed, will be carried over from the original statement.

If the user does not wish to replace the statement, he may change it into a comment by giving a carriage-return only in response to the message. Alternatively, if he does not want to replace the statement and wants it converted to BCD "as is," he may indicate this by responding with #carriage return; i.e., a pound sign and carriage return. This is useful where the target compiler may contain extended implementations; e.g., ENCODE/DECODE. (Refer to "Sample Non-Translatable Statements" below.)

Sample Translated Statements

The following are sample translations:

```
1)    9       LOGICAL KK001
```

This statement will be inserted into each translation to provide for EOF processing.

```
2)    100     ASCII ABC,NAME,XYZ
      100     INTEGER ABC,NAME,XYZ
```

```
3)    200     FILENAME FILE1,FILE2,FILE3,X
      200     INTEGER FILE1,FILE2,FILE3,X
```

All file name variables will be saved in a table, so that if equated in a quoted expression, the appropriate file code will be substituted.

```
4)    300     X = "FILNAM"
      300     X = 07
```

The file code substituted will be the first available one, or one provided by a previous file table file, or one designated by the file table modification.

```
5)    400     BACKSPACE "FILNAM"
      400     BACKSPACE 07
```

```
6)    500     ENDFILE "FILNAM"
      500     ENDFILE 07
```

```
7)    600     BEGINFILE "FILNAM"
      600     REWIND 07
```

There may exist some differences in REWIND processing in time-sharing and batch execution.

```
 8)    700     CLOSEFILE "FILNAM"
       700     REWIND 07


 9)    800     CALL SUBR (A,B,"QUOTES",X,Y)
       800     CALL SUBR (A,B,6HQUOTES,X,Y)


10)    900     DATA ABC/"QUOTES"/
       900     DATA ABC/6HQUOTES/


11)    1000#   120 FORMAT (2F6.2,"QUOTES",E12.5)
       1000#   120 FORMAT (2F6.2,6HQUOTES,E12.5)


12)    1200    IF (A.NOT.B) REWIND "FILNAM"
       1200    IF (A.NOT.B)
       1201   1REWIND 07


13)    1300    PRINT:"QUOTED DATA"
       1300    PRINT 32000
       1301#   32000 FORMAT (12H QUOTED DATA)


14)    1400    READ("FILNAM",150,END=500)A,B,C
       1400    CALL FLGEOF (07,KK001)
       1401    READ (07,150)A,B,C
       1402    IF (.NOT.KK001) GO TO 500


15)    1500    WRITE ("FILNAM",250)A,B,C
       1500    WRITE (07,250)A,B,C


16)    1600    X=Y;PRINT:"DATA";A=B
       1600    X=Y
       1601    PRINT 32001
       1602#   32001 FORMAT(5H DATA)
       1603    A=B
```

Sample Non-Translatable Statements

The following statements will be flagged by the translator as fatal errors:

```
 1)    100     A="QUOTES" where A has not been defined as a
               file name variable.

 2)    200     120 FORMAT (V)

 3)    300     ENCODE (a,n) list
```

4) 400 DECODE (a,n) list

5) 500 PRINT:A,B,C

6) 600 PRINT:"QUOTED DATA",X,Y,Z

7) 700 READ:A,B,C

8) 800 READ ("FILNAM"'100)A,B,C--random file processing

9) 900 WRITE ("FILNAM"'5)A,B,C--random file processing

The translator will not check the syntax of the input statements. Thus, the user should insure that his parentheses are balanced, expressions are not in mixed mode, etc. Translation will terminate when either an EOF is found on the input file or an END statement has been encountered.

WRITING SUBPROGRAMS IN ASSEMBLY LANGUAGE

It is possible, using GMAP, to write programs in assembly language which interface with Time-Sharing FORTRAN-produced object code. To do so, however, requires an understanding of the type of executable code generated by the Time-Sharing FORTRAN compiler; subroutine linkage and argument passing; and loading and linking of binary files in the Time-Sharing System. To this end, the following is a brief description of both the salient features of the object module produced by Time-Sharing FORTRAN, and the means by which compatible object modules using GMAP are generated.

Relocation

A standard GMAP-produced relocatable subroutine is composed of three collections of information: preface, or linkage, information; relocation information; and the object code to be loaded. In the batch environment, GELOAD uses the first two of these collections to assist in loading, relocation, and linking the third. When loading is finished, all linkage and relocation information is discarded, leaving only the object module in core for execution.

A standard Time-Sharing FORTRAN-produced relocatable subroutine is composed of only two collections of information: linkage information and the object code to be loaded. Output from the compiler does not include relocation information (i.e., "REL-BITS"); the code is itself relocatable. To illustrate, compare the following two versions of a hypothetical subroutine:

```
X    BSS    1           X    BSS    1
Y    BSS    1           Y    BSS    1
A    LDA    X           A    LDA    -2,IC
     STA    Y                STA    -2,IC
     TZE    B                TZE    3,IC
     STZ    X                STZ    -5,IC
     TRA    0,1              TRA    0,1
B    NULL               B    NULL
     ...                     ...
```

These two versions are functionally the same and, in the batch world, either could be used. They differ in that the first requires GELOAD to relocate references to locations X, Y, A, and B based on the load time origin of the subroutine. Once so modified (absolutized), this sequence is only executable when loaded with respect to that origin. The second version requires no address relocation at load time and can be executed from anywhere in core; the addresses are absolute quantities which, when coupled with the IC modification, address the desired locations.

This latter type of subroutine is probably best described as relocatable, but to avoid confusion with the current interpretation of that term, "floatable" or "self-relative" is generally used to describe it. In fact, a "relocatable" subroutine is as described for the former.

The technique used to write floatable subroutines involves interaction of the assembler's location counter with the computer's instruction counter. For example, the four instructions starting at location "A" of the above example could have been written as:

```
A     LDA     X-*,IC
      STA     Y-*,IC
      TZE     B-*,IC
      STZ     X-*,IC
```

thereby achieving the desired actual addresses of -2, -2, 3, and -5 respectively, while maintaining symbolic addressing.

To further facilitate writing routines of this sort, a feature has been added to the GMAP assembly language to recognize a dollar sign ($) in the modifier field as representing (-*,IC). For example, the instruction labeled "A" above can now be written:

```
A     LDA     X,$
```

Indirection is represented by the notation $*, e.g.,

```
A     LDA     X,$*
```

Note that this form of address modification precludes the use of a register (e.g., $AU) since the IC register is implied. For a more complete description of this GMAP feature, see the GE-600 Line Programming Reference Manual, CPB-1004.

Finally, the hypothetical subroutine is shown as it would be coded as a "floatable" module:

```
X     BSS     1
Y     BSS     1
A     LDA     X,$
      STA     Y,$
      TZE     B,$
      STZ     X,$
      TRA     0,1
B     NULL
      ...
```

Subroutine Linkage

In the batch environment, subroutine linkage is accomplished through use of SYMREFs and SYMDEFs. Symbols of this type are given a special type of relocation, and the Loader, upon encountering a SYMREF, transforms the specially coded address into an actual address. This type of linkage is not possible in time-sharing since relocation information is neither generated by the compiler nor recognized by the Loader.

In the time-sharing environment, linkage information is generated as part of the object module. Each module begins with linkage data pertaining to its main entry name. Linkage information for other entry names and external names are included in the object module body, threaded from the linkage for the main entry.

Before proceeding, it is necessary to define a few terms. A symbol which is an entry name is called a .SDEF.. A symbol which represents a location which is external to some object module is called a .SREF.. A collection of information describing a .SREF. or .SDEF. symbol for purposes of subroutine linkage is called a linkage block. That sequence of instructions, associated with an entry name, which accomplishes the necessary storage of registers and establishes the argument list pointer for internal usage is called the entry prologue.

A linkage block is a group of four words, which must begin at an even location, and contains the following information:

Word 1 - .SDEF. versus .SREF. indication
Word 2 - pointer to next linkage block
Words 3,4 - name encoded in ASCII

When the linkage block is for a .SDEF. symbol, Word 1 will contain an IC modified TRA instruction directed to the entry prologue for that .SDEF. symbol; for a .SREF., Word 1 is zero. When loaded, this .SREF. zero is changed to a TRA instruction to the first word of the matching .SDEF. linkage block, thus accomplishing the hook-up.

Word 2 contains a self-relative pointer to the next linkage block in this object module, if one exists, or is zero. That is, if the first of a pair of linkage blocks is at location M, and the second at location N, then the address field of Word 2 of the linkage block at M will contain the quantity N-M.

Words 3 and 4 contain, in ASCII, the .SDEF. or .SREF. symbol. Although Time-Sharing FORTRAN permits subroutine names of up to eight characters in length, names which must be known to both GMAP and FORTRAN subroutines should be limited to six characters. This restriction is based on the fact that use of the GE-supplied macros for defining .SDEF. and .SREF. symbols in GMAP requires that these symbols conform to the GMAP rules for symbol formation. Definition of eight character names in GMAP may be accomplished if necessary by hand-coding the linkage blocks.

Every object module is loaded beginning at a location modulo eight and has a size which is a multiple of eight. It begins with a linkage block for the first .SDEF. symbol. This first linkage block must in turn be followed by a word containing two size values: the size of the object module in the upper half, and the amount of required common storage in the lower half.

Inter-Subroutine Communication

Transfer from one subroutine to another is made using the TSX1 instruction. Control proceeds from the TSX1 to the linkage block for the instruction to the linkage block for the .SDEF. symbol of the same name in the object module containing that entry. This entry linkage block contains a TRA instruction to the prologue for that entry. The prologue is of the form:

```
     SREG    X,$
     EAX2    0,1
```

thus accomplishing the register safe-store, the establishment of the argument list pointer in Index Register 2, and the release of Index Register 1 for further subroutine calls. Location X represents an eight-word register safe-store area, normally adjacent to the prologue. When the prologue code is generated out-of-line, it may be concluded by a TRA to the actual first instruction of the entry.

As of this writing, Time-Sharing FORTRAN does not have a multiple entry statement and, as such, each subroutine has only one .SDEF. symbol. Thus the first 18 locations of all Time-Sharing FORTRAN compiled subroutines appear as follows:

```
LOC  INSTR  ADDR,MOD      EXPLANATION

 0   TRA    16,IC         LINK BLOCK WORD 1. GO TO PROLOGUE.
 1   ZERO   SREF1+1-*     POINTER TO LINK BLK OF 1ST SREF
 2   UASCI  2,NTRYNAME    ENTRY NAME IN ASCII
 4   ZERO   PSIZE,CSIZE   SIZES OF SUBROUTINE AND USED COMMON
 5   BSS    3             NOT USED
 8   BSS    8             REGISTER SAFE-STORE AREA
16   SREG   -8,IC         PROLOGUE: SAVE REGISTERS
17   EAX2   0,1           ESTABLISH ARGUMENT LIST POINTER
18   ...    ...           BEGINNING OF SUBROUTINE
```

In addition to the TSX1 instruction, the calling subroutine provides an argument list such that an instruction of the form:

```
     LDA     N,2*
```

loads the A register from the location addressed by the Nth argument pointer. Thus the burden is placed on the call to develop pointers to such various types of arguments as array elements, local variables, common variables, passed on arguments (from the caller's caller), FORTRAN statement numbers providing alternate returns, and the like.

An example of a simple call requiring no address computation would be one involving only arguments which are single variables, either local or in common, and an alternate return. In FORTRAN, this might have been written:

```
     CALL ALPHA (LOCAL,COMON,$100)
```

The calling sequence generated, including the linkage block to make this a more complete example, would be as follows:

```
     	    TRA      CALL,$
     	    EVEN
     SREFM  ZERO     0
     	    ARG      SREFN+1-*
     	    USACI    2,ALPHA
     CALL   TSX1     -4,IC
     	    TRA      4,IC
     	    ARG      LOCAL-CALL-1,2
     	    ARG      COMON
     	    TRA      SN100,$
     NEXT   ...      ...
```

The first instruction transfers around the four-word linkage block. The zero word indicates to the Time-Sharing Loader that the linkage is of the .SREF. type, and is replaced when loaded by a linking TRA to the corresponding .SDEF. linkage block. Following this is a self-relative pointer to the linkage block for the next .SREF. symbol in this object module. It is assumed that the first word of that block is labeled SREFN, just as this is labeled SREFM. Next is the identifier "ALPHA" encoded in upper-case ASCII code. This ends the linkage information.

The instruction labeled "CALL" is the actual call to "ALPHA". It is followed by an instruction, representing the normal return, which transfers around the argument list. Following are the three argument pointers.

The first argument pointer, for a local variable, relies on the fact that the called module has set Index Register 2 pointing to the beginning of the argument list (CALL + 1). Thus the effective address of the variable is derived by the expression:

```
     LOCAL - (CALL + 1) + (CALL + 1)
```

where the third term of this expression is in Index Register 2.

Common Storage is allocated absolutely at compile time, hence the pointer is simply the location symbol (LABEL).

Alternate returns are not simply pointers, but executed instructions, hence a standard IC modified transfer to the statement label is generated. These instructions occupy the argument position corresponding to their position in the CALL statement; the list of alternate returns is not sorted into a list separate from the arguments as is done in batch.

Argument pointers for variables which are themselves arguments to the calling routine require that the incoming pointer be planted into the argument list before calling. In general, if argument m is to be passed on as the nth argument, the following two instructions will be generated immediately preceding the call:

```
     EAA     M,2*
     STA     CALL+N+1,$
```

To illustrate, consider another call to ALPHA, later on in the above subroutine, of the form:

```
     CALL ALPHA (ARG2,ARG3,$100)
```

where ARG2 and ARG3 are the second and third arguments given to this calling module. This expansion would appear:

```
               EAA      2,2*
               STA      CALL1+2,$
               EAA      3,2*
               STA      CALL1+3,$
     CALL1     TSX1     SREFM,$
               TRA      4,IC
               ARG      **
               ARG      **
               TRA      SN100,$
     NEXT1     ...      ...
```

Return from a subroutine is accomplished by reloading the registers from the register safe-store area and transferring back via Index Register 1. Currently, Time-Sharing FORTRAN has but one entry; the register safe-store area is always at location 8 of the object module. The macros for use with GMAP, described below, enable definition of multiple entries, and the return function must always exit restoring the registers from the register safe-store area associated with the .SDEF. used for entry. Alternate returns, after reloading the registers, execute a TRA instruction addressing (n,1) where n is the argument number of the indicated alternate return.

Function calls are processed in exactly the same manner as described for subroutine calls, with the addition that a result is returned in a register. In Time-Sharing FORTRAN, the anticipated answer-back register varies with the type of value returned (which is in turn denoted by the type of the function name). The following is a list of register associations by data type:

Integer - Q Register
Real - EA Register
File Name - AQ Register
ASCII - A Register
Logical - Index Register 0

(Note. The logical value .TRUE. is represented by the binary value 1 and .FALSE. by 0. All other values in X0 are undefined.)

Returning a value in a register can be done by altering the register safe-store area, or hand-coding the return to set the appropriate registers after the restore operation. In simple functions which have no calls to other subprograms the register safe-store and restore can be omitted, and the target register(s) dealt with directly. Return would then be accomplished by a simple TRA 0,1.

Creating and Loading Binary Files

The Time-Sharing FORTRAN subsystem makes provision for saving object modules produced by the compiler, and for subsequent loading and execution of previously compiled object modules so saved. The format of these saved object modules is the same as that produced by GELOAD when it saves a link or a program on an H* type file. Thus, it is possible to assemble and load one or more subprograms in the batch world using GELOAD, and save the result in a permanent file accessible to the Time-Sharing FORTRAN subsystem.

There are two steps to this process -- the first is the creation of the file with properties appropriate to loadable perm-files; the second is the writing of the object module into this file.

File creation can best be done through use of the ACCESS subsystem of time-sharing. In response to the question

```
     FILE NAME,SIZE (BLKS), MAX SIZE?
```

reply with the required information plus an additional argument, "R", to designate that the file is random as opposed to linked.

For example, if a small (e.g., half-link) file called "RAND" is to be created, the response to this query would be:

```
     RAND,6,12,R
```

Other techniques are available for file creation (e.g., the FILSYS activity) but are not discussed here.

Writing of the object module into the perm-file is accomplished using GELOAD, with the options "SAVE" and "NOGO", and with appropriate control cards to select the created perm-file as the save file (H*). For greatest convenience, this process can be done using the CARDIN subsystem of time-sharing.

A typical deck setup, to write the file RAND, might look as follows:

```
$     IDENT      ...
$     USERID     ...
$     LOWLOAD    LSW
$     OPTION     NOGO,NOSETU,SAVE/SUBR1
        .
        .
      (Assorted source and object decks representing
       the object module to be loaded and saved)
        .
        .
$     NOLIB
$     EXECUTE
$     PRMFL      H*,WRITE,R,USERID/RAND
$     ENDJOB
```

The object module may be a mixture of source (GMAP) and object decks, but all must have been coded as "floatable" relocatable subroutines.

The "NOSETU" option and the $ NOLIB card are present to insure that nothing is taken from the batch subroutine library, which is not floatable and hence not usable in time-sharing.

Finally, consider that the file is to be loaded and executed in conjunction with a Time-Sharing FORTRAN program which is contained in the file JOB. The command employed to compile JOB load the object version of "JOB" with "RAND" (which is already an object module), and execute the combined object modules would look as follows:

```
RUN JOB;RAND
```

The combined object modules may in turn be saved in another file, by adding the SAVE clause to the RUN command.

```
RUN JOB;RAND=JOINT
```

This command does all that the first example did, plus the combined object modules are saved as one object module in the file "JOINT". Execution of that file is accomplished by:

```
RUN JOINT
```

The various options and properties of the RUN command are discussed in greater detail in Chapter 3.

Macro Package

To facilitate writing Time-Sharing FORTRAN compatible subroutines in GMAP, a collection of macros have been developed which define .SDEF. and the call, save and return functions, specify common variables, and in general insure adherence to the object module format required by the Time-Sharing Loader. This macro collection is named .TSFMA and is accessible via the LODM pseudo-operation of GMAP.

Before discussing the individual macros, consider the overall construction of the GMAP source program for this purpose.

1. The program normally begins with LBL, TTL, and other control pseudo-operations as required.
2. LODM .TSFMA brings the macros into the assembler's macro tables.
3. The main entry is defined.
4. Code unique to that entry is given.
5. Alternate entries and associated programming is given as required.
6. Local and common storage variables are defined and storage allocated.
7. The program concludes.

Step 3 above is accomplished using the .STRT. macro. Alternate entries (step 5) are defined using the .SDEF. macro. Calls to other object modules are accomplished using the .CALL. macro, which also generates the necessary .SREF. linkage block. Return from a subroutine is facilitated by the .EXIT. macro. The program is concluded (step 7) with the .END. macro.

Following are the descriptions of the individual macros and how to use them.

.STRT.

This macro must precede all executable instructions and data defining pseudo-operations. It initializes some assembly time variables and location counters which will be used in the generation of linkage blocks.

The first 18 instructions including the linkage block for the main entry, the entry prologue, and the required register safe-store area are generated. The entry name is defined as both a label and a SYMDEF (for identification purposes), corresponding to location 18 of the object module. For this reason, the entry name is limited to six characters in length.

.STRT. has one argument, which is the entry name, and is of the form:

```
     .STRT. NTRYNAME
```

.SDEF.

This macro is used to define alternate entries. It may appear anywhere in the program, after the .STRT. macro. It generates a linkage block and a prologue SAVE sequence terminated by an instruction of the form:

```
     TRA      NTRYNAME,$
```

for each entry name specified. It is the programmer's responsibility to define the label NTRYNAME elsewhere in the program, at the first instruction of that subroutine. For this reason, the entry name is restricted to symbols of six characters or less.

.SDEF. has one argument, which may be a list, providing one or more entry names and is of the form:

```
     .SDEF.     NTRNLNAM
            or
     .SDEF.     (NTRYNAME-1,NTRYNAME-2...,NTRYNAME-N)
```

.SREF.

This macro is used to define symbols which represent locations external to this subroutine. The user will not normally use this macro since it is generated as part of the .CALL. expansion. It may appear anywhere in the program after the .STRT. macro. It generates a linkage block and labels the first word of that block with the symbol may be no more than six characters long.

.SREF. has one argument, which may be a list, providing one or more external names, and is of the form:

```
.SREF.     XTRNLNAM
       or
.SREF.     (XTRNLNAM-1,XTRNLNAM-2...,XTRNLNAM-N)
```

.CALL.

This macro is used to call external subroutines. Its expansion includes the definition of the called name as a .SREF. symbol, and generation of a Time-Sharing FORTRAN compatible calling sequence.

.CALL. has two arguments. The first is the name of the .SREF. symbol to be called, and the second provides an argument list. The general form of the macro is:

```
.CALL.     ALPHA
       or
.CALL.     ALPHA(ARG-1,ARG-2...,ARG-N)
```

The first form is used for calls to subroutines requiring no arguments.

The subroutine name ALPHA will be defined (if it is not already) as a .SREF. symbol.

The argument list is composed of argument descriptors which specify both the argument and a qualifier as to its type. For example, if BETA is a variable in common, its argument descriptor would appear:

```
.COM.(BETA)
```

Arguments which are local variables are not qualified; an argument descriptor for a local variable is its label only.

The following four types are defined for use as argument qualifiers:

```
.COM.  - Variable in common
.ARG.  - Passed-on argument
.N.    - Integer arguments (18 bit max)
.RET.  - Alternate return
```

To illustrate, consider the following example:

```
.CALL. ALPHA(BETA,.COM.(GAMMA),.ARG.(2),.N.(1),.RET.(DELTA))
```

This represents a call to ALPHA, with argument 1 being a local variable, argument 2 a variable in common, argument 3 a hand-me-down of the second argument to this routine, argument 4 the quantity 1, and argument 5 an alternate return to the internal location DELTA. The actual expansion, not including linkage would be as follows:

```
EAA       2,2*
STA       5,IC
TSX1      ALPHA,$
TRA       6,IC
ARG       BETA-*+1,2
ARG       GAMMA
ARG       **
ARG       1,DL
TRA       DELTA,$
```

.EXIT.

This macro is used to return from a subroutine to its caller. Its expansion includes the restoration of all register contents for the caller, and a TRA instruction back to the indicated return.

.EXIT. has two arguments, either of which may be null. The first argument provides the entry name, back through which the return is to be made. When null, the main entry is assumed. The second argument chooses either the normal return or some alternate return.

The general form of this macro is:

```
.EXIT. NTRYNAME,RNUM
```

where NTRYNAME is the .SDEF. symbol which was used to call this routine (and hence, locates the associated register safe-store area), and RNUM is null or specifies an alternate return number. The alternate return number is the argument position which contains the alternate return TRA instruction, arguments being numbered from one to the number supplied.

The following four examples illustrate all variations of the argument list.

```
.EXIT.
```

is a normal return back through the main entry.

```
.EXIT.,2
```

is an alternate return back through the second argument given to the main entry.

```
.EXIT. COSINE
```

is a normal return back through the entry COSINE.

```
.EXIT. COSINE,2
```

is an alternate return via the second argument given to the entry COSINE.

.COMN.

This macro is used to define variables in common storage. It may appear anywhere after the .STRT. macro.

.COMN. has three arguments. The first is the identifier of the common storage variable. The second specifies the number of locations to be allocated to this variable; when null, one location is allocated. The third argument is used to perform modulo arithmetic on the common storage location counter before the allocation is done. When null, the next available location is the first allocated, regardless of the size specified by argument 2.

The general form of .COMN. is as follows:

```
.COMN. ID,BSS,MOD
```

The following examples illustrate use of this macro.

```
     .COMN. ALPHA
```

Defines ALPHA as a common storage variable requiring one location.

```
     .COMN. ALPHA,100,2
```

Defines ALPHA to be in common, and allocates 100 locations starting at the next available even location.

```
     .COMN. REGS,8,8
```

Allocates eight locations at the next modulo 8 address in common, and labels the first word of that area REGS.

.END.

This macro is used to terminate assembly. It must be the last line/card of the source program. Its expansion includes the determination of the program size and the size of common storage, and the generation of the end card. It has no arguments.

SAMPLE PROGRAM

Table 1 is an example of a subprogram written in GMAP. Table 2 is a sample Time-Sharing FORTRAN source file and Table 3 is a console dialogue showing the execution of these files/subprograms together.

Table 1. GMAP Subprogram - File GMAPS

```
100$      IDENT          SYZ,JOEDOAKS
110$      USERID         DOAKS$PASSWORD
120$      LOWLOAD        LSW
130$      OPTION         NOGO,NOSETU,SAVE/SUB1
140$      GMAP           NDECK
160       LODM           .TSFMA
170       PMC            OFF
180       .STRT.         GMAP1
190       LDA            1,2*
200       STA            INT,$
210       STA            COM2
220       EAX3           2,2*
230       SBLX3          1,DU
240       STX3           1,IC
250MOV    LDQ            **,AL
260       STQ            COM1-1,AL
270       SBLA           1,DL
280       TNZ            MOV,$
290       .CALL.         FP1(INT,.COM.(COM1))
300       .EXIT.
320       .SDEF.         (GMAP2,GMAP3,RAND)
330GMAP2  .CALL.         FP2(.RET.(ALT))
340       .CALL.         FP3
350ALT    .EXIT.         GMAP2
360GMAP3  .CALL.         FP1(.ARG.(1),.ARG.(2))
370       .EXIT.         GMAP3,3
400RAND   LDQ            STARTR,$
410       MPY            MAGIC,$
420       LLS            20
430       ARL            1
440       STA            STARTR,$
450       ANA            MASK,$
460       LDE            ZER,$
470       FAD            FZER,$
480       FST            1,2*
490       USE            RAND
500RANREG NULL
510       USE            PREVIOUS
520       LDX2           RANREG+1,$
530       TRA            0,1
540STARTR DEC            30517578125
550MAGIC  DEC            30517578125
560MASK   OCT            777777777400
570ZER    DEC            0
580FZER   DEC            0.0
590INT    BSS            1
600       .COMN.         COM1,15
610       .COMN.         COM2
620       .END.
630$      NOLIB
640$      EXECUTE
650$      PRMFL          H*,WRITE,R,DOAKS/GMAP
660$      ENDJOB
```

Table 2. Sample Time-Sharing FORTRAN Source File - GMAPT

```
100 COMMON GARAY(15),GCOM,FARAY(15)
110 ASCII FARAY,LINE(15)
120       READ 11,(LINE(I),I=1,15)
130   11  FORMAT (15A4)
140       LEN=15;  GO TO 25
150   20  LEN=LEN-1
160   25  IF (LINE(LEN) .EQ. "    ") GO TO 20
170       DO 40 I=1,LEN
180   40  FARAY(I)=LINE(I)
190       CALL GMAP1(LEN,FARAY)
200       CALL GMAP2
210       CALL GMAP3(LEN,FARAY,$90)
220       PRINT:"ERROR. NORMAL RETURN TAKEN FROM GMAP3."
230   90  CALL RANDT
240       PRINT:"SUCCESS"
250  100  STOP
260       END
270 SUBROUTINE FP1(LEN,ARAY)
280 ASCII ARAY(15)
290       PRINT 11,(ARAY(I),I=1,LEN)
300   11  FORMAT (" FP1 PRINTS ",12A4)
310       RETURN
320       END
330 SUBROUTINE FP2(*)
340 COMMON LINE(15),LEN
350 ASCII LINE
360       PRINT 11,(LINE(I),I=1,LEN)
370   11  FORMAT (" FP2 PRINTS ",12A4)
380       RETURN 1
390       END
400 SUBROUTINE FP3
410       PRINT:"ERROR. NORMAL RETURN TAKEN BY FP2."
420       RETURN
430 END
440 SUBROUTINE RANDT
450 DIMENSION NUMMIX(80)
460       PRINT:"     RANDOM NUMBER GENERATOR TEST"
470       DO 50 I = 1,4000
480       N=RAND(X)*79.99999+1.000005
490       IF (N .LE. 0) GO TO 20
500       IF (N .LE. 80) GO TO 50
510   20  PRINT 21,I,N
520   21  FORMAT (" RAND(",I4,") = ",I4)
530       N=40
540   50  NUMMIX(N) = NUMMIX(N) + 1
550       PRINT:"DISTRIBUTION OF RANDOM NUMBERS-4000 GEN."
560       DO 900 I=1,8
570       J=I*10
580       K=J-9
590       PRINT 801,K,J,(NUMMIX(L),L=K,J)
600  801  FORMAT (1X,I2,"-",I2,10I5)
610  900  CONTINUE
620       RETURN
630  END
```

Table 3. Console Dialogue Showing Execution of TSS FORTRAN Program with GMAP Subprograms

```
SYSTEM  ? CARDIN
OLD OR NEW-OLD GMAPS
READY FOR INPUT
*RUN
  SNUMB # 1566T
CARD FORMAT,DISPOSITION ?
NORM
*DONE
SYSTEM  ? FORTRAN
OLD OR NEW-OLD GMAPT
READY FOR INPUT
*JSTS 1566T
1566T-02 WAIT-CORE
*JSTS 1566T
1566T OUTPUT COMPLETE
*RUN *;GMAP
= DEMO TS FORTRAN MACRO PACKAGE
FP1 PRINTS DEMO TS FORTRAN MACRO PACKAGE
FP2 PRINTS DEMO TS FORTRAN MACRO PACKAGE
FP1 PRINTS DEMO TS FORTRAN MACRO PACKAGE
    RANDOM NUMBER GENERATOR TEST
DISTRIBUTION OF RANDOM NUMBERS - 4000 GEN.
 1-10   48   66   50   59   54   48   48   58   53   45
11-20   54   44   55   45   44   59   38   60   44   57
21-30   60   50   50   48   58   55   65   54   54   53
31-40   49   51   43   41   49   49   59   49   51   58
41-50   44   46   51   44   49   47   49   45   48   47
51-60   41   51   45   45   52   56   50   48   45   48
61-70   62   51   45   45   41   55   44   50   52   43
71-80   51   45   56   58   45   50   47   53   31   50
SUCCESS

PROGRAM STOP AT 250
*
```

TIME-SHARING FORTRAN LIBRARY GENERATOR PROGRAM/LIBRARY EDITOR SUBSYSTEM

The TSS FORTRAN Library Generator program and the TSS FORTRAN Library Editor subsystem, together, provide the user with the capability of producing his own load-time library of Time-Sharing FORTRAN subroutines. Such libraries are essentially collections of independent object subroutines either written in TSS FORTRAN or in GMAP language, the coding of the latter conforming to special TSS FORTRAN standards with respect to "floatability" and linkage conventions.

The Library Generator produces the TSS FORTRAN-loadable library file from one or more files of subroutines in GESAVE or GELOAD format.

The Library Editor allows the user to edit his file(s) of library subroutines in GESAVE format, prior to the use of the Library Generator program. The user can add, delete, replace, or copy individual subroutines on a master file.

The operations of the Library Generator and the Library Editor both follow. Since the Library Editor is not essential to the operation of the Library Generator, the latter's description is given first.

Library Generator Program

The TSS FORTRAN Library Generator (TSLG) Program is a GE-600 Line GECOS III batch program that is distributed on the System Software Library. It may be called via a $ PROGRAM card.

The TSLG permits a user to produce his own library file of TSS FORTRAN subroutines, complete with directory, in a form that is acceptable to the TSS FORTRAN loader. TSLG accepts collections of floatable subroutines, in one or both of the following formats:

- GESAVE format, as produced by a loader activity (H* file) and saved on a random permanent file, or as produced by the TSS FORTRAN compiler (savefile).

- GELOAD format, as produced by the Object Library Editor on a magnetic-tape file.

TSLG processes these subroutines so as to produce one or both of the following:

- A random-mass-storage file to be used as an individual user's own TSS FORTRAN library, accessible by the TSS FORTRAN loader from the permanent file system.
- A magnetic-tape file (Q*) to be loaded at system startup time as the installation's standard TSS FORTRAN library.

The user normally should not be concerned with the production of the latter. (TSLG may also be used by the installation for maintenance of the standard TSS FORTRAN subroutine library.)

The user's own library is stored on a permanent file named by the user on one of his TSLG-job $ PRMFL cards. At FORTRAN-run time, he identifies this library to the TSS FORTRAN subsystem via the ULIB option in the RUN command.

Subroutine Coding Requirements

Library routines may be coded in either TSS FORTRAN or GMAP. The individual library subroutines, when coded in GMAP language, must conform to TSS FORTRAN coding conventions; i.e., they must be compatible with the code produced by the TSS FORTRAN compiler. The conventions concern:

- Special linkage (global reference) symbols, which replace the normal SYMREF/SYMDEF symbols.
- Floatable, or self-relative, coding -- all location references IC-modified.
- Intra-subroutine communication, argument passing, etc.

The subroutines written in TSS FORTRAN language are compiled and saved on a permanent file, as are the GMAP assemblies. Compiler-saved output is automatically in GESAVE, or system-loadable, format.

Instructions for writing subroutines in GMAP to TSS FORTRAN standards, using special GMAP macros developed for this purpose, and on obtaining system-loadable (GESAVE) format on a permanent file, are described under "Writing Subprograms in Assembly Language" in this chapter.

Input and Output Files

Figure 2 illustrates the files utilized by TSLG. Descriptions of the files follow.

The principal inputs to TSLG, from the normal user's viewpoint, are as follows:

- UI, a disc/drum, permanent file of user's subroutines in GESAVE format.
- I*, a $ DATA file of control cards, on disc or drum.

Additional inputs are:

- R*, a labeled magnetic-tape file which contains the standard TSS FORTRAN library subroutines distributed with the system, in GELOAD object-library format, as produced by the Object Library Editor (a component of the System Editor).
- *Z, a magnetic-tape file identical in format to R*, containing installation-written subroutines additional to the standard subroutines on R*, or containing an edited selection of R*.

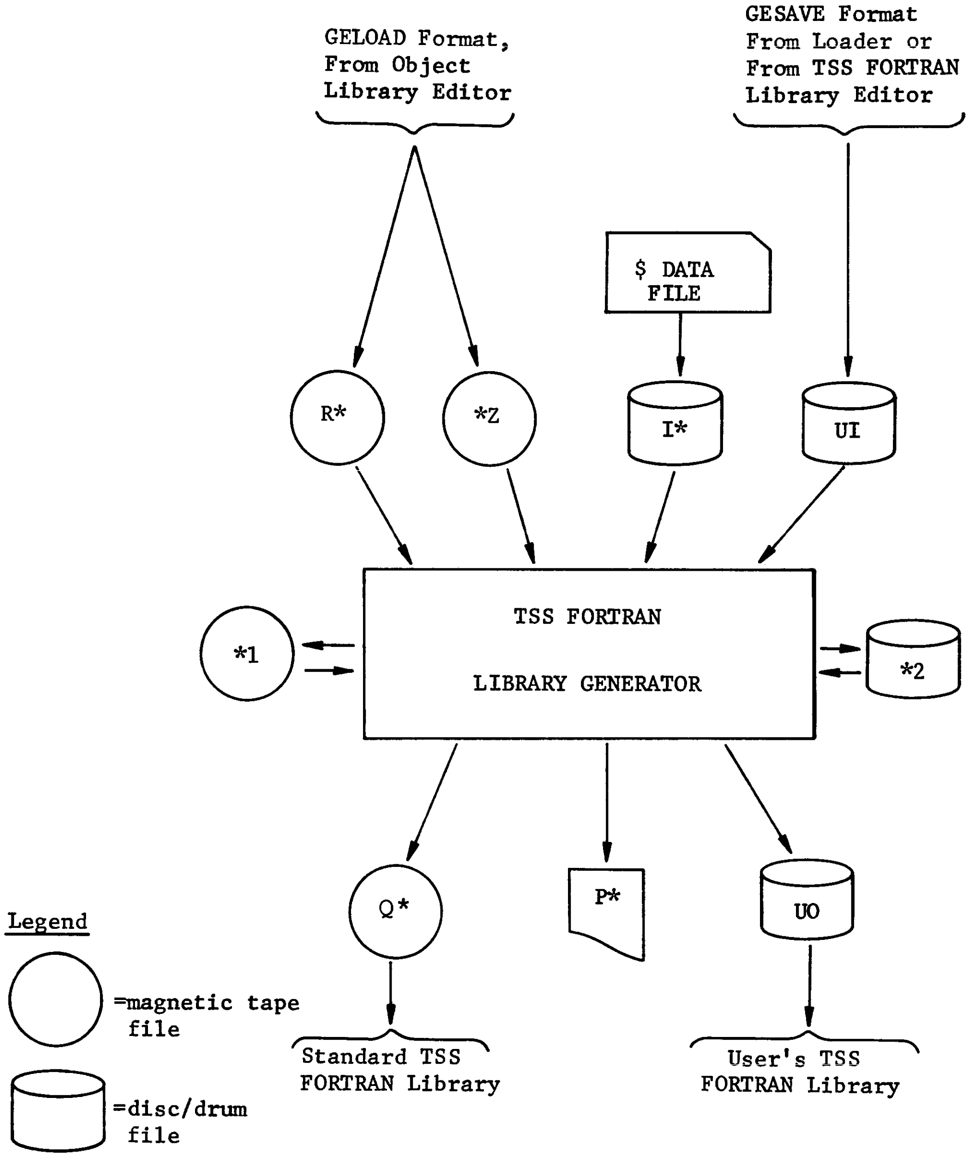


Figure 2. Files Used by TSS FORTRAN Library Generator

Each of the inputs described above are optional, except that the I* file (IN control card) must exist if either UI or *Z (or both) exist, or if output file UO is to be produced (OUT control card). The user will not normally be concerned with the R* input or Q* output; the I* file is the file generally required.

Two intermediate work files must be provided:

- *1, a magnetic-tape file.
- *2, a temporary disc/drum file used for error messages, to be written subsequently on the output-file P* (SYSOUT).

The possible output files are:

- UO, a disc/drum, permanent file containing the user's own TSS FORTRAN library, accessible from the permanent file system by the TSS FORTRAN loader.
- P*, a SYSOUT file containing a library storage map and error messages. The map and messages are described below.
- Q*, a labeled magnetic-tape file containing the standard TSS FORTRAN library, to be loaded by the startup routine at system-initialization time.

Either UO or Q*, or both, may be produced; it is assumed herein that only the UO file is to be produced; therefore it must be specified on an OUT control card (I* file). P* must always be provided.

The output file (Q*) is not rewound on open or close. All other magnetic-tape files are rewound when opened.

Either or both of the file codes, UI and UO, may be replaced by file codes of the user's choice, on his job control cards and on IN/OUT control cards.

Control Card and File Usage

Two control cards may appear on the I* ($ DATA) file -- an IN card and/or an OUT card. The IN card is used to specify which input file(s) -- of the set UI, *Z, and R* -- are to be processed and also, implicitly, in what order they are to be processed. The OUT card is used simply to specify which output file(s) -- of the set UO, Q* -- are to be produced.

If no IN card is present, the presence of R* only is assumed by TSLG. If multiple IN cards are present, only the last one is acknowledged. If no OUT card is present, the presence of Q* only is assumed by TSLG. If multiple OUT cards are present, only the last one is acknowledged.

If more than one input file is specified (e.g., UI and *Z), the content of each input file is processed and written on the output file in the order in which the input file-codes appear on the IN card. Thus, all routines on the input file corresponding to the first-specified file code will appear on the output file before those on the second input file, etc.

In general, TSLG processes entire input files only. Selective editing of UI files may be performed with LIBED; selective editing of a *Z file may be performed during the required Object Library Editor processing. See the manual, GE-600 Line System Editor, CPB-1138, for information pertaining to the Object Library Editor.

Input from R* and/or *Z is normally used by the installation to produce a modified Standard TSS FORTRAN library file, on Q*. Input from UI is normally used to produce a user's TSS FORTRAN library file, on UO. However, subroutines from any of the possible input files, or any combination thereof in any order, can be written to UO (or Q*).

The essential differences between the two libraries are that the standard library, on Q*, is initialized as a permanent file by the Startup routine at a subsequent system-startup time, and is automatically searched by the TSS FORTRAN loader if no user's library (ULIB) is specified at FORTRAN-run time. If one or more ULIB's are specified, the loader will search the standard library last in attempting to satisfy any subroutine-calls still outstanding. Thus, the standard library is referenced implicitly, and a user's library is referenced explicitly, with the latter having priority.

Control Card Formats

The control field of the IN/OUT control cards starts in column 8, and the variable field starts in column 16; multiple file codes are comma-separated.

- IN Control Card

```
1      8       16
|      |IN     |fc1, fc2, fc3
```

where fc is one of the set (UI, *Z, R*), normally only UI. File-code UI may be replaced by a code of the user's choice. At least one file code must appear.

- OUT Control Card

```
1      8       16
|      |OUT    |fc1, fc2
```

where fc is one of the set (UO, Q*), normally UO. File-code UO may be replaced by a code of the user's choice. At least one file code must appear.

Any card-type other than IN or OUT, as defined above, appearing on I* will cause TSLG to be aborted.

Program Description

The following exemplifies the general deck setup for a user's execution of TSLG. It is assumed that UI and UO are permanent files previously created (e.g., via the ACCESS subsystem) and that the user specified on the $ USERID card has permission to read UI and to write UO.

```
  $       SNUMB
  $       IDENT
  $       USERID      user-id$password
  $       PROGRAM     TSLG
  $       LIMITS      30,32k,0,5000
 *$       TAPE        *Z,X1D
  $       PRMFL       UI,R,R,catalog$password/filename
  $       PRMFL       UO,W,R,catalog$password/filename
  $       TAPE        *1,X2R
  $       FILE        *2,X3R,1R
  $       SYSOUT      P*
  $       DATA        I*
**        IN          UI (or UI,*Z or *Z,UI)
          OUT         UO
  $       ENDJOB
```

* Include only if applicable.
** Variable field specified as applicable.

Refer to the TSS FORTRAN Library Editor description below for a combined example of LIBED/TSLG usage in the production of a specific user's library file.

Storage Map and Error Messages

The FORTRAN library storage map is printed by GEOT. It contains the following information for each routine in the library:

- The sector address at which the routine is stored.
- The identification (name) of each SYMDEF/.SDEF. in the routine and an indication of its type (primary or secondary).
- The sector address contained in the Directory entry corresponding to each SYMDEF/.SDEF.. (This address will appear only if it differs from that of the current routine).
- The identification and corresponding sector address of SYMDEF/.SDEF. referred to by each .SREF.. (Undefined .SREF.'s are flagged with a U.)

Appended to the storage map are any applicable error or warning messages, as follows:

1. PROGRAM identity of deck ABORTED FOR INSUFFICIENT BUFFER SPACE.
2. RERUN WITH MORE CORE SPECIFIED ON $ LIMIT CARD.
3. SYMDEF'S DEFINED IN WRONG ORDER:
4. SYMREF'S UNDEFINED:
5. NO OBJECT CARD BEFORE card type CARD card identity DECK BYPASSED, PREVIOUS COMPLETE DECK identity of previous deck.
6. NO DKEND CARD IN DECK identity of current deck DECK BYPASSED PREVIOUS COMPLETE DECK identity of previous deck.
7. card type CARD card identity OUT OF ORDER IN DECK identity of current deck.
8. NO TEXT CARDS IN DECK identity of current deck DECK BYPASSED. PREVIOUS COMPLETE DECK identity of previous deck.

```
 9. card type CARD card identity IN DECK
    identity of current deck.
    DECK BYPASSED. PREVIOUS COMPLETE DECK
    identity of previous deck
    (something wrong with the card)

10. NONZERO RELOCATION BITS IN card type
    CARD card identity
    DECK BYPASSED.  PREVIOUS COMPLETE DECK
    identity of previous deck.

11. CHECKSUM ERROR IGNORED IN card type
    CARD card identity.

12. OCTAL CARD FORMAT ERROR IN DECK
    identity of current deck
    DECK BYPASSED.  PREVIOUS COMPLETE DECK
    identity of previous deck

13. CHECKSUM ERROR IGNORED IN WORD 0 OF CONTROL BLOCK
    FOR ROUTINE routine.
```

Program Aborts

Execution of TSLG may be aborted for the following reasons:

Reason Code	Meaning
0	Specified input or output file not present.
1	Premature EOF on input file R* or *Z.
2	Wrong routine read from *1.
3	Previously defined SYMDEF not found in directory.
4	Premature EOF on *S.
5	Insufficient buffer area provided. (Rerun with larger core specification on $ LIMITS card.)
6	Invalid device code specified in file control block for UI or UO.
7	Name of routine in UI control block does not agree with name in catalog.
8	Number of sectors specified in UI control block is inconsistent with total DCW word count.
9	Invalid control card in I*.

Library Editor Subsystem

General

The TSS FORTRAN Library Editor (LIBED) is a time-sharing subsystem that may be called at the subsystem-selection level (SYSTEM?) by the name LIBED. It allows the user to manipulate collections of independent subroutines or subprograms stored on permanent files in GESAVE; i.e., H* format. The user is permitted to combine several such collections into one file, to delete elements from such a collection, or to extract (copy) selected elements.

LIBED is specifically intended for the editing of TSS FORTRAN subroutine library files, such files to be subsequently processed by the TSS FORTRAN Library Generator (TSLG) program. However, the LIBED functions are equally applicable to any collection of object programs, subprograms, or subroutines in GESAVE format.

The LIBED subsystem provides for three processing functions -- LIST, APPEND, and DELETE -- and utilizes the control command DONE. The subsystem will operate on any of the following combinations of files:

- old master file (only)
- old master file and update file
- old master file and new master file
- old master file, update file, and new master file.

Each of these files must be previously created permanent files, and must be accessed explicitly prior to the use of LIBED.

A convenient means of pre-accessing these files is by the GET command, which may be given at the subsystem-selection level. The old-master and update files must be random files in GESAVE format; the new master file must simply be defined (or accessed) as a random file. The latter may be a newly created, 'empty' file (created most conveniently via the ACCESS subsystem), or it may contain previously stored data which will be overwritten.

If all three files are specified, processing will result in a modified new master file, and neither the old master file nor the update file will be changed. If a new master file is not specified, all modifications will take place on the old master file.

Use of Library Editor

Following selection of LIBED, at the subsystem-selection level, the user will be asked the question:

FILES?

Possible user-responses to this question are as follows:

- old-master-name
- old-master-name, update-name
- old-master-name, new-master-name
- old-master-name, update-name, new-master-name

Having received a valid reply to the FILES? question, the subsystem will respond with the message READY. In response to READY, the terminal user may type in one of the following commands:

- LIST filename

  where filename is the name of one of files specified in response to the FILES? question.

  The subsystem will list the names of all routines on the designated file along with the number of blocks required to contain each routine. The list will terminate with a report of the total number of blocks used on that file.

  Example of LIST format:

  ```
  LIST OLDMAS

  ROUTINE NAME              NO. OF BLOCKS

  XROUTN                         5
  YROUTN                         7
  ZROUTN                         6
  TOTAL NO. OF BLOCKS USED      18
  ```

- APPEND subr 1, subr 2,...,subr n

  where subr i is the name of a routine on the update file to be appended to the old master file to generate either a modified old master file or a new master file. When no routine name is supplied, the entire contents of the update file will be appended as above.

- DELETE subr 1, subr 2,....,subr n

  where subr i is the name of a routine to be deleted from the current master file (old master file, if no new file is designated, or new master file, if it were created prior to this command). If DELETE is given with a nonexistent routine name specified, a copy of old-master to new-master will occur.

- DONE

  This command terminates the subsystem, releases all files involved, and returns control to the subsystem-selection level.

Example of Combined LIBED and TSLG Usage

TSS FORTRAN subroutines to be used as input to LIBED (optional), and then to TSLG, may be written originally in TSS FORTRAN and compiled or may be written in GMAP (observing special coding restrictions). The latter is described under "Writing Subprograms In Assembly Language" in this chapter. Therefore, hypothetical TSS FORTRAN-generated subroutines will be used for the purpose of the following example. Text within brackets is not part of the printout but has been added to explain salient features.

1. To create user-library subroutines via the TSS FORTRAN subsystem:

```
SYSTEM ?ACCESS --- [Subsystem selection to create file
                        space]
FUNCTION?  CF
    CATALOG STRUCTURE TO WORKING LEVEL?

    FILE NAME,SIZE(IN BLCKS),MAX SIZE?  A1,3,3,R
    PASSWORD?
    GENERAL PERMISSIONS?
    SPECIFIC PERMISSION?
    LOGICAL RECORD SIZE?
SUCCESSFUL!
    FILE NAME,SIZE(IN BLCKS),MAX SIZE?  A2,2,3,R
    PASSWORD?
    GENERAL PERMISSIONS?
    SPECIFIC PERMISSION?
    LOGICAL RECORD SIZE?
SUCCESSFUL!
    FILE NAME,SIZE(IN BLCKS),MAX SIZE?  A3,6,6,R
    PASSWORD?
    GENERAL PERMISSIONS?
    SPECIFIC PERMISSION?
    LOGICAL RECORD SIZE?
    FILE NAME,SIZE(IN BLCKS),MAX SIZE?
SUCCESSFUL!


FUNCTION?
SYSTEM ?FORTRAN ---[Subsystem selection to create
OLD OR NEW-NEW        subroutines QUAD and NTRS]
READY
*010  SUBROUTINE QUAD(X,Y,Z)
*020  READ:A,B,C
*030  RR=B**2-4.0*A*C
*040  DD=SQRT(RR)
*050  X1=(-B+DD)/2.0*A
*060  X2=(-B-DD)/2.0*A
*070  PRINT:X1,X2
*080  RETURN
*090  END
*100  SUBROUTINE NTRS
*110  X=1.0
*120  2EX=EXP(X)
*130  EMX=1.0/EX
*140  XNEW=X+((EX+EMX)/2.0+COS(X)-3.0)/((EX-EMX)/2.0-SIN(X))
*150  PRINT:XNEW
*160  IF (ABS(X-XNEW) .LT. 1.E-6) STOP
*170  XNEW=X
*180  GO TO 2
*190  END
*RUN =A1(NOGO) ---[Place QUAD and NTRS in file A1]
*NEW ---[Create Subroutine FACT and AREA]
```

```
READY
*010  SUBROUTINE FACT(K,J,N)
*020  READ:K,J,N
*030  PRINT 88,K
*040 88FORMAT(1H0,I5," FACTORIAL IS")
*050  DO 99 I=1,N
*060  J=J-1
*070  K=K*J
*080 99CONTINUE
*090  PRINT:K
*100  RETURN
*110  END
*120  SUBROUTINE AREA(X,Y,Z)
*130  READ:A,B,C
*140  S=(A+B+C)/2.0
*150  AREA=SQRT(S*(S-A)*(S-B)*(S-C))
*160  PRINT:A,B,C,AREA
*170  RETURN
*180  END
*RUN =A2(NOGO) ---[Place FACT AND AREA in file A2]
*DONE
```

2. To merge subroutines on files A1 and A2 onto file A3:

```
SYSTEM ?ACCESS ---[Subsystem selection to determine
                        permissions]

FUNCTION?  AF
    CATALOG STRUCTURE TO WORKING LEVEL?

    FILE NAME$PASSWORD?  A1
    PERMISSIONS DESIRED?  R,W
SUCCESSFUL!
    FILE NAME$PASSWORD?  A2
    PERMISSIONS DESIRED?  R,W
SUCCESSFUL!
    FILE NAME$PASSWORD?  A3
    PERMISSIONS DESIRED?  R,W
SUCCESSFUL!
    FILE NAME$PASSWORD?


SYSTEM ?LIBED ---[Subsystem selection to edit files]
FILES?  A1,A2,A3 ---[old-master-name,update-name,
READY                    new-master-name]
LIST A1
ROUTINE NAME   # BLOCKS ---[Blocks refers to device block
    QUAD        0003               size]
    NTRS        0004

TOTAL # OF BLKS 0007
# BLKS AVAIL.   0006

READY
LIST A2
ROUTINE NAME   # BLOCKS
    FACT        0003
    AREA        0003

TOTAL # OF BLKS 0006

READY
LIST A3
NO DATA ON NEW FILE

READY
APPEND FACT ---[Append FACT from A2 to A1 and create
                  data for A3]


READY
LIST A3
ROUTINE NAME   # BLOCKS
    QUAD        0003
    NTRS        0004
    FACT        0003
```

```
TOTAL # OF BLKS 0010
# BLKS AVAIL.   0018

READY
DONE
```

The file A3 now contains all the routines that the user desires to be on his own library, when it is created. File A3 can now be processed by TSLG, creating another permanent file in actual subroutine-library format.

3. To convert file A3 into a user's subroutine library the following deck setup could be used:

```
$  SNUMB         ...
$  IDENT         ...
$  FILEDIT       OBJECT,INITIALIZE
$  TAPE          R*,X1S
$  DATA          *C,,COPY
$  ENDCOPY
$  PROGRAM       TSLG
$  USERID        user-id$password
$  LIMITS        30,32K,0,10000
$  PRMFL         UI,R,R,catalog/A3
$  PRMFL         UO,W,R,catalog/A4
$  TAPE          *Z,X1R
$  TAPE          *1,X2R
$  FILE          *2,X3R,1R
$  SYSOUT        P*
$  DATA          I*
   IN            UI,*Z
   OUT           UO
$  ENDJOB
```

A4 can now be specified as a user's library with the TSS FORTRAN RUN command; e.g.:

```
*RUN =(ULIB)A4
```

Error Messages

Possible error messages are as follows:

NO OLD FILE NAMED

User did not designate an old master file name. Control is returned to the subsystem-selection level.

FILE NOT RETRIEVED

One of files specified by the user has not been placed in the AFT table.

ILLEGAL COMMAND

Response to READY was not a valid command.

INVALID FILE NAMED

User designated a file that was not previously defined for the subsystem.

NO DATA ON NEW FILE

User requested a list of the new master file which had not been created yet in this subsystem.

ROUTINE xxxxxx NOT FOUND

In use of the DELETE or APPEND command, the user designated a routine name that could not be found on the appropriate file.

NO MORE ROOM ON CURRENT MASTER

The new master file (or the old master file if no new file was specified) has no more disc/drum space; current master will contain all information up to the point where this condition was found.

NO UPDATE FILE NAMED

The APPEND command was issued but no update file was specified.

DISC/DRUM ERROR

An unrecoverable error occurred while reading or writing subsystem files.

FILE IS NOT A RANDOM FILE

One of the files designated by the user is not a random file.

SYSTEM ERROR

Invalid condition occurred within subsystem.

FILE TOO LARGE FOR SUBSYSTEM

Table overflow in subsystem. At present, the subsystem can handle a file of 320 links, with 180 routines on it, if the block size is 64 words. With a block size of 40 words a file of 216 links containing 108 routines is the maximum.

Index

HONEYWELL INFORMATION SYSTEMS

Technical Publications Remarks Form*

TITLE: SERIES 600 GECOS TIME-SHARING SYSTEM GENERAL INFORMATION MANUAL

ORDER NO: CPB-1643A

DATED: JULY, 1971

ERRORS IN PUBLICATION:

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION:

(Please Print)

FROM: NAME ______________________ DATE ____________

COMPANY ______________________

TITLE ______________________

ADDRESS ______________________

____

*Your comments will be promptly investigated by appropriate technical personnel, action will be taken as required, and you will receive a written reply. If you do not require a written reply, please check here. □

CUT ALONG LINE

CUT ALONG LINE

POSTAGE STAMP NOT
NECESSARY IF MAILED
IN THE UNITED STATES

BUSINESS REPLY MAIL FIRST CLASS PERMIT NO. 39531, NEWTON HIGHLANDS, MASS.

POSTAGE WILL BE PAID BY:

HONEYWELL INFORMATION SYSTEMS

60 WALNUT STREET
WELLESLEY HILLS, MASS. 02181

ATTN: U.S. GROUP PUBLICATIONS, MS 050

Honeywell

The Other Computer Company:

Honeywell

HONEYWELL INFORMATION SYSTEMS

2452
5771
Printed in U.S.A.