Burroughs

B 6700 / B 7700

COMMAND AND EDIT (CANDE) LANGUAGE

INFORMATION MANUAL

B

THIS MANUAL REPLACES FORM NO. 5000318 DATED DECEMBER 1971

LIST OF EFFECTIVE PAGES

TOTAL NUMBER OF PAGES IN THIS MANUAL IS 112 CONSISTING OF THE FOLLOWING:

Page No.	Issue
Title	Original
A	Original
i thru iv	Original
1 thru 105	Original
Blank	Original

Burroughs - B 6700/B 7700 CANDE Language Manual

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

1.   INTRODUCTION

The COMMAND AND EDIT (CANDE) language provides generalized file preparation and updating capabilities in an interactive, terminal oriented environment. Execution of object programs with data input/output at a remote terminal is also provided by the CANDE Message Control System (MCS) program.

An attempt has been made to design and implement the B 6700 CANDE language to conform in its functional behavior to the existing B 5700 CANDE system. However, some points of difference exist to accommodate and/or efficiently work within the B 6700 operating system.

The discussion which follows assumes that the reader is acquainted with the B 6700 operating characteristics. Particular reference is made to the following documents:

B 6700 INPUT/OUTPUT SUBSYSTEM	Form No. 5000185
B 6700 NETWORK DEFINITION LANGUAGE	Form No. 5000078
B 6700 MASTER CONTROL PROGRAM	Form No. 5000086
B 6700 SYSTEM MISCELLANEA	Form No. 5000367

2.   SYNTAX SPECIFICATIONS

In many respects, CANDE may be regarded as a data processing language, comprising many commands the user may enter from his terminal. The detailed definition of the language is provided in later chapters of this document, as the individual commands are introduced. The current chapter presents the forms of expression, the metalanguage, used to define the CANDE language.

2.1   SYNTAX DIAGRAMS

The principal means of displaying CANDE command syntax is the syntax diagram. This method has been chosen because it affords a very concise and lucid exposition of syntax involving defaults, alternatives, and iterations; it is rigorous without being cumbersome. There are few formal rules to remember: the basic rule is that any path traced along the forward directions of the arrows will produce a syntactically valid command. The following examples illustrate the technique:

```
     ┌─,◄──┐
   ┌─▼►ROW─┬─►THE─┬►BOAT─┬►GENTLY┬►DOWN─┬►─ ─────────────┬►STREAM.
          └─►YOUR─┘      └───────┘      └─►THE─┬►OLD─┬►,─┬►MILL─┘
                                               └─────┴───┘
```
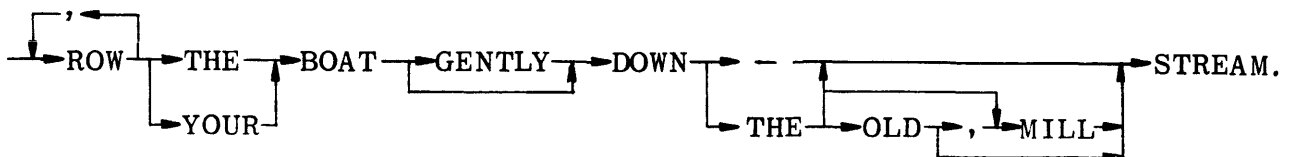
Valid productions from this syntax diagram include:

    ROW THE BOAT DOWN-STREAM.

    ROW. ROW. ROW YOUR BOAT GENTLY DOWN THE STREAM.

    ROW. ROW. ROW. ROW THE BOAT DOWN THE OLD STREAM.

    ROW YOUR BOAT DOWN THE MILL STREAM.

    ROW THE BOAT DOWN THE OLD. MILL STREAM.

The following convention is used to control the number of iterations:
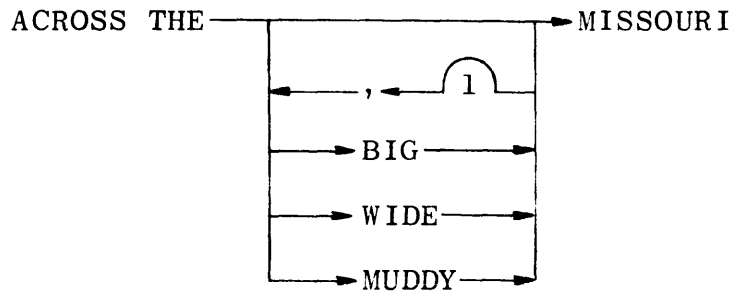
The "bridge" over the "1" can be crossed only one time, so a maximum of one comma (and two adjectives) may appear. Valid productions include:

        ACROSS THE MISSOURI

        ACROSS THE WIDE MISSOURI

        ACROSS THE BIG. MUDDY MISSOURI

        ACROSS THE MUDDY. WIDE MISSOURI

        ACROSS THE BIG, BIG MISSOURI

An "*" associated with the number under the bridge indicates that
the path must be crossed at least one time. If the previous ex-
ample is changed to the following:

```
                           ╭─1*─╮──────── , ──╮
                           │              ╱    │
ACROSS THE ───────┬───────►─BIG ────────►─┬───►MISSOURI
                  │                        │
                  ├───────► WIDE ─────────►┤
                  │                        │
                  └───────► MUDDY ────────►┘
```
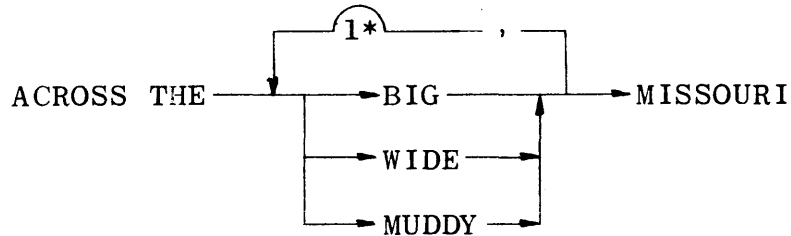
then proper syntax is obtained by crossing the bridge exactly
one time.

2.2   SYNTAX CONVENTIONS

CANDE commands are constructed of letters, digits, special char-
acters, and blanks. Letters and digits are alphanumeric charac-
ters; all other non-blanks are delimiters. Alphanumeric charac-
ters may be aggregated into such syntactic items as integers,
keywords and identifiers.
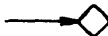
2.2.1   Keywords

Upper-case letters in syntax diagrams indicate keywords which
appear literally in the command. In many cases it is permissible
to abbreviate the keyword by its initial or first few letters.
In the syntax presentation. underscores mark the letters which
must appear, the rest are optional. Up to four letters are ex-
amined in most keywords; if more than the minimum number are
entered they must be correct through the fourth. CANDE keywords
may be entered in upper- or lower-case characters.

2.2.2   Blanks

Blanks in CANDE commands serve to separate syntactic items and
may appear freely anywhere except within certain text fields,
where they become significant characters.   Blanks are optional on
either side of a delimiter.   Whenever one alphanumeric item (key-
word, identifier, integer, etc.) follows another with no inter-
vening delimiter, they must be separated by at least one blank.
An exception is that a keyword and an integer may appear in jux-
taposition, in either order, without any separator.

2.2.3   End-of-statement

End of statement is indicated by either of two notations: The
form ———◄ indicates either semicolon or end of line; this form
is typical of most CANDE commands.   The form ———◄◇ denotes end
of line, which is necessary for some commands in which a semi-
colon would appear to be just another text character.

2.3   SYNTACTIC VARIABLES

Lower-case letters, words, and phrases in the syntax diagrams are
syntactic variables, which represent information to be supplied by
the user.   A particular variable may represent a single character,
a simple construct such as an integer or text string, or a rela-
tively complicated construct.   Most variables are defined where
used.   Several variables and types of variables which are frequently
encountered are defined here.

2.3.1 Identifier

An identifier is a string of characters used to represent some entity, such as a file or directory, a usercode, or a charge code. Identifiers in CANDE may vary in length from one through seventeen characters. In straightforward form, they are composed of letters and digits only; the characters must be adjacent. An identifier may appear between quotation marks, in which case it may contain any character except a quotation mark. Lower-case letters entered in an unquoted identifier will be translated to upper case.

Examples:

A	B3
RUMPELSTILTSKIN	172
"SUBJECT-INDEX"	95C

2.3.2 Integer

An integer is specified by a string of adjacent digits which represent the decimal value of the integer. Syntactic variables of this type occur frequently in CANDE. Some common examples are <s> (sequence number), <col> (column number), and <base> and <inc> (sequence base and increment).

2.3.3 Delimiters

In several cases, <delim> represents an arbitrary delimiter to be chosen by the user. The delimiter may generally be any non-alphanumeric character except one that occurs in the field being de-

limited or one that might have special significance in the context of its appearance. The context-dependent restrictions are detailed with each use of <delim>.

2.3.4   Text Fields

A text field, typically denoted <text> or <newtext>, is an arbitrary sequence of characters to be sought or placed in a file of data. In most text fields all characters are significant, and any character may appear except a specific delimiter.

2.3.5   Sequence Range

A <sequence range> specifies an inclusive range of sequence numbers which define a line or lines to be included (or excluded) in a particular operation.

Syntax:
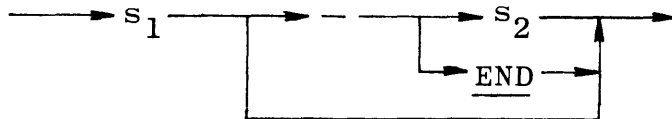
Semantics:

The integers $<s_1>$ and $<s_2>$ represent sequence numbers. If both $<s_1>$ and $<s_2>$ appear, the sequence range includes both those values and all values between; $<s_2>$ must exceed $<s_1>$. If $<s_1>$ appears alone, it defines a "range" comprising that single number. The keyword END, in place of $<s_2>$, represents the largest sequence number in the file.

2.3.6   Sequence Range List

A sequence range list specifies one or more sequence ranges, which must be disjoint. In some commands (e.g., LIST and MERGE) the ranges in the list must be in increasing order.

Syntax:
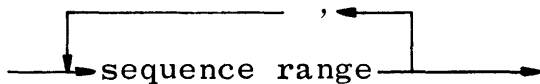
2.3.7   Column Range

A column range specifies an inclusive range of columns, defining a portion of a line.

Syntax:
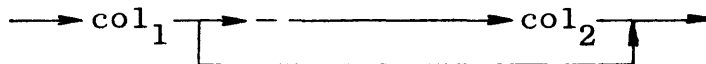
Semantics:

The integers $<col_1>$ and $<col_2>$ represent ordinals of positions on the line. (The first column is numbered 1.) If both appear, $<col_2>$ must exceed $<col_1>$, and they define the group of columns including both and all between. If $<col_1>$ appears alone, it defines a "range" comprising that single column.

2.3.8   Modifiers

A modifier is a standard system control statement. All control statements supported by the operating system and applicable to program compilation and execution are permissible, and must con-

form to B 6700 control statement specifications.

2.3.9   Logical Station Number

The logical station number <lsn> is a unique integer reference assigned by the datacom system to each station (terminal) in the datacom network.   This value is normally used when CANDE commands pertain to stations.

2.3.10   Station Name

A station name is an identifier specified in NDL to identify each station (terminal) in the datacom network.   This value may be used in CANDE commands pertaining to stations.

2.4   OTHER NOTATION

To assist in the simplicity and clarity of the discussion of the CANDE input commands and responses. certain notations are utilized. These notations are not a part of the language but are aids in its presentation.

2.4.1   Underline (_)

Messages and responses output by CANDE will be underlined; all messages not underlined are thus input by the user.

2.4.2   Broken Brackets (< >)

Left and right broken brackets are used to contain letters and/or digits representing a metalinguistic variable of the language.

3.   OPERATING FEATURES

This chapter presents the operating features which define the environment a user must work within to utilize the capabilities of the CANDE program. Consideration is given to both CANDE and B 6700 operating system aspects as viewed by the user.

3.1   USER IDENTIFICATION

Before the user can utilize the system capabilities, he must first present to the system his usercode and password. If the user is operating from a hardcopy device such as a teletype, the password may be typed into an area that the teletype has blacked out. This helps to ensure that only authorized persons gain access to the system. For screen devices, a similar degree of password security may be obtained by turning down the screen brightness before entering the password.

CANDE checks the user's identification against the system's file of authorized users. If the usercode/password entered is not correct, the user is informed by CANDE and asked to re-enter the information. If the usercode/password is correct, CANDE responds with an appropriate message and considers the station to be a valid user. The user is not able to use the system until after a successful log-in has been achieved.

The usercode is used to identify any created files; it may also be used for accounting purposes.

3.2   LOG-IN

To establish a connection for a teletype on a dial-up line, turn the teletype on and, by pushing the "ORIG" (originate) button, obtain a dial tone; then dial the computing center. When the computer accepts the call, the ringing changes to a high-pitched tone and the system types an initial message. If the phone keeps ringing, the system is temporarily unavailable; if a busy signal is heard, the system is already loaded to capacity. In either case, redialing will be necessary before a connection can be established. Terminals with acoustic couplers establish a connection in a similar manner; the phone is placed into the coupler when the high-pitched tone is heard.

To establish a connection for a terminal connected to a direct or leased line, turn the terminal on and enter the end-of-message command, or a valid usercode or usercode/password can be specified thereby completing the required log-in procedure.

For other devices, refer to the manual accompanying the device.

After a connection has been made, the system types a greeting message as follows:

B6700 CANDE <system release level> YOU ARE:<station name> (<lsn>)

Following the initial message, the system initiates the log-in sequence, which consists of the following steps:

a. The system types the following:

#ENTER USERCODE PLEASE

Then the user replies by entering his usercode, either by itself or followed immediately by a delimiter and his password.

b. If the user enters a usercode but no password, the system then responds with the following:

#ENTER PASSWORD PLEASE

For hardcopy terminal devices, it then blacks out 17 characters on the next line into which the user enters his password.

c. If the system recognizes the usercode and password, it proceeds to step d. If it does not recognize them, it types

#SECURITY ERROR, ENTER USERCODE PLEASE

and the log-in procedure begins again.

d. After completing the initial sequence for logging-in, a user may be asked to enter additional information such as a charge number (this is installation dependent).

e. If the user's previous session was aborted by a line disconnect or a system failure (H/L), CANDE created a recovery file through which the user may recover his workfile. If any recovery files exist at log-in time, CANDE will type:

#RECOVERY DATA:

    xxxx <workfilename> (<date>)

where

xxxx is the recovery index.

<workfilename> is the name of the recovered workfile, and

<date> is the date of the aborted session.

(Recovery of the desired workfile is achieved responding

with the RECOVER command, see Section 6.1.3.)

    f. Successful completion of log-in is noted by the

    message:

    #LOGGED IN <time> <date>

3.3   ENTERING DATA

Data and commands are entered in the form of messages from the
keyboard terminal or paper tape reader. Interpretation of the
characters comprising the message is governed by the installation-
dependent specification in the NDL.

3.3.1   Special Control Characters

Special characters to edit the message being prepared are also
specified in the NDL and are transparent to CANDE. The convention
supported in SYSTEM/SOURCENDL is:

end-of-message (ETX)   Control-C

backspace   Control-H

line delete   Rubout

control command   ?

3.3.2   Data Identification

If a message begins with one or more decimal digits it is assumed
to be a data record and is added to the workfile; otherwise, it
will be treated as a command.

3.3.3   CANDE Commands

The general group of commands to create and save files, to enter
data and perform editing functions, and to compile and execute
programs are CANDE commands. These inputs are processed in
serial order, and are queued if they cannot be immediately
executed.

3.3.4   Control Commands

Control commands are a separate class of input messages preceded
by the control command character (?). These commands provide capa-
bility of controlling and interrogating the user's operating en-
vironment as explained in a later chapter. Note, however, that
these commands are positioned at the head of the input queue for
that user, and therefore, will be processed before normal input.

3.3.5   Multiple Commands

A single line (record) may accommodate one CANDE command or multiple commands. Multiple CANDE commands, excluding control commands, may be entered on a single line if separated by a semi-colon (;). If a syntax or execution error occurs, all remaining commands in the record are ignored.

3.3.6   Queueing Input

Up to four input records will be queued if CANDE is not able to immediately handle a command. This will always occur if a compile or program execution is in process; it can also occur if CANDE is busy executing a previous command, e.g., MERGE.

Inputs after the fourth are discarded, in which case the user is notified.

3.4   RESPONSES

CANDE will respond in a positive manner to every command input except single-line entries. All responses will be preceded by a "#" character to indicate that the message is output from CANDE.

Commands which do not result in immediate output will be acknowledged by a # character.

For example,

        FIX 800 /EMD/END

will cause a response of the form:

    #
    ⎯

Commands which result in immediate action are noted by a specific response:

    COMPILE A/B

    #COMPILING

or

    UPDATE

    #UPDATING

Other responses include both comments and a "#" character. If several records are entered and followed by a LIST request, then the response is:

    LIST

    100 B

    200 C

    300 D

    #
    ⎯

Thus, the command is executed, and a # character output to signify end of operation.

3.5   FILES

A file is a collection of information which the user wishes to treat as a unit. It is the primary means by which he establishes continuity between one CANDE session and the next or between CANDE and other programs such as compilers.

3.5.1   Filename

A filename identifies a file in the B 6700 system; it is equiv-
alent to the B 6700 file attribute TITLE. File names, directory
structure, and file security through CANDE are treated in the
same manner and usually by the same mechanism as elsewhere in the
B 6700 system. (Exceptional handling by CANDE is required only
for object-code files: CANDE supplies an additional "hidden"
name, to permit the B 6700 MCP to distinguish source and object
files which appear to the user to have the same name.)

When a file is created through CANDE, the user supplies the
<filename> and the system adds the <usercode> to form the complete
filename.

Syntax:
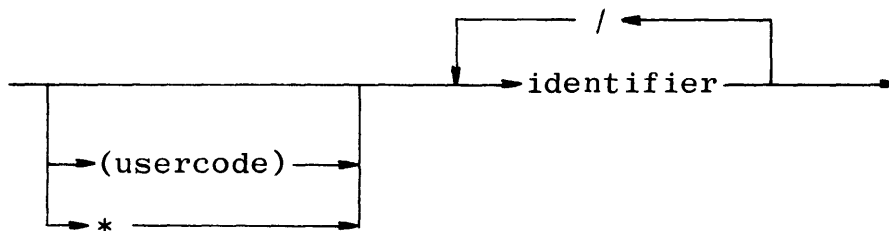
Semantics:
The simplest form, an identifier, or a list of identifiers sep-
arated by slashes, specifies a file in the user's library. A
maximum of 12 identifiers may appear, each of maximum length 17
characters; the last denotes a file of information and the rest
denote directories. The workfile name is limited to 136 charac-
ters as entered by the user.

The parenthesized (usercode) form of a filename refers to a file in another user's usercode directory; it forms part of the complete filename and distinguishes it unambiguously from files associated with other usercodes. Access to the file is subject to security constraints specified by the owner. Note that a user who references his own files in this manner is not recognized by CANDE as the owner.

Some files are stored outside the library of any particular user; these files may be referenced with the asterisk (*) form of a filename, subject to file security constraints.

If a user specifies a file, and no file with that name exists in his library, the system will access a file of that name, if one exists, in the system directory (outside any user library). Thus, if user A references file B/C and no file (A) B/C exists, he will access file * B/C if it exists. His access is constrained by the security attributes of that file, and he may not alter the file through CANDE.

The B 6700 system security disallows use of the parenthesized or asterisk form of filenaming when creating, removing or renaming a file.

The CANDE language is structured so that no file names are reserved by CANDE keywords or other syntactic considerations. (However, file names identical to keywords used in a command must be quoted to prevent ambiguity.) Two file names are reserved for

directories required by CANDE for editing and compiling a workfile and for recovery purposes; those are "OBJECT" and "CANDE". These identifiers may be used anywhere except for data or program files at the first level.

Examples:

W	A file in the user's library.
TEST/3	A directory and file in the user's library.
(SJ46)FIXIT	A file in another user's library.
*SYMBOL/PATCH	A directory and file in the system library.

3.5.2 Type

A file type indicates the purpose or intended disposition of the workfile. (The CANDE type is equivalent to the B 6700 file attribute FILEKIND.) Possible types which may be specified for a workfile are listed below; the corresponding B 6700 FILEKIND attribute mnemonic is listed in parentheses.

ALGOL	(ALGOLSYMBOL)
DCALGOL	(DCALGOLSYMBOL)
XALGOL	(XALGOLSYMBOL)
FORTRAN	(FORTRANSYMBOL)
XFORTRAN	(XFORTRANSYMBOL)
COBOL	(COBOLSYMBOL)
BASIC	(BASICSYMBOL)
PLI	(PLISYMBOL)
ESPOL	(ESPOLSYMBOL)
BINDER	(BINDERSYMBOL)
SEQ	(SEQDATA)
DATA	(DATA)

The first ten denote source files for the respective language
processors and the binder. A type SEQ file is **an** arbitrary data
file with sequence number in column 73 through 80 of each line. A
type DATA file has arbitrary data without sequence numbers.

3.5.3   Attributes

Files created for a user by CANDE are assigned the following file
attributes.

```
KIND = DISK
UNIT = WORD
MAXRECSIZE = 14
MINRECSIZE = 0
BLOCKSIZE = 420
AREAS = 15
AREASIZE = 450
SAVEFACTOR = 30
SECURITYUSE = IO
SECURITYTYPE = PRIVATE
```

Any file not created by CANDE, but presented to CANDE for proces-
sing, must satisy these conditions to prevent loss of data or in-
correct interpretation of the data.

3.5.4   Sequence Numbers

Files which are edited by CANDE must contain sequence numbers.
A sequence number is defined as a positive integer containing a
maximum of eight digits which must appear in the input data record.
The length of the sequence number must be equal to or less than
the sequence field width of that file. A sequence number is used

to identify the line for editing and to specify the position of the line in the file. Even if the lines are input out of order, CANDE arranges them in ascending numerical sequence. Except for type BASIC files, the sequence numbers are not considered to be a part of the data in a file. COBOL files are limited to six digits, BASIC files are limited to four, and the rest are limited to eight.

Pseudo sequence numbers are used to allow limited editing capabilities of type DATA files. All input records must be preceded by a sequence number of maximum length eight digits; the sequence number is not part of the data record. If the first data character is a digit, then all eight sequence digits must be provided; otherwise, leading zeros may be omitted. An update of the workfile automatically resequences the lines with a base and increment of 100, i.e., 100, 200, 300, etc.

3.5.5   User Library

The files that a user creates and saves are referred to as his user library; each file in the library must have a different name. All source files are saved unaltered as

USERCODE/<usercode>/<filename>

but the object files are entered into a separate directory:

USERCODE/<usercode>/OBJECT/<filename>

A user's library may also contain other files, such as recovery files, which CANDE generates and saves for the user.

3.6   ERROR MESSAGES

In general, errors will result in one or two line messages which are self-explanatory. Unrecognized CANDE commands are noted as "VERB REQUIRED" errors, while incorrect syntax gives a message followed by the item being scanned when the syntax error was noted.

Program execution errors are noted in typical B 6700 system fashion: fault condition, segment-address information and source line number.

All error messages are preceded by a "#" character.

4. CANDE MECHANISMS

4.1. WORKFILE

CANDE may be directed to read an arbitrary file for various purposes, but at a given time it will effect changes to the content of only one file, which is known as the workfile. The user may create a new workfile, using MAKE, or he may recall an existing file, using GET. He may augment or modify his workfile, by entering single lines or by invoking editing commands; he may examine or search any part of the file. He may compile and execute the file, in the case of programs, and may save the file (and any object version) in his permanent library.

A CANDE workfile may consist of one or two parts. The "tank" part exists for every workfile in use: as new lines and editing commands are entered, they are kept in a tank file (which belongs to CANDE). A "file" part exists for any but a brank new workfile; it is a separate, conventional disk file (in the user's library) which contains the bulk of the data. Whenever necessary, or upon demand, CANDE will UPDATE the workfile by applying the changes in the tank to the data in the file (if any) and writing a new file. The subdivision into tank and file is automatic and need not normally concern the user. He should be aware, however, that his corrections and commands are deferred (tanked) as long as possible, with the time-consuming UPDATE being invoked only when required to save, output or examine the file.

When a workfile is a program, it may acquire an object-code version by being COMPILEd or RUN. The object code may then be RUN, EXECUTEd or SAVEd; it is lost if unsaved and any changes are made to the workfile.

CANDE will create separate files in a user's library for any unsaved source or object version of the workfile. These files are:

> CANDE/TEXTn   an unsaved (source) workfile
>
> CANDE/CODEn   an unsaved object file

where n is a unique integer assigned by CANDE.

4.2   RECOVERY

Recovery of a user's workfile occurs whenever a station is disconnected (datacom problem) or a system failure (H/L) occurs, and changes to his workfile exist. (A saved workfile without pending changes is not recovered; the GET command may be used to recall the workfile.) The recovery mechanism involves editing a permanent CANDE disk file which contains the "tank" file portion of the user's workfile, and entering this information in a separate file in the user's library titled

> CANDE/RECVn

where n is a unique integer assigned by CANDE. This recovery file is then presented to the user when he logs-in following the failure.

It is possible to lose some of the user's input information since CANDE only updates the disk copy of the user's "tank" file periodically; normally, not more than four input records will be lost.

5.   PROGRAM EXECUTION

Program execution commands provide the ability to initiate system programs including standard compilers and utility routines, and to execute user-developed programs.   In general, the commands' syntax and semantics are similar to normal system definition; however, certain changes are required to distinguish user files from context-dependent keywords.

A program execution command may be specified across multiple input records.   This is accomplished via the use of a special line continuation character, the percent sign (%).   When used as a line continuation character, the percent sign must be the last item of the current input line; it may be entered at any point where blanks are permitted and insignificant (between items but not within them).   CANDE acknowledges the indicated command continuation by typing

    # %

and then awaits additional input.

Multiple CANDE commands may be specified in a single input record. However, potential conflicts exist between interpretation of the information as a modifier or as a separate CANDE command.   If the keyword following a semicolon after the execute command (or a modifier) unambiguously introduces a CANDE command, it is so treated; otherwise, it is treated as a modifier.

5.1   BIND/COMPILE

The BIND and COMPILE commands invoke the standard system binder
and compilers or any non-standard versions. Options provide the
ability to use an existing file or the workfile, to specify the
object filename, to specify the compiler name or the source file
type, and to specify label-equation and other control statements
for both the compile and resultant object program.

Syntax:
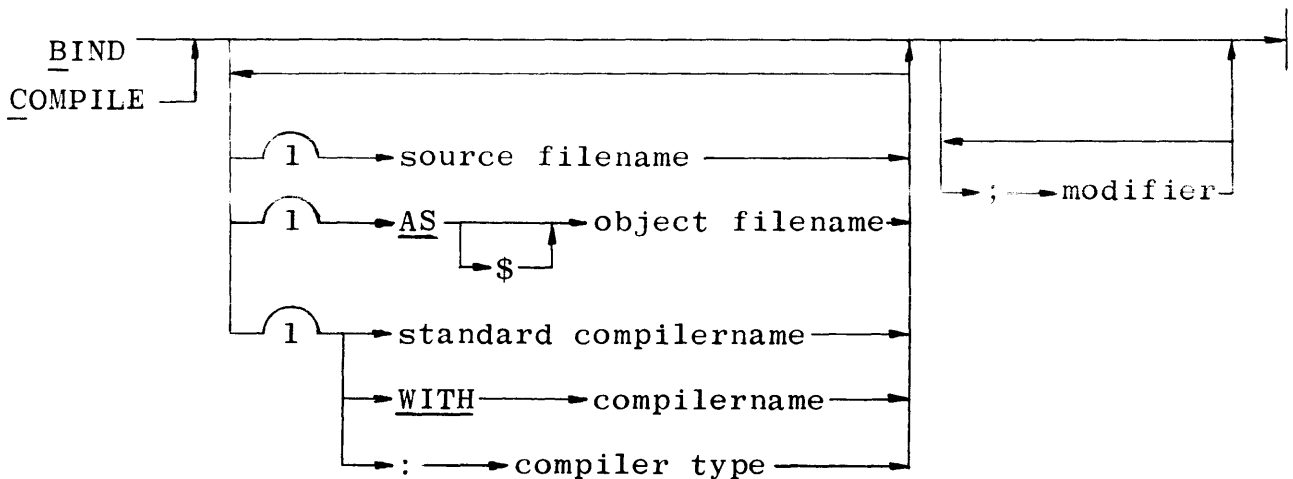
Semantics:

If <source filename> is not specified, then the workfile will be
used as the source file. If the <object filename> is not provided,
then the resultant object file will have the source filename. Per
CANDE convention, the object file will be placed in the user's
OBJECT file directory. i.e., USERCODE/<usercode>/OBJECT/<object
filename>. If a "$" precedes the <object filename>, the OBJECT
file directory is omitted, thus allowing normal system referencing.

(The use of (usercode) or "*" with the object filename is not allowed.)

A compiler may be specified, or by default, a standard compiler or the binder is used depending on the source file type; an error condition results if a compiler is not given for type DATA and SEQ files. If a <compilername> is provided, it may be a standard compilername or any non-standard compiler which is preceded by WITH; the source file type is ignored in this circumstance.

A <compiler type> may be specified if preceded by a colon.

The standard compilernames and compiler types are:

COMPILERNAME	COMPILER TYPES
ALGOL	ALGOL
XALGOL	XALGOL
DCALGOL	DCALGOL
FORTRAN	FORTRAN
XFORTRAN	XFORTRAN
COBOL	COBOL
BASIC	BASIC
PL/I	PL/I
ESPOL	ESPOL
BINDER	BINDER

Modifiers may be included for the compiler and/or the object program.

Note that in order to prevent ambiguity, it is necessary to disallow the use of standard compilernames and "AS" and "WITH" as <filenames> unless contained within quotes.

Examples:

C

Compile the workfile using the default compiler type and associate the object file with the workfile.

COMPILE A/B AS $C/D : DC

Compile the user file A/B with DCALGOL generating an object program named USERCODE/<usercode>/C/D

C B;OPTION FAULT ARRAY FILES

Compile the user file B using the default compiler type. The programdump option FAULT with options ARRAY and FILES is specified for the object file.

5.2 EXECUTE/RUN

The EXECUTE and RUN commands cause execution of an object program;
the RUN command will provide for compilation or binding of the
source file prior to execution if the object file is not available.

Syntax:

```
RUN ─────────┬───┬─────────────────┬──┬──────────────────────┬──┐
             │   └──► filename ─────┘  └──► ; ──── modifier ──┘  │
             │                                                   │
EXECUTE ─────┴──► $ ─┘
```

Semantics:

The workfile is assumed if no filename is provided. A "$" is
specified to indicate that the given filename is not in the OBJECT
file directory but is referenced as a normal system file, i.e.,
USERCODE/<usercode>/<filename>.

A <filename> in EXECUTE may refer to any file to which the user
is permitted access, but a <filename> in RUN must refer to a file
in the user's library.

Modifier specifications may be provided.

Examples:

    E

        Execute the object program associated with the workfile.

    RUN A/B; FILE LINE (KIND=REMOTE)

        Execute the program A/B and label-equate the output

        to the user's remote terminal.

5.3   REMOTE FILES

A user of CANDE must be aware, to some extent, of the special
nature of a remote file. The comments included herein are, of
necessity brief and restricted.

5.3.1   File Characteristics

An object program may treat a remote terminal as a file; this is
achieved by specifying the file attribute "KIND" as "REMOTE" in
the file declaration or by a control statement (label-equation).
Thus execution of an object program containing the file declara-
tion

     FILE F (KIND=PRINTER)

will cause any output to file F to be directed to the site
printer. However, if the execution command includes correct
label equation, i. e.,

     EXECUTE <object program> ; FILE F (KIND=REMOTE)

then the output is directed to the user's terminal.

Output may be directed to another CANDE terminal by also specify-
ing its <lsn> via the control statement STATION. For example,

     EXECUTE <object program> ; STATION=4 ; FILE F (KIND=REMOTE)

will assign the remote file F to the terminal whose <lsn> is 4.

Use of the TITLE file attribute to assign a station to a remote
file is only accomplished by specifying STATION = 0 in conjunction
with the TITLE. For example,

    EXECUTE <object program> ; FILE F (KIND=REMOTE, TITLE = M331)
       ; STATION = 0

will direct the remote file F to the terminal whose title is M331.

5.3.2 Multiple Files

More than one open file may be assigned simultaneously to a
terminal. However, there never can be more than one input or
input/output file open for a terminal at any one time. (Users
should therefore make careful use of the "MYUSE" file attribute
when using a terminal for multiple files which are simultaneously
open.) Also note that a single file may reference a family of
stations.

5.3.3 Control Messages

During execution of an object program referencing the user's
terminal as an input file, all input other than control messages
will go to the object job and will not be seen by CANDE. (See
Chapter 7 for discussion of control messages.)

6. COMMAND AND EDIT COMMANDS

The CANDE commands to create and build files, make inquiries, edit symbol files, etc., are described in this section. The commands are presented in groups according to general function. For example, all commands dealing with workfile manipulation are grouped together as are the output commands.

6.1   WORKFILE COMMANDS

6.1.1   GET

The GET command recalls an existing file as the workfile.
If an unSAVEd workfile already exists, the GET command is in
error.

Syntax:

GET→filename

```
          ┌──────────────────────────────┐
GET→filename─┤                            ├──┤
          │  ┌1┐──→ sequence range list──→│
          └──┤1┘──→ AS ──→workfilename ───┘
```
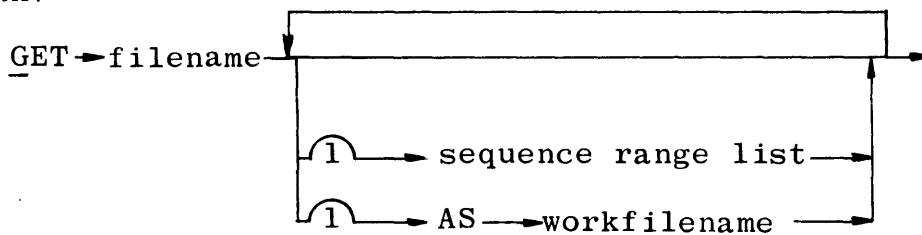
Semantics:

The <filename> may specify any file to which the user is per-
mitted read access. If desired, only a portion or portions of
the file may be recalled by providing a sequence range list.

A workfile name may be provided using the AS <workfilename>
capability; otherwise the name of the recalled file is used pro-
vided that (1) it is a file within the user's library, and (2)
there is no sequence range list specified. If either condition
is not met, the workfile remains unnamed; a name must be provided
later by a TITLE or SAVE command.

The type and other attributes of the workfile are set to those of the file specified by GET.

Note that the file is not actually read as a result of the GET command. It will be read when the first UPDATE occurs. The user must assure that the file he GETs remains present and unmodified until that time.

Examples:

    GET IF

        Get an existing user file named IF as the file part of a new workfile.

    GET IT 100-9540

        Get lines 100 through 9540 of the existing user file named IT as the file part of the workfile; the work-file remains unnamed.

    GET (HIS) NOTMYFILE AS WORKFILENAME

        Get another user's file named NOTMYFILE as the file part of the new workfile named WORKFILENAME.

6.1.2   MAKE

The MAKE command creates a new workfile.   If an unSAVEd workfile already exists, the MAKE command is in error.

Syntax:

MAKE ———►filename ┬─────────────────────────►┤
                  └─────────►type ─┘
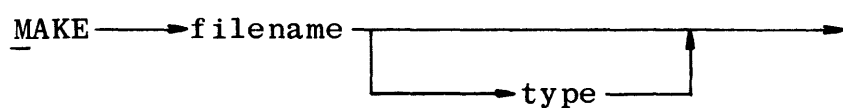
Semantics:

The <filename> must define a new file within the user's library; it becomes the name associated with the workfile.   A type SEQ file is assumed by default.

Examples:

    MAKE IT

        Make a new workfile with the name IT and a default
        type of SEQ.

    M NEWWORKFILE A

        Make a new workfile of type ALGOL called

        NEWWORKFILE.

6.1.3   RECOVER

The RECOVER command recalls a recovery file as the workfile.

Syntax:

```
RECOVER ──────────┬─────────────────────┬──────┤
                  └──► recovery index ───┘
```
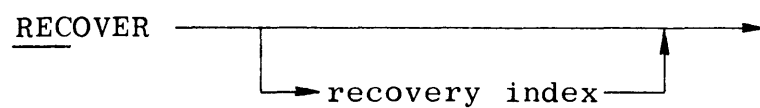
Semantics:

If a recovery index is not specified, CANDE will provide a list
of all recovery file:

≠RECOVERY DATA:

      xxxx <workfilename> (<date>)

where

    xxxx is the recovery index.

    <workfilename> is the name of the recovered workfile, and

    <date> is the date of the aborted session.

Recovery of a workfile is achieved by specifying the desired re-
covery index.

Example:

    REC   400

6.1.4   REMOVE

The REMOVE command may be used to remove the workfile or any file in the user's library. (The user is prevented by the B 6700 file security system from removing another user's files.)

Syntax:

```
                                        ,
        ┌──────────────────────────────────◄──────────────────┐
REMOVE ──┬───────────────────────────────────────────────────────┤├──
         │                                                   ▲
         │   ┌──►SOURCE──┐          ┌──►filename──┐
         │   │           ▲          │             │
         │   └──►OBJECT───┴──────────┴──►$ filename─┘
         │
```
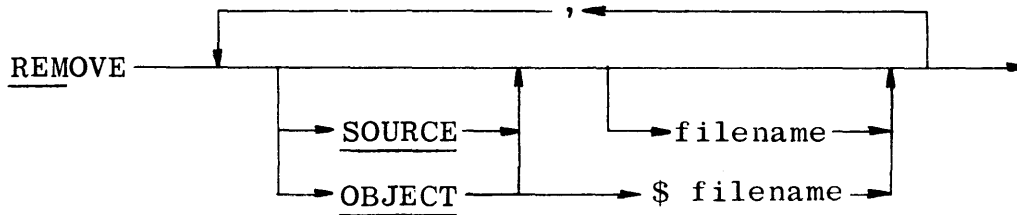
Semantics:

The semantics vary with the form of the command:

1. REMOVE

    The workfile and any associated object file are removed; no further editing may be performed until another MAKE or GET is performed. Note that the antecedents of the workfile are un-affected by this command; thus the sequence "GET A; REMOVE" has no effect upon the user's library file A.

2. REMOVE SOURCE

    The workfile is removed, but any object file remains available to be RUN, EXECUTEd, or SAVEd.

3. REMOVE OBJECT

    The object file is removed, the (source) workfile remains.

4. REMOVE filename

    If any file, including a directory, with the specified name exists in the user's library, it is removed. If an object

file named OBJECT/filename exists. it is also removed.

5. REMOVE SOURCE filename

   If a source file or directory with the specified name exists
   in the user's library. it is removed.

6. REMOVE OBJECT filename

   If an object file named OBJECT/filename or a directory named
   <filename> exists in the user's library. it is removed.

7. REMOVE OBJECT $ filename

   If an object file exists in the user's library as the name
   specified without prefacing by "OBJECT.". it is removed.

More than one file may be listed to be removed. Once the "SOURCE"
or "OBJECT" specification appears. it applies to all subsequent
file names until countermanded by the other specification.

Use of filenames "SOURCE" and "OBJECT" must be quoted to prevent
ambiguity with the keywords SOURCE and OBJECT respectively.

Note that removal of a directory effects removal of all files
included in that directory. Thus the files

    A/B/C
    A/B/D
    A/B/E/F
    A/B/E/G

will all be removed by the command

    REM A/B

When a file is removed, it immediately disappears from the system directory, so no program can find it. The actual data areas remain available to any program that already had that file open at the time of removal.

Examples:

REM

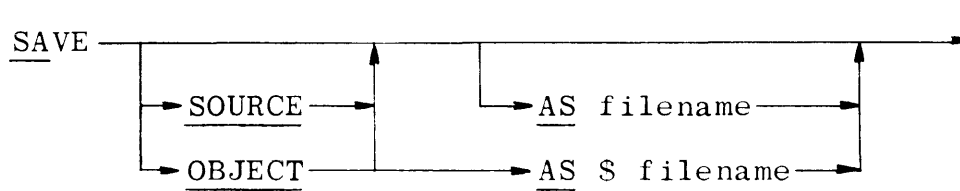Remove the workfile and any associated object file.

REMOVE SOURCE A, B, C/D, OBJECT A

Remove source files A, B, and C/D, and object file A from the user's library.

6.1.5   SAVE

The SAVE command causes the current workfile and/or its associated object file to be saved in the user's library.

Syntax:

Semantics:

The semantics vary for the several forms of this command:

1.   SAVE

The workfile is saved in the user's library using the current workfile name: any existing file of that name is removed.   If an object file is associated with the workfile, it is saved in the library using the current workfile name prefaced by "OBJECT/": any existing file of that name is removed.   This form is invalid if the workfile is unnamed.

2.   SAVE SOURCE

The workfile is saved as in (1): the object file is unaffected.

3.   SAVE OBJECT

The object file is saved as in (1): the workfile is unaffected.

4.   SAVE AS filename

The workfile is saved in the library under the name supplied, which must be a new file within the user's library.   If an object file is associated with the workfile, it is saved in

the library under the name OBJECT/filename; which must be a new filename within the user's library.

5. SAVE SOURCE AS filename

    The workfile is saved as in (4); the object file is unaffected.

6. SAVE OBJECT AS filename

    The object file is saved as in (4); the source file is unaffected.

7. SAVE OBJECT AS $ filename

    The object file is saved in the library under the name specified without prefacing by "OBJECT/", which must be a new filename in the user's library. The source file is unaffected. This form provides filename compatibility with object programs run from card decks at the site or via a remote-job-entry terminal.

A SAVE immediately following another SAVE implying the same name is redundant. If the name is different, the source file will be copied so that it may exist under both names. It is not possible to save an object code file under more than one name since CANDE cannot copy it; an attempt to do so will be rejected.

Note that after a workfile has been saved, it is still available for further editing. The recently saved file serves as the file part of the workfile; it therefore must remain present and unmodified until the next UPDATE.

An object file saved (but not saved AS) remains associated with the workfile and is available to be RUN or EXECUTEd until any changes to the workfile are entered.

Examples:

SAVE

> Save the workfile and object file. if it exists, in the user library.

SAVE SOURCE AS BACKUP

> Save the workfile with title BACKUP in the user library; the object file. if it exists. is unaffected.

SAVE OBJECT AS $SYSTEMFILE

> Save the object file associated with the current workfile as SYSTEMFILE in the user library, but do not use the OBJECT directory as per normal CANDE convention.

6.1.6   TITLE

The TITLE command changes the title (name) of the workfile or a file in the user's library.

Syntax:
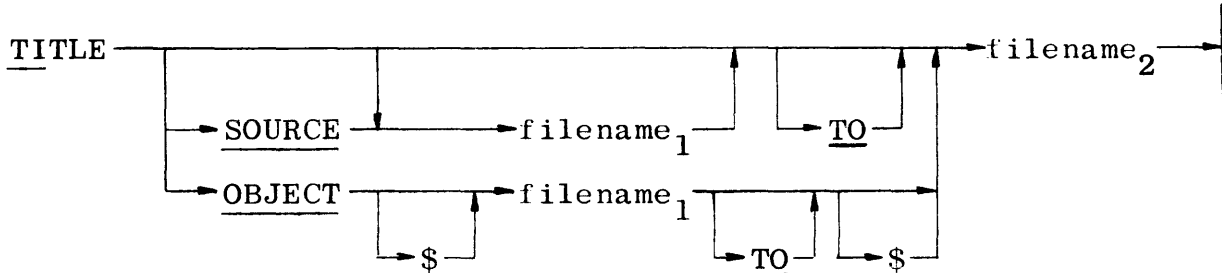
Semantics:

If <filename$_1$> is not specified, the name of the workfile is changed for any subsequent SAVE command. For a named file, the source file or object file, or both, may be changed.
For an object file, $ may be specified to indicate that the referenced file is not, or should not be in the user's OBJECT directory.

The name of the file is changed to <filename$_2$>, which must be a new file name within the user's library.

A <filename$_1$> of TO and SOURCE must be quoted.

Example:

    TI   SOURCE   A   TO   B
        Change the source file A to B.

    TITLE OBJECT TEST $ GOOD

        Change the user's object file TEST, stored as USERCODE/
        <usercode>/OBJECT/TEST to GOOD, stored as USERCODE/
        <usercode>/GOOD

6.1.7   TYPE

The TYPE command changes the file attribute FILEKIND of the workfile, or a file in the user's library.

Syntax:

Semantics:

The file attribute FILEKIND of a source file in the user's library or, by default, of the workfile is changed to the type specified. This attribute determines which standard compiler will be invoked by the COMPILE or RUN command if a specific compiler is not provided, and will dictate the position and length of the sequence numbers.

A filename of TO must be quoted.

Examples:

    TYPE DC

        Change the workfile to type DCALGOL

    TY BASIC

        Change the workfile to type BASIC

    TYP A A

        Change the user's source file named A to ALGOL

6.1.8 UPDATE

The UPDATE command forces immediate update action on the workfile. It is not necessary that the user explicitly invoke this command; it is invoked automatically whenever needed to permit another command to function properly. The user may wish to force UPDATE action, for instance to assure that the file part of his workfile is a private copy.

Syntax:

UPDATE ─────────────▶|

Semantics:

UPDATE forces all the changes/additions in the tank to be incorporated with the file part to produce a new file part for the workfile. In some cases, more than one pass through the user's workfile may be required to accomplish the UPDATE.

If the workfile already consists of a file part in the user's library with no changes pending, the UPDATE command has no effect.

6.1.9   WHAT

The WHAT command indicates the state of the workfile.

Syntax:

WHAT ————————————▶|

Semantics:

The amount of information provided varies depending on the current state of the workfile; as much of the information as is relevant (available) is indicated.   Output may include:

a.   TITLE (name)

b.   TYPE (filekind)

c.   number of records

d.   sequence number of last record

e.   object file present

f.   SAVEd or unSAVEd status

Example:

WHAT
#WORKFILE TEST: ALGOL, 78 RECORDS(THRU 830), SAVED
#OBJECT FILE PRESENT, UNSAVED

6.2   EDITING COMMANDS

6.2.1   DELETE

The DELETE command discards lines from the workfile.

Syntax:
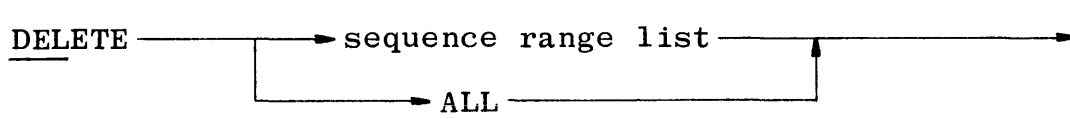
Semantics:

Those lines specified in the sequence range list are deleted from the workfile. If "ALL" is specified, the entire contents of the workfile is deleted, but the name and other attributes are pre-served. If the workfile is unnamed, it is removed by DELETE ALL.

Example:

DEL ALL

    Delete the entire contents of the workfile.

DELETE 0-599

    Delete lines 0 through 599 from the workfile.

DELE 25700-END

    Delete all lines from 25700 through the end of the workfile.

6.2.2   FIX

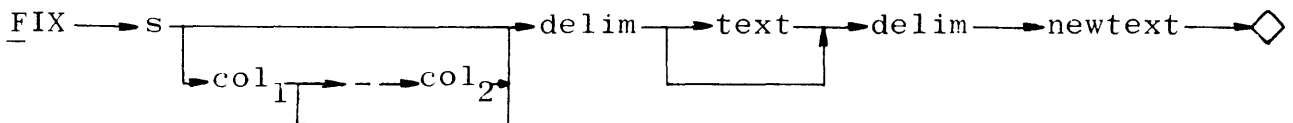The FIX command alters the contents of a line in the workfile, by inserting new text or replacing part of the line. The alteration may be controlled by specifying target text, column numbers, or a combination of these.

Syntax:

$$\underline{F}IX \longrightarrow s \begin{array}{c} \longrightarrow \\ \longrightarrow col_1 \longrightarrow - - \longrightarrow col_2 \end{array} \longrightarrow delim \longrightarrow text \longrightarrow delim \longrightarrow newtext \longrightarrow \diamondsuit$$

Semantics:

The integer <s> is the sequence number of the line to be modified. if there is no such line in the file, the command is disregarded.

The semantics vary for several forms of this command, depending upon the presence of column numbers and text:

1.   text, no columns:

The specified line is scanned from left to right for a literal appearance of the character string specified as text. When located, the target text is replaced by the character string specified as newtext. If the target is not found, the FIX is disregarded. Only one replacement is made on the line.

2. text. one column:

The specified line is scanned as in 1., beginning at col_1.

3. text, two columns:

The specified line is scanned as in 1., within the range col_1 through col_2.

4. no text, no columns:

The newtext is inserted at the beginning of the line.

5. no text, one column:

The newtext is inserted beginning at col_1.

6. no text, two columns:

The characters in columns col_1 through col_2 are replaced by the newtext.

If a column number outside the text field is specified. or if col_1 exceeds col_2, the FIX command is rejected.

The <delim> which bracket the target text may be any delimiter except hyphen or percent. The text may contain any characters except the delimiter. All characters. including blanks, are significant in the text; the text field is empty only if the two delimiters are in adjacent columns. The newtext field begins immediately after the second delimiter and runs to end of record. Blanks are significant; newtext is empty if end of record immediately follows the second delimiter. Adjustment for width of insertion or replacement is made by shifting the right-hand end of the text field left or right, deleting or adding terminal blanks. A

line overflow error is noted whenever adjustment shifts non-blank characters off the end of the line. Note that the entire line is shifted. whether or not column numbers were specified.

There is considerable functional overlap between the FIX and REPLACE commands. FIX is specialized for single replacements on single lines and has a very concise syntax. REPLACE is more general, with file-searching and output capability; its various options require a more elaborate syntax. FIX is more efficient to use when its capabilities are adequate.

Examples:

    FIX 398 1-39 :AB:ABB

        Locate the target text "AB" within columns 1 through 39
        of line 398, and replace it with the newtext "ABB".

    FIX 500 /END;/END ELSE

        For line 500, locate the text "END;" and replace it
        with the newtext "END ELSE".

    FI 10 40//% ADJUST LINKAGE

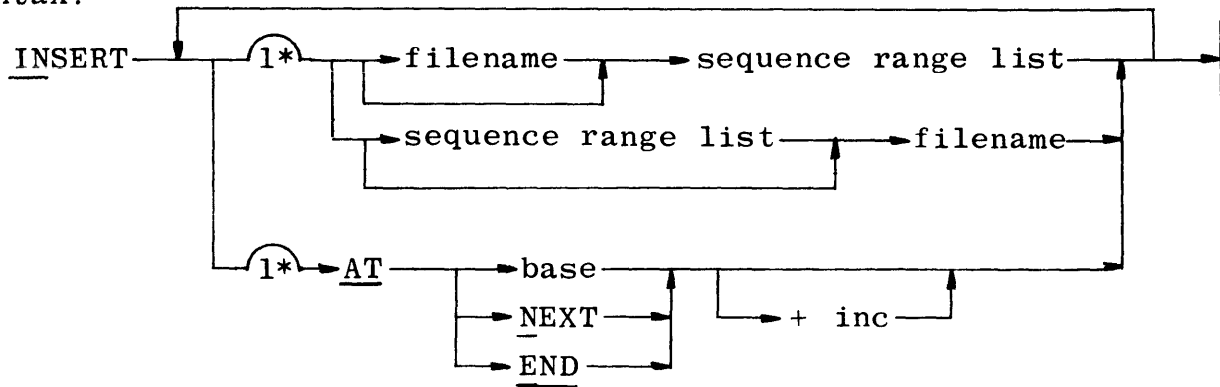   Insert newtext, beginning in column 40 of line 10.

    F 2 1-5//

        Delete the first 5 columns of line 2, shifting the image
        to the left.

6.2.3   INSERT

The INSERT command copies lines from the workfile, or from a file
in the user's library, and places the copies into the workfile
with new sequence numbers.

Syntax:

Semantics:

INSERT copies lines from any data file to which the user has read
access or from the workfile by default, and enters them in the
workfile as a contiguous block. The entire file will be copied
if a <filename> is specified and the sequence range list is omit-
ted. Use of <filename> "AT" is allowed if quoted.

New sequence numbers are determined by assigning an initial value
to the first line and incrementing that value for each succeeding
line. Three forms of specification are available:

1. An integer <base> specifies the initial value explicitly.

2. NEXT sets the initial value to the next sequence number that
    would have resulted from the most recent MOVE, INSERT, RESEQ,
    or SEQ command; the default value 100 is used if no such

command has appeared since MAKE, GET, or DELETE ALL.

3.   END sets the initial value to the largest sequence number cur-
rently in the file, plus the current specified (or default)
increment.

The increment <inc> for successive sequence numbers in the
block may be specified as an explicit integer. If none appears,
the increment from the most recent MOVE, INSERT, RESEQ or SEQ
command is used, or the default value 100 is used if no such
command has appeared since MAKE, GET, or DELETE ALL.

The range of new sequence numbers may not overlap any lines
already in the file, nor may the numbers exceed the largest
sequence number which may be expressed in the sequence number
field. The destination may not overlap the source when using
the workfile.

Examples:

    INSERT AT 20000 100500 - 101000 DATAFILE

        Copy from file DATAFILE the lines 100500-101000 and
        insert at 20000 using the default increment.

    INS 2670-2931 AT END

        Copy lines 2670 through 2931 of the workfile at the end
        of the workfile. If the highest sequence number in the
        workfile is 71900 and the last-used (or default) incre-
        ment is 100, then the copies will go to 72000, 72100, ...

6.2.4 MERGE/RMERGE

The MERGE and RMERGE commands cause a specified file, or portions thereof, to be merged with the workfile, with the result becoming the new workfile. The two commands differ in precedence consider- ations when a record in the merge file has the same sequence number as a record in the workfile: MERGE keeps the workfile record and discards the other; RMERGE does the reverse.

Syntax:

```
MERGE ──┬──┬────╭─1*─╮────────────►filename───────────┬──────┬──┤
RMERGE─┘  └─╭─1─╮──────────►sequence range list───────┘
```
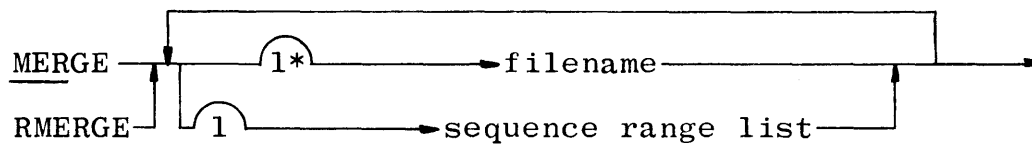
Semantics:

The <filename> may specify any data file the user is allowed to read. If a sequence range list appears, only the selected lines are merged and the rest of the file is ignored. The sequence num- bers of lines in the merge file remain unchanged as they enter the workfile.

Examples:

MERGE SYMBOL

> Merge the workfile with the complete file SYMBOL, keeping the workfile copy of records with identical sequence num- bers.
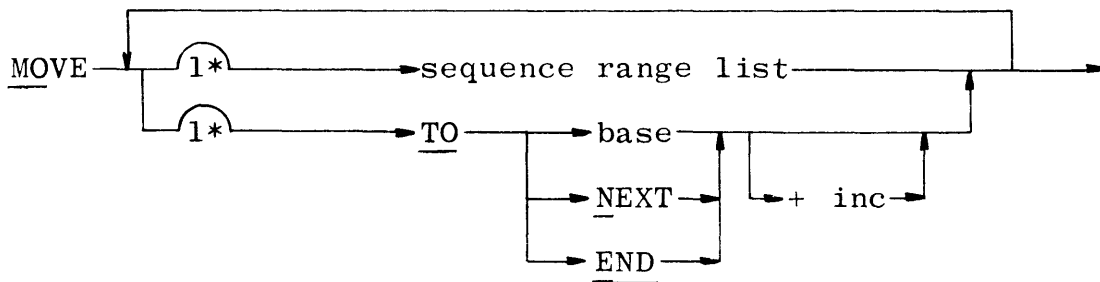
RM 400-END PATCHES

Merge the lines from 400 to end of file PATCHES with the workfile, discarding the workfile copy of duplicate sequence numbers.

6.2.5   MOVE

The MOVE command moves lines from one point to another within the
workfile, and changes their sequence numbers.

Syntax:

```
MOVE ──┬──⟨1*⟩──────────►sequence range list─────────────────┤
       └──⟨1*⟩──────────►TO───┬──►base──────┐  ┌──────────┐
                              ├──►NEXT──┤  └──+ inc──┘
                              └──►END───┘
```

Semantics:

The lines in the sequence range list are deleted after being
entered in the workfile as a contiguous block.

New sequence numbers are determined by assigning an initial value
to the first line and incrementing that value for each succeeding
line.   Three forms of specification are available:

1.   An integer <base> specifies the initial value explicitly.

2.   NEXT sets the initial value to the next sequence number that
     would have resulted from the most recent MOVE, INSERT, RESEQ
     or SEQ command; the default value 100 is used if no such com-
     mand has appeared since MAKE, GET, or DELETE ALL.

3.   END sets the initial value to the largest sequence number cur-
     rently in the file, plus the current specified (or default)
     increment.

The increment <inc> for successive sequence numbers in the
block may be specified as an explicit integer. If none appears,
the increment from the most recent MOVE, INSERT, RESEQ or SEQ
command is used, or the default value 100 is used if no such
command has appeared since MAKE, GET, or DELETE ALL.

The range of new sequence numbers may not overlap any lines
already in the file, nor may the numbers exceed the largest
sequence number which may be expressed in the sequence number
field. The destination may not overlap the source.

Examples:

MOVE 234-417 TO 595+5

    Remove lines numbered 234 through 417 and reinsert them
    at 595, 600, 605, ...

MOVE 1400 TO 1150

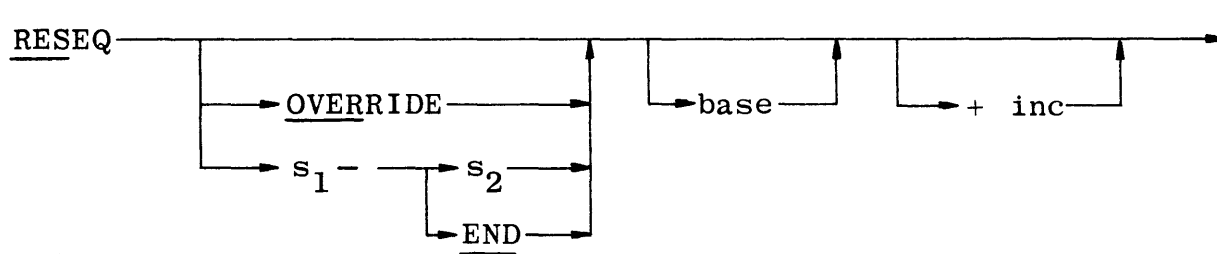   Remove line 1400 and insert it as line 1150.

MOVE 16700-END, 5100-5250 TO 1710 + 10

    Move lines numbered 16700 and beyond, followed by the
    lines from 5100 through 5250, numbering them 1710, 1720,
    1730, ...

6.2.6   RESEQ

The RESEQ command assigns new sequence numbers to lines in the

workfile, without changing the order of appearance of any lines.

Syntax:

```
RESEQ ─────────┬──────────────────────────────┬──────┬───────┬──────────┐
               │        ┌──►OVERRIDE──────┐    │  └──►base──┘  └──►+ inc──┘
               ├────────┤                 ├────┤
               └──► s₁ ─┴──┬──► s₂ ──────┤
                           └──►END───────┘
```

Semantics:

An initial value is assigned as the sequence number of the first

line to be renumbered; the value is incremented for each subsequent

line.

If a range of sequence numbers is provided, only the specified

part of the workfile is renumbered; by default, the entire work-

file is renumbered.

An initial value may be specified as the integer <base>; otherwise

the starting value of the sequence range is used. The increment for

successive new sequence numbers may be specified as the integer

<inc>. If none appears, the increment from the most recent MOVE,

INSERT, RESEQ, or SEQ command is used, or the default value of 100

is assumed if no such command has occurred since a MAKE, GET or

DELETE ALL command.

If sequence bounds are provided, the new sequence numbers must fall within the bounds; RESEQ will not change the order of appearance of the lines. Thus s_1 must be less than or equal to <base> and the <base> plus the number of lines times the <inc> must not exceed s_2 + <inc>.

OVERRIDE causes the sequence numbers currently in the file to be ignored. This option may be used only with a complete file with no pending changes. The expected application is GET <filename>; RESEQ OVERRIDE, thereby sequencing a file which previously was unacceptable to CANDE due to sequence errors.

Resequencing of workfiles of type BASIC requires special processing since the sequence numbers (line numbers) are part of the symbolic information. A separate program SYSTEM/RESEQBASIC is initiated by CANDE to perform this function. This program resequences all or part of the workfile, and updates the symbolic information to include new sequence numbers where appropriate, e.g., ON statements. References within the symbolic information to non-existent sequence numbers are noted at the user's terminal as warnings, but are not fatal errors.

Examples:

RESEQ

> The entire workfile is renumbered; the initial value is 100. If no increment has been specified for this workfile, 100 is used, so the new numbers are 100, 200, 300, . . . ,

RESEQ 10000000 + 10000

> The entire workfile is renumbered; the lines are numbered
> 10000000, 10010000, 10020000, ...

RESEQ 4700-13699 + 50

> Lines numbered 4700 through 13699 are assigned new numbers
> beginning at 4700 and incrementing by 50.

RESEQ 212-252 215+5

> Lines numbered 212 through 252 are renumbered; if there
> are eight such lines they become 215, 220, 225, ...,
> 245, 250. This RESEQ command would cause a range-overlap
> error if there were more than eight lines.

6.2.7   Single-Line Entry

Any line beginning with a digit is a "command" to enter a new line of text at the sequence number specified, or to replace or delete the line already at that sequence number.

Syntax:
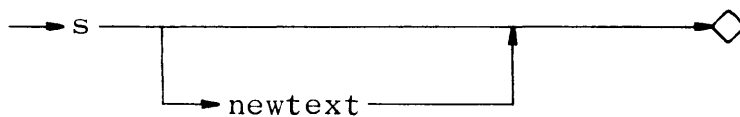
Semantics:

The line consists of an integer sequence number, <s>, followed immediately by an optional newtext field supplying the contents of the new line.   The sequence number begins in the first column of the input line, and runs until a non-digit character is encountered, or until the maximum number of digits for a sequence number have been entered; newtext field begins in the column immediately following.

If there is already a line of text at the specified sequence number, that line is replaced by the newtext, if present, or is deleted if newtext is empty.   Otherwise, a new line at that number is created with the newtext.   An entry with a new sequence number and no newtext is disregarded.

Examples:

    100THIS IS A SINGLE LINE ENTRY

    215AND SO IS THIS

    50ME TOO
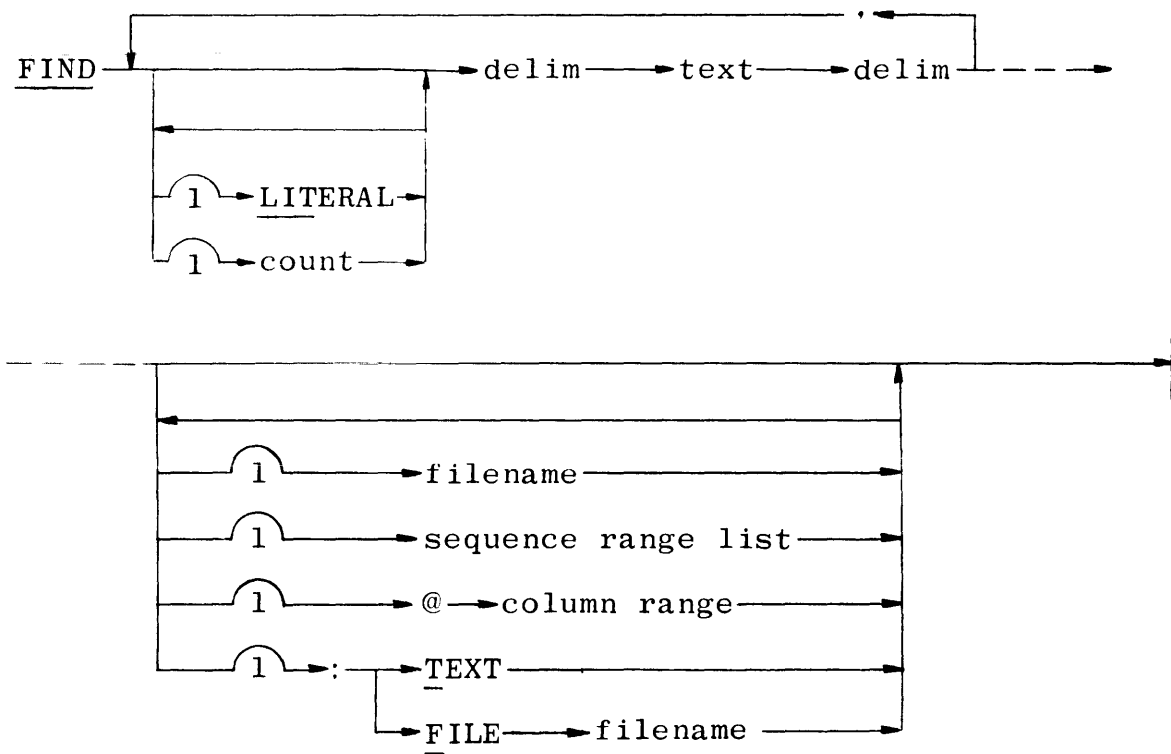
    100I REPLACE THE PREVIOUS LINE 100

    00000300716 I AM A LINE BEGINNING "716" AT SEQ 300

6.3 SEARCH COMMANDS

6.3.1 FIND

The FIND command searches a file for appearances of specific text. Output indicating the result of the search may be directed to the terminal or a new file.

Syntax:

```
                                                    ,
                                              ┌─────←─────┐
       ┌──────────────────────────────────┐  │           │
FIND ──┴──┬──────────────────────┬──→ delim ──→ text ──→ delim ──┴ ─ ─ ─→
          │  ┌─────────────────┐ │
          └──┤                 ├─┘
             ├──( 1 )─→ LITERAL ──┤
             └──( 1 )─→ count ────┘


──── ─ ─ ──┬─────────────────────────────────────────────┬──────────────┤
           │ ┌───────────────────────────────────────┐   │
           ├─┤                                        ├──→┤
           │ └──( 1 )─────────→ filename ─────────────┘   │
           ├────( 1 )─────────→ sequence range list ──────┤
           ├────( 1 )──────→ @ ─→ column range ───────────┤
           └────( 1 )──→ : ─┬──→ TEXT ────────────────────┤
                            └──→ FILE ──→ filename ────────┘
```

Semantics:

One or more targets may be sought in one FIND; each is specified in a text field between two delimiters, <delim>. The delimiters surrounding the target may be any delimiter except percent. The text may contain any characters except the delimiter. Dif-

ferent text fields may be bracketed by different delimiters. The text field may not be empty.

For purposes of the search, the text field may contain either tokens or a literal string. The default mode is token; literal mode is invoked by the keyword LITERAL, and applies to the following text field only. In token mode, a file is considered to be a sequence of tokens, where a token is any group of adjacent alphanumeric characters (digits and letters) or any non-alphanumeric non-blank graphic character. Any number of blanks may separate tokens; at least one blank must separate adjacent alphanumeric tokens. The text field is considered as one or more tokens; the search is successful whenever the same sequence of tokens is found in a line of the file. In literal mode, each line of the file is considered as an arbitrary string of characters, including blanks. The search is successful whenever the sequence of characters in the text field is found in a line of the file.

If an integer <count> appears, the search for the associated text will be terminated after it has been found the specified number of times.

By default, the workfile is searched. Specification of <filename> causes searching of the specified file, which may be any file the user is allowed to read. One or more sequence ranges may be selected from any file; otherwise the whole file is searched.

A column range, if provided, indicates that only the specified part of each line is to be examined. By default, the total text field of each line is examined.

Two output options are available:

1. TEXT

    Output is directed to the terminal and lists the lines where the search succeeded, including both sequence number and text.

2. FILE filename

    Output is directed to the specified file, which must be a new file in the user's library. The output is the entire line, in line-image form.

By default, output is directed to the terminal and lists the sequence numbers of lines at which the search was successful; an asterisk flags lines with more than one find.

Examples:

FIND /FILEID/ , <DISK<   660-1000,10000-END

    Search the workfile in the sequence ranges 660 through

    1000 inclusive and 10000 through the end of the workfile,

    and locate all uses of the tokens FILEID and DISK; the

    sequence numbers of any lines containing either token are

    to be listed on the terminal.

FIND #VARIABLE# FILENAME @ 1-72   :T

    Search the complete user file FILENAME in columns 1 through

    72 inclusive, looking for the target text VARIABLE.   The

    sequence number and text of any lines containing the target

    text are to be printed on the terminal.

FIND LIT 10 #IT#, #END# A:F AB

    Search the user file A for the first 10 occurrences of

    the literal target text IT and all occurrences of the

    token target text END.   The complete lines containing any

    of the targets are to be entered in the new user file AB.

6.3.2 REPLACE

The REPLACE command scans line-by-line through the workfile or selected portions, replacing certain target text with new text. Several options are available to treat possible line overflow due to addition of characters by the REPLACE command. Output modes may be specified to direct results to the terminal or a new file.

Syntax:
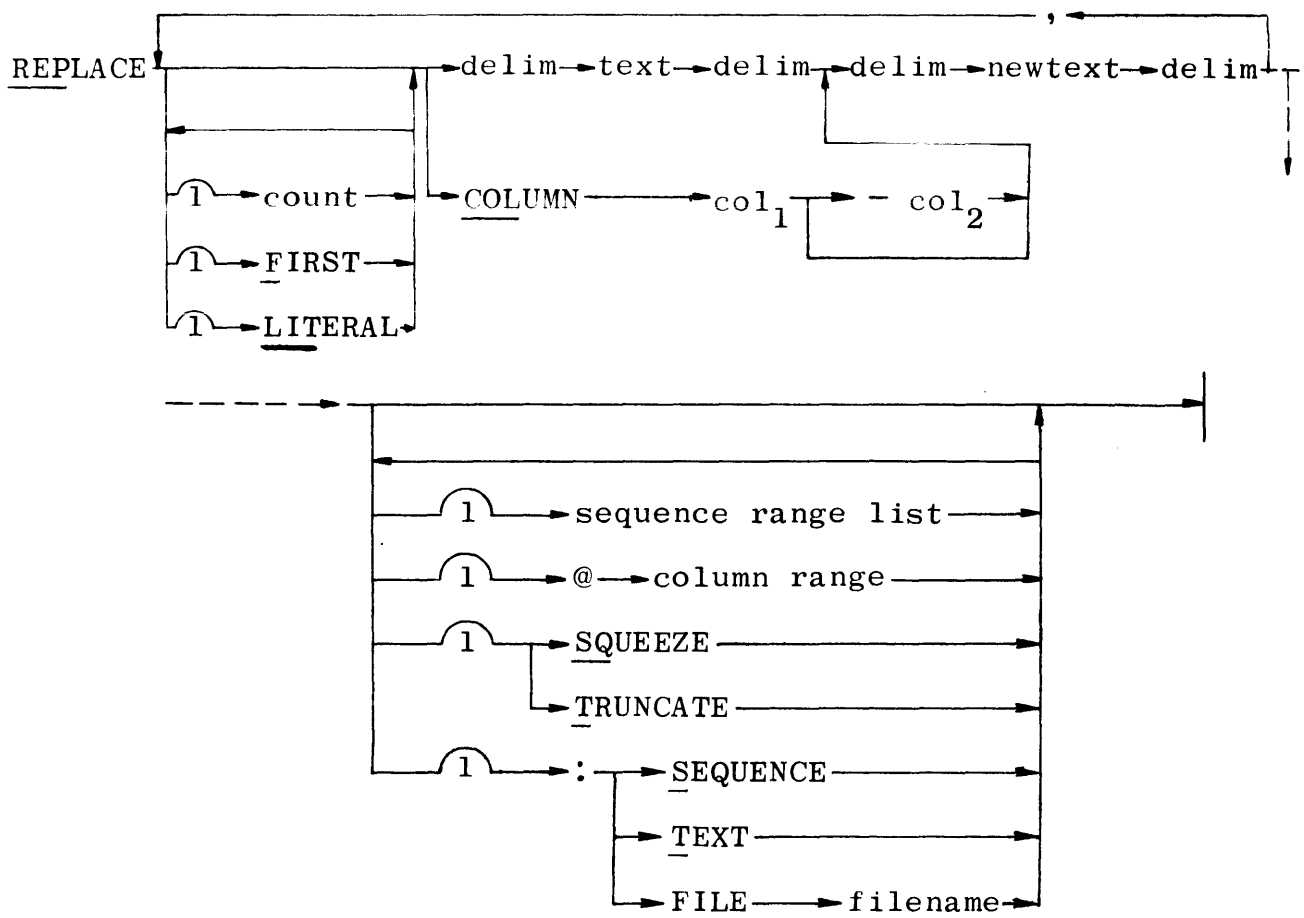
Semantics:

One or more replacements may be specified in one REPLACE; each is
defined by a target and a substitution. The target specification
may contain a text field or column numbers; the substitution speci-
fication always contains a newtext field, which may be empty.
The <text> and <newtext> fields are bracketed by delimiters,
<delim>, which may be any delimiter except percent and hyphen.
The <text> or <newtext> may contain any characters except the
delimiter. Different fields may be bracketed by different de-
limiters.

Target Specifications:

The form <COLUMN col_1> causes the <newtext> to be inserted at
the column specified. The form <COLUMN col_1 - col_2> means that
the characters in columns col_1 through col_2 are to be replaced by
<newtext>. If <text> form is used, <text> must be present, and
it specifies a target to be sought.

For purposes of the search, a text field may contain either tokens
or a literal string. The default mode is token; literal mode is
invoked by the keyword LITERAL, and applies for that text only.
In token mode, each line in the file is considered to be a se-
quence of tokens, where a token is any group of adjacent alpha-
numeric characters (digits and letters) or any non-alphanumeric
non-blank graphic character. Any number of blanks may separate
tokens; at least one blank must separate adjacent alphanumeric

tokens. The text field is considered as one or more tokens; the search is successful whenever the same sequence of tokens is found in a line of the file. In literal mode, each line of the file is considered as an arbitrary string of characters, including blanks. The search is successful whenever the sequence of characters in the text field is found in a line of the file.

If the keyword FIRST appears, only the first appearance of the text on any line is sought and replaced. If an integer <count> appears, the replacement will be terminated after it has occurred the specified number of times.

One or more sequence ranges may be selected from the workfile; otherwise the whole workfile is searched.

The replacement may be restricted to affecting a portion of the line only by specifying a column range.

Substitution Specifications:
Whenever a target is found, the <newtext> (which may be empty) is substituted. Adjustment for different length <text> and <newtext> is made by shifting the right-hand end of the line (or column range) to the left or right, deleting or adding terminal blanks. A line overflow error is detected whenever the adjustment would shift non-blank characters off the end of the line (or column range).

Two options are available for the treatment of overflow errors:

1. SQUEEZE

    If possible, strings of multiple blanks are shortened to
    make room for the substitution. If insufficient squeezing is
    possible, the line is left unsqueezed and the substitution is
    skipped.

2. TRUNCATE

    The characters pushed off the right-hand end of the line are
    lost.

    By default, the replacement is not performed; the text is left
    in the line without substitution.

    The user is notified at his terminal of all line overflows and
    their disposition.

Output Specifications:

Three output options are available:

1. SEQUENCE

    Output is directed to the terminal and lists the sequence
    numbers of lines where replacement occurred; an asterisk
    flags lines with more than one replacement.

2. TEXT

   Output is directed to the terminal and lists the modified

   lines, including both sequence number and newtext.

3. FILE filename

   Output is directed to the specified file, which must be a new

   file in the user's library. Each modified line is output, in

   line-image form.

   By default, the REPLACE statement produces no output.

   Note that the FIX command, which has substantial functional

   overlap with REPLACE, is a better choice for most single-line

   alterations.

Examples:

   REP   .AB..ABB.

      Search the entire workfile for the tokens "AB" and

      replace them with the newtext "ABB".

   REPLACE COL 1-5 // 130-180

      Replace the contents of columns 1 through 5 in lines 130

      to 180 inclusive with empty newtext. This results in a

      left shift of 5 columns with the contents of columns 1-5

      being lost.

   REP 10 F * I1 * * I1 (INDEX,ARRAY)* :T

      Replace the first occurrence of I1 of every line by

      I1 (INDEX,ARRAY); a maximum of 10 replacements is speci-

      fied. Print the sequence number and text of the lines for

      which replacement occurs. If an overflow occurs, do not

      perform the replacement.

6.4   INPUT/OUTPUT COMMANDS

6.4.1   LIST

The LIST command displays the contents of the workfile or some other file to the user at his terminal. Options are available for selecting specific lines and columns and for presenting the display in several formats.

Syntax:
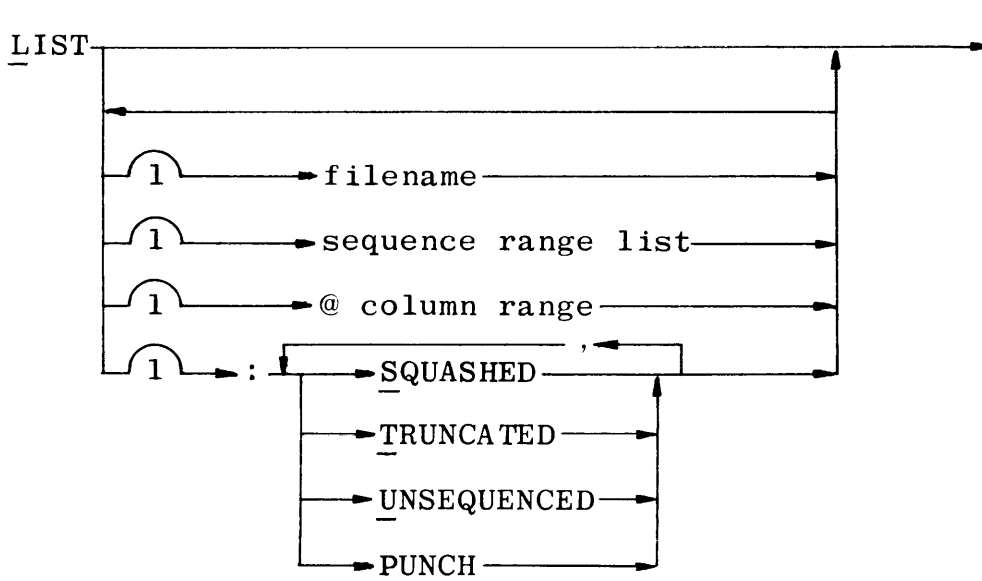
Semantics:

By default, the contents of the workfile are listed; if a file-name is specified, it is listed if the user has access privileges.

One or more sequence ranges may be selected; otherwise, the whole file is listed.

A column range, if provided, indicates that only the specified part of each line is to be listed.

Several output format options are available. If none is speci-
fied, the default format is used: the sequence number of the
line appears (pseudo sequence numbers for type DATA files) with
leading zeros omitted, followed by a blank and the text of the
line. Lines too long for the terminal are split. The options
and their effects are:

1. SQUASHED

    Any group of multiple blanks is reduced to a single blank.

2. TRUNCATED

    A line too long for the terminal is truncated to one terminal
    line.

3. UNSEQUENCED

    The lines are listed without sequence number.

4. PUNCH

    This option outputs the line to the paper tape punch. The
    user must turn on the paper tape punch if it is not in AUTO-
    START. The system sends an X-ON character followed by 10 rub-
    outs, the name of the <filename>, or the workfile by default,
    followed by 40 additional rubouts. Each line of data is
    ended with a carriage return, a line feed, DC_1 character, and
    a rubout. An X-OFF is punched at the end of the tape. The
    tape can be read back to the system using the TAPE command by
    positioning it in the group of 40 rubouts. Note that the
    sequence numbers irclude leading zeros.

Examples:

    L

        List the entire contents of the workfile.

    LIST (OTHERUSER) A/B   0-10000

        List lines 0 through 10000 of the file A/B which is

        owned by user OTHERUSER.

    LIST   80000000-END @ 1-40 "A-B"

        List from file "A-B", the sequence range 80000000

        through end of file. Only columns 1 through 40 of each

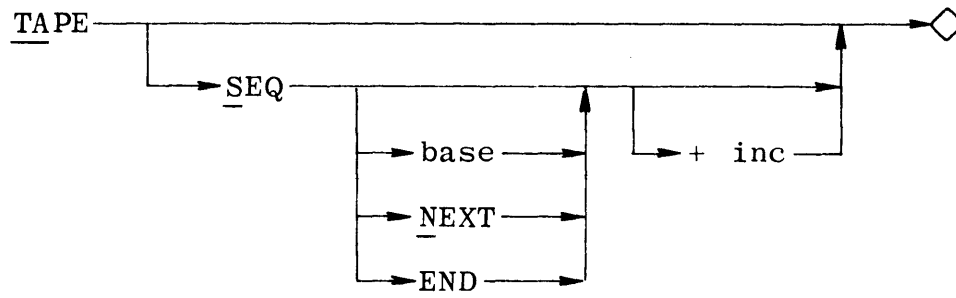   line are to be listed.

    L :P

        Output the contents of the workfile to the paper tape

        punch.

6.4.2   TAPE

The TAPE command initiates reading input from paper tape.

Syntax:

```
TAPE ─────────────────────────────────────────────────◇
      └──►SEQ──┬──────────────►──┬─────────────►──┘
              ├──►base──────►──┘  └──►+ inc──┘
              ├──►NEXT──────►──┤
              └──►END ───────►──┘
```

Semantics:

In response to the TAPE command, CANDE responds with a "#OK"
and sends an X-ON character which starts the tape reader if it is
set to AUTO-START. If the tape reader is not set to AUTO-START,
the user must manually start the reader after the #OK message.
The tape input mode is terminated whenever the reader is stopped,
either manually during input or automatically at the end of the
tape.

Renumbering of the input records is provided by the SEQ option.
This command is presented in Section 6.4.3; the semantics and de-
faults are unchanged when used in conjunction with the TAPE command.
Note that the input records are treated as type DATA records in
this circumstance; all information on the tape, including any se-
quence numbers, is considered to be data, and new (additional)
sequence numbers are supplied during input.

When resequencing of the input data is not requested, the data is treated the same way as ordinary input. Each line must have a sequence number, but the lines may be out of order. Corrections, in the form of re-entered lines may be included.

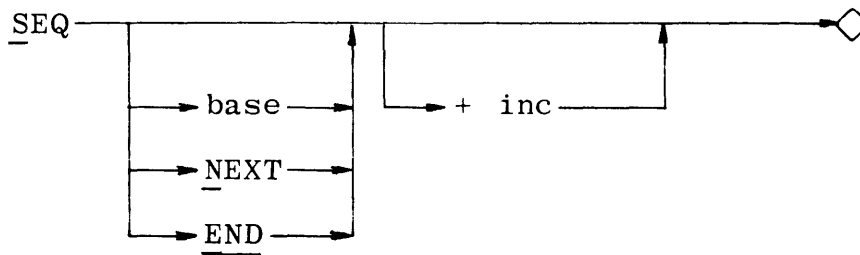Examples:

```
TAPE
TAPE SEQ END + 100
TA S 10 + 10
```

6.5   EDITING MODE COMMANDS

6.5.1   SEQ

The SEQ command invokes automatic sequence mode, causing the
system to provide the sequence number for each new line. With
the exception of @ (MARGIN), and ?(control character), no commands
are recognized in automatic sequence mode.

Syntax:

```
SEQ ─────┬──────────────┬──────────────────────◇
         │              │            ┌───────┐
         ├─► base ──────┤  └─► + inc ─┘
         ├─► NEXT ──────┤
         └─► END ───────┘
```

Semantics:

An initial value is assigned as the sequence number of the next
line to be entered; the value is incremented for each subsequent
line. The initial value may be specified in three ways:

1.   An integer <base> specifies the initial value directly.

2.   NEXT sets the initial value to the next sequence number that
     would have resulted from the most recent SEQ, MOVE, INSERT,
     or RESEQ command. If no such command has appeared since MAKE,
     GET or DELETE ALL, the default value 100 is used.

3. END sets the initial value to the largest sequence number in the workfile, plus the specified (or default) increment. If the workfile is empty, the initial value is the increment.

If no specification appears, END is used if the workfile is not empty, else 100 is used by default.

The increment applied to successive sequence numbers may be specified explicitly by an integer, <inc>. If none appears, the increment from the most recent SEQ, MOVE, INSERT or RESEQ command is used. If no such command has appeared since MAKE, GET or DELETE ALL, the default value 100 is used.

In automatic sequence mode on a teletype or analogous device, the datacom processor types the new sequence number at the beginning of each line; the user then enters the text desired for that line. On devices with multi-line input capability, a new sequence number is displayed following each transmission; the sequence number is incremented but not displayed for subsequent lines within a transmission block.

Sequence mode is terminated by entering end-of-text immediately following the sequence number; CANDE sends a "#" character to note return to normal mode.

Normal CANDE commands are not recognized in sequence mode, since they are indistinguishable from text entries. The @ form of the

MARGIN command is recognized if it is the first character in the input line. A line may begin with @ if it is entered twice as @ @.

If the line begins with the control character (?), the input is processed as a control command.

Although the sequence number is generated by the datacom processor, the line is otherwise treated as a single-line entry: if the new sequence number matches one in the workfile, the older line is replaced.

Examples:

S 10 + 10

Enter automatic sequence mode using an initial and increment value of 10.

S END + 1000

Enter automatic sequence mode using an initial value of the largest sequence number in the workfile plus the current increment of 1000.

6.5.2   MARGIN/@

The MARGIN command controls the entry of text at the left margin of a line by inserting a specified number of blanks ahead of the data being entered.   A margin specification may be in absolute column numbers or relative to the previous margin; it may be effective for all subsequent lines entered at the terminal, or for a single line.   The @ form must be used in automatic sequence mode; either form may be used in normal mode.

Syntax:
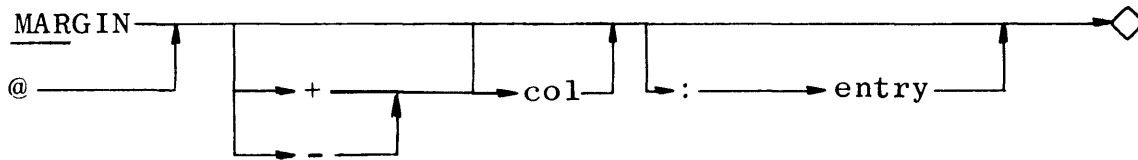
Semantics:

If an unsigned integer <col> appears, the margin is set to that column, relative to column one at the beginning of the line.   The value of <col> must fall within the text field of the line.

If a signed integer <col> appears, the margin is moved left (-) or right (+) by the number of columns specified; the resulting value must fall within the text field of the line.

If no <col> appears, the margin is set to its default position at the first column of the text field.

If no colon and entry appears, the new margin specification applies to all subsequent entries from the terminal, until a new margin specification is entered or a GET, MAKE, or TYPE is performed. If a colon appears, the new margin specification applies only to the single-line entry which follows.

In automatic sequence mode, the @ must be entered as the first character (after the system-supplied sequence number). If no colon and entry appears, the margin specification for subsequent entries is changed, and the sequence number is repeated for the next line. If a colon appears, the subsequent single-line entry is made at the sequence number provided; the user enters only the text, as in any other entry in automatic sequence mode.

Line overflow may occur as a result of indentation. The user is notified at his terminal, and the resulting line is truncated.

Examples:

    MAR 20 : 4700A:=B;

        Enter a line with A:=B beginning at column 20 with se-
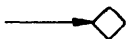   quence number 4700.

    @ + 5

        Change the indentation position to the current starting
        position plus 5 columns and retain this indentation posi-
        tion for all following input.

6.6   ENVIRONMENT COMMANDS

6.6.1   BYE

The BYE command terminates the user's current session.

Syntax:

BYE ──────◆──◇

Semantics:

A CANDE session is terminated by the BYE command; switched lines are disconnected. If the workfile has not been saved or removed, the user is notified of this condition. The workfile may then be removed or saved, and the BYE command re-entered.

System resource usage information is provided for reference.

Example:

    BYE

    # OFF AT <time> <date>

    # ET = <x> PT = <x> IO = <x>

where <x> is a time indication.

6.6.2   CHARGE

The CHARGE command is used to specify a charge code at the
start of a user's session or to change the charge code during
the session.

Syntax:

   CHARGE ─────────▶ charge code ───────────────────▶|

Semantics:

The new charge code is an arbitrary string which may not exceed
17 characters.   A new charge code number may be entered as desired,
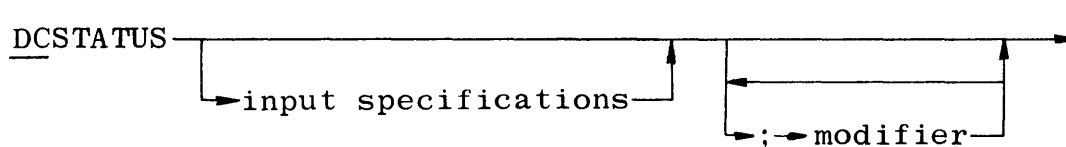thus providing charge breakdown at any point during the CANDE
session.

Examples:

        CHARGE   340-20-7700

        CHAR   J111/LSP

6.6.3   DCSTATUS

The DCSTATUS command causes execution of SYSTEM/DCSTATUS which provides general information regarding the status of the datacom network.

Syntax:

```
DCSTATUS ─┬──────────────────────────────┬─┬────────────────────┬─►│
          └──►input specifications──┘        │◄─────────────┐     │
                                             └──►;──►modifier─┘
```

The input specifications consist of the string of standard commands for SYSTEM/DCSTATUS.   Reference is made to the B 6700

System Documentation of SYSTEM/DCSTATUS to describe this information.   If the input specifications are omitted, the status of the user's terminal is provided.

Modifiers may be specified to alter the normal operation of the program SYSTEM/DCSTATUS.   For reference, the single output file is titled "LINE".
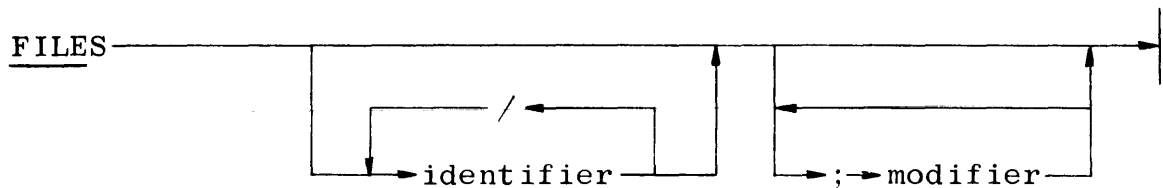
Example:

DCSTATUS

Cause execution of SYSTEM/DCSTATUS to obtain the datacom status of the user's terminal.

6.6.4   FILES

The FILES command results in the execution of SYSTEM/LISTFILES
which provides a list of the name and type of files in the user's
library.

Syntax:

Semantics:

If a filename is not given, all files in the user's library are
listed. A filename may be provided at the directory or datafile
level; if a directory is specified, all files referenced by that
directory are processed.

Modifiers may be specified if desired. For reference, the single
output file is titled "LINE".

Example:

         FILES   A

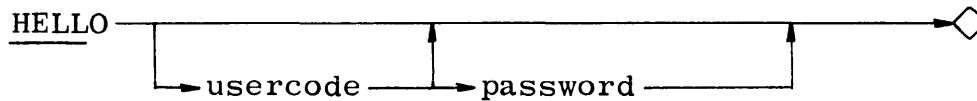   # RUNNING

             A : DIRECTORY

             . B : ALGOLSOURCE

             . C : BASICSOURCE

6.6.5 HELLO

The HELLO command initiates a new user session without disconnecting the station.

Syntax:

```
HELLO ──┬─────────────────────────────────────────────┬──▶◇
        └─▶usercode ──┬─▶password ──┐
                      └─────────────┘
```

Semantics:

The current user's session is terminated by implicitly performing the BYE command and then invoking the normal log-in process to initiate the new user's session. As with the BYE command, this command is in error if the current user's workfile has not been saved or removed. Switched lines will not be disconnected as part of the implicit BYE.

Usercode and password may be included to complete the log-in procedure or this information will be requested in the normal log-in manner.

If the station is not currently logged-in, then only the normal log-in procedure is invoked.

Examples:

HELLO

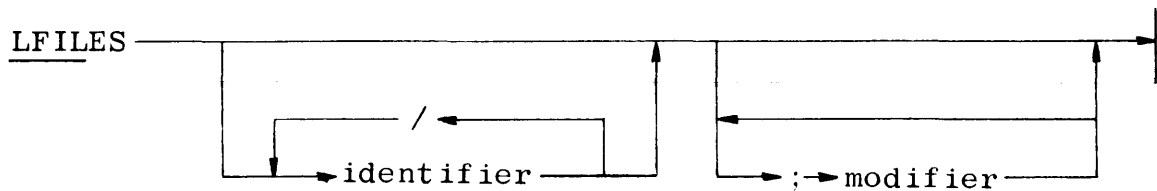Initiate the log-in procedure for a new user; any existing user is logged-off.

HELLO A/B

Log-off any existing user and log-in user A/B.

6.6.6   LFILES

The LFILES command results in the execution of SYSTEM/LISTFILES which provides a list of the names and B 6700 file attributes of files in the user's library.

Syntax:

Semantics:

If a filename is not given, all files in the user library are listed. A filename may be provided at the directory or datafile level; if a directory is specified, all files referenced by that directory are processed.

Modifiers may be specified if desired. For reference, the single output file is titled "LINE". The data file attributes listed are:

FILEKIND	LASTRECORD
FILETYPE	AREAS
INTMODE	AREASIZE
UNITS	DATE
MAXRECSIZE	LASTACCESSDATE
MINRECSIZE	SAVEFACTOR
BLOCKSIZE	

Example:

LFILES   CANDE   (a directory file)

Cause execution of SYSTEM/LISTFILES to obtain the attributes of all files in the CANDE directory.

6.6.7   LOG

The LOG command results in the execution of SYSTEM/LOGOUT which provides a log of job performance.

Syntax:

LOG ———————► input specifications ———————►|

Semantics:

The input specifications consist of the string of standard commands for SYSTEM/LOGOUT. Reference is made to the B 6700 System Documentation of SYSTEM/LOGOUT to describe this information.

Examples:

    LOG MIX 2437

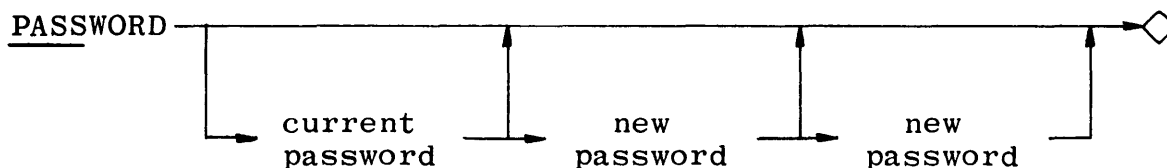   Obtain a log of all events for mix 2437.

    LOG JOB A/B

        Obtain a log of all events for the job titled A/B.

6.6.8   PASSWORD

The PASSWORD command allows a user to change his password.

Syntax:

Semantics:

To change passwords, a user must enter his current password, the new password, and for verification purposes, repeat the new password. All or part of the required information may be entered with the PASSWORD command; any information not provided initially will be requested by CANDE.
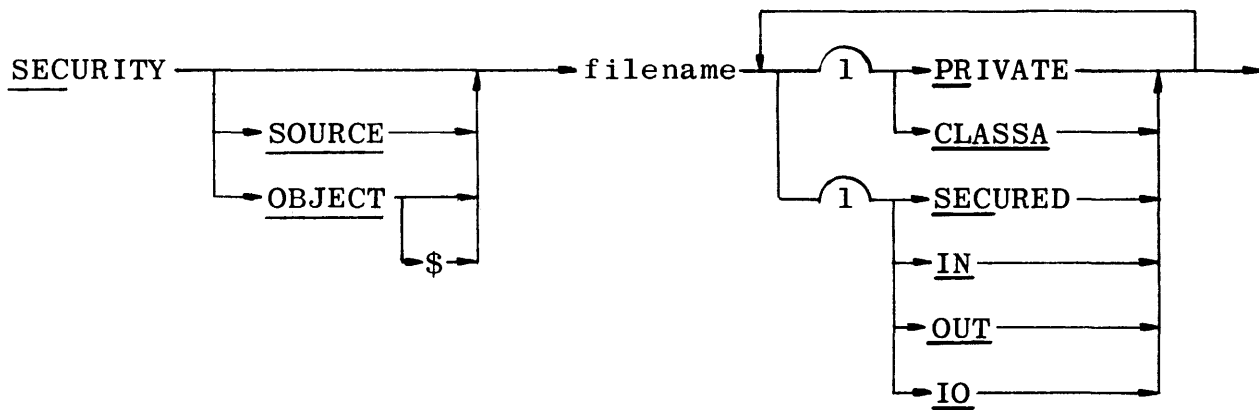
Example:

        PASS   OLDPASS   NEWPASS   NEWPASS

            The user current password, OLDPASS, is replaced with the new password, NEWPASS.

6.6.9   SECURITY

The SECURITY command allows specifying the security attributes of a user's file.

Syntax:

Semantics:

The source file only, or the object file only, or, by default, both the source and object files named <filename> may be specified. An object file not in the user's OBJECT directory is specified as OBJECT $<filename>.   The file must be in the user's library;   the B 6700 security prevents a user from changing the security attributes of a file which he does not own.

The security options which may be specified are identical to the B 6700 system file attributes which pertain to security, SECURITYTYPE and SECURITYUSE.   Reference is made to the formal system documentation to explain these attributes in detail.

Examples:

SECURITY   SOURCE   "A-B"   CLASSA , IO

> Set the security attributes of the source file "A-B" to
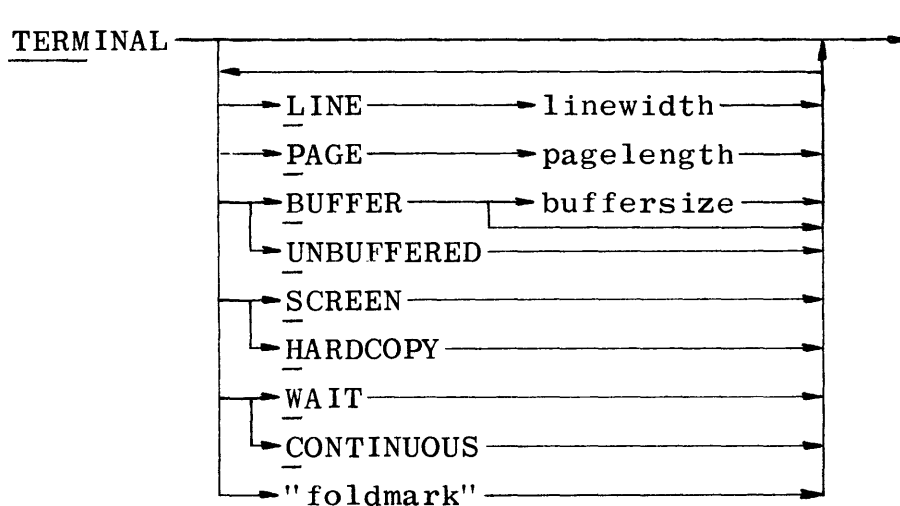>
> CLASSA and input/output access allowed.

SEC   OBJECT $ GOOD/CODE   SEC

> Set the security attribute SECURITYUSE to SECURED for
>
> the user object program named GOOD/CODE which is not in
>
> the user OBJECT directory.

6.6.10 TERMINAL

The TERMINAL command specifies attributes of the user's terminal, which determine the mode of transmission of data by CANDE. (Most of the specifications are significant only during the execution of CANDE output commands, such as LIST or FIND.)

Syntax:

```
TERMINAL ─────────────────────────────────────────────────┤

              ┌──► LINE ─────────────► linewidth ──────┐
              ├──► PAGE ─────────────► pagelength ──────┤
              ├──► BUFFER ──────┬────► buffersize ──────┤
              └──► UNBUFFERED ──┴───────────────────────┤
              ┌──► SCREEN ──────────────────────────────┤
              └──► HARDCOPY ────────────────────────────┤
              ┌──► WAIT ────────────────────────────────┤
              └──► CONTINUOUS ──────────────────────────┤
              └──► "foldmark" ──────────────────────────┘
```

Semantics:

LINE specifies the width of the terminal line in characters, as the integer linewidth.

PAGE specifies the number of lines per page or screen, as the integer pagelength.

BUFFER indicates that the terminal is buffered and optionally specifies the number of characters the terminal can receive and display at once, as the integer buffersize. If buffersize is absent, the previous or default specification is used. A buffersize of zero means unlimited capacity, e.g., teletype.

UNBUFFERED indicates that the terminal is unbuffered. (The buffer-size is not lost, but becomes ignored.)

SCREEN indicates that the terminal has a volatile display, whereas HARDCOPY indicates that the terminal produces permanent output.

WAIT causes CANDE to wait after sending each page, until the user signals that he wants the next; CONTINUOUS causes CANDE to send continuously until the end of the output. The user's signal is a null (empty) line (or any non-control input; the text, if any, is ignored).

When an output line exceeds the terminal width, CANDE either truncates the line or "folds" it by printing it on several lines. A special foldmark character is printed as the last character of a truncated or interrupted line, and as the first character of a continuation line. The user may specify the foldmark character by including it in quotes in the TERMINAL specification command. (If the quotes enclose more than one character, the first is used; any character including blank and quote may be specified.)

A TERMINAL specification with no parameters causes the current parameter values to be displayed.

Defaults:

Default values of linewidth, pagelength, buffersize, and SCREEN/HARDCOPY are specified in the NDL description of the terminal and station; the input-buffer declaration in NDL is used by CANDE as

a default output buffersize. If pagelength exceeds one and buffer-size exceeds linewidth, and the device has SCREEN set, CANDE assumes the station is buffered and will WAIT for page-turn.

Pragmatics:

For teleprinting devices, there is seldom value in PAGE, BUFFER or WAIT specifications. The user may use LINE to take advantage of terminals that are wider than their default specifications (72 characters for teletypes), and he may wish to change the foldmark.

PAGE, BUFFER and WAIT specifications are primarily intended for CRT display devices. Linewidth and pagelength become screen width and depth, respectively; buffersize specifies the maximum number of characters the terminal can display at once (which may be less than the product of linewidth times pagelength).

For a buffered terminal, CANDE sends an entire page in one or a few transmissions; lines are sent one at a time to an unbuffered terminal. Since one character per line is generally needed for new-line action and one line per page is generally needed for end-of-page action, CANDE puts useful graphic output in linewidth-1 positions on pagelength-1 lines for buffered terminals. CANDE sets toggle #3 in the new-page message to permit NDL action such as screen clear to be programmed.

Though they are most useful together, BUFFER and WAIT attributes are independent and may be used in any combination. When BUFFER and CONTINUOUS are used together, CANDE sends page after page

without stopping. (Depending upon the terminal and line discipline: this "continuous" output may in fact be interrupted by user or terminal action involving "local" or "keyboard" versus "receive" states.) If UNBUFFERED and WAIT are selected together. CANDE sends pagelength lines one at a time. and then waits for an input.

The SCREEN attribute is used by CANDE to suppress creation of blackout spaces for password entry.

CANDE forces UNBUFFERED state if pagelength is one or less or if buffersize is smaller than linewidth: CANDE forces CONTINOUS state if pagelength is one or less.

Users must exercise care and discretion in specifying TERMINAL attributes. since some combinations are unuseable with some terminals. NDL specifications, or line disciplines. Misuse may cause errors which result in not-ready or other dire conditions at the terminal. Note that specification of buffersize to CANDE for output does not affect the NDL-declared limitation on input buffer capacity.

The buffersize value limits the total number of characters sent by CANDE, including a CR and LF between lines. (The datacom processor may send additional characters to some devices.) The limit does not necessarily correspond exactly to the terminal capacity, since some devices do not treat all characters alike. For example, the B9353 (BIDS) terminal uses two buffer positions to store certain graphics: the buffersize must be set smaller in proportion to the relative occurrence of these characters.

Examples:

TERMINAL   HARD   L 120

The terminal is specified to be a hardcopy device with a
line width of 120 characters.

TERM   B   SCR   "/"   P 8

The terminal is specified to be a buffered, screened
device with 8 lines per page capability. A "/" character
is to be used to denote the foldmark for output lines
which exceed the terminal width.

7.   CONTROL COMMANDS

Control commands constitute a special class of commands to pro-
vide immediate and direct control of the user's CANDE environment.
These commands are not queued-up for processing behind normal
CANDE commands, but are executed immediately upon being entered.
Thus it is possible to realize the result of a control command
before previously entered input.   These commands may be used,
where appropriate, before log-in is completed.

Most control commands are preceded by a control character.   This
is an installation-determined character specified in the Network
Definition Language.   The following discussion will use the char-
acter defined in the standard NDL, i.e., the question mark, "?".

7.1   BREAK

The BREAK command terminates the current output to the terminal.

Syntax:

> ?BRK ──◆◇
>
> <break key>

Semantics:

All output queued for the terminal is "flushed" when the BREAK

command is input. If the output is from an object program, then

a break-on-output condition occurs for all object programs which are

currently using the terminal as an output file.

7.2   DS

The DS command will cause termination of any program scheduled or initiated by a user, or any CANDE editing or output activity.

Syntax:

?DS ———◆◇

Semantics:

Normal DS'ed termination occurs when a user DS's his program initiated through the terminal. Following this action, the terminal is available for further input commands.

A CANDE command may also be terminated using the DS command.

7.3   DENY /END

The DENY and END commands provide for terminating the usage of a
terminal as a remote file.

Syntax:

　　?DENY ────◆◇

　　?END ────◆◇

Semantics:

The DENY command will cause an end-of-file indication on the
next input or output request by any object programs using the
terminal as a remote file; the END command will similarly provide
an end-of-file condition for input request only. If multiple
object programs have opened the terminal as a remote file, all
will receive the specified end-of-file action.

7.4   MCS

The MCS command provides a user the ability to transfer to a MCS other than SYSTEM/CANDE.

Syntax:

?MCS ——→ message control system name ——→◇
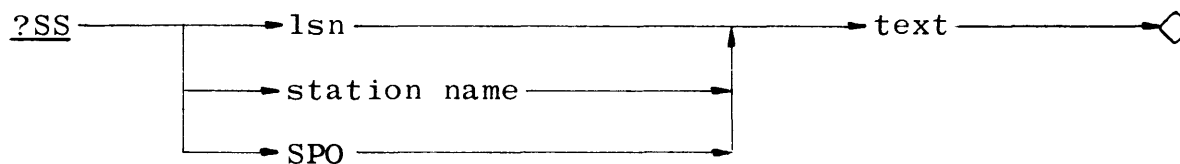
Semantics:

The specified <message control system name> may be any standard message control system (MCS) program which is either active or inactive, or a non-standard MCS which is active. The complete MCS name (TITLE) must be specified, i.e., SYSTEM/MCSII.

7.5   SS

The SS command provides the ability to send a message to another
station.
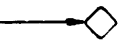
Syntax:

Semantics:

The receiving station may be referenced by <lsn> or station name.
SPO is the designation of the CANDE control terminal.

The <text> may consist of any character string up to the maximum
of the sending or receiving terminal's line size.

7.6   STATUS

The STATUS command provides the current status of a user's
program.

Syntax:

?STATUS ———◇

Semantics:

If the user has a running program, the elapsed time, processor
time, and input/output time are indicated.

The information is presented in the form:

ET <x> : CP <x> : IO <x>

where <x> is the time in hours, minutes and seconds.

7.7   WHERE

The WHERE command provides the station name and <lsn> of a user.

Syntax:

?WHERE ──────➤ usercode ─────────➤◇

Semantics:

If the specified usercode is currently logged-in, then his station
name and <lsn> are provided; otherwise the user is so notified.

Examples:

        ?WHERE   "A-B"

                # "A-B" ON M331 (5)

                Usercode "A-B" is currently logged-in on a station named

                M331 and lsn 5.

        ?   WHERE   BAKER

                # BAKER   NOT   ON

7.8   WRU

The WRU (who are you) command provides identification of the
operating datacom system and the user's terminal.

Syntax:

?WRU ─────◆
<wru key>

Semantics:

The information provided includes the associated MCS name (CANDE),
the user's station name and <lsn>.