A User's Guide

to

H O P E

A Human Oriented Program Editor

Management:

Henry Ledgard

Design:

Andrew Singer
Jon Hueras

Implementation:

Daryl Winters

March 1978

This work was supported by the National Science Foundation.

This is a COINS Technical Report numbered 78-05, University of Massachusetts Amherst, 01003.

PREFACE

The Human Oriented Program Editor (HOPE) is a program for a time-sharing computer system that modifies program text (or other text) using a small set of user requests.

This prototype editing system was designed in the belief that most existing editors have such a limited set of requests that they are ineffective for general purpose editing, or such a large and powerful set of requests that the requests cannot be easily handled by the average user. HOPE was designed to be very simple and moderately powerful. This effort is part of an on-going project to promote better human engineering of computer systems.

We would like to express our appreciation to Dr. Conrad Wogrin and the University Computing Center for their support and interest in this project.

We would like to thank Joe Kasprzyk and Jeffrey Horn for their work on an earlier version of this system, and Neal Ogden for his suggestion of an appropriate compound term for the acronym HOPE.

Finally, we are grateful to the U.S. Army Research Office, Durham, and the National Science Foundation for their support of this effort.

1. Editing Concepts

Text editing is a process of creating, maintaining, and updating files of text such as programs, data files, chain letters, and so forth. The editing requests make it possible to insert, delete or substitute text, and to change the layout and spacing of text.

Files

A file is a named collection of information that the editor maintains for you. Most files will consist of text and many of these will be programs. When a file is created, you must give it a name. From then on, both you and the editor refer to the file by that name.

The Current File

In order to make changes to a file, the editor must be told what file will be modified. The editor always assumes that requests are performed on a "current file." Initially, there is no current file. Once the editor is notified which file will be the current file, all future requests will be performed on that file. The current file can be changed by you at any time.

The Current Line

In making editing requests, you must always have some means of specifying what it is you want changed. The editor always assumes that a request is made relative to a "current line." Initially, the current line is the first line of the file. Thereafter, the last line you see printed is the current line.

Prompting and HOPE's Messages

The editor prints two prompting characters to indicate that it is ready for a response. The characters indicate what type of response is expected from you.

Prompt	Response Expected From You
--	A Request
++	New-Text Input
//	A Security Check Answer

Security checks are used to prevent you from accidently overwriting an existing file or from leaving the editor without saving your file.

In addition, to further aid in understanding the interaction between you and the editor two characters will prefix all lines that the editor displays (ß indicates a blank).

Prefix	Editor Display To You
**	A Message
ßß	A Listing of Text

Grammatical Notation

Every language has a grammar, and the editor's request language is no exception. In the descriptions that follow, we employ the following notation to describe the grammatical form of each request:

[1] Keywords

Words shown in upper case are keywords. Some keywords introduce a request, other keywords qualify a request. Any keyword (except PRESERVE and DESTROY) may be abbreviated by its first letter. For example, the request:

```
ASSUME FILE IS XYZ
```

may be entered:

```
ASSUME FILE I XYZ
ASSUME F I XYZ
A F I XYZ
```

If not abbreviated, a keyword must be spelled out correctly.

[2] Objects

Words shown in lower case are names for the objects that you supply to make a request specific. These words may denote the name of a file or a piece of text.

[3] Specifying Text:

References to pieces of text require brackets to delimit the text. This notation has the form:

```
=text=
```

The given text is any actual sequence of characters. The symbol "=" represents a special delimiting character. This character may not be a letter, digit, semi-colon, or space. (See Appendix A for a list of acceptable special characters.) Since this character indicates both the beginning and ending of the desired text, it must be a character which does not appear in the text itself. If a delimiting character is the last character on a line, it may be omitted.

[4] Alternatives

Keywords or objects that are grouped together and separated by slashes "/" denote mutually exclusive alternates. For example:

```
HERE/FIRST
```

means that either the keyword HERE or the keyword FIRST may be specified, but not both.

[5] Options

Keywords and objects that are in parentheses represent parts of a request whose use is optional. For example:

```
(n/ALL)
```

means that either n (a number), the keyword " ALL ", or neither may be used.

[6] Blank Spaces

Between keywords, their first letter abbreviations, or other objects, spaces are optional. For example:

```
FORWARD 5
```

may be written as

```
FORWARD5
F 5
F5
```

[7] Multiple Requests

More than one request can be entered on a single line by separating the requests with a semi-colon ";". For example, by entering the line

```
FORWARD 5;   DELETE
```

or in the abbreviated form

```
F5; D
```

both the FORWARD request and the DELETE request will be performed before the editor asks for more requests. If any problem or error is detected in an initial request, the remaining requests will not be performed.

2. Request Summary

Editing Requests

```
FORWARD     (n/ALL)     (=text=)
BACKWARD    (n/ALL)     (=text=)
LIST        (n/ALL)     (=text=)
DELETE      (n/ALL)     (=text=)
TRANSFER    (n/ALL)     (=text=)
COPY        (n/ALL)     (=text=)     INTO  file-name
MAKE        (n/ALL)     =text=new-text=
INSERT      (=text=  /  FILE  file-name)   (AFTER/BEFORE/OVER)
```

Assume Requests

```
ASSUME      FILE      IS   file-name
            MARGIN    IS   special-symbol/NULL
            ELLIPSIS  IS   special-symbol/NULL
            JOKER     IS   special-symbol/NULL
            TOP       IS   HERE/FIRST
            BOTTOM    IS   HERE/LAST
            ALPHABET  IS   NORMAL/ASCII
```

General Requests

```
SHOW        (assumption/CATALOG)
PRESERVE    (file-name)
DESTROY     (file-name)
GRIPE
UNDO
QUIT
```

3. Editing Requests

Moving Forward:

To move the current line forward, you say

FORWARD (n/ALL) (=text=)

The variations of the FORWARD request are as follows:

[1] FORWARD

The editor moves the current line forward one line.

[2] FORWARD n

The editor moves the current line forward n (where n is a number) lines.

[3] FORWARD ALL

The editor moves the current line forward to the last line in the file.

[4] FORWARD =text=

The editor moves the current line forward to the next line containing an occurrence of the specified text.

[5] FORWARD n =text=

The editor moves the current line forward to the n-th line containing an occurrence of the specified text.

[6] FORWARD ALL =text=

The editor moves the current line forward to the last line containing an occurrence of the specified text.

Moving Backward:

To move the current line backward say

BACKWARD (n/ALL) (=text=)

The BACKWARD request is exactly the reverse of the FORWARD request. Note that BACKWARD ALL takes you to the first line in the file.

Displaying Lines:

To display one or more lines of text just say

LIST (n/ALL) (=text=)

The variations on the LIST request are similar to the FORWARD request. The essential difference is that the search for the specified text will include the current line (FORWARD and BACKWARD do not). The variations of the LIST request are as follows:

[1] LIST

Only the current line is displayed on the terminal.

[2] LIST n/ALL

The editor displays the next n (for ALL) lines including the current line.

[3] LIST n/ALL =text=

The next n (or ALL) lines containing the specified text are displayed.

A long listing may be terminated by using the request interrupt characters as given in Appendix 2.

Deleting Lines:

To delete one or more lines of text say

DELETE (n/ALL) (=text=)

This request is virtually identical to the LIST request with the important difference that the lines containing the specified text are not displayed but are removed from your file.

Moving Lines:

To move one or more lines of text out of your file and into another file just say

TRANSFER (n/ALL) (=text=) INTO file-name

This request removes the specified lines of text from your file and puts them into the named file. The lines that are removed will replace the previous contents of the file. Before the request is performed a security check is made to prevent an accidental erasure of the named file.

Duplicating Lines:

To make a copy of one or more lines of text from your file and place them in another file say

COPY (n/ALL) (=text=) INTO file-name

This request is identical to the TRANSFER request except that the lines are not removed from your file. Instead, copies of the specified lines are placed into the named file.

Inserting Lines:

To insert new lines of text into the currently assumed file say

INSERT (=text= / FILE file-name) (AFTER/BEFORE/OVER)

The variations of the INSERT request are as follows:

[1] INSERT (AFTER/BEFORE/OVER)

The editor will repeatedly prompt you for lines of input from the terminal until you interrupt the editor. The lines you type will be inserted after, before, or instead of the current line.

[2] INSERT =text= (AFTER/BEFORE/OVER)

The lines specified by text will be inserted after, before, or instead of the current line.

[3] INSERT FILE file-name (AFTER/BEFORE/OVER)

The contents of the named file will be inserted after, before, or instead of the current line.

If AFTER, BEFORE, or OVER is not specified, AFTER is assumed. A combination of TRANSFER (or COPY) and INSERT requests may be used to move (or repeat) large sections of text from one place to another place in your file.

Changing Text Within A Line:

In order to change a piece of text in one or more lines say

MAKE (n/ALL) =text=new-text=

This request is different from all other requests in that it operates on text within lines rather than on whole lines themselves. The variations of the MAKE request are as follows:

[1] MAKE =text=new-text=

The editor will search for the next occurrence of text (starting with the current line) and replace that occurrence with new-text. The last line that contains new-text will become the current line.

[2] MAKE (n/ALL) =text=new-text=

The editor will replace the next n (or ALL) occurrences of the specified text with new-text. The last line which contains new-text will become the current line.

If no new text is given after the second bracket, each occurrence of text will be removed from within the individual lines.

4. Assume Requests

Assume requests affect the actions of the previously described editing requests by modifying the environment in which the requests are performed. Each request will be discussed separately.

Specifying a File:

All of the editing requests in the previous section depend on having a "currently assumed" file to edit. To specify that a new (or existing) file is to be edited simply say

```
ASSUME FILE IS file-name
```

All subsequent requests will use this file as your file.

Referring to a Line Boundary:

Sometimes it is useful to refer to text at the left or right margin of a line. To do this you must first define a special symbol to represent a margin by saying

```
ASSUME MARGIN IS special-symbol
```

For example, if "$" is your margin symbol, then

```
=$XXX=
```

refers to a piece of text "XXX" at the beginning of a line,

```
=YYY$=
```

refers to a piece of text "YYY" at the end of a line,

```
=YYY$XXX=
```

refers to a piece of text "YYY" at the end of a line followed by a piece of text "XXX" at the beginning of the next line, and

```
=$ABC$=
```

refers to a piece of text "ABC" that makes up a whole line.

A special-symbol may consist of one to three characters. The special-symbol you choose may not contain a letter, digit, semi-colon, or space. You may redefine the margin symbol at any time, or you can say

```
ASSUME MARGIN IS NULL
```

which means that no special-symbol will be interpreted as a margin.

Referring to Long Pieces of Text:

When referencing a long piece of text, it is tiresome to have to type it all out when only a few details identify it uniquely. In prose we use three dots as an ellipsis to indicate that a piece of text has been omitted. For example, we might quote the previous sentence as: "In prose, we ... indicate that a piece of text has been omitted." With the editor you may also omit pieces of text using a special symbol as an ellipsis.

If you have defined "..." as your ellipsis symbol, then

```
=XXX...YYY=
```

refers to any piece of text starting with "XXX " and ending with "YYY" and

```
=XXX...YYY...ZZZ=
```

refers to any piece of text starting with "XXX", ending with "ZZZ", and having "YYY" somewhere in between.

You define the special ellipsis symbol by saying

```
ASSUME ELLIPSIS IS special-symbol
```

The ellipsis may be redefined at any time, or you can say

```
ASSUME ELLIPSIS IS NULL
```

in which case no special symbol will be defined as the ellipsis.

Referring to Character Positions:

Sometimes when referencing existing text, it is convenient to refer to a character position rather than a specific character. For example, suppose you wanted to find misspellings of the word "PASCAL." You might want to refer to something like "P_SC_L," where the underscores "_" indicate that any single character is acceptable. You can define a special symbol which has this "wild card" meaning by saying

```
ASSUME JOKER IS SPECIAL-symbol
```

The joker may be redefined by any time, or you can say

```
ASSUME JOKER IS NULL
```

in which case no symbol will be defined as the joker.

Putting Limits on the Range of Text:

Sometimes you may wish to limit the range of editing requests. To do this say

```
ASSUME TOP IS HERE/FIRST
ASSUME BOTTOM IS HERE/LAST
```

If HERE is specified as the TOP (or BOTTOM), then the current line becomes a boundary that all subsequent editing requests may not cross. The TOP is the first text line in the file that you can reference, and BOTTOM is the last. If the TOP (or BOTTOM) is specified as FIRST (or LAST), then the boundary will be the first (or last) line in your file.

Assuming a new TOP or BOTTOM voids any previous boundary; when you assume a new file, the boundaries are set to the first and the last line of the entire file.

Using the Ellipsis and Joker in MAKE Requests

You may use the joker and ellipsis symbols in the new-text of a MAKE request. In these cases, the character or string represented by the symbol is identical to the character or string in the same position as in the text string. For example, if $ is the margin symbol and _ is the joker symbol,

```
MAKE 10 /$_ _ _/$_ _ _  /
```

adds a space after the first three characters of the next ten lines.

Changing the Character Set

Some installations may offer different character sets to be used in your files. In order to correctly reference and modify those files you will have to indicate which ALPHABET to use.

```
ASSUME ALPHABET IS ASCII
```

will allow you to generate and reference your files with the ASCII character set (upper and lower case characters). All future requests will use that character set unless you say

```
ASSUME ALPHABET IS NORMAL
```

at which point you will be able to use the NORMAL character set (upper case characters used in program text). When you start up the editor, the character set is assumed to be NORMAL.

5. General Requests

General requests are used to interact with the system in various ways. Each request will be described.

Showing the Editing Assumptions

To see the current status of any editing assumption just say

```
SHOW assumption
```

("assumption" is replaced by the name of the particular assumption that you wish to see displayed.) In addition, you may want to know what files have been preserved for you by the editor. To see a summary of all files in your catalog say

```
SHOW CATALOG
```

To see the state of all assumptions simply say

```
SHOW
```

This will show all assumptions along with a listing of your catalog.

Preserving the Current File

When editing a file, your changes will not be made permanent unless you say so. To have the current file preserved for subsequent use say

```
PRESERVE (file-name)
```

replacing any other permanent copy of that file. If you specify a file name, the current file will be preserved with that name.

Destroying a File

When you are finished with a file that was preserved for you by the editor, you may want to remove it from your catalog of files. To do this just say

```
DESTROY (file-name)
```

If you don't specify a file name, the preserved version of the current file will be removed from your catalog.

Complaints, Comments, and Suggestions

To let the people in charge of the editor know of some particular complaint or comment of yours simply say

```
GRIPE
```

The editor will repeatedly prompt you for the lines of your message until you interrupt it. Your message will be saved and at regular intervals all messages sent by you and others will be examined by the people responsible for maintaining the editor.

Undoing a Request

The effect of any editing request or editing assumption can be undone. If a request does not perform as you expected it would say

UNDO

This will undo the effects of the last valid request that was completed. The effects of TRANSFER, COPY, PRESERVE, and DESTROY on the permanent file cannot be undone. (The changes to the current file made by TRANSFER and COPY can be undone.)

Leaving the Editor

To leave the editor and return to the system monitor say

QUIT

Before allowing you to leave the editor, a security check will be made to verify that you have preserved the current file. If you do not wish to preserve the file, the changes you have made will be ignored.

APPENDIX A: SAMPLE EDITING SESSION

The following is a sample editing session. The numbers in parentheses are references to explanations of the actions of particular requests (following the editing session). Note that each request is spelled out in full with all optional parts included. (This is not necessary for correct interpretation by HOPE -- See Section 1.)

Text Editing Session

```
 BATCH1
/HOPE2

**WELCOME TO HOPE  FEBRUARY 03, 1978  1:46 PM3
--ASSUME ALPHABET IS ASCII4
--ASSUME FILE IS POEM5
**CURRENT LINE IS
  What am I?6
--LIST ALL7
  What am I?

  They chose me from my brothers: "That's the
  actual number of lines
  Nicest one," they said,
  Candle in my head;
  And they carved me out a face and put a
  Night was dark and will
  But when they lit the fuse, then I jumped!8
--BACKWARD /actual/9
  actual number of lines
--DELETE 110
  Nicest one," they said,
--FORWARD 111
  Candle in my head;
--TRANSFER 1 INTO HOLD12
**A NEW FILE WILL BE CREATED FOR YOU
  And they carved me out a face and put a
--INSERT FILE HOLD AFTER13
**INSERT FILE HOLD AFTER CURRENT LINE
  Candle in my head;
--ASSUME MARGIN IS $14
--ASSUME ELLIPSIS IS ...15
--INSERT /$/16

--INSERT17
**ENTER NEW TEXT

++And they set me on the doorstep, oh, the
++18
```

1 The BATCH subsystem is entered.

2 The user requests to start up HOPE.

3 HOPE responds with a message indicating it is ready for requests.

4 The alphabet is changed to upper and lower case characters.

5 An existing file POEM becomes the current file.

6 The current line is the first line of the file.

7 The entire file is displayed.

8 The current line is the last line displayed.

9 The current line is moved backward.

10 The current line is deleted and the current line is moved to the following line.

11 The current line is moved forward.

12 The current line is transferred out of the current file and placed in file HOLD.

13 A copy of the contents of file HOLD is inserted after the current line and the current line is moved forward to the line inserted.

14 The special symbol $ will now be interpreted as a margin symbol whenever it appears in a text string.

15 The special symbol ... will be the ellipsis symbol.

16 A blank line is inserted after the current line.

17 A sequence of lines will be inserted after the current line. The first line is requested by the prompting characters.

18 An interrupt is entered to terminate new text input.

```
**END OF NEW TEXT
  And they set me on the doorstep, Oh, the
--FORWARD 1
  Night was dark and wild
--MAKE /will/wilmars/ 19
  Night was dark and wilmars
--UNDO 20
**UNDO MAKE REQUEST COMPLETE
**CURRENT LINE is
  Night was dark and wild
--MAKE /will/wild;/
  Night was dark and wild;
--FORWARD 1
  But when they lit the fuse, then I Jumped!
--MAKE /fuse,,,Jumped!/candle, then I'Smiled!/ 21
  Smiled!
--BACKWARD ALL 22
  What am I?
--LIST ALL
  What am I?

  They chose me from my brothers: "That's the
  Nicest one," they said,
  And they carved me out a face and put a
  Candle in my head;

  And they set me on the doorstep, Oh, the
  Night was dark and wild;
  But when they lit the candle, then I
  Smiled!
--PRESERVE POEM 23
**ORIGINAL FILE POEM WILL BE LOST AFTER PRESERVE REQUEST
**SHOULD THE REQUEST BE PERFORMED ANYWAY?
//YES 24
--QUIT 25
**END OF HOPE SESSION  1:52 PM

RETURN;HOPE.
/
```

19 One string of characters in the current line is changed.

20 The effect of the previous request was not what the user intended. The request is "undone", returning the current line to what it was before the last request was performed.

21 The current line is altered and a new line is added by use of the margin character.

22 The current line is moved backward to the first line of the file.

23 The user wishes to preserve the changed made to the current file with the name POEM.

24 A security check is made because file POEM already exists. The user indicates the file can be changed.

25 The user wishes to end the editing session.

APPENDIX B: USING HOPE AT UMASS

This is your guide to the use of HOPE at the University of Massachusetts. If you are familiar with the sign-on procedures and the use of computer terminals, this appendix should be all you need to utilize HOPE at UMASS. If not, you should turn to Appendix B before reading further.

Starting HOPE

After you are signed on, you are ready to use the editor. When the system monitor requests

```
RECOVER/SYSTEM:
```

you must enter the correct subsystem by saying

```
BATCH
```

You can then enter the editor by simply saying

```
HOPE
```

The command HOPE can also be entered in response to any system request. At this point the editor will be made active and you should see

```
**WELCOME TO HOPE  JANUARY 1, 1978  1:01 AM
```

You are now in the editor and can enter any HOPE request in response to the prompt.

After quitting HOPE, you may leave the system monitor by typing BYE.

Use of Permanent Files

Since the editor maintains your files for you while you are editing them, your files must reside in your permanent file catalog before you enter the editor. The editor will not be able to edit a "local" file. When you preserve a file, all of your work will be returned to your catalog. Thus, after you leave the editor you will not have a "local" copy of any file that you were working with in the editor.

Restrictions

Some restrictions are imposed on the use of HOPE at UMASS. The characters "@" and "↑" are not allowed to appear in sequence in any file that you are editing. If the characters are found together, the editor will ask if it can insert a blank between the two in order to allow you to edit the file. The only characters that can be used as special symbols are

```
< > ? $ + - * ! ( ) = , . # [ ] % " ' _ /
```

The ELLIPSIS, MARGIN, and JOKER symbols have special restrictions. ELLIPSIS symbol may not appear as the first (or last) element in a text string in any request. If the MARGIN symbol appears on the left (or right) of a text string in a MAKE request, it must also appear on the left (or right) in the new text string. The JOKER symbol will not match the characters "@" and "↑" if they appear in your file file.
Currently the length of an individual file that you can use in the editor is restricted to approximately 40 pages of text where each page is about 50 lines containing 30 characters per line. (This does not restrict you to page or line lengths of that size; a line in a file may be of any length up to 150 characters long.)

General Information

Even though all previous examples show a request in uppercase letters, you may enter the request in either upper or lowercase letters (or a mixture of both). For example

```
fOrWArD aLL
```

will be correctly interpreted as "FORWARD ALL."

APPENDIX C: USE OF TERMINALS AT UMASS

This is your guide to the use of the computer terminals at the University of Massachusetts. No doubt the most difficult part is finding a terminal that is not already in use. At the end of this appendix you will find detailed information about each of the three types of terminals you are most likely to encounter at UMASS.

Sign-On Procedures

To use the terminal you must first "sign-on." By this we mean you have to establish communications between your terminal and the computer. A terminal can be connected to the computer in one of two ways. The first is a terminal that can be connected to the computer through the phone lines. In this case, a phone is connected to each terminal by means of a small box called a coupler. A coupler has a cradle for holding the telephone handset. The second is a terminal that is directly connected to the computer by means of a direct line. You can recognize this by seeing if the terminal has a wire (other than the power cord) running from it to the wall. This is referred to as a hard-wired terminal.

Connecting a Hard-Wired Terminal

After turning on the power to a hard-wired terminal, connection is automatically established between your terminal and the computer. You may now follow procedures for after the connection is made.

Connecting a Phone-Coupled Terminal

The first step is to turn on the terminal and coupler. If they do not turn on, find another terminal. You must next dial a number which will connect you to the computer. That number is 5-1600. After a ring, you should hear a high-pitched tone; you now have approximately ten seconds to respond to the computer.

After Connection is Made

Now that the computer is in contact with your terminal it is waiting for you to tell it what type of terminal you are using. This information is provided by typing what are called "sign-on" characters. The proper characters are listed at the end of this appendix for each terminal. After typing the sign-on characters for your terminal, the computer should respond by typing

```
78/01/01. 01.01.01.
UMASS NOS 1.2-452
USER NUMBER:
```

If the computer does not respond, you've waited longer than ten seconds to type in the sign-on characters -- hang up and try again. If all's well, type your user-number, a comma, your password, and a carriage return. Alternately you may type your user-number and a carriage return; wait for the computer to blacken out a space for your password and then type your password and a carriage return. If the computer responds with

```
IMPROPER LOG-IN, TRY AGAIN
```

you did something wrong while typing in your user-number or password. You can re-enter your user-number and password to try again. If you've entered a correct user-number and password, the system will print

```
TERMINAL: nnn, USERS mm
RECOVER/SYSTEM:
```

where "nnn" is your terminal number and "mm" is the number of users currently using the system. You may now refer to Appendix A for further information concerning the use of HOPE.

Sign-On and Control Characters

At the end of this appendix you will find a list of control characters for each terminal. These characters may be used to delete the last n characters on a line, to delete an entire line, or to interrupt the actions of the editor. To delete the last n characters on the line you are currently typing, type the character-delete character n times. For example, if * is the particular character-delete character for your terminal, QUO*IT will be interpreted as QUIT after you press the carriage return. If you begin typing a line and then wish to start over, type the line-delete character for your terminal. The editor will then allow you to retype the line. To stop the action of the editor, type the interrupt character for your terminal. For example, if you have requested the editor to list a file and then decide to halt the listing, type the interrupt character.

A summary of sign-on and control characters for each terminal follows. (CR stands for carriage return and CTRL-C (or CTRL-H) means push the buttom marked CTRL and at the same time press the "C" (or "H") character.)

Control Characters for the Decwriter

The Decwriter is a modern, high-speed terminal with a wide printing surface. It can be recognized by its white wedge-shaped body and its dot matrix print.

Sign-on:	CR CR
Character delete:	(line-feed) backspace
Line delete:	ESC or CTRL-C
Interrupt:	CTRL-C

Control Characters for the ITY-38

The Teletype model 38, or ITY-38, is an APL terminal which prints on a wide carriage. It can be distinguished by its print mechanism which is enclosed in a curved see-through plastic top.

Sign-on:	) CR
Character delete:	(line-feed) backspace
Line delete:	ESC or CTRL-C
Interrupt:	CTRL-C

Control Characters for the Beehive

The Beehive is a video terminal which uses a TV-like screen. There are several different models of video screen terminals, but they will all accept the same control characters.

Sign-on:	CR CR
Character delete:	CTRL-H
Line delete:	ESC or CTRL-C
Interrupt:	CTRL-C

Sign-off Procedures

After you end the HOPE session (or any time that you are through using the system) you may "sign-off." By this we mean you will terminate communications between your terminal and the computer. To do this, simply type

BYE

After the system response, hang up the phone (if you used a phone-coupled terminal) and turn off the terminal.