Van Dam's HT System: 'Edit Phase'

A MANUAL
FOR THE
EDIT PHASE
OF THE
HYPERTEXT EDITING SYSTEM
FOR THE
IBM SYSTEM/360 MODEL 50

ANDRIES VAN DAM

CENTER FOR
COMPUTER & INFORMATION SCIENCES
BROWN UNIVERSITY
PROVIDENCE, RHODE ISLAND

TABLE OF CONTENTS

21 January, 1969

0. PREFACE

The current Hypertext Editing System and its manuals (EDIT phase and FORMAT phase) are preliminary releases of a first stage of development. Implementation of this first version was begun towards the end of May, 1968 and we started doing useful work using a constantly evolving system towards the end of September. (A 130 page manual on the Brown University Student Operating System was finished in November; all our software documentation since then, including these manuals, has been done on the Hypertext Editing System.)

Needless to say, there are many developments and improvements still to be made, both to the system and to its documentation (and there are even a few lingering bugs). We expect to provide updates to the current version for the next several months, and then start implementation of the next version -- a multi-user, multi-station system with many changes and new features.

A paper describing the system, to be delivered at the second Computer Graphics Conference at the University of Illinois in March, 1969 (available upon request), describes general background, philosophy, and ideas to be incorporated in the next version; these manuals serve only as preliminary reference manuals.

Feel free to use the current system and its documentation for experimenting or producing manuscripts -- let us know about any problems you may encounter so that we may explain or fix them. Good luck!

Acknowledgements: The work reported in this manual was supported in part by a contract between the International Business Machines Corporation and Brown University. The manual was edited by Steven Carmody, Kenneth Denzel, David Rice, and Robert Wallace.

1. INTRODUCTION

The Brown University Hypertext Editing System is a computer system for text handling and display, having a number of related uses.

It is both a sophisticated system for the composition and manipulation of manuscripts, and a reading machine on which to browse and query written materials having complex structures. By a Hypertext we mean strings of text arbitrarily interwoven and connected, i.e., nonlineal text. There are two essentially distinct phases for manipulating hypertexts:

In the EDIT phase, a user can create and manipulate a Hypertext using three classes of functions: EDIT commands, STRUCTURE commands, and TRAVEL commands. Combined, these functions allow a user to create new text, to edit text which has been previously typed into the system, and to browse through text which already exists. He may also structure a text in any manner desired, either while it is being created, or after the entire text has been written as a Hypertext.

In the FORMAT phase, the user can format (i.e., specify the lay-out of) a previously edited Hypertext for conversion to conventional hard copy form, via IBM's TEXT360 program and a high speed line printer.

Reference material on the system is divided into two manuals, one for the EDIT phase, and one for the FORMAT phase. The user should become familiar with the system by first reading the U. of Illinois paper, and then trying simple functions such as INSERT,DELETE, and SUBSTITUTE. (He can get started just by trying the simple functions, of course.)

2. PRELIMINARIES FOR THE EDIT PHASE

2.1 SCREEN AREAS

There are three areas on the scope screen. These are:

HYPERTEXT DISPLAY AREA: The actual Hypertext is displayed in this area, 40 lines of text (or blanks) at a time. This is the area where the user does all of his creating and editing of text.

ANNOTATION AREA (or TAG AREA): In this area the LINK and TAG explainers are displayed. Also, any labels or explainers that are typed in by the user appear in this area, as he is typing them, and prior to his accepting them.

PROMPT AREA: In this area, below the annotation area, the system "prompts" the user as to his possible alternative actions and informs him of any errors he has made. In the EDIT phase this is a system area, and the user never types in it; in the FORMAT phase, he may choose options or type in parameters in this area.

2.2 BASIC EDIT MODE -- HOME PLATE

When a display first appears, the system will be in basic EDIT mode, with the prompt: "HIT FUNCTION KEY OF DESIRED ACTION". The user then pushes the desired function key, indicates with the light-pen where on the screen (i.e., with respect to which character(s)) he wants the indicated function to occur, and finally accepts or rejects the results.

2.3 PROMPTS AND MESSAGES

The prompts and messages which appear in the prompt area at the bottom of the scope display are important aids to the user. They explain the options that are available to him in the current state of the system. For example, the prompt will specify what the user is to do next in order to complete the function he has specified.

The general format of the prompt is:

FUNCTION: DIRECTIONS.

The directions generally include the accept/reject options described below. the term "string" in the directions denotes an arbitrarily long sequence of characters, blanks, or special symbols. Anything from a single character to a full screen of text is thus a string; naturally, if the

string whose endpoints must be identified is a single character, it must be lightpenned twice.

If the user does something wrong, the prompts have been designed to direct him out of difficulty. It is always wise to read the prompt before proceeding, i.e, to make sure that the correct function key has been pressed, that the system has processed the interrupt, and that it is waiting for the user's next action. As a further indication that the system is ready and waiting, the function key which was pressed lights up.

If the user attempts to do something which the prompt does not require, no operation will occur. Thus if the user tries to type in his INSERT before he lightpens the place where it is to begin, the keyboard will not function; if the user lightpens the screen when he is supposed to be typing, the screen will blink but nothing else will happen.

2.4 ACCEPT VERSUS CANCEL

With most of the EDIT functions (operations or commands), the user has the option of accepting or rejecting the function before its results are actually entered into the Hypertext data structure, i.e. while he is in the process of exercising the function, or even after he has viewed its results.

The user ACCEPTs an operation, e.g. an INSERT, by simultaneously pressing the ALTERNATE CODING (ALTN CODING) key, located at the extreme left in the upper row of the alphanumeric keyboard, and the END key, also located in the top row of the keyboard (the key for the numeral 5).

The user rejects (CANCELs) a function during any of its intermediate and final stages by pressing the CANCEL function key on the function keyboard (this is not the same as pressing the alphanumeric key for the numeral 0). Pressing the CANCEL key cancels the function completely[1], results in no operation occurring, and brings the user back to basic EDIT mode status, so that he can activate his next function. Thus the CANCEL option can be employed as a panic or reset button in case the user feels he has done something wrong or is dissatisfied with his results.

2.5 HYPERTEXT AREAS AND SCOPE PAGES

A Hypertext area is a continuous, arbitrarily long piece of text (or the "space" reserved for it) which is terminated either by an "end area line" (* * * END * * *) on the screen, or by a BRANCH depar-

---

1. There is as yet no partial reset of the last phase within a function.

ture point (i.e. BRANCH explainer). A "scope page" or "screen" is a "window" which permits the user to view (up to) 40 lines of a Hypertext area. If a Hypertext area is less than 40 lines long it can be displayed in its entirety as one display page. If it is not, the user must move the window by exercising one of the traveling functions.

2.6 STARTING A NEW AREA

There are only two methods for obtaining a "blank" (i.e., new) area of Hypertext: by creating a brand new data set (see appendix A), or by using the NEWAREA key while exercising a MAKEBRANCH or a MAKELINK function (see section 5.2). Initially, the only "text" in a new area is the "start new area line":

***** START *****,

and the "end area line":

* * * END * * *.

After the user has created a new area, he may travel to it, but before he does any editing in this area, it is wise that he place a LABEL in this new section of the Hypertext. (See MAKELABEL). The label should be made following the last asterisk of the start line since the end line is not pen-detectable. After the label is made, the user should DELETE all of the start line, being careful not to also DELETE the label at the same time. If the start line is not DELETEd it will appear in any hard copy printout; the end line will never appear.

3. GOOD SCOPE TECHNIQUES AND PRECAUTIONS

3.1 PROPER USE OF THE LIGHTPEN

When pointing the lightpen at a character, the user should be sure to center the two dots of light directly over it so that parallax doesn't cause him to point at the wrong character. This is particularly important for dim characters such as the period. Furthermore, after the character has been "seen" (at which point the screen will blink briefly) the user should check to see that the correct character was indeed identified before proceeding.

3.2 LACK OF RESPONSE AND LOCKED KEYBOARD

If the user operates either of the keyboards but the editing system fails to respond, he should try simultaneously pressing the ALTERNATE (ALTN) CODING key and SHIFT key located at the bottom left edge of the alphanumeric keyboard. This action clears the "locked" condition of the alphanumeric keyboard which may result from a mistaken operation of this keyboard. As long as the alphanumeric keyboard is locked, the Editing program will fail to respond to depressions of the function keys.

If the keyboard is clear and the system still does not respond, the /360 may be busy processing some one else for a few seconds -- relax! If the keyboard is clear and the /360 is not looping or preoccupied, the user may have locked himself out by pressing function keys before previous interrupts had been properly processed by OS -- it gets confused easily. His only recourse at this point is to reload the editing system.

3.3 CURSOR, JUMP KEY, AND ADVANCE KEY (ALPHANUMERIC KEYBOARD)

Hitting the JUMP key (by accident), located on the far left of the alphanumeric keyboard, will have no effect except locking the keyboard if the "cursor" (the horizontal line (typing bar) which represents the current position of the "typewriter carriage") is not displayed on the screen. If, however, the cursor is located in the main display area it will probably "jump" down to the beginning of the prompt area when the JUMP key is hit. At this point the user should hit the JUMP key until the cursor is back at the location at which it was initially displayed, from where the user may move it by retyping or using the ADVANCE key.

If convenient, it is always best (easiest) to CANCEL the function when the JUMP key is hit and begin again. If the JUMP key does displace the cursor and the user accepts the function, all kinds of graphical junk is likely to result on the screen, and it is best to leave via ABEND and start afresh.

3.4 LEAVE FUNCTION

All EDITing and FORMATing is done on a work data set which was copied from the user's permanent data set when he initially requested the data set by name. Consequently there are several procedures for updating the user's permanent data set from the work data set.

The LEAVE function allows the user to lightpen one of the following options for cleaning up or finishing up the current Hypertext session:

1) COPY - This allows the user to save all his work on disk but does not end the session. The work data set which he has been editing overwrites (replaces) his permanent data set on disk. The program returns to the page in the work data set from which the COPY function was activated. Frequent COPYing is desirable -- if the /360 gets clobbered, all the work done since the last COPY is lost. (We suggest using COPY every 10 to 15 minutes, or while the user is scratching his head wondering what to do next -- COPYing takes less than 20 seconds for medium sized data sets). While the COPYing is taking place, a message ("THIS SPACE AND TIME FOR RENT") will appear.

2) FINISH - Like COPY this allows the user to save on disk his work done during the current session. The program halts, ending the session. The same message as in COPY appears. Naturally the system may be reloaded now and work done on the same or a different data set.

3) ABEND - This allows the user to scratch all his work since the last COPY for this session and ends the session (used when the user has messed everything up completely, when he is demo-ing, or just fooling around). At Brown this action requires a reply by the operator before the "monitor" reappears (the partition is being re-initialized).

4) RETURN - For non Brown University users this is the same as ABEND; at Brown, this will return the user to the monitor without operator intervention, after an appropriate clean-up as in ABEND. However, a little space is lost after each RETURN, so that after many RETURNs, when the monitor is in a small partition (98K), the system might return to the monitor with a completion code of 80A; don't worry about this -- continue as usual.

3.5 MYSTERIOUS GLITCHES

If the user experiences difficulties such as causing the bombout message to appear on the screen ("THIS PROGRAM WORKS, BUT SOME PEOPLE DON'T KNOW HOW TO USE IT..."), or getting locked out while both the alphanumeric keyboard and the /360 are ok, he should have the system reloaded and try again (in case of the bombout message, pressing CANCEL or RETURN may suffice). Occasionally we have found that when totally inexplicable things are happening, re-IPLing magically cures them. Particularly in a multi-programmed environment such strange things do occasionally happen, and are seldom repeatable -- grin and bear it, remembering that genuine bugs are usually repeatable. If the situation can be made to repeat itself, describe it, and send us a dump.

4. EDIT COMMANDS

4.1 INSERT

To enable direct text insertion from the scope keyboard, the user must first depress the INSERT function key. The cursor (the typing bar) will appear automatically at the end of the text in the display area, provided there is an end of area line displayed:

*** END ***.

If the user is working on a new area, the cursor may or may not appear after the last asterisk (*) in the start area line:

**********START**********,

but he may start typing there regardless.

The user may now type in continuous text until he approaches the bottom of the "display" page (about 35 lines). If he desires to continue, the user accepts the INSERT (see below), SCROLLs forward as many lines as he wants to, or more quickly, one page forward and then backward by line until the text from the preceding page appears at the top of the screen. The user now reinitiates (presses) the INSERT function and can type in another page without interruption.

When the INSERT is finished the user accepts it by depressing the ALTERNATE CODING (ALTN CODING) key and the END key simultaneously. The text, which wraps around from the right scope margin to the left as it is typed, will then be formatted as ragged-right (i.e., words will not be broken). (The user should not type strings of more than 73 characters without a blank! - if he does, the display from that point on will be blank and all below is lost, in which case it is best to ABEND.)

If an INSERT is to be made within existing text, the user must lightpen the point at which he wishes the INSERT to begin. As he types, the text spreads to form as large a gap as needed for him to type his insertion. If this latter option is chosen, a maximum of 255 characters, approximately three and one half lines, can be typed in at one time. Again, the user accepts the insert by pressing the ALTERNATE CODING and END keys.

If the user tries to go beyond the 255 character limit, the cursor will stop at that point and further typing by the user will simply result in typing over the same single character. If the user ACCEPTs the INSERT at this time he will probably only get the first few words he typed in, the rest

being lost. However, if he notices that he has gone to the very end, he can BACKSPACE one space and then ACCEPT, and the first 255 characters will be INSERTed correctly. (The BACK-SPACE key is in the upper right hand-hand portion of the alphanumeric keyboard.)

Notes:

1) The user can get around the 255 character limitation in one of two ways, depending upon the length of the INSERT. If the INSERT is reasonably short the user can perform a sequence of INSERTs and ACCEPTs. However, if the INSERT is quite long, the user can SCROLL to the blank portion of the area (just above the end area line), and type the INSERT. He can then REARRANGE it into place when he has finished typing. If the end area line is not displayed, he can travel to another area (e.g. an area reserved for this purpose), type the INSERT into this area, and finally REARRANGE it into place.

To insert long segments of text the user might prefer to use CARD INPUT (Appendix E).

2) If at any time an error is made in the INSERT (prior to ACCEPT) the user can BACKSPACE the cursor one character at a time, type over the error and then ADVANCE the cursor one character at a time by retyping the correct letters. Using the SPACE bar to pass any following correct characters will result in their being replaced by blanks; the ADVANCE key should be used instead to advance past the correct characters.

3) If the user discovers that he has made an error in typing at some place a number of character spaces from his "present location" (i.e., where the cursor is), a much quicker way of moving the cursor through the text is by pressing the CONTINUE key and then tapping the BACKSPACE key until the error is reached by the cursor, releasing the CONTINUE key, and then typing over the error. The CONTINUE and ADVANCE keys may then be used to reposition the cursor back to where he was typing.

4) Since an INSERT can be made only directly following a character already positioned on the screen and not in some blank area, the user must first type in the INSERT behind this preceding character, and then, if necessary, position the new string properly using FORMAT-ting functions such as PARA-GRAPH or SKIP LINES.

5) In the EDIT phase any special characters such as LINK asterisks (*) or BRANCH explainers, labels, or TAG and FORMAT markers are treated as literal character strings dur-ing EDITing operations and are therefore not differentiated from the text proper. Thus one may insert after such a marker as if it were ordinary text; similarly, using DELETE (see below), one may (inadvertently) remove such markers - be careful!

Inserting after after a label (lightpenning the character string) may mess up the display while typing is proceeding, but everything will be properly ACCEPTed.

6) On long inserts, it helps to use the SWITCH PHASE key (see section 4.11) to paragraph the text so as to make it more legible and flicker-free.

7) The user may INSERT and ACCEPT only one string with each operation of the INSERT function key.

4.2 DELETE

The user may DELETE a string of characters by lightpenning, in either order, the end points (i.e., characters) of the string to be DELETEd. This portion of the text will be removed from the display, and a blank space will appear where the DELETion occurred. The user has the option of ACCEPTing the DELETion or CANCELing it in the usual way. If it is ACCEPTed, the endpoints of the text will be closed up; if it's CANCELLed, the DELETEd string will reappear.

The user may DELETE a maximum of a whole display of text since he may not travel between the lightpenning of the DELETion endpoints.

Any LINK or BRANCH points of departure embedded in the string will also be DELETEd, and their entry points (i.e., the sections of text they lead to) will be lost (unless the user restructures JUMPs to them or has previously LABELEd them). Similarly, any labels, or TAG or FORMAT markers internal to the string will be DELETEd.

4.3 DELETE SINGLE CHARACTER

This function is basically the same as the previous delete function with the limitation that only one character may be DELETEd, but with the advantages that it requires only one lightpenning to DELETE the character, and that any number of single DELETions can be made with each operation of the function.

The user does not have the option of ACCEPTing or CANCELing each DELETion or group of DELETions performed as a whole -- all DELETions done in this mode are irrevocable, being accepted as they occur. Only the CANCEL function key ends the mode; the user may then go on to select another function.

4.4 REARRANGE

This function allows the user to move or "REARRANGE" a portion of the text from one point to another. The user lightpens (in any order) the end points of the string to be moved and then lightpens the point at which this string is

to be inserted. The text being REARRANGEd is displayed at the indicated point and removed from its current location so that the user can see how it looks. The user may reposition the indicated insertion point by lightpenning any number of points in succession until he finds a position that suits him. When the user ACCEPTs the REARRANGE, the indicated string is INSERTed at the last indicated point and DELETEd from its original position.

The user may REARRANGE up to one whole scope page of text. However, if he REARRANGEs more than 255 characters, only the first 255 will be displayed for his approval, on the screen at the insertion point, prior to ACCEPTing. After ACCEPTing the operation, however, the whole REARRANGEd section will appear correctly.

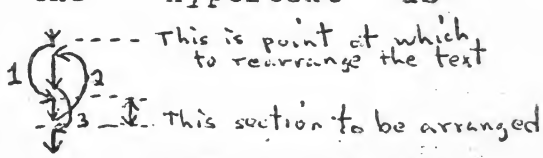
After having lightpenned the endpoints of the string to be REARRANGEd, the user may travel to other sections of the text. Once he has finished traveling, the prompt message will tell him to lightpen the point at which he wants the REARRANGEd string to be inserted.

Since one usually REARRANGEs whole words or phrases rather than pieces of words, the REARRANGE function automatically inserts a blank between the point of insertion (i.e., the character which was lightpenned) and the leading character of the string. To REARRANGE within a word therefore, it is usually faster to SUBSTITUTE for the incorrect portion of the word. Also, when a string is REARRANGEd, the blanks which were used to pad out the right margin of each line are not removed and will appear between words. They are removed, however, when the text is printed, so don't worry about them.

Notes:

1) In the current version, REARRANGE (and COPY and BORROW as well) has the (odd) property of changing special symbols such as FORMAT characters (# and $, see FORMAT Manual Section 2.2), LINK asterisk (*), TAG marker (@), and BRANCH explainers, into literal characters. This means that any special meanings are lost, and if text including these markers must be moved, it is less confusing to DELETE them soon after moving. If a BRANCH explainer is part of the REARRANGEd string it will be moved as normal text, with the possibility of a few bonus characters (true for COPY or BORROW as well). Also, if text including PARAGRAPHing formats is REARRANGEd, it may look a little misplaced prior to ACCEPTance; after ACCEPTance, the PARAGRAPH formats are deleted, i.e., turned into #'s.

2) If a large, formatted section needs to be REARRANGEd, it is better not to REARRANGE it but to fake it by restructuring the Hypertext as follows:



3) If one REARRANGEs a piece of text which contains a label, the characters of the label will become part of the REARRANGEd text, with the possibility of a few bonus characters and extra blanks as well.

4) REARRANGing a string into itself will clobber the system and is therefore not recommended.

5) The user may REARRANGE one string with each operation of this function.

4.5 COPY

This function is similar to REARRANGE in that the indicated character string is inserted at the indicated point. The only difference is that the string is not deleted from its original position.

As above, the user may repostion the indicated insertion point so as to move the copied string around until he has decided upon the correct point for its insertion. To copy text from another data set, use MERGE (Appendix F).

4.6 BORROW

This function is similar to COPY with one added facility which indicates that the string was BORROWed: the orginal string is sandwiched by a "-/" in front of the first character and a "/" after the last, whereas the copied (BORROWed) string, to distinguish it, is sandwiched before and after by a "+/" and a "/" respectively. These additional characters are now part of the text, and hence are printable (or editable, for that matter).

The user has the added option of tacking on an explainer (or citation) which will appear after the original string and after the BORROWed copy wherever it is inserted into the text. When the user exercises this option to enter or change the explainer (the default explainer is a blank) he lightpens the words "CHANGE EXPLAINER" in the prompt message, types in the new explainer (a maximum of 255 characters) and ACCEPTs it as usual. He must then CANCEL in order to BORROW or to execute any other function.

In the current system the explainer is purged after the scope session(i.e., when the program is terminated) or when the explainer is changed and replaced by another explainer. Any occurrences of the explainers remain in the text, naturally. An explainer is displayed immediately following the closing "/" of the BORROWed string and is terminated by the characters "//". Thus the completed "BORROW" looks like:

+/BORROWED STRING/EXPLAINER//.

An explainer could be used as a citation to indicate the

source of the string, and, perhaps, when and by whom the BORROW was made. This is handy, for instance, when trying to arrange notes from diverse sources into a coherent structure.

Note: The explainer will also be printed on any hard copy.

4.7 SUBSTITUTE

This function can be considered an ordered combination of the functions DELETE and INSERT. The user lightpens the end points of the string to be SUBSTITUTEd for and this string is DELETEd from the display (and from the text if the SUBSTITUTion is subsequently ACCEPTed). The user then types in the substitution string and this is INSERTed in place of the string which was DELETEd.

The user may SUBSTITUTE any one string for another with each operation of this function; the two strings do not have to be of equal length. Up to a whole display page can be deleted but the replacement string may not be longer than 255 characters. If a longer SUBSTITUTion is desired, the user may first DELETE and then use the procedures described at the end of 4.1 INSERT for long insertions.

One of the common uses of SUBSTITUTE is to INSERT a character string directly in front of (i.e., without intervening blanks) an existing string. For example, to put a left parenthesis in front of the first letter, "H", of "HYPERTEXT", SUBSTITUTE the string "(H" for the "H". Also, to close up the space in

"LIGHT PEN",

one would substitute "TP" for "T P".

4.8 MAKE INSTANCE

This function allows the user to "subroutine" or quote a piece of text by name. The endpoints of the string which is to become the INSTANCE are lightpenned, and a label (independent of normal text labels) which will be used to catalogue the INSTANCE must be typed in and ACCEPTed. The original text remains unchanged on the display screen, but a copy of the INSTANCE and its label is stored for future use (during that current scope session only, in the current system).

While this function is much like the COPY function, it does not require the user to lightpen the endpoints of the string each time it is to be inserted; he need only "call" the INSTANCE by using its label and lightpen the point of its insertion, as shown below. As an example, a separator line of asterisks is a useful INSTANCE.

4.9 INSERT INSTANCE

This function allows the user to insert at any point in the text an INSTANCE which he has previously "subroutined". The user light-pens the point at which he wishes the INSTANCE to be inserted and then types in and ACCEPTs the label of the particular INSTANCE to be inserted. As with the INSERT function the text plus the inserted INSTANCE is shown on the screen and the user may now ACCEPT the insert. Only one INSTANCE can be inserted with each operation of the function.

4.10 SUPPRESS

This function allows the user to SUPRESS (or redisplay) the appearance on the screen of either FORMAT markers (# or $) or labels (character strings with lines through them). He must lightpen the appropriate part(s) of the prompt and then exit from the function by hitting the CANCEL key. He may undo either SUPPRESSion by calling for the function again, and lightpenning the appropriate part(s) of the prompt: the prompt acts like a flip-flop. Note that in the FORMAT phase this function is located in a slightly different position, and does not give a prompt -- the button itself is the flip-flop there (for FORMAT markers only).

4.11 SWITCH PHASE

This key flip-flops between the EDIT and FORMAT phases. When switching to the FORMAT phase, the system is automatically in PARAGRAPH & CAP1 mode (see FORMAT Manual, Section 4.13). When switching phases, make sure the switch has been fully completed by checking the prompt, before operating any function keys.

5. STRUCTURE COMMANDS

Once the text has been entered into the Hypertext Editing System it can be structured into separate sections using the three structure commands below. It is also possible to add "empty" areas to the Hypertext structure which may be filled with text using EDIT commands.

5.1 MAKELABEL

The user may label a section of the text by lightpenning the point at which he wants the label to be inserted and then typing in a unique, one to six character label which will identify this particular section of the text. (The label can be composed of any keyboard characters.) The label will appear after the lightpenned point with a line through it (identifying it as a label). This label will also be entered in the label table so that it can be accessed by the GETLABEL function. The text at the point where the actual label character string is inserted remains continuous with the exception that the label appears on the screen, possibly in the middle of a character string if the user put it there. Labels do not appear in hard copy.

To delete a label the user may operate either of the DELETE functions, DELETE one or more of the characters within the label, and ACCEPT the DELETion. The entire label will be DELETEd and will be purged from the screen display of that area. However, it will remain in the label table and will be "undefined" (see GETLABEL).

The present system does not check for duplicate labels, and only the first of duplicate labels is accessible.

5.2 LINKS AND BRANCHES

The next two functions, MAKELINK and MAKEBRANCH, are quite similar in their structuring. Their basic differences become apparent most readily in hard copy printout, where a LINK is treated as a footnote.

A number of differences also exist in their screen operation and locations. The BRANCH is a physical tie to another area of

the Hypertext.[2] There may be more than one BRANCH at the end of a page (forming a "BRANCH MENU"); the user is forced to take one of them to continue with the text. The LINK differs in that it provides the user with the option to travel to the LINKed section, while the default case is for him to continue reading text on that page.

Both LINKs and BRANCHes are symbolized by an asterisk (*) at their point of occurrence (point of departure) in the text, and by associated explainers. An explainer is a small string (less than 255 characters in length) used as an annotation, teaser, instruction to the reader, meta-footnote, etc.: the LINK explainer appears in the annotation area, while the BRANCH explainer appears in-line, next to the BRANCH asterisk (*).

5.3 MAKE BRANCH

This function allows the user to create a BRANCH to some other section of the Hypertext:

1) The initial prompt tells the user to lightpen the point of departure from which he wishes the BRANCH to be made.

2) The next prompt message tells the user the four options that he now has: "MAKEBRANCH: TYPE LABEL, HIT NEWAREA KEY, LITEPEN TEXT, OR TRAVEL." The first option allows him to type in the label (probably, though not necessarily, previously defined) of the area to which he wants the BRANCH to go (ACCEPTing it as usual). The second alternative is to press the NEWAREA key which sets up a new Hypertext area for the user. (Until the user adds new BRANCHes into this new area or creates a label within the area, he can not access it in any other way than by taking this particular BRANCH -- see section 2.6). These first two are the most straightforward methods for making branched Hypertexts.

The user also has the option of (traveling arbitrarily and then) lightpenning the point in the text to which he wants the BRANCH structured. Note that the prompt message may change, depending on the mode of travel, but after any traveling has been completed, the MAKEBRANCH prompt returns. Also, the lightpen may be used both to travel, and to point into the text after traveling has been completed.

For instance, if the BRANCH is to be made to a label but the user has forgotten the label, he may use the GETLABEL function, lightpen the label in the label table, and, after having decided that this is

---

2. One is allowed to create a LINK or BRANCH which loops back into the same area -- this causes no problems for browsing, but must be done carefully so as to keep the Print Program from looping. (See Printing, Part 3, of the Format Manual.)

indeed the desired label, may then either type in the label or lightpen the actual label (the character string with the horizontal line through it) ; he could then RETURN to the BRANCH point of departure.

3) The user must then type in and ACCEPT a mnemonic BRANCH "explainer" which he may use to describe where (or why) the BRANCH is going, ACCEPTing it as usual. If the user types in no explainer (i.e., he ACCEPTs immediately), the default explainer is just the BRANCH asterisk (*).

If the user ACCEPTs the operation, the BRANCH will be structured, and the BRANCH asterisk (*) followed by the BRANCH explainer will appear at the lightpenned point on the screen. Any text on the same Hypertext page following the BRANCH now disappears from the screen and, unless the user has labeled this text, or structured another JUMP into the text, he will not be able to get to it without deleting the BRANCH.

To delete a BRANCH the user operates either of the two DELETE functions and DELETEs any character(s) of the BRANCH explainer. The BRANCH will be DELETEd and, if no other BRANCHes exist at the BRANCH point, any text following it in the same area (or the * * * END * * * line) will reappear.

Notes:

1) Once a BRANCH is DELETEd, the user cannot in general get to the section where that BRANCH led unless that section had been previously labeled explicitly, or unless another BRANCH or LINK leads to it.

2) More than one BRANCH can appear at the end of a screen page, providing for a BRANCH "menu". the user can choose from the menu which area he desires to go to (see BRANCH).

3) In the current version, BRANCH explainers may not be edited in any way: at best no operation will occur and at worst the BRANCH will be deleted and all following BRANCHes will not be displayed. If it is necessary that one or a "menu" of BRANCHes be edited, the user must recreate them with the MAKE BRANCH function. It is entirely possible to insert a new BRANCH in the middle of an existing menu by attaching it during the MAKE-BRANCH operation to any character of the previous BRANCH explainer.

4) The user should LINK or BRANCH to a BRANCH MENU not by lightpenning the first BRANCH in the menu, but by labelling the top of the menu or lightpenning the character preceding the menu.

5.4 MAKE LINK

To make a LINK, the user follows the same actions as in making a BRANCH. thus:

1) The user lightpens a point where the LINK point of departure is to occur.

2) A prompt message appears telling the user the options that are now available: "MAKE-LINK: TYPE IN LABEL, HIT NEWAREA KEY, LITE-PEN TEXT, OR TRAVEL." The user can now type in a label, press the NEWAREA key, or (travel and) lightpen a section of the text.

3) He then types in and ACCEPTs a LINK explainer. The default explainer is an asterisk (*).

To repeat, a LINK differs from a BRANCH in that:

1) The LINK is represented as a point of departure asterisk within the text at the insertion point. Text in the same Hypertext Area following the point of departure remains on the screen, and the explainer appears at the bottom of the screen in the annotation area. To take a LINK, one lightpens the LINK *.

2) When hard-copy printout of the text is made, LINKs are treated as footnotes and appear at the bottom of the printed page in which the LINKed point appears. The text to which the LINK was made is not printed like ordinary Hypertext: if the area LINKed to is more that ten lines long (at the current printed column width), the remaining text will be wrapped into the main text following the link point. (See section 3.1 of the FORMAT manual).

Thus nested or looping LINKs to arbitrary sections of Hypertext are allowed as far as the EDIT mode and on-line browsing are concerned, but are not recommended for printout--the user must temporarily restructure the Hypertext in accordance with the restrictions in section 3.1 of the FORMAT manual.

5.5 MAKE TAG

This function allows the user to make marginal notes to himself or an on-line reader. Upon pressing this function key the user is asked to lightpen the point at which he wishes the TAG to be made. The user then types in (and ACCEPTs) his marginal note (TAG explainer) which appears at the bottom of the screen in the annotation area for his future reference; following the lightpenned point a TAG marker "@" will appear.

Tag markers and the corresponding margin notes do not appear in any hard-copy printout; they are simply for the benefit of the on-line Hypertext reader and editor.

5.6 SCROLLING EXPLAINERS

There may be several TAGs and LINKs appearing on the same scope display and since only one explainer (the one closest to the top of the scope page) at a time can be displayed in

the annotation area at the bottom of the screen, there are two functions which allow the user to "SCROLL" through the explainers of the LINKs and TAGs on that displayed page.

5.6.1 EXPLAINERS FORWARD

Upon operation of this function the explainer list is moved forward, displaying the next explainer for that display page. If the last explainer is displayed this function will have no operation. Only 6 explainers may be SCROLLed consecutively. To see any remaining ones on that page, the user must travel, for instance by SCROLLing a line.

5.6.2 EXPLAINERS BACKWARD

This function is the reverse of the above, i.e., the explainer list is moved back one. If the first explainer on the screen page is displayed this function will have no operation.

5.6.3 Scrolling Markers Off the Screen

To get positive identification of which explainer belongs to which marker (* or @), the user can SCROLL FORWARD one line at a time until the explainer changes. This is also the way to find where the marker for a particular explainer is located.

6. TRAVEL COMMANDS

There are a group of functions which provide the user with the ability to "travel" through a Hypertext by SCROLLing or by jumping (via BRANCH, LINK, and GETLABEL described in Section 5. Structure Commands).

6.1 SCROLLING

If the Hypertext area is longer than one display page the user may operate the SCROLL functions to move the display window backward or forward. These scroll functions will work only within a "Hypertext" area; to travel between areas requires a jump (LINK, BRANCH, or GETLABEL).

6.1.1 SCROLL FORWARD BY LINE

The user may scroll the screen display forward one line by pressing this functon key, one line for each operation of the key. When the end of an area is reached (either at a BRANCH explainer or the "end area line"), lines will simply be scrolled off the top and blank lines fed in at the bottom.

6.1.2 SCROLL BACKWARD BY LINE

The user can also SCROLL BACKWARD one line at a time, one line for each operation of the function key. When the beginning of the area has been reached no change will occur on the scope display with this function. Depending on how the page has been edited, especially with regard to the PARAGRAPH function, SCROLL BACKWARD BY LINE may actually scroll back several lines at a time.

6.1.3 SCROLL FORWARD BY PAGE

SCROLL FORWARD BY PAGE allows the user to SCROLL FORWARD through the text by a whole display page. The last line of the previous display becomes the first of the present one. When the end of the Hypertext area is reached this function will have no effect on the scope display.

6.1.4 SCROLL BACKWARD BY PAGE

SCROLL BACKWARD BY PAGE is exactly the reverse of SCROLL FORWARD BY PAGE. (At least) one whole display page is SCROLLed BACKWARD. When the beginning of the Hypertext area

is reached, this function will have no affect on the scope display.

6.2 LINK

The LINK function allows the user to travel to and thus display a LINKed-to area of his choosing. When this function key is pressed, the user is asked to lightpen the asterisk (*) within the text representing the LINK point of departure. If the user does not lightpen a LINK asterisk, an error message will appear at the bottom of the screen asking him to "try again". Upon successful lightpenning of a LINK asterisk, "a LINK is taken" to the point to which that LINK was structured. Thus the new display on the screen will be the section of text pointed at by the LINK. If the user tries to LINK to an undefined label, (i.e. either the LINK was structured to a label not yet created or the label has since been DELETEd) an error message will appear at the bottom of the screen.

6.3 BRANCH

This function allows the user to BRANCH to another section of the text. When this function key is pressed the user is asked to lightpen the explainer of the BRANCH he wishes to take. If the user does not lightpen a BRANCH explainer, an error message will appear at the bottom of the screen asking him to try again for a BRANCH explainer. Upon the successful lightpenning of an explainer, "a BRANCH is taken" to the point to which that BRANCH was structured. If the user tries to BRANCH to an undefined label (i.e., the BRANCH was structured to a label which had not yet been defined, or which has since been DELETEd), an error message will appear at the bottom of the screen.

6.4 GETLABEL

This function allows the user to access any of the labeled sections he has previously structured within the text. When this function key is pressed the text presently on the screen is replaced by columns of labels (the one-to-six character strings structured in the MAKE LABEL function) arranged alphabetically. The user lightpens the label of the section to which he desires to travel. Currently, if a label has been previously DELETEd it will remain in the table, but when lightpenned a message will appear at the bottom of the screen telling the user that the label is nonexistent and asking him to try again. When a "legal" label is lightpenned, the section with that label will be displayed on the screen (the label and the text following will start in the upper left hand corner of the screen).

GETLABEL provides a very fast way of moving around "in" a Hypertext, and then RETURNing to the previous place(s) visited-- as such it is often preferred to SCROLLing around which does not allow RETURNing. Consequently, the user is advised to use labels copiously, say, one per subsection of his manuscript. In this manner he creates an on-line "index".

6.5 RETURN

Each time the user executes one of the three jump functions, i.e. either:

1) takes a branch,
2) takes a link, or
3) retrieves a labeled section,
("gets" or "takes" a label)

the screen configuration from which he traveled is remembered (i.e., placed at the top of a pushdown stack). On each operation of the RETURN function the display on the screen reverts to the last Hypertext section from which the user executed one of the jumps.

Therefore the RETURN function may be operated successively until the user returns to his starting point (i.e., until the stack is exhausted). Further RETURNs result in no further operation. Note that SCROLLing commands are not remembered.

7. APPENDICES

Note for all JCL which follows:

1) Whenever "VOLUME=SER=AVDPAK" is specified below, substitute for AVDPAK your disk volume name.

2) The DSNAMEs shown may be changed, as long as the change is uniformly made wherever used.

3) The delimiters "<" and ">" are used to show where user information is needed.

4) If your Disk Unit is a 2311, double all the disk SPACE allocations indicated.

APPENDIX A: BASIC SYSTEM CONFIGURATION

The minimum requirements for running the system are:

1) a 360/50 with PCP, MFT or MVT, including BSAM and GAM (Graphic Access Method). however, variations must be made when moving from one configuration to another;

2) a 2311 or 2314 Disk Unit;

3) a 2250 Mod 1 with an 8K buffer and the Graphic Design Feature, or a Mod 3 scope; and

4) a TN (or equivalent) printer chain, required for operating systems which cannot handle characters not on a normal print chain, useful for other systems to get both upper and lower case from the TEXT360 program.

APPENDIX B: INITIAL CREATION OF SYSTEM

The Hypertext System is generated by running the following two jobs. The first (//UNLOAD) unloads the system from the included tape and places it in HTEXTSYS. The second (//SETUP) initializes the various works data sets used by the system.

```
//UNLOAD JOB (ACCOUNTING INFO),'UNLOAD TAPE',MSGLEVEL=1
// EXEC PGM=IEHMOVE
//SYSPRINT DD SYSOUT=A
//DD1 DD UNIT=2400,VOLUME=SER=<TAPE VOLUME>,                    X
//             DSNAME=HTEXTSYS,LABEL=(1,SL),DISP=OLD
//DD2 DD UNIT=DISK,VOLUME=SER=AVDPAK,DISP=(NEW,KEEP),           X
//             DSNAME=HTEXTSYS,SPACE=(TRK,(20,5,5)),            X
//             DCB=(RECFM=U,BLKSIZE=3072,LRECL=80)
//SYSUT1 DD VOLUME=SER=AVDPAK,DISP=OLD,UNIT=DISK
//SYSIN DD *
   COPY PDS=HTEXTSYS,FROM=2400=(<TAPE VOLUME>,1),               X
             TO=2314=AVDPAK,FROMDD=DD1
/*
```

```
//SETUP JOB (ACCOUNTING INFO),'CREATE WORKAREAS',MSGLEVEL=1
// EXEC PGM=IEBGENER
//NEWWORK DD DSNAME=WORKAREA,UNIT=DISK,DISP=(NEW,KEEP),         X
//             VOLUME=SER=AVDPAK,SPACE=(TRK,(<MAX>,0),,CONTIG),X
//             DCB=(RECFM=F,BLKSIZE=7200,LRECL=7200)
//NEWMERGE DD DSNAME=MERGE,UNIT=DISK,DISP=(NEW,KEEP),           X
//             SPACE=(TRK,(15,3)),VOLUME=SER=AVDPAK,            X
//             DCB=(RECFM=FB,BLKSIZE=1600,LRECL=80)
//NEWOUT DD DSNAME=T360IN,UNIT=DISK,DISP=(NEW,KEEP),            X
//             SPACE=(TRK,(50,5)),VOLUME=SER=AVDPAK,            X
//             DCB=(RECFM=FB,BLKSIZE=1600,LRECL=80)
//NEWPRTOT DD DSNAME=PRTFILE,UNIT=DISK,DISP=(NEW,KEEP),         X
//             DCB=(RECFM=FB,LRECL=132,BLKSIZE=132),            X
//             SPACE=(TRK,(50,10)),VOLUME=SER=AVDPAK
//NEWDICT DD DSNAME=DICT,DISP=(NEW,KEEP),                       X
//             DCB=(,BLKSIZE=2000),UNIT=DISK,                   X
//             SPACE=(TRK,(10,5)),VOLUME=SER=AVDPAK
/*
```

Note 1: For the SPACE parameter of the NEWWORK DD card, <MAX> should be replaced by the maximum number of tracks (text data set size) you will be using, with an upper limit of 260 tracks.

Note 2: If a 2311 Disk Unit is used instead of a 2314, change the DCB parameter on the //NEWWORK DD card to:

DCB=(RECFM=F,LRECL=3600,BLKSIZE=3600)

Note 3: The DSNAME for the //NEWOUT DD card (T360IN) is specified as SYS1.P1518.AVD.T360IN on the SYSIN card for the TEXT360 job specified in Appendix A of the Format Manual. Make sure that the two DSNAMES are the same.

Note 4: The DSNAME for //NEWPRTOT (here PRTFILE) should be the same as that for TEXT360's //PRTOT card (there SYS1.P1518.AVD.PRTFILE).

Note 5: Finally, the DSNAME for //NEWDICT (here DICT) should be the same as that for TEXT360's //DICT card (there SYS1.DICT).

APPENDIX C: NORMAL HYPERTEXT RUN

This is the JCL needed to run the Hypertext Editor on the scope:

```
//SCOPE JOB (ACCOUNTING INFO),'THE GROUP',MSGLEVEL=1
//JOBLIB DD   DSNAME=HTEXTSYS,UNIT=DISK,DISP=OLD,                X
//           VOLUME=SER=AVDPAK
// EXEC PGM=HYPRTEXT
//HFILE DD  DSNAME=DUMMY,UNIT=DISK,DISP=OLD,                     X
//           VOLUME=SER=AVDPAK
//WORKFILE DD DSNAME=WORKAREA,UNIT=DISK,DISP=OLD,               X
//           VOLUME=SER=AVDPAK
//OUT DD DSNAME=T360IN,UNIT=DISK,DISP=OLD,                      X
//           SPACE=(TRK,(50,50)),VOLUME=SER=AVDPAK
//SCOPE DD UNIT=SCOPE
//SYSPUNCH DD DSNAME=MERGE,UNIT=DISK,DISP=OLD,                  X
//           VOLUME=SER=AVDPAK
/*
```

The HFILE DD card above has a dummy DSNAME parameter which will be replaced by the DSNAME typed in by the user at the beginning of the scope session.

The WORKFILE DD card refers to the scratch area onto which the data set is copied during the work session. This data set must absolutely and positively be contiguous.

The OUT DD card refers to the data set onto which the intermediate output from the Print Program is placed. This data set later serves as

the input for the TEXT360 program which is run to get hard copy printout.

The SCOPE DD card refers to the scope unit to be used on this run of the program.

The SYSPUNCH DD card refers to the data set which will temporarily store the text to be moved (merged) from one data set to another. This data set later serves as input to the MERGE program.

APPENDIX D: CREATING DATA SETS

The following job initializes a data set of specified name and size:

```
//CREATE JOB (ACCOUNTING INFO),'CREATE DS',MSGLEVEL=1
//JOBLIB DD DSNAME=HTEXTSYS,UNIT=DISK,DISP=OLD,                    X
//             VOLUME=SER=AVDPAK
// EXEC PGM=CREATE,PARM='< N TRACKS >'
//CREFILE DD UNIT=DISK,VOLUME=SER=AVDPAK,DISP=(NEW,KEEP),          X
//             DSNAME=<NEW DATA SET>,SPACE=(TRK,(< N TRACKS >,0))
/*
```

The parameter (PARM=) on the EXEC card indicates the number of tracks on the disk of primary allocation. Approximately 1 track should be allowed for each expected page of printout (2 for 2311's). This number must also be included in the SPACE parameter of the CREFILE DD card.

The data set named on the CREFILE card is the one initialized. If this program is run on an already existing data set, that data set is re-initialized and whatever was in the data set is lost.

All editing is done on the system's work data set, and only if the user explicitly requests it, by using the COPY or FINISH options of the LEAVE function, is the permanent data set replaced with the updated work data set.

APPENDIX E: CARD INPUT

The following job allows the user to insert text into a data set through the card reader:

```
//CARDS JOB (ACCOUNTING INFO),' CARD INPUT ',MSGLEVEL=1
//JOBLIB DD DSNAME=HTEXTSYS,UNIT=DISK,DISP=OLD,                X
//             VOLUME=SER=AVDPAK
// EXEC PGM=CHARGEIT,PARM='<NUM,CHAR>'
//WORKFILE DD UNIT=DISK,VOLUME=SER=AVDPAK,DISP=OLD,            X
//             DSNAME=<EXISTING DATA SET>
//SYSIN DD *
<LABEL>
   .
   .
Text goes here
   .
   .
/*
```

The DSNAME for the WORKFILE DD card must be the name of the data set into which the insertion is to be made.

The LABEL after the SYSIN DD card is the one to six character label with which the user wants to identify the insertion. The new text is not linked to the rest of the text, but its label may be referenced with a BRANCH, GETLABEL, etc. Note: this cannot be a label that already exists in the data set.

The parameters (PARM=) on the EXEC card are:

1) NUM-- the number of columns beginning with column 1 in which the text to be read in has been punched;

2) CHAR-- a single character (e.g., <, *, ¬, etc.) that will be interpreted as a PARAGRAPH & CAP1 format every time it occurs in the punched input text.

The assumed values are: NUM=80 and CHAR=<; no PARM need be specified. If you want to change NUM while using the assumed CHAR, specify PARM='<NUM>'. If you want to change CHAR while using the assumed NUM, specify PARM=',<CHAR>'. As examples, to use only 72 columns of each card, and to use * for paragraphs, you would specify PARM='72,*'; to use all 80 columns but specify ¬ for paragraphs, you would use PARM=',¬'.

APPENDIX F: MERGING DATA SETS

This job will MERGE part or all of one data set into another data set:

```
//MERGE JOB (ACCOUNTING INFO),'2ND STEP OF MERGE',MSGLEVEL=1
//JOBLIB DD DSNAME=HTEXTSYS,UNIT=DISK,DISP=OLD,                X
//             VOLUME=SER=AVDPAK
// EXEC PGM=MERGE
//HFILE DD  DSNAME=DUMMY,UNIT=DISK,DISP=OLD,                   X
//             VOLUME=SER=AVDPAK
//WORKFILE DD DSNAME=WORKAREA,UNIT=DISK, DISP=OLD,             X
//             VOLUME=SER=AVDPAK
//SCOPE DD UNIT=SCOPE
//SYSPUNCH DD DSNAME=MERGE,UNIT=DISK,DISP=OLD,                 X
//             VOLUME=SER=AVDPAK
/*
```

To do a MERGE, the user starts by using the normal Hypertext job (//HYPRTEXT JOB) with the data set containing the text to be merged. The user presses the MERGE function key. As the prompt message indicates, he may now (or at any time during the merging process) change the BRANCH LEVEL (from its default value of the 1st branch) by lightpenning "CHANGE BRANCH OPT.". (see Sect. 3.2.3 Branch Level Option of the Format Manual for a description of branch levels.) He may now (travel and) lightpen the start and end points of the text to be merged. The Prompt Area will return to the "home-plate" message: "HIT FUNCTION KEY OF DESIRED ACTION", signifying the completion of the first step which copies the selected text to a work area.

To merge the text from the work area into another (or the same) data set (the second and final step), he next runs the //MERGE job. The user is asked to type in the name of the data set into which the text will be merged, hit the "JUMP" key on the keyboard (not any of the jump function keys), and type in the label to be associated with the new text. As with CARD INPUT, the new text is not linked to the rest of the text, but it may be referenced (via its label) with a BRANCH, LINK, or GETLABEL.

APPENDIX G: TAPE/DISK MOVES

These are useful for saving infrequently used data sets on tape. They are standard IBM Utility Programs.

1) Disk to Tape

```
//COPY JOB (ACCOUNTING INFO),'DISK TO TAPE',MSGLEVEL=1
// EXEC PGM=IEBGENER
//SYSUT1 DD UNIT=DISK,VOLUME=SER=AVDPAK,DISP=OLD,                    X
//             DSNAME=<HYPERTEXT DATA SET>
//SYSUT2 DD DSNAME=<TAPE DATA SET>,UNIT=<TAPE UNIT>,LABEL=(#,NL),    X
//             DCB=(RECFM=F,LRECL=7200,BLKSIZE=7200),                X
//             DISP=(NEW,KEEP),VOLUME=SER=<TAPE VOLUME>
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=A
/*
```

= File Sequence Number on the tape

2) Tape to Disk

```
//COPY JOB (ACCOUNTING INFO),'TAPE TO DISK',MSGLEVEL=1
// EXEC PGM=IEBGENER
//SYSUT1 DD DSNAME=<TAPE DS>,UNIT=<TAPE UNIT>,LABEL=(#,NL),          X
//             DCB=(RECFM=F,LRECL=7200,BLKSIZE=7200),                X
//             VOLUME=SER=<TAPE VOLUME>,DISP=OLD
//SYSUT2 DD DSNAME=<HYPERTEXT DATA SET>,UNIT=DISK,DISP=OLD,          X
//             VOLUME=SER=AVDPAK
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=A
/*
```

= File Sequence Number on the tape

Special note - if a 2311 Disk Unit is used rather than a 2314, change the DCB parameter in both jobs to:

```
DCB=(RECFM=F,LRECL=3600,BLKSIZE=3600).
```

APPENDIX H: FUNCTION KEY LABEL AND NUMBER

	CANCEL 0	LEAVE 1	Unused 2	SWITCH PHASE 3	
RETURN 4	NEWAREA 5	MERGE 6	SUPPRESS 7	SINGLE DELETE 8	Unused 9
GETLABEL 10	MAKE LABEL 11	MAKETAG 12	EXPLAINER FORWARD 13	EXPLAINER BACK 14	Unused 15
LINK 16	MAKE LINK 17	MAKE INSTANCE 18	INSERT INSTANCE 19	COPY 20	REARRANGE 21
BRANCH 22	MAKE BRANCH 23	BORROW 24	DELETE 25	SUBSTI-TUTE 26	INSERT 27
SCROLL	BACK PAGE 28	BACK LINE 29	FORWARD PAGE 30	FORWARD LINE 31	SCROLL

Compliments of the

HYPERTEXT EDITING SYSTEM

Brown University

21 January, 1969

20 January, 1969

TABLE OF CONTENTS

20 January, 1969

LIST OF FIGURES

1. INTRODUCTION

The Format Phase of the Hypertext Editing System provides the user with the facility to format and print onto hard copy his entire Hypertext created in the Edit Phase. It is assumed that you are familiar with the functions of the Edit Phase.

All of the text perusing (traveling) options available in the Edit Phase are included, and the appropriate function keys are the same. The current formatting options available, described in detail below, are sufficient to format a Hypertext so that it can be immediately printed in hard copy using a high speed printer (possibly followed by photo-offsetting). Provisions are also included for printing supplemental listings, such as a table of contents or a list of figures.

Rough draft copies, corresponding line for line with the screen appearance, may also be obtained (see Sec. 3.2.2, Quickie Printout). This mode prints all characters as capitals (if a regular print chain is used), and requires only a minimum of formatting (e.g., paragraphing, skipping lines and indenting).

The Format Manual will be most useful if the actions described here are tried on-line.

The reference source for the details of Parts 4 and 5 of this manual, Figures 3 and 4, and the Appendices is IBM's T3XT360 Reference Manual and Operating Guide.

Acknowledgements: The work reported in this manual was supported in part by a contract between the International Business Machines Corporation and Brown University. The manual was edited by Andries van Dam.

2. GENERAL REMARKS

The Format Phase of the system is entered from the Edit Phase via a function key (SWITCH PHASE). You may switch back and forth between phases using this key, editing and formatting at will. When the Format Phase is initially entered, the system will automatically be in the PARAGRAPH & CAP1 mode (see Sec. 4.13 below). This makes initial paragraphing of an unedited text quite facile.

The switching between format modes is accomplished solely by pressing any of the function keys. There is never any need to use the CANCEL function key to end a format mode (unlike the Edit Phase). However, as a "fail-safe" function, the CANCEL function key will always return the Format Phase to its initial state, whose prompt message is "THIS IS THE FORMATTING PHASE: HIT DESIRED FUNCTION KEY." Any formatting done up to this point, however, will already have been accepted in the data structure.

All of the traveling commands that are offered in the Edit Phase are applicable here, and they are controlled by the same function keys. Keep in mind that one may always travel while in an "edit" format mode.

2.1 SPECIFICATION AND TYPES OF FORMATS

Two types of formats and methods of inserting them are used in the system-- "edits" and "alters".

When specifying an "edit" format, you must first choose the format and any options dealing with that format, and then you are then free to apply this format to any and all desired points of your Hypertext. You initially press one of the "edit" format function keys (e.g. INDENT). You might then be asked to choose, with the lightpen, which of several related formats you desire (e.g. REGULAR INDENT or HANGING INDENT). Any choices applicable are displayed in the Prompt Area at the bottom of the screen. As a final step in the specification process you might be asked to type

in a number[1] (e.g., 6 meaning an indentation of six spaces).

After having designated an "edit" format, you may apply it to any points of your text with the lightpen. The specified format will remain active as you travel to other areas of your Hypertext via the travelling commands. Thus you could capitalize your entire Hypertext having pressed the capitalize key (CAP1) only once. Note that this differs from the Edit Phase where, in most cases, one function key hit allows only one action.

Each time a format is indicated by penpointing at the text display, the format is immediately entered into the Hypertext data structure. Thus, if a mistake is made, the incorrect format must be deleted (see Sec. 2.3, Interrogation and Deletion of Formats). When specifying multiple capitals (MULT CAPS) or underscoring (UNDERSCORE) formats, which require pairs of lightpen hits, the formats are entered into the text only after each pair of hits. They are, in fact, entered as two separate formats: a begin capitalization or underscore, and an end capitalization or underscore. Thus, to preserve symmetry, both should be deleted if you choose to delete one of them. If you end this mode after making an odd number of hits, the last lightpen hit is ignored.

The second method of inserting formats is used when entering "alter" formats, so-called because they "alter" the overall format of the printed copy (e.g., formatting the pages in double columns with a specified running title). Typically, many types of alterations are specified together at only a few places in the text, in particular at the start of the text. When you select the ALTER function key, you will be asked to lightpen a point in the text. After choosing the point, you may select (from a menu on the display), as many of the "alter" formats as you wish specified at that point. Each time you make a choice, the format is entered into the data structure but of course it may be interrogated and deleted later.

---

1. You must then either accept this number (ALTN CODING & ACCEPT) or reject it (by pressing any of the function keys for your next formatting choice, or CANCEL).

2.2 SYMBOLIZATION OF FORMATS

Most of the formatting specified by the user will not actually appear on the scope-displayed text. Many of the options would be impratical to implement (e.g., performing line justification and hyphenation on-line) and a few are impossible (e.g., displaying a single column of text 80 spaces wide on a display unit whose width is only 74). Thus, in general, the specified formatting will only be represented by a marker. The markers used are the number sign (#) and the dollar sign ($).

Several considerations resulted in the use of only two marker types to represent the formats. Primarily there are not as many readily discernable markers as format options. If 50 different marker types were used, it would be quite difficult to distinguish between markers and text. In addition, a typical user could not be bothered to learn what marker stands for what option.

Each time a formatting option is applied to the Hypertext with the lightpen, a number sign (#) or a dollar sign ($) will appear on the scope display as the "format symbol" or "marker" corresponding to that format. In addition, INDENT (REGULAR), PARAGRAPH, PARAGRAPH & CAP 1, and SKIP will be formatted on the display. Whenever a formatting option is specified by pen-pointing at an actual character of the text, the format symbol (and therefore the format option) will in general precede that character. The exceptions are RIGHT BRACKET, SUPERSCRIPTS, END KEEPS, and END TABLE, all of which immediately follow the lightpenned character. If any marker or symbol (e.g., a link marker or another format symbol) is pointed to with the lightpen, the format symbol will always appear before it.

A distinction should be made between the characters, # and $, and the format symbols, # and $. The format symbols will not be printed out, while the characters will. If there is ever any question as to which is which on the display, the function key labeled SUPPRESS can be utilized. Pressing this key suppresses the display of all format markers, thus any # or $ remaining on the display is an actual character of the text. The SUPPRESS function key acts like a "flip-flop" switch, i.e., the next time it is pressed, all the format symbols will reappear. A particular # or $ in question could also be interrogated (see Sec. 2.3, Interrogation and Deletion of Formats, below). If you try to interrogate a character of the text, the prompt area will give a message indicating that you lightpenned a

non-format character. Also, in general, if several formats have been specified together, they should each be interrogated to ensure that they are in the desired and correct order (explained in Part 4 where applicable).

Format symbols are not distinguished from regular text in the Edit Phase-- they may be deleted, substituted for, inserted after, etc. Act with caution, however; if, for instance, one inserts a blank or a special character between a PARAGRAPH & CAP1 format and its operand character, that character will no longer be capitalized. Furthermore, in the current system, any formats occurring in a string that is rearranged, copied or borrowed will be destroyed (they are transformed magically into the characters # and $).* The user is therefore advised to delay serious formatting until his editing is virtually finished. Rough draft hard copies can be easily obtained with the Quickie Printout feature (see Section 3.2.2), which requires only minimal formatting.

The order in which the options are specified can be important. As described below, a single capital format (CAP1) will only be effective if it immediately precedes a letter, whereas multiple capitals format (MULT CAPS) will capitalize all letters (and only letters) within its jurisdiction. Be logical when formatting, and be sure to insert formats in the order desired, e.g., a SKIP of 5 lines before an INDENT of 8 spaces.

2.3 INTERROGATION AND DELETION OF FORMATS

A facility is available which allows you to inquire as to the meaning of, and to delete if desired, any formats specified previously. This is done by pressing the INQUIRE/DELETE function key and lightpenning a character of the text display. If the character lightpenned is a format symbol (# or $), all the information about it will be displayed at the bottom of the display. If you lightpen it a second time in succession, it will be deleted. Otherwise, you can lightpen any other symbol to inquire as to its meaning. If at any time you lightpen a character of the actual text, you will be told that it is not a format symbol.

Format symbols may also be deleted in the Edit Phase with either the DELETE or SINGLE DELETE functions, although this is done without prior knowledge of the format's meaning.

---

*This will be cleaned up in the next release of the system.

2.4 NORMAL FLOW OF EVENTS

Normally, you create and edit your manuscript, using only the PARAGRAPH & CAP1 format option (automatically active when the format phase is entered) and possibly the SKIP and INDENT options, getting a number of Quickie Printouts along the way, until most of your editing is finished.

At that time, you would probably apply some "Alter" formats to the "start" of your manuscript and apply the fancier "Edit" formats to the body in order to prepare for the final Full Printout. Remember that "Alter" formats may be used anywhere in your manuscript to alter its appearance (changing margins, switching from single to double column, etc.).

3. PRINTING

3.1 CONVENTIONS

Due to the non-lineality of a Hypertext, certain conventions have been established so that the text can be printed lineally. As the default option, the first branch at every branch point is taken during the printing. However, the user can designate a different branch level to be taken by the Printer (see Branch Level Option, Sec. 3.2.3, below).

At print time (though not necessarily at any other time), a linked section of a Hypertext is assumed to be a footnote and is printed as such,[2] with the convention that the footnote is terminated when another link, a branch, or the END of the Hypertext fragment (area) is reached. A maximum of ten lines of footnotes (at the current line length) is allowed per printed column. If more than ten lines are included, the first ten lines will be printed as a footnote while the rest will be included in the main text, following the point at which the footnote (link) occurred.

Normally, if the user desires an expanded footnote in his Hypertext, but only wishes the first few lines printed in hard copy, he would signify the end of the text for the printed footnote by inserting, at the end, a link to any point on the scope page. Thus the complete link would be shown on the display, but only the designated portion would be printed as a footnote.

If, however, he wished for the entire linked section to be printed (in-line), he would have to make a branch to the linked section and another branch back to the main-line text from the end of the linked section.

---

2. This is a sample printed footnote. It was created as a linked section of the FORMAT MANUAL Hypertext (in a NEWAREA) and is terminated in this example by the END of this text area (which is right here).

The Hypertext Printer reinitializes itself after completing a printout (signified by an audible fweeeep at the scope), so that in the same session you may get several different printouts, provided the TEXT360 "Final Print" program is run in between each request for a printout (in the batch processing partition say). Several different Hypertexts can be "linked" together by using the page numbering facility described in Section 5.9.

The second (and final) stage for a printout of a Hypertext is through IBM's TEXT360 printing program. The Hypertext Printer is in actuality a translator which transforms all the text and formatting of a Hypertext into a lineal string of characters recognizable by the TEXT360 program, thus eliminating the need for learning the mysterious TEXT360 format codes, and reducing the probability of typographical errors. This design philosophy achieves two goals: first, you do not have to know anything about the TEXT360 program or text formatting to produce reasonably pretty hard copies; and secondly, if you desire quite elaborate printout, the only concepts required are those of column widths, margins, tabs, and running heads and foots, or precisely those concepts understood by most competent secretaries.

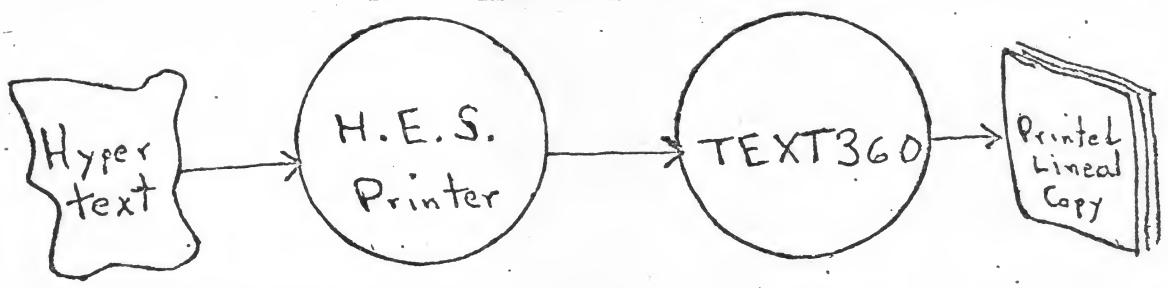


Figure 1. Flow of Printing Process

A separate deck must be run for the TEXT360 program (consisting solely of Job Control Language). This program may vary from installation to installation, but a sample of the program as run at Brown University is included as Appendix A.

3.2 OPTIONS

3.2.1 Full Printout

After having formatted a Hypertext using the Format Phase facilities, the full printout option causes a hard copy to be printed with all the specified formats plus any default options, with upper and lower case letters and any special characters specified (provided a special print chain ["TN" or equivalent] is used).

The FULL PRINTOUT is one of the alternatives which can be selected with the lightpen after pressing the PRINT function key.

3.2.2 Quickie Printout

A facility is provided for obtaining "rough draft" hard copies of a Hypertext, requiring only a minimum of "edit" formatting. The copy will be in ragged-right format, in a single column 74 spaces wide (i.e. screen width), double spaced, with no hyphenation. The only formattings used are PARAGRAPH, PARAGRAPH & CAP1, SKIP and INDENT (REGULAR), or precisely those formats which are displayed on the scope; consequently, the copy will correspond, line for line, with the text as presented on the scope display, except that extra blanks between words will be removed.[3]

The QUICKIE PRINTOUT may be chosen with the lightpen after pressing the PRINT function key. This option is useful for making edit changes on hard copy and then entering them into the scope copy, which will match the hard copy almost line for line.

---

3. TEXT360 always removes extra blanks between words-- if you want them kept, use the AS-IS format described in Sec. 4.1 below.

3.2.3 Branch Level Option

As was mentioned in Section 3.1, you may specify any branch level to be taken during printing. This is done by selecting the PRINT OPTIONS function key, lightpenning "SPECIFY BRANCH OPTION" and typing in the number of the desired branch level.

For example, if you type in a "4", the Printer will determine at each branch point if at least four options are available. If so, the fourth branch will be taken. If, however, fewer than four branches exist at that point, the first one will be followed. The default branch level is one.

4. "EDIT" FORMAT COMMANDS

"Edit" format types are basically those which one wishes to specify at many different points in the text. The Prompt Area of the display will tell you exactly what should be done next. The types of actions necessary were explained above in Sec. 2.1, Specification and Types of Formats. The options available, arranged alphabetically, are described in detail below. Each section heading refers to an "edit" function key label (name). See Appendix D for the correspondence of function key label and number.

4.1 AS-IS

The AS-IS function key is used when a string of text is to be printed as it appears in the Hypertext, without line justification and without removing or inserting any extra blanks between words. This format is specified by lightpenning the start of all pieces of text wished printed "as-is".

The AS-IS mode begins a new line and is terminated by the next formatting that starts a new line.[4] [It is not affected by the right margin.] If the length (i.e., the number of characters) of text under control of the AS-IS format exceeds the current column width (see Fig. 2), the text is truncated, and an appropriate error message is provided in the user reference material printed at the end of the hard copy.

A common use of the AS-IS format is to line up two columns of numbers. The AS-IS format is specified at the beginning of each line; if the numbers were typed in so that they were lined up, they would be printed out the same way.

For example, if the following table were formatted on the screen as shown (the #'s on the first line represent SKIP 1, INDENT 10, AS-IS, and two CAP1's, left-to-right; on the second line, SKIP 1, INDENT 10, and AS-IS; and on the last

---

4. The formats which start a new line are AS-IS, CENTER, HEADINGS, INDENT, NEW COL/PAGE, PARAGRAPH, PARAGRAPH & CAP1, SKIP and the "alter" formats.

four lines the #'s represent INDENT 10 and AS-IS; the blanks preceding the 0, the 1 and the 2 in the first column were inserted last, through the Edit Phase),

```
### #X        . #Y

### 0          0
## 1           1
##-1           1
## 2           4
##-2           4
```

the result on the printed copy would be:

```
 X          Y

 0          0
 1          1
-1          1
 2          4
-2          4
```

This same effect could also be achieved using the TAB option, explained below.

4.2 CAP1

This mode allows the user to capitalize a single letter. If three or more consecutive letters are to be capitalized, it is quicker to use MULT CAPS. The single capital format must immediately proceed the character to be capitalized.[5] This format is combined with a paragraph under the PARAGRAPH & CAP1 option, described below. If both a CAP1 and a BEGIN MULT CAPS are specified for the same character, the character in question will be capitalized only once.

---

5. The only exception to this rule is that the single capital can be followed by "BEGIN UNDERSCORE" and the next character will be capitalized.

4.3 CENTER

The CENTER format is used to center a particular piece of text between the current left and right margins (see Fig. 2). This format starts a new line. Termination of the text to be centered is indicated by the next format request (other than MULT CAPS, UNDERSCORE or CAP1). If the text being centered exceeds the column width, it will not be centered, i.e., the centering will be no-oped.

Aesthetical note: if there are blanks between the end of the line being centered and the terminating format symbol, the centering will be incorrect (i.e., blanks at the end are considered to be part of the piece being centered).

As an example, consider:

```
#   CENTER ME   #
```

The three blanks after the first format symbol (i.e. representing CENTER) will be ignored, but the two blanks between "ME" and the second # (the symbol for any terminating format) will be included in the centered line.

4.4 DRAW TABLE

This function key, in conjunction with DRAW LINES below, provides an effective method of drawing tables. Tables (or charts) are usually combinations of tabular text (see also Sec. 4.16 Tab (Execution) and Sec. 5.8 Tab Settings), horizontal and vertical lines, and their intersections (the proper intersection at the junction of a horizontal and vertical line is automatically provided).

The line drawing formats do not cause a new line to be started. When a new line is required it must be specified by the user. Care must also be taken to ensure that a text character does not occupy the same position as a line character. When a conflict exists between a text and a line character, the text character overrides the line character. The line characters will, however, override the blank characters of a text line. For example, if the text line "as illustrated below." is to be followed by a horizontal line and the user fails to request a new line, the result on the printed copy will be:

as-illustrated-below.---

The table formats require the specification of column positions. These positions are always measured from the leftmost position (position 1) of the column width.

The Begin Table format is chosen by lightpenning "BEGIN A TABLE" after pressing the DRAW TABLE function key. You will be asked to type in a series of numbers, separated by commas:

nn,nn,...,nn

This format will cause a horizontal line to be printed starting at the leftmost column position specified by the nn's and ending at the rightmost column position specified by the nn's. The horizontal line occupies the space equivalent to one text line. This format will also start a vertical line in each of the column positions specified. The column positions need not be entered in ascending order. The vertical lines started by this format continue until an End Table format (explained below) is encountered.

If, instead of typing in a series of numbers, the user immediately presses the accept key (ALTN CODING & END), this will cause the column positions specified in the last Begin Table format to be used. This "null" Begin Table format can be inserted next to a regular one, in which case the column positions of the regular one will be combined with those of the previous one. For example, if a Begin Table (10,20,30) is later followed by Begin Table (null) Begin Table (5,15, 40), the result of the last two is effectively Begin Table (5,10,15,20,30,40).

A subsequent Begin Table format can be specified while a previous one is still operative. This subsequent code ends all vertical lines of the first before starting the new table. In this fashion, a table of varying format can be constructed by using the Begin Table format rather than the horizontal and vertical line drawing formats described in the next section.

A table started with the Begin Table format should be ended using the End Table format activated by lightpenning "END A TABLE" after pressing the DRAW TABLE function key, and then lightpenning the end of the table. This format causes all vertical lines started by the Begin Table format to be ended and a horizontal line to be printed between the leftmost and rightmost vertical lines of the table.

4.5 DRAW LINES

In conjunction with the DRAW TABLE formats described in the previous section, these format options enable elaborate tables to be printed on the hard copy. This function key controls four options: "DRAW HORIZ LINE SEGS," "DRAW HORIZ LINE OF BEG TABLE," "DRAW VERTICAL LINES," and "END VERT LINES OF ABOVE."

If "DRAW HORIZ LINE SEGS" is chosen with the lightpen, the user will be asked to type in a series of numbers, separated by commas, designating column positions between which to draw horizontal line segments. The numbers are specified, as usual, as:

nn,nn,...,nn,nn

The first nn specifies the column position in which the first segment is to begin and the second nn indicates the column position in which it is to end, and similarly for the remaining pairs of nn's. The total number of column positions specified must be a multiple of two. If only nn,nn is specified, a single, continuous line will be drawn. After specifying the desired column positions, the user can apply this format to his text.

If "DRAW HORIZ LINE OF BEG TABLE" is chosen, and subsequently entered into the text with the lightpen, a horizontal line will be printed between the leftmost and rightmost column positions of the most recent begin table format. The intersections at the vertical lines are printed automatically by the program. The line occupies one text line or a portion thereof.

If "DRAW VERTICAL LINES" is chosen with the lightpen, the user will again be asked to type in a sequence of numbers, separated by commas, corresponding to column positions in which vertical lines are to be printed. As before, the numbers are specified in the form:

nn,nn,...,nn

and need not be specified in ascending order.

If, instead of typing in a series of numbers, the user immediately presses the accept key (ALT'N CODING & END), this "null" Vertical Lines format will cause the column positions specified by the previous Vertical Lines format to be used. The null Vertical Lines format could be inserted immediately

before a normal Vertical Lines format, in which case the column positions of the regular one will be combined with those of the previous one, analogous to the case of the Begin Table format.

And finally, if "END VERT LINES OF ABOVE" is chosen, and subsequently entered into the text at an appropriate point, all vertical lines started by the Vertical Lines format will be ended, in the line in which the format is specified. This format does not end vertical lines started by the Begin Table format; these must be ended by the End Table format.

4.6 FIGURES, LIST OF

This option provides the facility to have printed a List of Figures as a supplemental listing, similar to the Table of Contents. The List of Figures is automatically created if the user chooses to include at least one figure, and will be made up of the name of the figure (specified by pen-pointing at the start of the name after having pressed the LIST of FIGURES function key, and terminated by the next "edit" or "alter" format code other than MULT CAPS, CAP1 or UNDERSCORE) and its page number, derived by the TEXT360 program from its own output. As an example, see the List of Figures for this manual.

4.7 HEADINGS

This feature allows the user to format headings in a variety of ways and enables the construction of a fairly elaborate Table of Contents, automatically created by the system if the user desires. The heading types (specified by typing in a number), and their in-line appearance, are given in Table 1, below.

Heading types 1 through 4 are "stand-alone" headings, i.e., the text that follows them begins on a separate line. The start of the heading is pointed to with the lightpen and the heading is terminated by the next "edit" or "alter" format code other than capitals or underscore.

As examples, the heading 4.7 HEADINGS for this section was specified as a type 3 heading and was terminated by the first paragraph of this section. The headings for the main

sections of the manual, such as 4. "EDIT" FORMAT COMMANDS were specified as type 1 headings. An example of a type 4 heading is 3.2.1 FULL PRINTOUT. See the Table of Contents (TC) for examples of how each of these headings is included.

Headings 5 and 6 are "run-in" headings. The text that follows them continues on the same line. The format of headings 5 and 6 continues until a colon or period is reached.

The type 1 heading begins a new page, is right- or left-justified on the top of the page (right-justified on odd-numbered pages, left-justified on even-numbered pages), and is printed in the Table of Contents left-justified, preceded by 1 blank line. Heading type 1 is restricted to one line of text and therefore must not exceed the column width (twice the column width plus three if in two-column format). If it does, it will be truncated in the Table of Contents and in the main text.

Heading types 2-4 are also included in the Table of Contents, as follows: type 2 is included exactly like type 1; type 3 begins a new line and is indented 2 spaces; type 4 also begins a new line, but is indented 4 spaces. The Table of Contents will be printed in single-column format, at the end of the main text. Although the headings are automatically formatted in the actual body of text as indicated below, they must be formatted as you wish them printed in the Table of Contents (e.g., only initial capitals) by inserting the desired capitalization in the in-line heading.

Head Type	Stand-alone	All Cap	Under-score	Skips Before	Skips After	New Page	In TC
1	x	x	x	0	5	x	x
2	x	x	x	3	2		x
3	x	x		3	2		x
4	x		x	3	2		x
5		x	x	1	0		
6			x	1	0		

Table 1. Heading Sets and Their In-Line Appearance

4.8 INDENT

The INDENT mode provides two types of indentations. The user will be asked to make a choice, using the lightpen, between "REGULAR INDENTATION" and "HANGING INDENTATION". After making this choice, the user types in the number of spaces to indent, and finally (after hitting ALTN CODING & END) he lightpens the points which he wants indented.

A regular indentation begins a new line which is indented the user-specified number of spaces from the current left margin. A regular indentation of 0 spaces will begin a new line, left justified, with no intervening blank lines. A regular indentation is also displayed on the scope, indented up to a maximum of 40 spaces.

A hanging indentation begins a new line, but only the subsequent lines are indented the specified number of spaces (again with respect to the current left margin). This "hanging" indent mode continues until it is ended by the next format that begins a new line (see Footnote 4).

Both a regular and a hanging indentation can be specified together (only 1 new line is begun).

> For example, if a SKIP of 1, a REGULAR INDENT of 5 and a HANGING INDENT of 10 were specified at the same point in the text, as has been done here, the first line would be indented 5 spaces and all following lines would be indented 10 spaces.
>
> Conversely, if a SKIP of 1, a REGULAR INDENT of 10 and a HANGING INDENT of 5 were specified together, again as has been done here, the first line would be indented 10 spaces while all following lines would be indented only 5 spaces.

Restriction: the number of characters per line (i.e., the right margin minus the left margin, minus either of these indentations) should not be less than twenty. If this restriction is violated, the indentation will not be made for the complete number of spaces specified and a error message will appear at the end of the printed copy, as explained in Appendix C.

4.9 KEEPS

A keep is a reserved area (blank, or including text) of the printed copy which the user wants not to be split between columns or pages. This is used for instance to leave space for diagrams or pictures in the final printed copy. When this format is chosen (by pressing the KEEPS function key), the user will be presented with four choices: "BEGIN A REGULAR KEEP," "BEGIN A FLOATING KEEP," "BEGIN A TWO-COLUMN KEEP," and "END ANY OF THE KEEPS."

The facility to insert keeps, while extremely important, seems to present the most problems to the TEXT360 program--therefore extra care should be used when selecting them. It also appears that the floating keep does not perform as indicated. It is recommended that only the regular and two-column keeps be used.

The regular and floating keeps are similar. If there is room in the current column, the designated space (e.g., 20 lines if a skip of 20 is used) will be reserved beginning at the point selected. (The keep only declares the start of the reserved area, and the desired space must then be reserved, by using the SKIP format and/or by including actual text.) However, if there is not room in the current column to save the desired number of lines, the two keeps are handled differently. The regular keep will begin at the start of the next column (or page) and will leave the remainder of the previous column empty. The floating keep will also begin in a new column, but text following the keep will be used to complete (fill-up) the previous column.

Immediately preceding the keep designation, the user should specify, with the skip format, by how many lines his keep should be separated from the rest of the text. If no separator is desired, a skip of 0 should be specified. Also, a blank character (representing a dummy character string) should be inserted (through the Edit Phase) between the keep designation and the format specifying the length of the keep.

For example, to reserve 10 blank lines on the printed copy, separated from the main text by 1 line, the order of the formats would be: 1) SKIP 1 line; 2) BEGIN A KEEP; 3) insert at least one blank character; 4) SKIP 10 lines; and 5) END THE KEEP. Of course, the blank character must be entered last, while in the Edit Phase.

To reserve an area across the full page width in the two-column format, a two-column keep should be specified. The column width is temporarily altered to equal twice the current column width plus three (the width of the gutter, the separator between the two columns). The reserved area will be placed at either the top or the bottom of that section of text depending on which column contains the request for the keep (see Figure 3). If the keep format occurs in the left column of the printed section (usually a page), then the two-column keep will appear at the top of the section; if the format occurs in the right column, the keep will appear at the bottom of the section. Using the AS-IS format, the user can flow any text contained in the keep across the full width of the augmented column. This can also be done by increasing the right margin to the temporary width. When the keep is released (ended), the column width will automatically be reset to its previous value.

All of the above keeps must be released by lightpenning "END OF KEEP" after pressing the KEEPS function key, and then pointing with the lightpen at the last character of the desired "kept" area. None of these keeps should be longer than the specified number of lines per page (50 is assumed). To leave blank columns or pages, see NEW COL/PAGE below.

4.10 MULT CAPS

This mode enables the user to capitalize words or phrases of his Hypertext without having to capitalize each letter separately. He specifies the text to be capitalized by pen-pointing at the first and last letters of the desired piece, in any order. Any non-alphabetic characters included in this piece are not capitalized. If only one or two consecutive letters are to be capitalized, CAP1 is more efficient.

Caution: if a single capital (CAP1) or an "alter" format is specified within a portion of text under control of MULT CAPS (i.e., before the END MULT CAPS format is encountered), the multiple capitals mode will end at the point at which CAP1 or the "alter" format is specified.

4.11 NEW COL/PAGE

Two choices are available with this function key. The user will be asked to choose between "BEGIN NEW COLUMN" (a new column will be started) and "BEGIN NEW PAGE" (a new page will be started). These formatting options are fairly obvious.

If the user specifies "BEGIN NEW COLUMN" and his text is being printed in single-column format, a new page will be started.

It is advisable to specify a BEGIN NEW PAGE format at the start of a text, following the column and margin definition formats (if any) described in Sec. 5.4 TEXT COLUMNS and Sec. 5.5 MARGINS. This allows lines to be skipped at the start of a text and guarantees that your "text box" will not look strange (a result of having a default column width of 74 spaces).

In order to force blank columns and/or pages in the printed Hypertext, these formats must be specified in series with at least one blank (representing a dummy character string) separating each one (inserted through the Edit Phase). N successive occurrences of the formats will leave N-1 blank columns and/or pages. For example, assume each # represents "Begin New Column;" then the sequence

# # # #

would cause three blank columns to be printed since four new columns have been started.

4.12 PARAGRAPH

This option begins a new paragraph in the printed text, i.e., a blank line is skipped and the following text is indented three spaces. (See also Sec. 3.2.4, Paragraph Option.) The paragraph is displayed on the scope.

4.13 PARAGRAPH & CAP1

This option combines the PARAGRAPH and CAP1 functions described above. A new paragraph is begun and the first character of the paragraph will be capitalized. The paragraph is displayed on the scope.

This mode is automatically activated when the Format Phase is entered-- so the user may readily flip back and forth between the two Phases, paragraphing an unedited text for initial readability.

4.14 SKIP

This feature enables a user to skip a designated number of lines between sections of a Hypertext. The number typed in by the user is the number of lines to be left blank between the pieces of Hypertext. The skip is also displayed on the scope, although at most 4 lines will be skipped on the display.

To begin a new line, with no blank lines between (i.e., to do a "Carriage Return"), a skip of zero should be specified. If a skip is immediately followed by another format that begins a new line (see Footnote 4), an additional line is not skipped. For example, a skip of 2 lines followed by an indentation of 5 spaces will cause only 2 lines to be left blank on the printed copy.

4.15 SPEC CHARS

A choice of special characters not available on the standard 2250 keyboard is offered to the user. He will be asked to choose between "LEFT BRACKET" ([), "RIGHT BRACKET" (]), "PLUS OR MINUS" (±), and "SUPERSCRIPT" (e.g. ^{0} or ^{5}). The possible digits that can be superscripted are 0 through 9. However, these can be combined in series to form any positive decimal number desired, e.g. $[X \pm Y]^{52}$.

When using a superscript to indicate the position in the

main text of a footnote,[6] the superscript should be entered so as to immediately precede the link marker (an * on the display) so that no blanks will appear between the text and the footnote marker. You, the user, must put the same number on the footnote (link) itself.

4.16 TAB (EXECUTION)

The TAB feature allows text to be positioned at (tabbed to) pre-set positions on a line. The tab stops can be set at any column positions desired, as explained in Sec. 5.8, Tab Settings. As default values, the tab settings are at five space intervals across the line. That is, tab position 1 is at space 5, position 2 at space 10, etc., through position 20 at space 100. Of course, if the current column width is less than 100, only those settings corresponding to spaces less than or equal to the column width are applicable. The tab positions are in relation to the column width and are not restricted by the current left or right margins (see Fig. 2).

Since the user has the option of printing leaders (periods) between the tabbed portions of text, he must choose, with the lightpen (from a menu in the Prompt Area of the display), between "REGULAR TAB" (i.e., with blanks between tabbed sections) and "TAB WITH LEADERS" (i.e., with periods between tabbed sections). These two types of tabs can be varied on the same line. After choosing, with the lightpen, which of the tabs he wants, the user will type in the number corresponding to the position to which he wishes to tab. He may then lightpen the starts of pieces of text which are to begin (or possibly end, again, see Tab Settings) at that position.

Each line of tabular text must be preceded by an AS-IS format, one "as-is" per line. Since the text is under control of the AS-IS format, the end of the last tabbed fragment should not overflow the column width. If it does, the text will be truncated, and an appropriate error message will be included in the user reference material printed at the end of the hard copy.

---

6. As has been done in this manual; for example the superscripted 6 to indicate this footnote.

In addition, the user may specify that tabular text and normal (flowing) text are to be printed on the same line, the normal text possibly continuing onto the next line. This has the effect of nullifying the as-is format. This is designated by lightpenning "END TABULAR TEXT," one of the options of the TAB function key, and then lightpenning the start of the normal text. No other formatting is necessary to end the tabular mode.

As an example of the tabbing operation, consider the same example given in Sec. 4.1, AS-IS. Here the format symbols have different meanings: in the first line, the $ represents the user's choice of SETTAB at 1=10,2=11,3=22, and the six #'s represent SKIP 1, AS-IS, TAB 2, CAP1, TAB 3, and CAP1, respectively; in the second line, they represent SKIP 1, AS-IS, TAB 2, and TAB 3; in the third and fifth lines, AS-IS, TAB 2 and TAB 3; and in the fourth and sixth lines, AS-IS, TAB 1 and TAB 3.

```
$#####X##Y
##0#0
##1#1
##-1#1
##2#4
##-2#4
```

Having done this it would hopefully look something like this in your printout:

X	Y
0	0
1	1
-1	1
2	4
-2	4

4.17 UNDERSCORE

A means of underlining particular words or phrases of a Hypertext is provided, requiring pairs of lightpen hits (in any order), analogous to the method used in MULT CAPS. If this mode is ended after an odd number of lightpen hits, the last hit is ignored.

5. "ALTER" FORMAT COMMANDS

As explained in General Remarks (Part 2), the "alter" formats are selected with the lightpen from a menu which appears after hitting the ALTER function key and lightpenning the desired point in the text. They are explained here in the order in which they appear in the menu, opposite choices grouped together via an indent. To return to regular formatting, with all previously selected options applied, lightpen "RETURN TO NORMAL FORMATTING OPERATIONS," the first choice. Remember that due to default options, you need not specify a single option, i.e., you need not go into this mode at all.

5.1 LINE AND COLUMN JUSTIFICATION

The next four choices allow you to control line and column justification as you desire.

Line Justification

To accomplish line justification, all text lines are spaced out to the line length by inserting blank characters between words. As many blanks as are necessary to justify the line are inserted. In an attempt to distribute white space equally across the column, an equal number of blanks are inserted between pairs of words in the line. If this is not enough to justify the line, one additional blank is inserted between each word until the justification is accomplished, alternately beginning from the left and then from the right for each successive line (see also Hyphenation, Sec. 5.2 below).

Lines preceding a format that begins a new line, column or page are not justified. Also, a line consisting of one word is left-justified.

The line justification feature is assumed "on" as the default case and need be specified only after having previously specified "SUPPRESS LINE JUSTIFICATION."

Column Justification

When the column justification feature is "on" (the assumed option), all the text columns under control of a given "lines per page" format (see below, Sec. 5.3, Page Depth) are made equal in length (i.e., "vertical" justification is performed) by spacing the columns, with blank lines, out to the page length.

Column justification can be "turned off" by lightpenning "SUPPRESS COLUMN JUSTIFICATION."

5.2 HYPHENATION

This feature allows the last word of a line to be hyphenated if necessary. Words of less than six characters are not hyphenated. Since a word can be hyphenated incorrectly (as was done here!), there is a facility for the creation of your own hyphenation dictionary which will be checked before attempting any hyphenation.

When line justification and hyphenation are both operative (they don't have to be specified together), line justification is attempted first. This line justification differs from that described above (Sec. 5.1, Line and Column Justification) in that at most one blank is inserted between consecutive words. If this fails to justifiy the line, hyphenation is attempted. It is possible that a word cannot be hyphenated, in which case the word is moved to the next line and the original line is justified by inserting additional blanks according to the normal formula.

The hyphenation feature is assumed to be "on" and need be specified only if you have previously specified "SUPPRESS HYPHENATION."

The hyphenation dictionary is constructed using the next two features:

To add words to the hyphenation dictionary, choose "INCLUDE WORDS IN THE HYPHENATION DICT." from the Alter Formats Menu, and then type in the correctly hyphenated words, separating each by a comma. For example, if HYPHENATION and HYPERTEXT had been incorrectly hyphenated in the hard copy, you would type in:

HY-PHE-NA-TION,HY-PER-TEXT

A maximum of twenty words may be active (included) in the dictionary at one point in the Hypertext. Words are accumulated when entered via this format. The words specified must be at least six, but not more than twenty-four, characters in length, not including hyphens.

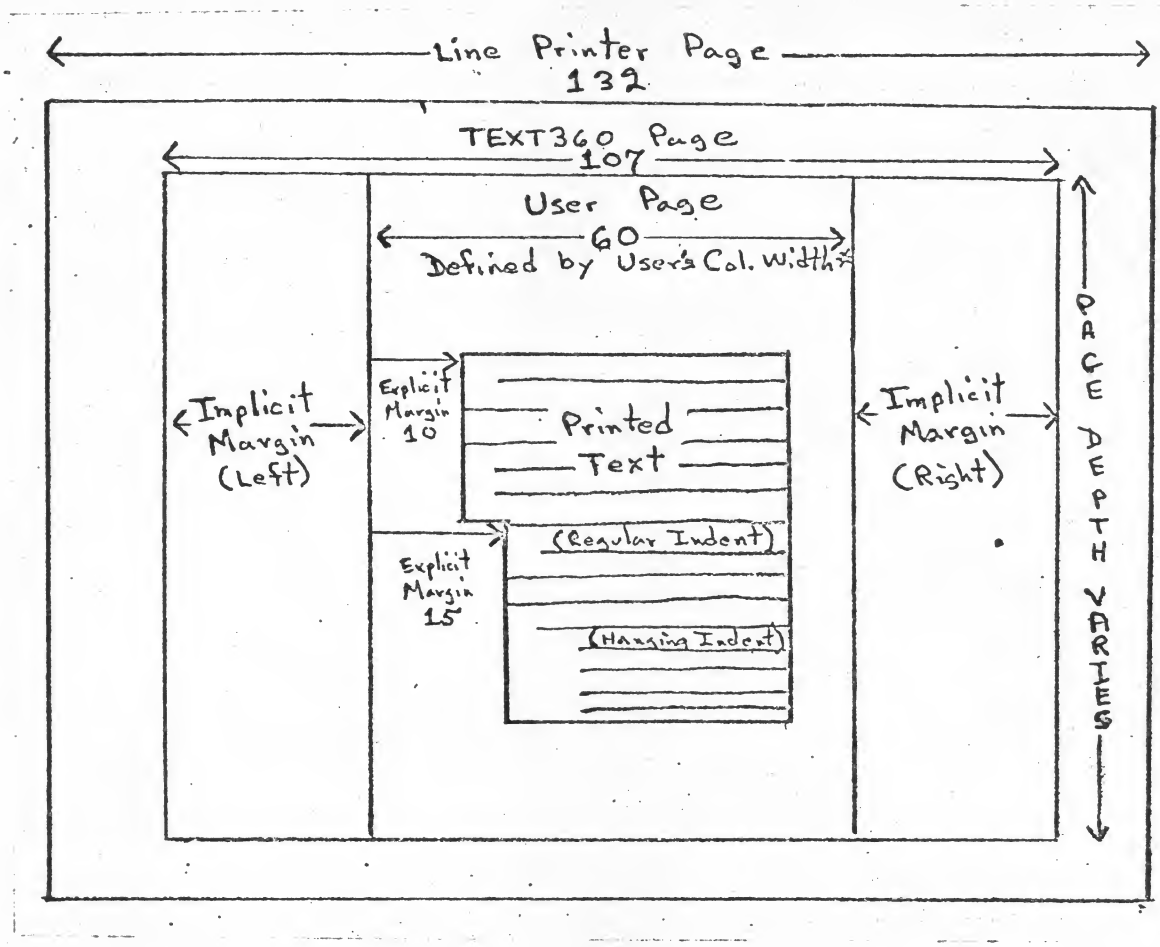
To clear out all words previously specified from the dictionary and enter new ones, choose "CLEAR OUT OLD HYPHENATION DICT. AND ADD NEW WORDS." The old words currently active in the dictionary can only be seen by interrogating previous format markers used to enter them. All the conventions and restrictions mentioned above are the same with regard to any new words typed in. It is important to remember that this option will clear out all words in effect at that point, while the first option only adds new words.

5.3 PAGE DEPTH

The page depth is the number of lines, including blank lines, that can appear in a text column. The number of lines may vary from 25 lines through 75. The assumed number is 50, the value used for this manual. The page depth may be changed by lightpenning "SPECIFY NUMBER OF LINES PER PAGE," and typing in the number of lines you wish printed per page. Different page depths can be specified throughout the text. Only the text column is affected by this specification. Any running heads and foots are adjusted to the page depth.

5.4 TEXT COLUMNS

The page margins of a printed copy are determined by several factors. Figure 2 shows a typical breakdown of the various elements involved.



*i.e. Col. Width for single column;
2 x Col. Width + 3 for double column.

Figure 2. Column Width, Margins and Indentations

The "Line Printer" Page (the outer box) is 132 spaces wide. Of course if narrow paper (say with a width of only 86 positions) is used, all 132 positions would not be printed. The TEXT360 Page (107 spaces wide) is the maximum user column width allowed. Within this box, the user defines his own column width. The implicit margins will be the space left between the User Page and the TEXT360 Page. These margins can be varied by moving the paper on the printer to the left or right of center (of the carriage). The User Page is automatically centered in the TEXT360 Page.

All formatting is done within the User Page, with respect to the left-most position (position 0) of the user's column. The explicit (i.e. settable) margins are described in detail in the following section (5.5 MARGINS).

Text can be printed in one or two columns. A particular column format does not have to remain in effect throughout the entire text. It may be respecified as originally entered, respecified with a new width, or alternated with the inoperative format (e.g., from a one-column to a two-column format). This does not cause a new page to be started. When a change is encountered, the preceding text is formatted as a "short" page, and the rest of the page is formatted as specified.

For example, if text is being printed in two-column format and a column format is encountered, the page depth is recalculated, the text is divided between the columns, and if specified, column justification is performed. Then two lines are skipped and the remainder of the page is formatted as specified by the column format. This effect is illustrated in Figure 3. If there is not room left for at least five lines (per column), a new page will be started.

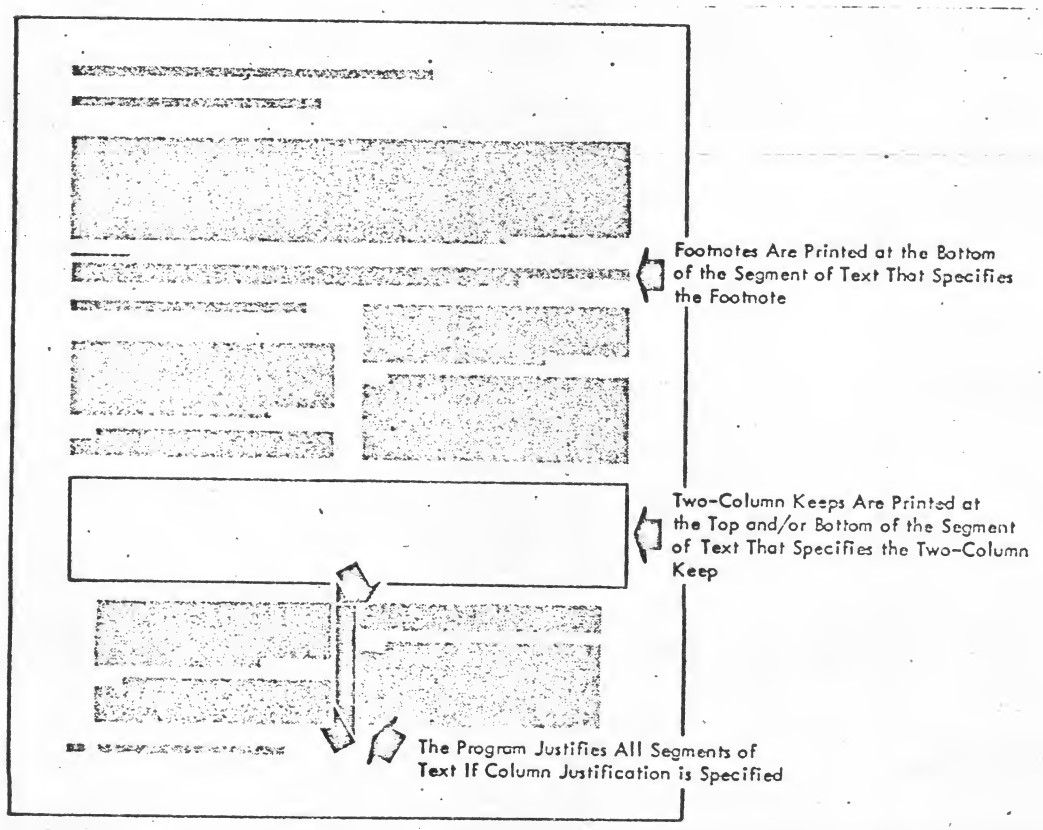


Figure 3. Column Format Change

A single column is specified by lightpenning "PRINT TEXT IN ONE-COL. FORMAT AND SPECIFY WIDTH" and typing in a number corresponding to the number of characters per line (the width). Single column with a width of 74 characters is the assumed format, but the width may vary from 20 characters through 107.

The double column format is specified by lightpenning "PRINT TEXT IN TWO-COL. FORMAT AND SPECIFY WIDTH" and then typing in the desired width (the width of one of the two, equal columns), which may vary from 20 characters through 52.

5.5 MARGINS

An (explicit) margin (see Sec. 5.4, Text Columns) is obtained by temporarily decreasing the column width. The left and right margins are in relation to the left-most position of the column width (position 0). (See Figure 2.)

The margins remain in effect until respecified, and therefore must be reset if no longer required.

The left margin is specified by lightpenning "SPECIFY LEFT MARGIN" and typing in a number. If "10" were typed in, a left margin of ten spaces would be left for each line. Initially, the left margin is set to zero, and consequently there is no margin. The left margin may be set to different values in a text and must be reset to zero when no longer desired. The left margin must not exceed 87 spaces and, finally, the number of characters per line (i.e., the right margin minus the left margin) must not be less than 20.

The right margin is specified by lightpenning "SPECIFY RIGHT MARGIN" and typing in a number, this number indicating the column position in which the line must end. The column position is measured from the left-most position of the column width, not from the current left margin setting. Thus the length of the actual (explicit) right margin is equal to the length of the column width minus the number specified as the right margin.

As an example, assume that the column width is set to 60 and the right margin to 55, as has been done here. Then a right margin of five spaces is left to the right of each line.

If, in addition, the left margin is set to 10

(again as has been done here), only 45 characters are printed per line.

The right margin is initially set to 74, the value of the default single-column width, thus leaving no right margin. The right margin specified should not be greater than the column width. If it is, it will be reset to the column width, and an error message will be printed in the user reference material included at the end of the printed copy.

Text under control of an AS-IS format is printed to the full column width and not to the setting of the left and right margins.

5.6 TEXT-LINE SPACING

Text can be printed either single- or double-spaced. Single spacing causes the text to be printed with no extra blank space between consecutive lines. The double spaced format causes a blank line to be inserted between each pair of consecutive non-skip lines. Only one extra blank line is added to any set of skip lines.

Text is assumed to be single-spaced, therefore "SINGLE SPACE TEXT" need be specified only to return from the double-spacing mode.

Double-spacing is specified by lightpenning "DOUBLE SPACE TEXT." The extra blank lines that are added are counted when computing the number of lines per page. Thus, double-spacing reduces the amount of text on a page to approximately one-half the amount on a single-spaced page. Note that a Quickie Printout will always be double-spaced.

5.7 RUNNING HEADS AND FOOTS

A facility is provided for including running heads (title, subtitle, and date) and running foots on each printed page of the hard copy. The running heads are printed at the top of each page and are aligned with the greatest column width specified on the page.

A title (e.g. HYPERTEXT EDITING SYSTEM above) is specified by lightpenning "SPECIFY A TITLE FOR TOP OF EACH PAGE"

and typing in the desired text to be used as the title. The title is printed in upper case, with six intervening lines between it and the first line of the text page. It is normally right adjusted on odd-numbered pages and left adjusted on even-numbered pages.

A subtitle (e.g. "ALTER" FORMAT COMMANDS above) is specified by lightpenning "SPECIFY A SUBTITLE FOR TOP OF EACH PAGE" and typing in the desired subtitle. The subtitle is printed in upper case, with three blank lines between it and the first line of the text body, and is aligned with the title.

The date (e.g. the date printed above) is printed at the top of each page, on the same line as the subtitle, if "PRINT DATE AT TOP OF EACH PAGE" is lightpenned. The current date is always printed, on the same line as the subtitle, left-justified on odd-numbered pages and right-justified on even-numbered pages. If the user desires to specify his own date, he could do so by entering it as a subtitle.

A running foot (e.g. FORMAT MANUAL at the bottom of each page), which is printed at the bottom of the page, can be specified for the odd-numbered pages, the even-numbered pages, or both. The foot is printed in upper case with two blank lines between it and the last line of the page. On even-numbered pages, the foot is printed two spaces to the right of the page number, while on odd-numbered pages it is two spaces to the left. (The page number is also aligned to the greatest column width specified on the page.)

To specify a running foot to be printed at the bottom of all even-numbered (left-hand) pages, lightpen "SPECIFY A RUNNING FOOT FOR EVEN-NUMBERED PAGES" and type in the desired "even" foot. This feature has not been used in this manual.

To specify a running foot for the odd-numbered (right-hand) pages, lightpen "SPECIFY A RUNNING FOOT FOR ODD-NUMBERED PAGES" and type in the desired "odd" foot.

The reason that the running heads and foots of this manual are not formatted differently for odd- and even-numbered pages is that at the start of the manual the "Right-Hand Pages" format, described below in Sec. 5.10, was specified. As a result, the "odd" foot FORMAT MANUAL is printed at the bottom of every page.

The maximum length of the heads and foots is determined by the column width. If the page is formatted in one column, the maximum length equals the width; if it is formatted in two columns, the maximum length is twice the column width plus three. It should be noted, however, that the maximum length is computed on a per page basis; if various column widths are used, the maximum length fluctuates accordingly, and the smallest column width used within a text should be considered when determining the length of a running head or foot.

5.8 TAB SETTINGS

In conjunction with the TAB (Execution) format described in Part 4, a facility is provided for specifying tab settings. Tabs are set by lightpenning "SPECIFY TAB SETTINGS" and typing in the tab settings in the form:

n=nn,n=nn,...,n=nnR

where "n" indicates the tab number and "nn" or "nnR" indicates the corresponding column position in which the text is to begin or end. "nn" indicates the column position in which text is to begin while "nnR" indicates the position in which text is to end (right-adjusted). The two types of tab stops can be in any order. Leading zeros can be omitted from the column positions. Tab settings are permanently entered into the text until changed.

As default tab settings, tab stops are at five-position intervals from column position 5 to 100 (i.e., 1=5,2=10,3=15,...,20=100). When you specify tab settings, the specified settings override the default settings; however, only the default settings are overridden. For example, if

1=3,2=9,3=38R

were specified, the default settings for number 4 through 20 are still operative. If subsequently

1=4

is specified, tabs 2 and 3 are still set at positions 9 (left-justed) and 38 (right-justed) respectively. When a SET TAB format marker is interrogated, only the tab settings specified for that format will be described.

The tab number must be a value from 1 through 20 and the corresponding column positions must be a value from 1 through 107.

5.9 PAGE NUMBERING

Pages will normally be numbered sequentially beginning with page one. However, a facility is provided for specifying another number with which to begin numbering or to change the numbering at any point in the text. This option is chosen by lightpenning "SPECIFY PAGE NUMBER FROM WHICH TO BEGIN (RE)NUMBERING" and typing in the page number (a value between 1 and 255).

This facility may be used to "link" different Hypertexts together, by specifying that the page numbering of the second text begin where the first leaves off, and similarly for any others.

If 5 were specified at the start of the text, the first page will be numbered 5. If later in the text, at hard copy page 25, 56 were specified, pages would be numbered

...23,24,56,57,...

The value specified must be a number that is equal to or greater than the number of the page being processed; otherwise, the format is ignored. Renumbering is generally not used until after a printout has been obtained so that you would know what page number you want.

5.10 RIGHT-HAND PAGES

As a default option, printed pages are formatted as odd-numbered (right-hand) pages and their even-numbered (left-hand) backup pages, as in a bound book. To print all pages as right-hand pages (as has been done in this manual) lightpen "PRINT ALL PAGES AS RIGHT-HAND PAGES." This would normally be specified at the start of a text.

When this option is specified, the running heads and foots and the type 1 headings will all be formatted as if on odd-numbered pages. In addition, any running foot specified for even-numbered pages will be ignored, while an odd-page running foot will be printed on every page.

6. APPENDICES

APPENDIX A: SAMPLE TEXT360 PROGRAM

Col. 72

```
//TEXT360 JOB (1518,DEB,10,5,,T,3),'SAMPLE JCL'
//JOBLIB DD DSNAME=SYS1.T360,DISP=OLD
//STEP1   EXEC  PGM=PMAINT
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=VA,LRECL=137,BLKSIZE=141)
//INTER   DD DSNAME=&WORK1,DISP=(,PASS),SPACE=(CYL,(15,15)),          X
//              DCB=(BLKSIZE=2400,LRECL=600,RECFM=FB),UNIT=SYSDA
//EXCEPTN  DD  DSNAME=&EXCFILE,DCB=(,BLKSIZE=300),                    X
//              SPACE=(TRK,(100,20)),DISP=(NEW,PASS),                 X
//              UNIT=(SYSDA,SEP=INTER)
//SYSIN DD DSNAME=SYS1.P1518.AVD.T360IN,DISP=OLD,UNIT=2314,           X
//              VOLUME=SER=AVDPAK,                                    X
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600)
//STEP2  EXEC  PGM=BLDLIN
//SYSPRINT DD  SYSOUT=A,DCB=(RECFM=VA,LRECL=137,BLKSIZE=141)
//INTER   DD  DSNAME=*.STEP1.INTER,UNIT=SYSDA,                        X
//              DISP=(OLD,DELETE),VOLUME=REF=*.STEP=.INTER,           X
//              DCB=(BLKSIZE=2400,LRECL=600,RECFM=FB)
//EXCEPTN DD DSNAME=*.STEP1.EXCEPTN,DISP=(MOD,PASS),UNIT=2314,        X
//              DCB=(BLKSIZE=300),VOLUME=REF=*.STEP1.EXCEPTN
//LAYFILE  DD  DSNAME=&WORK2,UNIT=(SYSDA,SEP=(INTER,EXCEPTN)),        X
//              DCB=(,BLKSIZE=1056,RECFM=FB,LRECL=176),               X
//              SPACE=(CYL,(20,20)),DISP=(NEW,PASS)
//STEP3  EXEC  PGM=PLAOUT
//SYSPRINT DD  SYSOUT=A,DCB=(RECFM=VA,LRECL=137,BLKSIZE=141)
//LAYFILE  DD DSNAME=*.STEP2.LAYFILE,DISP=(OLD,DELETE),UNIT=SYSDA,    X
//              DCB=(BLKSIZE=1056,RECFM=FB,LRECL=176),                X
//              VOLUME=REF=*.STEP2.LAYFILE
//EXCEPTN DD DSNAME=*.STEP1.EXCEPTN,DISP=(MOD,PASS),UNIT=SYSDA,       X
//              DCB=(BLKSIZE=300),VOLUME=REF=*.STEP1.EXCEPTN
//MASTER  DD  DSNAME=&MASFILE,UNIT=(SYSDA,SEP=(LAYFILE,EXCEPTN)),     X
//              DISP=(,PASS),SPACE=(CYL,(20,20)),                     X
//              DCB=(BLKSIZE=132)
//PRTOF DD DSNAME=SYS1.P1518.AVD.PRTFILE,UNIT=2314,DISP=(OLD,PASS),X
```

```
//              DCB=(RECFM=FB,LRECL=132,BLKSIZE=132),                    X
//              SPACE=(TRK,(100,10)),VOLUME=SER=AVDPAK
//STEP4  EXEC  PGM=POSTPR
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=VA,LRECL=137,BLKSIZE=141)
//PRTOT DD DSNAME=*.STEP3.PRTOT,DISP=(MOD,PASS),UNIT=SYSDA
//EXCEPTN DD DSNAME=*.STEP1.EXCEPTN,VOLUME=REF=*.STEP=.EXCEPTN,        X
//              DISP=(OLD,DELETE),UNIT=SYSDA,DCB=(BLKSIZE=300)
//MASTER  DD  DSNAME=*.STEP3.MASTER,VOLUME=REF=*.STEP3.MASTER,         X
//              DISP=(OLD,DELETE),DCB=(BLKSIZE=132),UNIT=SYSDA
//DICT    DD   DSNAME=SYS1.DICT,DISP=OLD,DCB=(,BLKSIZE=2000)
//T360PNT  DD  SYSOUT=A,DCB=(,LRECL=132,BLKSIZE=132)
//STEP5  EXEC  PGM=PRNTPGM
//T360PT DD DSNAME=*.STEP3.PRTOT,DISP=(OLD,KEEP),UNIT=SYSDA
//T360PNT  DD  SYSOUT=A,DCB=(,LRECL=132,BLKSIZE=132)
/*
```

The DSNAME for SYSIN (here SYS1.P1518.AVD.T360IN) must correspond to the DSNAME for the DD card OUT of the Hypertext Editing System program, described in the Appendices of the Edit Phase Manual.

APPENDIX C: TEXT360 ERROR MESSAGES

At the end of a printed document, the TEXT360 program includes a section called ALTER CODES, EXPANDED EDIT CODES, FOOTNOTES, AND ERROR MESSAGES. This listing can contain four distinct types of entries. When applicable, it contains every "alter" code specified and its location in the text, every footnote line and its location, every line that has only a partial representation in the Edit Code Field of the printed text (see Fig. 5), and any error messages that are issued by TEXT360. The error messages are the only entries which are in general of interest to the Hypertext user.

(In addition to the footnote lines, the program also lists the text line that contains the footnote. If the footnote is embedded within a text line, the portion that precedes and the portion that follows the footnote are assigned a line number.

Any "edit" code that is partially represented in or overflows the Edit Code Field of the printout causes an entry in this listing. The program lists the complete "edit" code and the text line in which it appears.)

For the quick recognition of error messages, the error messages are set off from the remaining classes of entries. The error message informs the user of the type of error, the corrective action (if any) taken by the program, and the location at which the error occurred. The text line in error is printed following the error message. The TEXT360 error messages are listed below. From the given page and line number, and the type of error detected, you should be able to discover your mistake. If not, call us. The correction should then be made in your Hypertext.

APPENDIX F: TEXT360 ERROR MESSAGES

The TEXT360 program can detect certain error conditions within the input text. If an error is detected, the program issues the appropriate error message in the error message listing (see "User Reference Material"). The program lists the error message number, the error message, and in the line immediately following the error message, the program lists the page and line number at which the error occurred and the erroneous text.

This appendix lists, in ascending order, the error messages that can be printed by the program. Each error message is accompanied by a brief explanation of the error message and the action the program takes. The program action affects only the print tape; the erroneous data, as specified in the input text, is written on the master tape, and it must be corrected by the user.

T360.001 NO DELIMITER FOLLOWING INSERT CODE

Explanation: Terminal + delimiter is missing..

Program Action: The program ignores the insert code.

T360.002 UPDATE CODE OUT OF SEQUENCE

Explanation: The page and line number of the code is less than that of the one that precedes it.

Program Action: The program ignores the code that is out of sequence by skipping to the next code.

T360.003 WORD REPLACE EXCEEDS 20 CHARACTERS

Explanation: The "old" portion of the +pppllRold+new code exceeds 20 characters.

Program Action: The program ignores the complete replace code.

T360.004 WORD REPLACE NOT FOUND AT SPECIFIED LOCATION

Explanation: The "old" portion of the +pppllRold+new code cannot be located at the page and line specified.

Program Action: The program ignores the code.

T360.005 COPY SPECIFIED AS AN UPDATE TO COPIED TEXT

Explanation: This message is issued when a request to copy text into text that has been copied is received. This type of copy must be specified as three separate copies.

Program Action: The program ignores the nested copy code.

T360.006 INVALID COPY CODE OR COPYOFF SYNTAX

Explanation: The +pppl11C or +COPYOFF code is specified incorrectly.

Program Action: The program ignores the code.

T360.007 UPDATE CODE HAS IMPROPER CLOSING DELIMITER

Explanation: The update code is specified incorrectly.

Program Action: The program ignores the update code.

T360.008 SPECIFIED PAGE-LINE NUMBER NOT ON OLD MASTER TAPE

Explanation: The page and line specified for the update code is not on the master tape.

Program Action: The program ignores the update code.

T360.009 INVALID ENCODED CHARACTER

Explanation: The input contains a character not defined in the TEXT360 character set.

Program Action: The program ignores the invalid character.

T360.010 TEXT360 CANNOT REPOSITION TO START OF COPY

Explanation: The TEXT360 program cannot reposition the old master tape to the location of processing before the copy was initiated.

Program Action: The program continues to process from the copied location.

T360.011 ALTER CODE PROCESSING OUT OF SYNCHRONIZATION

Explanation: The program, as the result of improper input, is processing out of synchronization (i.e., text as codes and codes as text).

Program Action: The program can determine after a certain amount of text has been processed that it is out of synchronization and corrects itself.

T360.012 REMAINING INPUT NOT PROCESSED BECAUSE OF ERRONEOUS UPDATE SPECIFICATION

Explanation: The update to the master file is specified in such a way that the program cannot process it.

Program Action: The program prints the first card of the flushed input immediately following the error message.

T360.013 TOO MANY SEARCH AND/OR REVISE REQUESTS; REQUEST IGNORED

Explanation: When the SEARCH or REVISE code was encountered, 60 such codes were operative.

Program Action: The program ignores the code.

T360.014 INVALID SEARCH OR REVISE REQUEST FORMAT

Explanation: The SEARCH or REVISE code is either specified incorrectly or the character count exceeds the maximum of 20.

Program Action: The program ignores the code.

T360.015 SEARCHOFF OR REVISEOFF REQUEST NOT FOUND

Explanation: The SEARCHOFF or REVISEOFF code is not specified exactly the same as the code it is to end.

Program Action: The program ignores the code.

T360.101 INVALID n VALUE SPECIFIED FOR -Hn- CODE; -I0- ASSUMED

Explanation: The -Hn- code is specified as -H0- or -Hn- where n is greater than 6.

Program Action: The invalid -Hn- code is accepted by the program; however, it is processed as a -I0- code.

T360.102 n NOT SPECIFIED FOR -Hn- CODE; -I0- ASSUMED

Explanation: The -H- edit code is an invalid code. The code must be specified with a numeric character from 1 to 6.

Program Action: The invalid -H- code is accepted by the program; however, it is processed as a -I0- code.

T360.103 n NOT SPECIFIED FOR -?n- CODE; -I0- ASSUMED

Explanation: The -?- is an invalid code. The code must be specified with a numeric character from 1 to 15.

Program Action: The invalid -?- code is accepted by the program; however, it is processed as a -I0- code.

T360.104 INVALID n SPECIFIED FOR -?n- CODE; -I0- ASSUMED

Explanation: The -?n- code is specified of as -?0- or -?n- where n is greater than 15.

Program Action: The invalid -?n- code is accepted by the program; however, it is processed as a -I0- code.

T360.105 x NOT SPECIFIED FOR -Xx- CODE; NO EXCEPTION GENERATED

Explanation: The x value of the -Xx- code is not specified.

Program Action: The invalid -X- code is accepted by the program; however, the program does not assign an exception value to the text line.

T360.106 -Snn- EXCEEDS PAGE DEPTH; CODE SET TO -S1-

Explanation: The skip specified by the -Snn- code is greater than the page depth.

Program Action: The program sets the -Snn- code to -S1-.

T360.107 n VALUE GREATER THAN 20 SPECIFIED IN -Tn- CODE; CODE SET TO -T1-

Explanation: The n value specified in the -Tn- code is greater than 20.

Program Action: The program sets the -Tn- code to -T1-.

T360.120 INVALID ALTER CODE

Explanation: The alphabetic portion of the alter code does not represent a valid code.

Program Action: The alter code is accepted by the program; however, the only action initiated by the code is to start a new line. The code is assigned a page and line number and is printed in the alter code listing.

T360.121 NO NUMERIC VALUE FOLLOWING ALTER CODE; MAXIMUM ASSUMED

Explanation: The numeric value of the alter code is not specified.

Program Action: The program assumes the maximum parameter of the code.

T360.123 COLUMN WIDTH GREATER THAN MAXIMUM; SET TO MAXIMUM

Explanation: The number of characters specified for the column width exceeds the maximum number that can be specified.

Program Action: The program sets the column width equal to the maximum number of characters that can be specified.

T360.124 COLUMN WIDTH LESS THAN MINIMUM; SET TO 20

Explanation: The number of characters specified for the column width is less than 20.

Program Action: The program sets the column width to 20.

T360.125 MARGIN EXCEEDS MAXIMUM, SET TO MAXIMUM

Explanation: The margin specified by the +MARGIN code exceeds the maximum that can be specified.

Program Action: The program sets the margin to the maximum.

T360.126 LINE LENGTH GREATER THAN COLUMN WIDTH; SET TO COLUMN WIDTH

Explanation: The number of characters specified for the line length exceeds the number of characters specified for the column width.

Program Action: The program sets the line length equal to the column width.

T360.127 LINE LENGTH LESS THAN MINIMUM; SET TO 20

Explanation: The number of characters specified for the line length is less than 20.

Program Action: The program sets the line length to 20.

T360.128 PAGE NUMBER SPECIFIED AFTER PAGE PROCESSED; CODE IGNORED

Explanation: The +PAGEnn+ or +REPAGEnn+ code is entered after the page specified by the code has been processed.

Program Action: The program accepts the code, but it is not processed.

T360.129 CLOSING DELIMITER NOT SPECIFIED IN ALTER CODE, DELIMITER ASSUMED

Explanation: The alter code is entered with an opening delimiter but the closing delimiter is not specified.

Program Action: The program assumes the presence of a +.

T360.130 n NOT SPECIFIED FOR Jn CODE WITHIN +DEFINE; J IGNORED

Explanation: The n value of the Jn code within the DEFINE code is not specified.

Program Action: The erroneous portion of the DEFINE code is ignored by the program; the remainder of the code is processed.

T360.131 HANGING INDENTION VALUE WITHIN +DEFINE CODE TOO LARGE

Explanation: The line length minus the margin and hanging indention is less than 20.

Program Action: The program assumes a hanging indention of zero. The zero is assumed only at the location at which the error message occurred.

T360.132 INDENTION VALUE WITHIN +DEFINE CODE TOO LARGE

Explanation: The line length minus the margin and indention is less than 20.

Program Action: The program assumes an indention value of zero. The zero is assumed only at the location at which the error occurred.

T360.133 n NOT SPECIFIED FOR I CODE WITHIN +DEFINE; I IGNORED

Explanation: The n value of the In code within the +DEFINE code is not specified.

Program Action: The erroneous portion of the +DEFINE code is ignored by the program; the remainder of the code is processed.

T360.134 n NOT SPECIFIED FOR S CODE WITHIN +DEFINE; S IGNORED

Explanation: The n value of the Sn code within the +DEFINE code is not specified.

Program Action: The erroneous portion of the DEFINE code is ignored by the program; the remainder of the code is processed.

T360.135 INVALID ENTRY WITHIN +DEFINE CODE; INVALID ENTRY IGNORED

Explanation: An unacceptable code is specified within the +DEFINE code.

Program Action: The invalid portion of the DEFINE code is ignored by the program; the remainder of the code is processed.

T360.136 = ASSUMED IN +DEFINE CODE

Explanation: The = is not specified within the DEFINE code.

Program Action: The program assumes the presence of an =.

T360.137 ? NOT FOUND IN +DEFINE CODE; CODE IGNORED

Explanation: The ? is not specified within the DEFINE code.

Program Action: The program ignores the code.

T360.138 NO NUMERIC VALUE FOLLOWING ? IN +DEFINE CODE

Explanation: The numeric value that must follow the ? in the DEFINE code is not specified.

Program Action: The program accepts the code; however, it does not initiate any action. The code is assigned a page and line number and is printed in the alter code listing.

T360.139 INVALID NUMBER FOLLOWING ? IN +DEFINE CODE; CODE IGNORED

Explanation: The ? in the DEFINE code is followed by a 0 or a number greater than 15.

Program Action: The program ignores the code.

T360.140 INVALID TAB NUMBER SPECIFIED IN +SETTAB CODE

Explanation: A tab number (indicated by n in the +SETTABn=nn+ code) is not specified or specifies a non-numeric character.

Program Action: The program accepts and processes the code up to the point of the error; the remainder of the code is ignored. The complete code is assigned a page and line number and is printed in the alter code listing.

T360.141 INVALID COLUMN POSITION SPECIFIED IN +SETTAB CODE

Explanation: A column position (indicated by nn+ in the +SETTABn=nn code) is not specified or specifies a non-numeric character.

Program Action: The program accepts and processes the code up to the point of the error; the remainder of the code is ignored. The complete code is assigned a page and line number and is printed in the alter code listing.

T360.142 TAB NUMBER WITHIN +SETTAB GREATER THAN 20

Explanation: The SETTAB code specifies a tab number greater than 20.

Program Action: The erroneous tab number and its column position are ignored by the program. The remainder of the code is processed. The complete code is assigned a page number which is printed in the alter code listing.

T360.143 COLUMN POSITION WITHIN +SETTAB GREATER THAN 107

Explanation: The SETTAB code specifies a column position greater than 107.

Program Action: The erroneous column position and its tab number are ignored by the program; the remainder of the code is processed. The complete code is assigned a page and line number and is printed in the alter code listing.

T360.144 PAGE DEPTH GREATER THAN MAXIMUM; SET TO 75

Explanation: The number of lines specified for the page depth exceeds 75.

Program Action: The program sets the page depth to 75.

T360.145 PAGE DEPTH LESS THAN MINIMUM; SET TO 25

Explanation: The number of lines specified for the page depth is less than 25.

Program Action: The program sets the page depth to 25.

T360.146 HYPHENATION DICTIONARY OVERFLOWED

Explanation: The maximum of 20 words that can be entered in the hyphenation dictionary has been exceeded.

Program Action: The first 20 words submitted to the dictionary are entered. The remaining words are ignored.

T360.151 INSUFFICIENT INFORMATION FOLLOWING -G- CODE

Explanation: The -G- code is specified with only one column position.

Program Action: The -G- code is accepted by the program; however, no line is drawn. The code is assigned a page and line number and is printed in the alter code listing.

T360.152 NO COLUMN POSITIONS SPECIFIED WITH -G- CODE

Explanation: The line parameter is not specified with the -G- code. (-G- does not print the last specified horizontal line.)

Program Action: The -G- code is accepted by the program; however, no processing takes place. It is assigned a page and line number and is printed in the alter code listing.

T360.153 ODD NUMBER OF COLUMNS SPECIFIED WITH -G- CODE

Explanation: The number of column positions specified by the -G- code is not a multiple of 2.

Program Action: The complete code is accepted by the program; however, the last unpaired column position is not processed.

T360.154 -Qnn- CODE FOR A NONEXISTENT VERTICAL LINE

Explanation: The -Qnn- code specifies the ending of a non-existing vertical line.

Program Action: If the code specifies the ending of only one vertical line, the code is accepted by the program, but it is not processed. If the code specifies the ending of several lines, the erroneous portion of the code is ignored; the remainder of the code is processed.

T360.155 NO COLUMN POSITION SPECIFIED WITH -BB- CODE

Explanation: The -BB- code is specified without column positions.

Program Action: The code is accepted by the program, but it is not processed.

T360.156 NO COLUMN POSITION SPECIFIED WITH -VV- CODE

Explanation: The -VV- code is specified without column positions.

Program Action: The code is accepted by the program, but it is not processed.

T360.157 INSUFFICIENT INFORMATION FOLLOWING -B- CODE

Explanation: The -B- code is specified with only one column position.

Program Action: The -B- code is accepted by the program; but it is not processed.

T360.158 NUMBER OF VERTICAL LINES STARTED BY -B- AND/OR -BB- CODES EXCEEDS 22

Explanation: The total number of vertical lines started by the -B- or -BB- code exceeds 22.

Program Action: The program processes the first 22 lines requested; the remaining requests are ignored by the program.

T360.160 ATTEMPT TO OVERLAY VALID CHARACTER WHILE CONSTRUCTING A TABLE

Explanation: The -B- or -E- edit code specifies the column position that already contains text.

Program Action: The program suppresses the line character.

T360.161 ATTEMPT TO PLACE VERTICAL LINE BEYOND COLUMN WIDTH

Explanation: The -V- code specifies a column position greater than the column width.

Program Action: The program accepts the code, but it is not processed.

T360.162 ATTEMPT TO OVERLAY VALID CHARACTER WHILE DRAWING HORIZONTAL LINES

Explanation: The -G- code specifies a horizontal line to be printed in a line position that already contains text.

Program Action: The program suppresses the line character.

T360.163 BEGIN TABLE LINE DRAWN BEYOND COLUMN WIDTH

Explanation: The -B- code specifies a column position greater than the column width.

Program Action: The program accepts the complete code; however, the table line is drawn to the column width only.

T360.164 ATTEMPT TO END NONEXISTENT TABLE

Explanation: An -E- code is specified while no -B- code is operative.

Program Action: The program accepts the code, but it is not processed.

T360.165 TOO MANY VERTICAL LINES SPECIFIED

Explanation: Explanation: The total number of vertical lines specified by the -V- or -VV- code exceeds 22.

Program Action: The program processes the first 22 lines requested; the remaining lines are ignored.

T360.167 ATTEMPT TO OVERLAY VALID CHARACTER WHILE DRAWING VERTICAL LINE

Explanation: The -V- edit code specifies a column position that already contains text.

Program Action: The program suppresses the line character.

T360.181 AS-IS TEXT EXCEEDS 140 CHARACTERS

Explanation: The length of the text that is under the control of one -A- code exceeds 140 characters.

Program Action: The program starts a new line with the extra characters.

T360.182 INVALID EDIT CODE

Explanation: The alphabetic portion of the edit code does not represent a valid edit code.

Program Action: The edit code is accepted by the program, but the invalid portion of the code is not processed.

T360.183 EDIT CODE PROCESSING OUT OF SYNCHRONIZATION

Explanation: The program, as the result of improper input, is processing out of synchronization (i.e., text as codes and codes as text).

Program Action: The program can determine after a certain amount of text has been processed that it is out of synchronization and correct itself.

T360.184 AS-IS TEXT EXCEEDS COLUMN WIDTH

Explanation: The number of printable characters of the text line exceeds the column width.

Program Action: The program truncates (on the print tape only) the text at the column width.

T360.186 LINE LENGTH MINUS MARGIN AND INDENTION IS LESS THAN 20

Explanation: The line length minus the margin and indention reduces the printable line to less than 20 characters.

Program Action: The program sets the indention to 0.

T360.187 LINE LENGTH MINUS MARGIN AND HANGING INDENTION IS LESS THAN 20

Explanation: The line length minus the margin and hanging indention reduces the printable line to less than 20 characters.

Program Action: The program sets the hanging indention to the maximum indention that permits a 20 character line.

T360.188 MORE THAN 6 EDIT CODES IN COMBINATION

Explanation: Because normal text formatting seldom requires a combination of more than 6 edit codes, this message is issued when the combination exceeds 6. This message usually indicates processing out of synchronization (i.e., text as edit codes).

Program Action: The first six codes are processed; starting at the seventh code through to the next edit code, the input is processed as text.

T360.189 KEEP EXCEEDS PAGE DEPTH; KEEP RELEASED

Explanation: The space required for the specified regular or floating keep is greater than that of one column.

Program Action: The program releases the portion of the keep that exceeds the length of the column.

20 January, 1969

APPENDIX D: FUNCTION KEY LABEL AND NUMBER

	CANCEL 0	PRINT OPTIONS 1	PRINT 2	SWITCH PHASE 3	
RETURN 4	SUPPRESS 5	DRAW TABLE 6	DRAW LINES 7	ALTER 8	INQUIRE/ DELETE 9
GETLABEL 10	LIST of FIGURES 11	SPEC CHARS 12	TAB 13	HEADINGS 14	CENTER 15
LINK 16	KEEPS 17	AS-IS 18	SKIP 19	INDENT 20	PARAGRAPH & CAP1 21
BRANCH 22	UNDER-SCORE 23	MULT CAPS 24	NEW COL/PAGE 25	PARAGRAPH 26	CAP1 27
	BACK PAGE 28	BACK LINE 29	FORWARD PAGE 30	FORWARD LINE 31	

20 January, 1969

Compliments of the

HYPERTEXT EDITING SYSTEM

Brown University

20 January, 1969

A HYPERTEXT EDITING SYSTEM FOR THE /360[1]

Steve Carmody, Theodor H. Nelson,
David Rice and Andries van Dam
(Brown University; The Nelson Organization, Inc;
Brown University; Brown University)

I INTRODUCTION

The Brown University Hypertext Editing System is a computer system for text handling and display, having a number of helpful and intertwined uses.

It is both a sophisticated system for the composition and manipulation of manuscripts, and a reading machine on which to browse and query written materials having complex structure.

By Hypertext we mean strings of text arbitrarily interwoven and connected, i.e., nonlineal text.

In the Edit Phase, a user can create and manipuate Hypertexts; in the Format Phase he can format a Hypertext and convert it to conventional hard copy manuscript form.

---

1. For a fuller explanation of the system, its goals and its planned improvements, see the full paper by the same title, submitted to the 1969 SJCC.

II THE EDIT PHASE

The Hypertext editing functions currently available to the user on our 2250 MOD I display console fall into three main categories: Edit Commands, Travel Commands, and Structure Commands. Combined, these functions allow a user to create new text, to edit text which has been previously typed into the system, to browse through text which already exists, and to structure a text in any manner desired, either while it is being created, or after the entire text has been written as a Hypertext.

EDIT COMMANDS

INSERT:
Allows the user to type new text into the place he wishes to insert it. As he does this, the text spreads to form as large a gap as is needed for him to type his insertion. There is also an offline program which will accept card image data.

DELETE:
Allows the user to delete arbitrarily sized text strings and structures from a continuous piece of text.

DELETE SINGLE CHARACTER:
Same as DELETE above only the user need only activate the function key once and then may continuously delete any number of single characters.

REARRANGE:
Allows the user to move an arbitrary length string from one section of a Hypertext to another (cf. COPY).

COPY:
Allows the user to copy text from one section of the manuscript to another section (differs from REARRANGE in that COPY does not delete the string from where it was originally).

BORROW:
Same as COPY above, only the user has the option of supplying a citation or "explainer," which then appears after the string of text copied and after each subsequent insertion of the copied string in the text.

SUBSTITUTE:
Allows the user to replace a text string already in the manuscript by typing a new one in.

MAKETAG:
Allows the user to make "marginal notes" to himself. These are displayed at the bottom of the screen in a protected area, but are not printed in the hard copy.

INSTANCE:
Allows the user to "subroutine" a piece of text by giving it a name. It is called where needed by using the function INSERT INSTANCE.

INSERT INSTANCE:
Allows the user to retrieve a previously created instance and insert it into the text.

Two functions which make a text more readable are:

PARAGRAPH:
Allows the user to paragraph any section of the text on the scope. Multiple paragraphing at the same point provides a convenient method of skipping lines on the display.

UNPARAGRAPH:
Allows the user to unparagraph any point that he has previously paragraphed.

STRUCTURE COMMANDS

MAKELABEL:
Allows the user to identify a particular section of text with a label which can later be retrieved via the GETLABEL command.

MAKELINK:
Allows the user to tie together different pieces of the text, or to make links[2] to new areas which themselves have all the properties of a Hypertext.

MAKEBRANCH:
Similar to MAKELINK in that it allows a user to structure a Hypertext. However, a branch point stops the lineal flow of text, and at that point the user must decide which fragment of the text he is to follow. Any number of branches are possible, forming a branch menu (as in programmed texts).

TRAVEL COMMANDS

LINK:
Allows the user to follow a link to a different part of the text.

BRANCH:
Allows the user to take a branch (possibly one of many available at this point) to a different part of the text.

GETLABEL:
Allows the user to access a complete list of all the labels in the text, and the ability to transfer to any of these labels.

RETURN:
Allows the user to return via a pushdown stack to the last point from which he either 1) took a branch, 2) took a link, or 3) retrieved a label. A user may make many such jumps before retracing any.

A continuous piece of text is treated as if it were on a single long piece of paper, allowing the user to move forward or backward through it by SCROLLing:

SCROLL FORWARD BY LINE:
Allows the user to move the screen image forward one line at a time.

SCROLL BACKWARD BY LINE:
Allows the user to move the screen display backward one line, as might be expected.

---

2. One interpretation of a link is that of a super-footnote (i.e., an optional fragment of text which the user can reference as desired). Or a Link may be an autonomous piece of text from which the user may choose not to return.

SCROLL_FORWARD_BY_PAGE:
Similar to the above, a page being thought of as one full screen display.

SCROLL_BACKWARD_BY_PAGE:
Similar to above.

III THE FORMAT PHASE (FORMATTING AND PRINTING)

Having created a Hypertext, it is desirable to be able to print it out in a meaningful and final form.[3] Therefore, the facility to indicate simple or complex formatting directly from the Scope, and to print out the Hypertext in hard copy form, has been implemented. The actual printing is done by IBM's TEXT/360 text printing program.

The formatting options available to the user are those recognizable by IBM's TEXT/360. Included among them are capitalization and underscoring, paragraphing, indenting, skipping lines, automatically entering headings into a table of contents, formatting variable page and column layout, arbitrary running heads and foots, special characters not available on the 2250, tables, etc.

The formats are designated by a marker appearing on the screen, which can later be interrogated and deleted if desired. The facility to suppress these markers on the display is also available, making the text more readable.

All the travelling commands described in Part II above are available while formatting, and the user can flip between the editing and formatting phases via a function key.

Since a Hypertext is meant to be non-lineal, certain conventions are necessary to produce a lineal printed text. As the system is currently implemented, the first level of a linked section of Hypertext is printed as an inline footnote. At every branch point, the default branch taken is the first. However, the user can, if he wishes, specify a different branch level to be taken. For example, if he specifies the fifth branch, at every branch point the branch menu will be examined: if at least five branches exist, the fifth will be taken. However, if less than five are present, the printer routine will take the first branch.

Since these conventions are somewhat limiting, a design is underway to print out the entire Hypertext, with suitable cross-references, including a map designating the location of each section in the output.

Also available is the facility to print out an image of the current screen display, without detailed formatting (e.g., no line justification). This form of output would be analogous to a rough-draft copy.

---

3. This paper itself was naturally produced by the system.

IV PAGING

The Hypertext Editing System utilizes a paging system to retieve from disk pages corresponding to an area of the Hypertext selected by the user. Pages not currently being viewed by the user are stored back on disk only when new pages are requested and core is full.

A nice characteristic of the paging system is its ability to accomodate variable size pages up to 7200 bytes long. This allows better storage usage (both in core and in the data set) and greater speed in page manipulation and updating.

When a user begins a session with the editing system, he designates his Hypertext data set, which is then copied onto a scratch area on disk. All his editing and formatting changes are entered into this work area and are only made permanent when the user specifies COPY TO DISK or FINISH, at which point his original data set is replaced by the newly edited version. This technique allows another accept-reject level for the user, and in addition, (as a precaution when running in a multi-programming environment) this procedure eliminates complete destruction of his data set in case the operating system should fail.

8 December, 1968

Compliments of the

HYPERTEXT EDITING SYSTEM

Brown University

8 December, 1968

H.E.S. 7