A research center for augmenting human intellect*

by DOUGLAS C. ENGELBART
and WILLIAM K. ENGLISH

Stanford Research Institute
Menlo Park, California

1 SUMMARY

1a This paper describes a multisponsor research center at Stanford Research Institute in man-computer interaction.

1a1 For its laboratory facility, the Center has a time-sharing computer (65K, 24-bit core) with a 4.5 megabyte swapping drum and a 96 megabyte file-storage disk. This serves twelve CRT work stations simultaneously.

1a1a Special hardware completely removes from the CPU the burden of display refreshing and input sampling, even though these are done directly out of and into core.

1a1b The display in a user's office appears on a high-resolution (875-line) commercial television monitor, and provides both character and vector portrayals. A relatively standard typewriter keyboard is supplemented by a five-key handset used (optionally) for entry of control codes and brief literals. An SRI cursor device called the "mouse" is used for screen pointing and selection.

1a1b1 The "mouse" is a hand-held X-Y transducer usable on any flat surface; it is described in greater detail further on.

1a2 Special-purpose high-level languages and associated compilers provide rapid, flexible development and modification of the repertoire of service functions and of their control procedures (the latter being the detailed user actions and computer feedback involved in controlling the application of these service functions).

1b User files are organized as hierarchical structures of data entities, each composed of arbitrary combinations of text and figures. A repertoire of coordinated service features enables a skilled user to compose, study, and modify these files with great speed and flexibility, and to have searches, analyses data manipulation, etc. executed. In particular, special sets of conventions, functions, and working methods have been developed to air programming, logical design, documentation, retrieval, project management, team interaction, and hard-copy production.

2 INTRODUCTION

2a In the Augmented Human Intellect (AHI) Research Center at Stanford Research Institute a group of researchers is developing an experimental laboratory around an interactive, multiconsole computer-display system, and is working to learn the principles by which interactive computer aids can augment their intellectual capability.

2b The research objective is to develop principles and techniques for designing an "augmentation system."

2b1 This includes concern not only for the technology of providing interactive computer service, but also for changes both in ways of conceptualizing, visualizing, and organizing working material, and in procedures and methods for working individually and cooperatively.

*Principal sponsors are: Advanced Research Projects Agency and National Aeronautics and Space Agency (NAS1–7897), and Rome Air Development Center F30602–68–C–0286.

2c The research approach is strongly empirical. At the workplace of each member of the subject group we aim to provide nearly full-time availability of a CRT work station, and then to work continuously to improve both the service available at the stations and the aggregate value derived therefrom by the group over the entire range of its roles and activities.

2d Thus the research group is also the subject group in the experiment.

2d1 Among the special activities of the group are the evolutionary development of a complex hardware-software system, the design of new task procedures for the system's users, and careful documentation of the evolving system designs and user procedures.

2d2 The group also has the usual activities of managing its activities, keeping up with outside developments, publishing reports, etc.

2d3 Hence, the particulars of the augmentation system evolving here will reflect the nature of these tasks—i.e., the system is aimed at augmenting a system-development project team. Though the primary research goal is to develop principles of analysis and design so as to understand how to augment human capability, choosing the researchers themselves as subjects yields as valuable secondary benefit a system tailored to help develop complex computer-based systems.

2e This "bootstrap" group has the interesting (recursive) assignment of developing tools and techniques to make it more effective at carrying out its assignment.

2e1 Its tangible product is a developing augmentation system to provide increased capability for developing and studying augmentation systems.

2e2 This system can hopefully be transferred, as a whole or by pieces of concept, principle and technique, to help others develop augmentation systems for aiding many other disciplines and activities.

2f In other words we are concentrating fully upon reaching the point where we can do all of our work on line—placing in computer store all of our specifications, plans, designs, programs, documentation, reports, memos, bibliography and reference notes, etc., and doing all of our scratch work, planning, designing, debugging, etc., and a good deal of our intercommunication, via the consoles.

2f1 We are trying to maximize the coverage of our documentation, using it as a dynamic and plastic structure that we continually develop and alter to represent the current state of our evolving goals, plans, progress, knowledge, designs, procedures, and data.

2g The display-computer system to support this experiment is just (at this writing) becoming operational. Its functional features serve a basic display-oriented user system that we have evolved over five years and through three other computers. Below are described the principal features of these systems.

3 THE USER SYSTEM

3a Basic Facility

3a1 As "seen" by the user, the basic facility has the following characteristics:

3a1a 12 CRT consoles, of which 10 are normally located in offices of AHI research staff.

3a1b The consoles are served by an SDS 940 time-sharing computer dedicated to full-time service for this staff, and each console may operate entirely independently of the others.

3a1c Each individual has private file space, and the group has community space, on a high-speed disc with a capacity of 96 million characters.

3a2 The system is not intended to serve a general community of time-sharing users, but is being shaped in its entire design toward the special needs of the "bootstrapping" experiment.

3b Work Stations

3b1 As noted above, each work station is equipped with a display, an alphanumeric keyboard, a mouse, and a five-key handset.

3b2 The display at each of the work stations (see Figure 1) is provided on a high-resolution, closed-circuit television monitor.

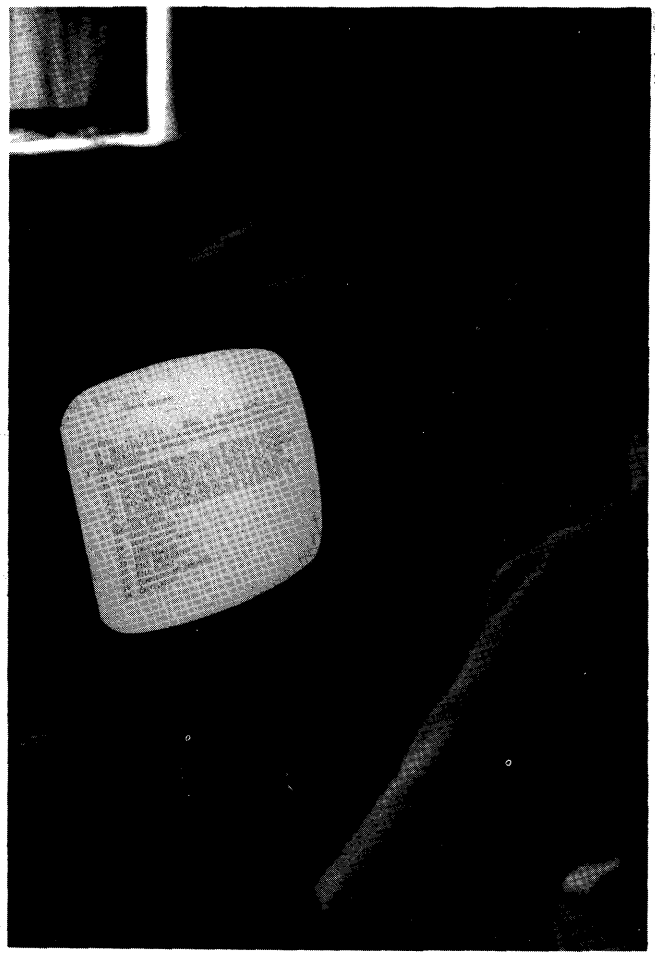


FIGURE 1—Typical work station, with TV display, typewriter keyboard, mouse, and chord handset

3b3 The alphanumeric keyboard is similar to a Teletype keyboard. It has 96 normal characters in two cases. A third-case shift key provides for future expansion, and two special keys are used for system control.

3b4 The mouse produces two analog voltages as het wo wheels (see Figure 2) rotate, each changing in proportion to the X or Y movement over the table top.

3b4a These voltages control—via an A/D converter, the computer's memory, and the display generator—the coordinates of a tracking spot with which the user may "point" to positions on the screen.

3b4b Three buttons on top of the mouse are used for special control.

3b4c A set of experiments, comparing (within our techniques of interaction) the

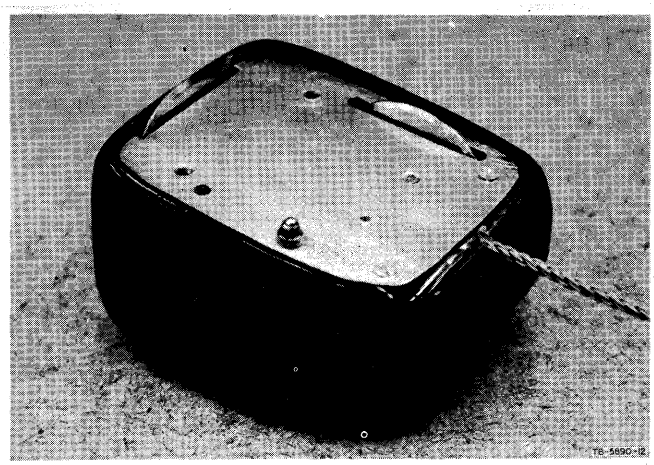


FIGURE 2—Underside of mouse

relative speed and accuracy obtained with this and other selection devices showed the mouse to be better than a light pen or a joystick (see Refs. English 1 and English 2).

3b4c1 Compared to a light pen, it is generally less awkward and fatiguing to use, and it has a decided advantage for use with raster-scan, write-through storage tube, projection, or multiviewer display systems.

3b5 The five-key handset has 31 chords or unique key-stroke combinations, in five "cases."

3b5a The first four cases contain lower- and upper-case letters and punctuation, digits, and special characters. (The chords for the letters correspond to the binary numbers from 1 to 26.)

3b5b The fifth case is "control case." A particular chord (the same chord in each case) will always transfer subsequent input-chord interpretations to control case.

3b5c In control case, one can "backspace" through recent input, specify underlining for subsequent input, transfer to another case, visit another case for one character or one word, etc.

3b5d One-handed typing with the handset is slower than two-handed typing with the standard keyboard. However, when the user works with one hand on the handset and one on the mouse, the coordinated in-

terspersion of control characters and short literal strings from one hand with mouse-control actions from the other yields considerable advantage in speed and smoothness of operation.

3b5d1 For literal strings longer than about ten characters, one tends to transfer from the handset to the normal keyboard.

3b5d2 Both from general experience and from specific experiment, it seems that enough handset skill to make its use worthwhile can generally be achieved with about five hours of practice. Beyond this, skill grows with usage.

3c Structure of Files

3c1 Our working information is organized into files, with flexible means for users to set up indices and directories, and to hop from file to file by display-selection or by typed-in file-name designations. Each file is highly structured in its internal organization.

3c1a The specific structure of a given file is determined by the user, and is an important part of his conceptual and "study-manipulate" treatment of the file.

3c2 The introduction of explicit "structuring" to our working information stems from a very basic feature of our conceptual framework (see Refs. Engelbart1 and Engelbart2) regarding means for augmenting human intellect.

3c2a With the view that the symbols one works with are supposed to represent a mapping of one's associated concepts, and further that one's concepts exist in a "network" of relationships as opposed to the essentially linear form of actual printed records, it was decided that the concept-manipulation aids derivable from real-time computer support could be appreciably enhanced by structuring conventions that would make explicit (for both the user and the computer) the various types of network relationships among concepts.

3c2b As an experiment with this concept, we adopted some years ago the convention of organizing all information into explicit hierarchical structures, with provisions for arbitrary cross-referencing among the elements of a hierarchy.

3c2b1 The principal manifestation of this hierarchical structure is the breaking up of text into arbitrary segments called "statements," each of which bears a number showing its serial location in the text and its "level" in an "outline" of the text. This paper is an example of hierarchical text structure.

3c2c To set up a reference link from Statement A to Statement B, one may refer in Statement A either to the location number of B or to the "name" of B. The difference is that the number is vulnerable to subsequent structural change, whereas the name stays with the statement through changes in the structure around it.

3c2c1 By convention, the first word of a statement is treated as the name of the statement, if it is enclosed in parentheses. For instance, Statement 0 on the screen of Figure 1 is named "FJCC."

3c2c2 References to these names may be embedded anywhere in other statements, for instance as "see(AFI)," where special format informs the viewer explicitly that this refers to a statement named "AFI," or merely as a string of characters in a context such that the viewer can infer the referencing.

3c2c3 This naminga nd linking, when added to the basic hierarchical form, yields a highly flexible general structuring capability. These structuring conventions are expected to evolve relatively rapidly as our research progresses.

3c3 For some material, the structured-statement form may be undesirable. In these cases, there are means for suppressing the special formatting in the final printout of the structured text.

3c4 The basic validity of the structured-text approach has been well established by our subsequent experience.

3c4a We have found that in both off-line and on-line computer aids, the concep-

tion, stipulation, and execution of significant manipulations are made much easier by the structuring conventions.

3c4b Also, in working on line at a CRT console, not only is manipulation made much easier and more powerful by the structure, but a user's ability to get about very quickly within his data, and to have special "views" of it generated to suit his need, are significantly aided by the structure.

3c4c We have come to write all of our documentation, notes, reports, and proposals according to these conventions, because of the resulting increase in our ability to study and manipulate them during composition, modification, and usage. Our programming systems also incorporate the conventions. We have found it to be fairly universal that after an initial period of negative reaction in reading explicitly structured material, one comes to prefer it to material printed in the normal form.

3d **File Studying**

3d1 The computer aids are used for two principal "studying" operations, both concerned with construction of the user's "views," i.e., the portion of his working text that he sees on the screen at a given moment.

3d1a Display Start

3d1a1 The first operation is finding a particular statement in the file (called the "display start"); the view will then begin with that statement. This is equivalent to finding the beginning of a particular passage in a hard-copy document.

3d1b Form of View

3d1b1 The second operation is the specification of a "form" of view—it may simply consist of a screenful of text which sequentially follows the point specified as the display start, or it may be constructed in other ways, frequently so as to give the effect of an outline.

3d1c In normal, off-line document studying, one often does the first type of operation, but the second is like a sissors-and-staple job and is rarely done just to aid one's studying.

3d1d (A third type of service operation that will undoubtedly be of significant aid to studying is question answering. We do not have this type of service.)

3d2 Specification of Display Start

3d2a The display start may be specified in several ways:

3d2a1 By direct selection of a statement which is on the display—the user simply points to any character in the statement, using the mouse.

3d2a2 If the desired display start is not on the display, it may be selected indirectly if it bears a "marker."

3d2a2a Markers are normally invisible. A marker has a name of up to five characters, and is attached to a character of the text. Referring to the marker by name (while holding down a special button) is exactly equivalent to pointing to the character with the mouse.

3d2a2b The control procedures make it extremely quick and easy to fix and call markers.

3d2a3 By furnishing either the name or the location number of the statement, which can be done in either of two basic ways:

3d2a3a Typing from the keyboard

3d2a3b Selecting an occurrence of the name or number in the text. This may be done either directly or via an indirect marker selection.

3d2b After identifying a statement by one of the above means, the user may request to be taken directly there for his next view. Alternately, he may request instead that he be taken to some statement bearing a specified structure relationship to the one specifically identified. For instance, when the user identifies Statement 3E4 by one of the above means (assume it to be a member of the list 3E1 through 3E7), he may ask to be taken to

3d2b1 Its successor, i.e., Statement 3E5

3d2b2 Its predecessor, i.e., Statement 3E3

3d2b3 Its list tail, i.e., Statement 3E7

3d2b4 Its list head, i.e., Statement 3E1

3d2b5 Its list source, i.e., Statement 3E

3d2b6 Its subhead, i.e., Statement 3E4A

3d2c Besides being taken to an explicitly identified statement, a user may ask to go to the first statement in the file (or the next after the current location) that contains a specified word or string of characters.

3d2c1 He may specify the search string by typing it in, by direct (mouse) selection, or by indirect (marker) selection.

3d3 Specification of Form of View

3d3a The "normal" view beginning at a given location is like a frame cut out from a long scroll upon which the hierarchical set of statements is printed in sequential order. Such a view is displayed in Figure 1.

3d3b Otherwise, three independently variable view-specification conditions may be applied to the construction of the displayed view: level clipping, line truncation, and content filtering. The view is simultaneously affected by all three of these.

3d3b1 Level: Given a specified level parameter, L (L = 1, 2, . . . , ALL), the view generator will display only those statements whose "depth" is less than or equal to L. (For example, Statement 3E4 is third level, 3E second, 4B2C1 fifth, etc.) Thus it is possible to see only first-level statements, or only first-, second-, and third level statements, for example.

3d3b2 Truncation: Given a specified truncation parameter, T (T = 1, 2, . . ., ALL), the view generator will show only the first T lines of each statement being displayed.

3d3b3 Content: Given a specification for desired content (written in a special high-level content-analysis language) the view generator optionally can be directed to display only those statements that have the specified content.

3d3b3a One can specify simple strings, or logical combinations thereof, or such things as having the word "memory" within four words of the word "allocation."

3d3b3b Content specifications are written as text, anywhere in the file. Thus the full power of the system may be used for composing and modifying them.

3d3b3c Any one content specification can then be chosen for application (by selecting it directly or indirectly). It is compiled immediately to produce a machine-code content-analysis routine, which is then ready to "filter" statements for the view generator.

3d3c In addition, the following format features of the display may be independently varied: indentation of statements according to level, suppression of location numbers and/or names of statements, and separation of statements by blank lines.

3d3d. The user controls these view specifications by means of brief, mnemonic character codes. A skilled user will readjust his view to suit immediate needs very quickly and frequently; for example, he may change level and truncation settings several times in as many seconds.

3d4 "Freezing" Statements

3d4a One may also pre-empt an arbitrary amount of the upper portion of the screen for holding a collection of "frozen" statements. The remaining lower portion is treated as a reduced-size scanning frame, and the view generator follows the same rules for filling it as described above.

3d4b The frozen statements may be independently chosen or dismissed, each may have line truncation independent of the rest, and the order in which they are displayed is arbitrary and readily changed. Any screen-select operand for any command may be selected from any portion of the display (including the frozen statements).

3d5 Examples

3d5a Figures 3 and 4 show views generated from the same starting point with different level-clipping parameters. This example happens to be of a program written in our Machine-Oriented language (MOL, see below).

3d5b Figure 5, demonstrates the freezing feature with a view of a program (the same one shown in Figure 8) written in our Control Metalanguage (CML, see below). Statements 3C, 3C2, 2B, 2B1, 2B2, 2B3, and 2B4 are frozen, and statements from 2J on are shown normally with L = 3, T = 1.

3d5b1 The freezing here was used to hold for simultaneous view four different functionally related process descriptions. The subroutines (+ BUG1SPEC) and (+ WAIT were located by use of the hop-to-name feature described above.

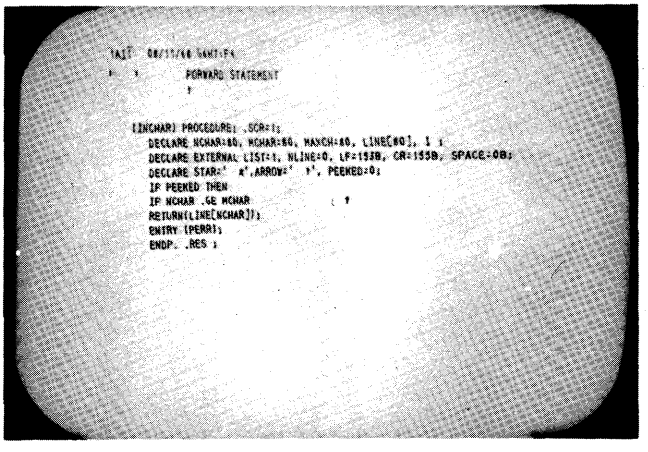


FIGURE 3—View of an MOL program, with level parameter set to 3 and truncation to 1

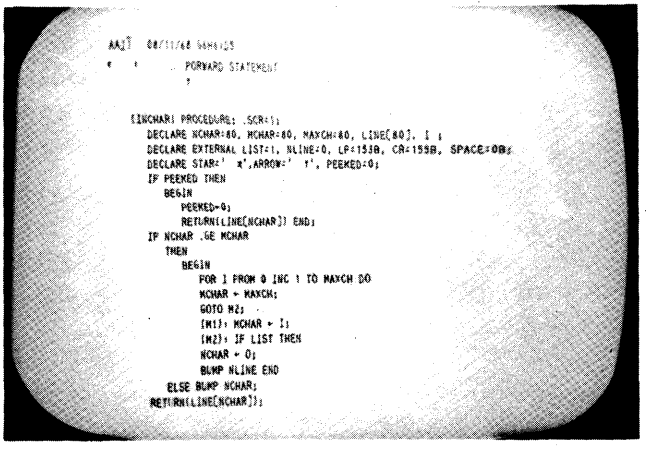


FIGURE 4—Same program as Figure 3, but with level parameter changed to 6 (several levels still remain hidden from view)

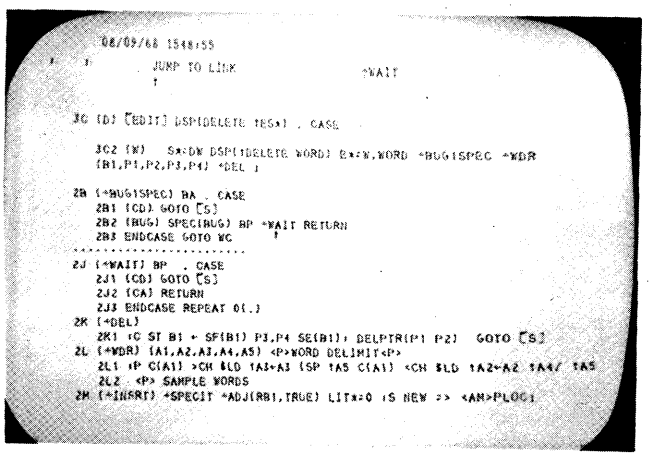


FIGURE 5—View of CML program, showing six frozen statements and illustrating use of reference hopping

3e **File Modification**

3e1 Here we use a standard set of editing operations, specifying with each operation a particular type of text entity.

3e1a Operations: Delete, Insert, Replace, Move, Copy.

3e1b Entities (within text of statements): Character, Text (arbitrary strings), Word, Visible (print string), Invisible (gap string).

3e1c Entities (for structure manipulation): Statement, Branch (statement plus all substructure), Group (sublist of branches), Plex (complete list of branches).

3e2 Structure may also be modified by joining statements, or breaking a statement into two at a specified point.

3e3 Generally, an operation and an entity make up a command, such as "Delete Word." To specify the command, the user types the first letter of each word in the command: thus "DW" specifies "Delete Word." There are occasional cases where a third word is used or where the first letter cannot be used because of ambiguities.

3f **File Output**

3f1 Files may be sent to any of a number of different output devices to produce hard copy—an upper/lower-case line printer, an

on-line high-quality typewriter, or paper tape to drive various typewriters.

3f1a In future it will be possible to send files via magnetic tape to an off-line CRT-to-film system from which we can produce Xerox prints, Multilith masters, or microform records.

3f2 Flexible format control may be exercised in this process by means of specially coded directives embedded in the files—running headers, page numbering, line lengths, line centering, suppression of location numbers, indenting, right justification (hyphenless), etc., are controllable features.

3g Compiling and Debugging

3g1 Source-code files written in any of our compiler languages (see below), or in the SDS 940 assembly language (ARPAS, in which our compiler output is produced) may be compiled under on-line control. For debugging, we have made a trivial addition to the SDS 940's DDT loader-debugger so as to operate it from the CRT displays. Though it was designed to operate from a Teletype terminal, this system gains a great deal in speed and power by merely showing with a display the last 26 lines of what would have been on the Teletype output.

3h Calculating

3h1 The same small innovation as mentioned above for DDT enables us to use the CAL system from a display terminal.

3i Conferencing

3i1 We have set up a room specially equipped for on-line conferencing. Six displays are arranged in the center of a square table (see Figure 6) so that each of twenty participants has good visibility. One participant controls the system, and all displays show the same view. The other participants have mice that control a large arrow on the screen, for use as a pointer (with no control function).

3i2 As a quick means of finding and displaying (with appropriate forms of view) any desired material from a very large collection, this system is a powerful aid to presentation and review conferences.

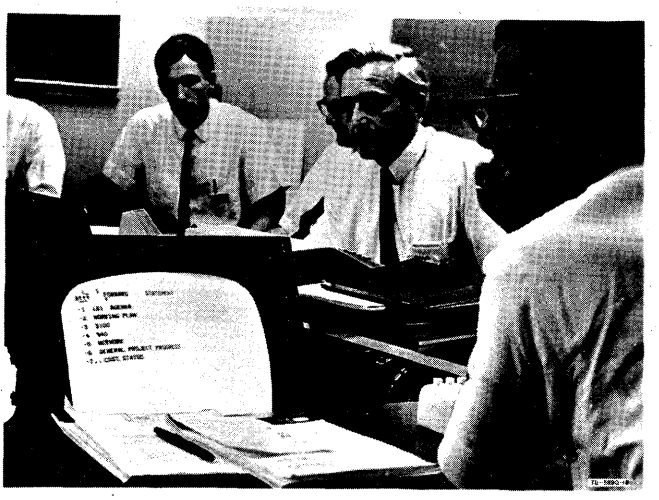


FIGURE 6—On-line conference arrangement

3i3 We are also experimenting with it in project meetings, using it not only to keep track of agenda items and changes but also to log progress notes, action notes, etc. The review aid is of course highly useful here also.

3i4 We are anxious to see what special conventions and procedures will evolve to allow us to harness a number of independent consoles within a conference group. This obviously has considerable potential.

4 SERVICE-SYSTEM SOFTWARE

4a The User's Control Language

4a1 Consider the service a user gets from the computer to be in the form of discrete operations—i. e., the execution of individual "service functions" from a repertoire comprising a "service system."

4a1a Examples of service functions are deleting a word, replacing a character, hopping to a name, etc.

4a2 Associated with each function of this repertoire is a "control-dialogue procedure." This procedure involves selecting a service function from the repertoire, setting up the necessary parameter designations for a particular application, recovering from user errors, and calling for the execution of the function.

4a2a The procedure is made up of the sequence of keystrokes, select actions, etc.

made by the user, together with the interspersed feedback messages from the computer.

4a3 The repertoire of service functions, together with their control-dialogue procedures, constitutes the user's "control language." This is a language for a "master-slave" dialogue, enabling the user to control application of the computer's capabilities to his own service.

4a3a It seems clear that significant augmentation of one's intellectual effectiveness from the harnessing of computer services will require development of a broad and sophisticated control-language vocabulary.

4a3b It follows that the evolution of such a control language is a very important part of augmentation-system research.

4a4 For the designer of user systems, it is important to have good means for specifying the nature of the functions and their respective control-dialogue procedures, so that a design specification will be

4a4a Concise, so that its essential features are easily seen

4a4b Unambiguous, so that questions about the design may be answered clearly

4a4c Canonical, so that information is easily located

4a4d Natural, so that the form of the description fits the conceptual frame of the design

4a4e Easy to compose, study, and modify, so that the process of evolutionary design can be facilitated.

4a5 It is also important for the user to have a description of the service functions and their control-dialogue procedures.

4a5a The description must again be concise, unambiguous, canonical, and natural; furthermore, it must be accurate, in that everything relevant to the user about the service functions and their control-dialogue procedures is described, and everything described actually works as indicated.

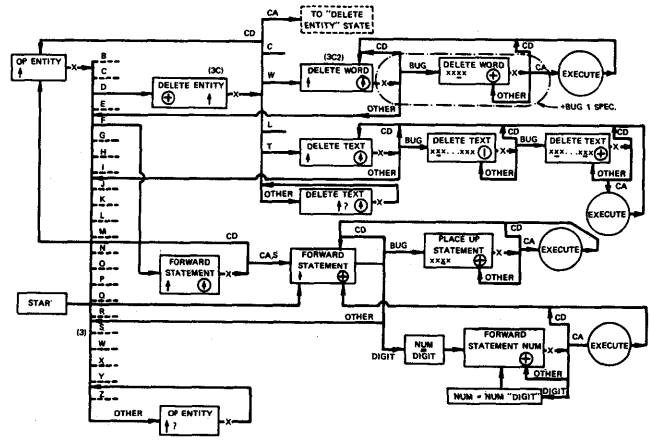


FIGURE 7—State-chart portrayal of part of the text-manipulation control language

4b State-Chart Representation of Control-Language Design

4b1 Figure 7 shows a charting method that was used in earlier stages of our work for designing and specifying the control-procedure portions of the control language. Even though limited to describing only the control-dialogue procedures, this representation nonetheless served very well and led us to develop the successive techniques described below.

4b2 Figure 7 shows actual control procedures for four service functions from the repertoire of an interactive system: Delete Word, Delete Text, Place Up Statement, and Forward Statement.

4b2a The boxes contain abbreviated descriptions of relevant display-feedback conditions, representing the intermediate states between successive user actions. Both to illustrate how the charting conventions are used and to give some feeling for the dynamics of our user-system control procedures, we describe briefly below both the chart symbols and the associated display-feedback conventions that we have developed.

4b2a1 The writing at the top of each box indicates what is to be shown as "command feedback" at the top of the display (see Figures 3, 4 and 5).

4b2a1a An uparrow sometimes ap-

pears under the first character of one of the words of Command Feedback.

4b2a1a1 This indicates to the user that the next character he types will be interpreted as designating a new term to replace that being pointed to—no uparrow under Command Feedback signifies that keyboard action will not affect the command designation.

4b2a1b "Entity" represents the entity word (i.e., "character," "word, "statement," etc.) that was last used as part of a fully specified command.

4b2a1b1 The computer often "offers" the user an entity option.

4b2a2 The circle in the box indicates the character to be used for the "bug" (the tracking spot), which alternates between the characters uparrow and plus.

4b2a2a The uparrow indicates that a select action is appropriate, and the plus indicates that a select action is inappropriate.

4b2a3 The string of X's, with underlines, indicates that the selected characters are to be underlined as a means of showing the user what the computer thinks he has selected.

4b2b There is frequently an X on the output line from a box on the chart. This indicates that the computer is to wait until the user has made another action.

4b2b1 After this next action, the computer follows a branching path, depending upon what the action was (as indicated on the chart) to reach another state-description box or one of the function-execution processes.

4c The Control Metalanguage

4c1 In search for an improvement over the state chart, we looked for the following special features, as well as the general features listed above:

4c1a A representational form using structural text so as to harness the power of our on-line text-manipulation techniques for composing, studying, and modifying our designs.

4c1b A form that would allow us to specify the service functions as well as the control-dialogue procedures.

4c1c A form such that a design-description file could be translated by a computer program into the actual implementation of the control language.

4c2 Using our Tree Meta compiler-compiler (described below), we have developed a next step forward in our means of designing, specifying, implementing and documenting our on-line control languages. The result is called "Control Metalanguage" (CML).

4c2a Figure 8 shows a portion of the description for the current control language, written in Control Metalanguage.

4c2a1 This language is the means for describing both the service functions and their control-dialogue procedures.

4c2b The Control Metalanguage Translator (CMLT) can process a file containing such a description, to produce a corresponding version of an interactive system which responds to user actions exactly as described in the file.

4c3 There is a strong correspondence between the conventions for representing the control procedures in Control Metalanguage and in the state chart, as a comparison of Figures 8 and 7 will reveal.

4c3a The particular example printed out for Figure 8 was chosen because it specifies some of the same procedures as in Figure 7.

4c3b For instance, the steps of display-feedback states, leading to execution of the "Delete Word" function, can readily be followed in the state chart.

4c3b1 The steps are produced by the user typing "D," then "W," then selecting a character in a given word, and then hitting "command accept" (the CA key).

4c3b2 The corresponding steps are outlined below for the Control Metalanguage description of Figure 8, progressing from Statement 3, to Statement 3c, to

Statement 3c2, to Subroutine + BUG-SPEC, etc.

4c3b3 The points or regions in Figure 7 corresponding to these statements and subroutines are marked by (3), (3C), (3C2), and (+ BUG1SPEC), to help compare the two representations.

4c3c These same steps are indicated in Figure 8, starting from Statement 3:

4c3c1 "D" sets up the state described in Statement 3C

4c3c2 "W" sets up the state described in Statement 3C2

FIGURE 8—Metalanguage description of part of control language

```
3  (wc:) zap  case

   3A (b) [edit]  dsp(backward ↑es*) .  case
   •
   •
   •

   3B (c) [edit] dsp(copy ↑es*) :s true => <am>adj1: . case

      3B1   (c)  s*=cc dsp(↑copy character) e*=c,character +bug2spec
      +cdlim(b1,p1,p2,p3,p4) +cdlim(b2,p5,p6,p7,p8)
      +cpchtx(b1,p2,p4,p5,p6) ;

      3B2  (w)  s*=cw dsp(↑copy word) e*=w,word +bug2spec
      +wdr2(b1,p1,p2,p3,p4) +wdr2(b2,p5,p6,p7,p8)
      +cpwdvs(b1,p2,p4,p5,p6) ;

      3B3  (l)  s*=cl dsp(↑copy line) e*=l,line +bug2spec
      +ldlim(b1,p1,p2,p3,p4)  +ldlim(b2,p5,p6,p7,p8) :c st b1←sf(b1) p2,
      rif :p p2>p1 cr: then (cr) else (null) ,  p5 p6, p4 se(b1): goto
      [s]

      3B4  (v)  s*=cv dsp(↑copy visible) e*=v,visible +bug2spec
      +vdr2(b1,p1,p2,p3,p4) +vdr2(b2,p5,p6,p7,p8)
      +cpwdvs(b1,p2,p4,p5,p6) ;
      •
      •
      •
      3b10  endcase +caqm ;

   3C (d) [edit] dsp(delete ↑es*) . case

      3C1 (c)  s*=dc dsp(↑delete character) e*=c,character +bug1spec
      +cdlim(b1,p1,p2,p3,p4) +del;

      3C2 (w)   s*=dw dsp(↑delete word) e*=w,word +bug1spec +wdr
      (b1,p1,p2,p3,p4) +del ;

      3C3 (l)   s*=dl dsp(↑delete line) e*=l,line +bug1spec...
      •
      •
      •
```

4c3c3 The subroutine +BUG1SPEC waits for the select-word (1) and CA (2) actions leading to the execution of the delete-word function.

4c3c3a Then the TWDR subroutine takes the bug-position parameter and sets pointers P1 through P4 to delimit the word in the text data.

4c3c3b Finally, the + DEL subroutine deletes what the pointers delimit, and then returns to the last-defined state (i.e., to where S* = DW).

4d Basic Organization of the On-Line System (NLS)

4d1 Figure 9 shows the relationships among the major components of NLS.

4d2 The Tree Meta Translator is a processor specially designed to produce new translators.

4d2a There is a special language—the Tree Meta Language—for use in describing the translator to be produced.

4d2b A special Tree Meta library of subroutines must be used, along with the output of the Tree Meta Translator, to produce a functioning new translator. The same library serves for every translator it produces.

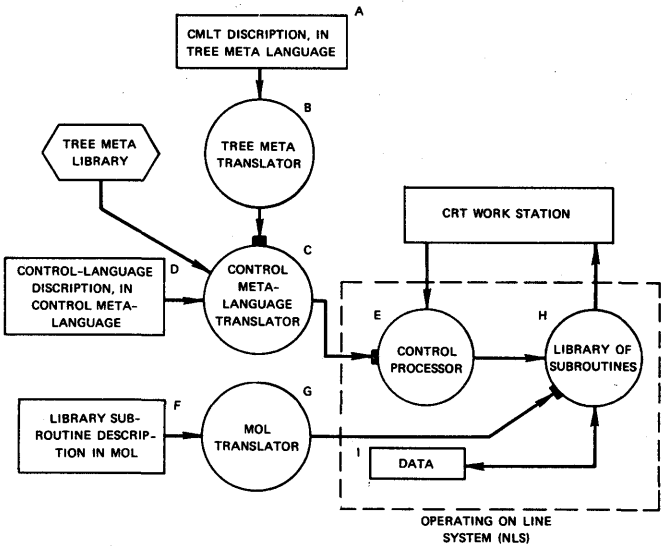


FIGURE 9—Basic organization of NLS showing use of compilers and compiler-compiler to implement it

4d3 For programming the various subroutines used in our 940 systems, we have developed a special Machine-Oriented Language (MOL), together with an MOL Translator to convert MOL program descriptions into machine code (see Ref. Hay1 for a complete description).

4d3a The MOL is designed to facilitate system programming, by providing a high-level language for iterative, conditional, and arithmetic operations, etc., along with a block structure and conventions for labeling that fit our structured-statement on-line manipulation aids.

4d3a1 These permit sophisticated computer aid where suitable, and also allow the programmer to switch to machine-level coding (with full access to variables, labels, etc.) where core space, speed, timing, core-mapping arrangements, etc., are critical.

4d4 The NLS is organized as follows (letters refer to Figure 9):

4d4a The Control Processor (E) receives and processes successive user actions, and calls upon subroutines in the library (H) to provide it such services as the following:

4d4a1 Putting display feedback on the screen

4d4a2 Locating certain data in the file

4d4a3 Manipulating certain working data

4d4a4 Constructing a display view of specified data according to given viewing parameters, etc.

4d4b The NLS library subroutines (H) are produced from MOL programs (F), as translated by the MOL Translator (G).

4d4c The Control Processor is produced from the control-language description (D), written in Control Metalanguage, as translated by the CMLT (C).

4d4d The CMLT, in turn, is produced from a description (A) written in Tree Meta, as translated by the Tree Meta Translator (B).

4d5 Advantages of Metalanguage Approach to NLS Implementation

4d5a The metalanguage approach gives us improved means for control-language specification, in terms of being unambiguous, concise, canonical, natural and easy to compose, study and modify.

4d5b Moreover, the Control Metalanguage specification promises to provide in itself a users' documentation that is completely accurate, and also has the above desirable characteristics to facilitate study and reference.

4d5c Modifying the control-dialogue procedures for existing functions, or making a reasonable range of changes or additions to these functions, can often be accomplished solely by additions or changes to the control-language record (in CML).

4d5c1 With our on-line studying, manipulating and compiling techniques, system additions or changes at this level can be thought out and implemented (and automatically documented) very quickly.

4d5d New functions that require basic operations not available through existing subroutines in the NLS library will need to have new subroutines specified and programmed (in MOL), and then will need new terms in CML to permit these new functions to be called upon. This latter requires a change in the record (A), and a new compilation of CMLT by means of the Tree Meta Translator.

4d5d1 On-line techniques for writing and modifying the MOL source code (F), for executing the compilations, and for debugging the routines, greatly reduce the effort involved in this process.

5 SERVICE-SYSTEM HARDWARE (OTHER THAN SDS 940)

5a In addition to the SDS 940, the facility includes peripheral equipment made by other manufacturers and equipment designed and constructed at SRI.

5b All of the non-SDS equipment is interfaced through the special devices channel which connects to the second memory buss through the SDS memory interface connection (MIC).

5b1 This equipment, together with the RAD, is a significant load on the second memory buss. Not including the proposed "special operations" equipment, the maximum expected data rate is approximately 264,000 words per second or one out of every 2.1 memory cycles. However, with the 940 variable priority scheme for memory access (see Pirtle[1]), we expect less than 1 percent degradation in CPU efficiency due to this load.

5b2 This channel and the controllers (with the exception of the disc controller) were designed and constructed at SRI.

5b2a In the design of the hardware serving the work stations, we have attempted to minimize the CPU burden by making the system as automatic as possible in its access to memory and by formatting the data in memory so as to minimize the executive time necessary to process it for the users.

5c Figure 10 is a block diagram of the special-devices channel and associated equipment. The major components are as follows:

5c1 Executive Control

5c1a This is essentially a sophisticated multiplexer that allows independent, asynchronous access to core from any of the 6 controllers connected to it. Its functions are the following:

5c1a1 Decoding instructions from the computer and passing them along as signals to the controllers.

5c1a2 Accepting addresses and requests for memory access (input or output) from the controllers, determining relative priority among the controllers, synchronizing to the computer clock, and passing the requests along to memory via the MIC.

5c1b The executive control includes a comprehensive debugging panel that allows any of the 6 controllers to be operated off-line without interfering with the operation of other controllers.

5c2 Disc File

5c2a This is a Model 4061 Bryant disc, selected for compatibility with the continued 940-system development by Berkeley's Project GENIE, where extensive file-handling software was developed.

5c2b As formatted for our use, the disc will have a storage capacity of approximately 32 million words, with a data-transfer rate of roughly 40,000 words per second and average access time of 85 milliseconds.

5c2c The disc controller was designed by Bryant in close cooperation with SRI and Project GENIE.

5c3 Display System

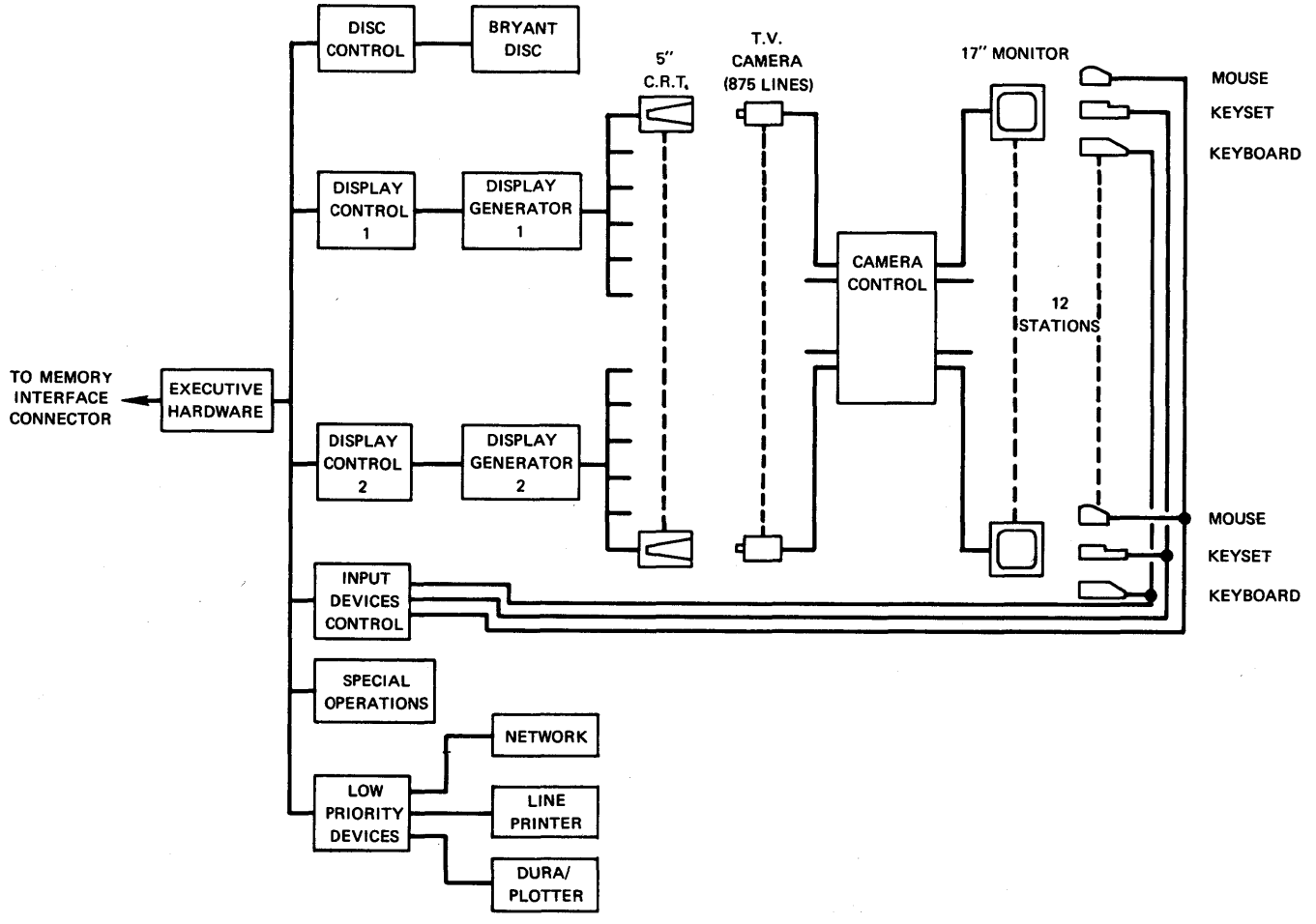
5c3a The display systems consists of two identical subsystems, each with display controller, display generator, 6 CRT's, and 6 closed-circuit television systems.

5c3b The display controllers process display-command tables and display lists that are resident in core, and pass along display-buffer contents to the display generators.

5c3c The display generators and CRT's were developed by Tasker Industries to our specifications. Each has general character-vector plotting capability. They will accept display buffers consisting of instructions (beam motion, character writing, etc.) from the controller. Each will drive six 5-inch high-resolution CRT's on which the display pictures are produced.

5c3c1 Character writing time is approximately 8 microseconds, allowing an aver-

FIGURE 10—Special devices channel



age of 1000 characters on each of the six monitors when regenerating at 20 cps.

5c3d A high-resolution (875-line) closed-circuit television system transmits display pictures from each CRT to a television monitor at the corresponding work-station console.

5c3e This system was developed as a "best solution" to our experimental-laboratory needs, but it turned out to have properties which seem valuable for more widespread use:

5c3e1 Since only all-black or all-white signal levels are being treated, the scan-beam current on the cameras can be reduced to achieve a short-term image-storage effect that yields flicker-free TV output even when the display refresh rate is as low as 15 cps. This allows a display generator to sustain about four times more displayed material than if the users were viewing direct-view refreshed tubes.

5c3e2 The total cost of small CRT, TV camera, amplifier-controller, and monitor came to about $5500 per work station—where a random-deflection, display-quality CRT of similar size would cost considerably more and would be harder to drive remotely.

5c3e3 Another cost feature which is very important in some system environments favors this TV approach: The expensive part is centrally located; each outlying monitor costs only about $600, so terminals can be set up even where usage will be low, with some video switching in the central establishment to take one terminal down and put another up.

5c3e4 An interesting feature of the video system is that with the flick of a switch the video signal can be inverted, so that the image picked up as bright lines on dim background may be viewed as black lines on a light background. There is a definite user preference for this inverted form of display.

5c3f In addition to the advantages noted above, the television display also invites the use of such commercially available devices as extra cameras, scan converters, video switches, and video mixers to enrich system service.

5c3f1 For example, the video image of a user's computer-generated display could be mixed with the image from a camera focused on a collaborator at another terminal; the two users could communicate through both the computer and a voice intercom. Each user would then see the other's face superimposed on the display of data under discussion.

5c3f2 Superimposed views from cameras focused on film images or drawings, or on the computer hardware, might also be useful.

5c3f3 We have experimented with these techniques (see Figure 11) and found them to be very effective. They promise to add a great deal to the value of remote display terminals.

5c4 Input-Device Controller

5c4a In addition to the television monitor, each work-station console has a keyboard, binary keyset, and mouse.

5c4b The controller reads the state of these

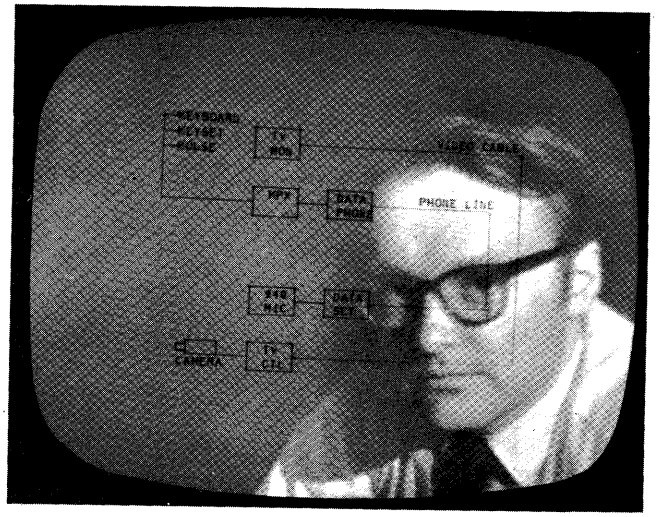


FIGURE 11—Television display obtained by mixing the video signal from a remote camera with that from the computer-generated display

devices at a preset interval (about 30 milliseconds) and writes it into a fixed location table in core.

5c4b1 Bits are added to information from the keyboards, keysets and mouse switches to indicate when a new character has been received or a switch has changed state since the last sample. If there is a new character or switch change, an interrupt is issued after the sample period.

5c4b2 The mouse coordinates are formatted as a beam-positioning instruction to the display generator. Provisions are made in the display controller for including an entry in the mouse-position table as a display buffer. This allows the mouse position to be continuously displayed without any attention from the CPU.

5c5 Special Operations

5c5a The box with this label in Figure 10 is at this time only a provision in the executive control for the addition of a high-speed device. We have tentative plans for adding special hardware here to provide operations not available in the 940 instruction set, such as character-string moves and string-pattern matching.

5c6 Low-Priority Devices

5c6a This controller accommodates three devices with relatively low data-transfer rates. At this time only the line printer is implemented, with provisions for adding an on-line typewriter (Dura), a plotter, and a terminal for the proposed ARPA computer network.

5c6a1 The line printer is a Potter Model HSP-3502 chain printer with 96 printing characters and a speed of about 230 lines per minute.

6 REFERENCES

6a (English 1) W K ENGLISH D C ENGELBART B HUDDART
Computer-aided display control
Final Report Contract NAS 1-3988 SRI Project 5061 Stanford Research Institute Menlo Park California July 1965

6b (English2) W K ENGLISH D C ENGELBART M L BERMAN
Display-selection techniques for text manipulation
IEEE Trans on Human Factors in Electronics Vol HFE-8 No 1 1967

6c (Engelbart1) D C ENGLEBART
Augmenting human intellect: A conceptual framework
Summary Report Contract AF 49 638 1024 SRI Project 3578 Stanford Research Institute Menlo Park California October 1962

6d (Engelbart2) D C ENGELBART
A conceptual framework for the augmentation of man's intellect
In Vistas in Information Handling Vol 1 D W Howerton and D C Weeks eds Spartan Books Washington D C 1663

6e (Hay1) R E HAY J F RULIFSON
MOL940 Preliminary speifications for an ALGOL *like machine-oriented language for the SDS 940*
Interim Technical Report Contract NAS 1-5940 SRI Project 5890 Stanford Research Institute Menlo Park California March 1968

6f (Pirtle1) M PIRTLE
Intercommunication of Processors and memory
Proc Fall Joint Computer Conference Anaheim California November 1967