Final Report

COMPUTER-AIDED DISPLAY CONTROL

By: W. K. ENGLISH D. C. ENGELBART BONNIE HUDDART

Prepared for:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LANGLEY RESEARCH CENTER
LANGLEY AIR FORCE BASE, VIRGINIA

CONTRACT NAS 1–3988

STANFORD RESEARCH INSTITUTE

MENLO PARK, CALIFORNIA

SRI

July 1965

Final Report

COMPUTER-AIDED DISPLAY CONTROL

Prepared for:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LANGLEY RESEARCH CENTER
LANGLEY AIR FORCE BASE, VIRGINIA

CONTRACT NAS 1-3988

By: W. K. ENGLISH D. C. ENGELBART BONNIE HUDDART

SRI Project 5061

Approved: TORBEN MEISLING, MANAGER
SYSTEMS ENGINEERING LABORATORY

JERRE D. NOE, EXECUTIVE DIRECTOR
ENGINEERING SCIENCES AND INDUSTRIAL DEVELOPMENT

Copy No. 180....

ABSTRACT

30204

This report presents the initial results of a continuing research study on the computer-aided human control of computer displays. Specifically, this project has been concerned with exploring methods of improving a person's ability to compose and modify text presented on a computer-driven cathode ray tube display.

This report includes a description of the on-line system for text manipulation, developed in part by this project and used in the preparation of this report. An approach to the analysis and evaluation of techniques for the control of computer displays is developed and the results of some preliminary evaluative experiments are described.

ILLUSTRATIONS

FOREWORD

This report summarizes the status of one project within a multiproject program at Stanford Research Institute, aimed at increasing the intellectual effectiveness of problem-solving human beings.

This report differs markedly from other technical reports. A glance at its pages will reveal many stylistic differences; not so readily apparent are the reasons for the differences and the methods by which the report was prepared.

Viewed as a whole, the program is an experiment in cooperation of man and machine. The comprehensible part of man's intellectual work involves manipulation of concepts, often in a disorderly cut-and-try manner, to arrive at solutions to problems. Man has many intellectual aids (e.g., notes, files, volumes of reference material, etc.) in which concepts are represented by symbols that can be communicated and manipulated externally. We are seeking to assist man in the manipulation of concepts--i.e., in his thinking, by providing a computer to aid in manipulation of these symbols. A computer can store and display essentially any structure of symbols that a man can write on paper; further, it can manipulate these symbols in a variety of ways. We argue that this service can be made available to help the on-going intellectual process of a problem-solving man; the service can be instantly available to perform tasks ranging from the very smallest to the very largest.

To make the most of this service, we believe that man will significantly alter his way of structuring and manipulating his working records and his ways of thinking and working. These altered facets of his problem-solving "system" will provide better coupling between the processes of the mind and the services of the computer.

One promising approach to exploring for increased value in man-machine "systems" would be for a group to:

(1) Develop an initial set of experimental aids;

(2) Apply these aids to their daily work;

(3) Use the experience thus accumulated to generate needs and possibilities for improvement;

(4) Improve the system (with new conventions, computer processes, methodology, etc.); and

(5) Apply the improved system in their daily work, using the new experience to generate new needs and new

possibilities for improvement, and so on.

The process sketched above is essentially what is being done in this multiproject program.

Our initial focus has been on computer-aided text* manipulation. There are several reasons for this:

(1) Text is representative of our speech and much of our conscious reasoning about nontextual records; it is the basic fabric in which most of the interpersonal collaboration in system development work such as ours takes place.

(2) Text is applicable as a representation of our thoughts and actions at all levels of our working system (e.g., from coding for the computer up to long-range planning for the research program). This promises us a comprehensive integration of our aids into our way of working--an important factor in our basic approach to exploring computer augmentation.

(3) A coordinated, working system for usefully manipulating text is relatively easy to implement. For the same resources, a wider collection of useful working aids may be implemented for text than for graphics, for instance.

(4) An effective system for handling the text of working records (planning, design, reference, etc.) will provide a sound structure in which later to embed manipulation techniques for other symbols e.g., graphics, mathematics, and chemical formulas. (Except in unusual cases of specialization, instances of a professional person's usage of these symbols are actually quite isolated in the context of his total working system when compared to the "text" manipulation he does.)

The vehicle for our study and experimentation has been a combination of on-line and off-line systems.

*By "text" we mean generally information represented by strings of characters. This includes mathematical equations, programming statements, etc.

Our on-line system, incorporating a CDC 160A computer and a CRT display, allows about 17,000 characters of working data to be written on a drum. Any portion of this material can be displayed on the CRT; the current working size of the display is 16 lines of 63 characters each. Basic manipulation operations of scan, delete, insert, replace, move, and copy, can be performed on entities of character, word, line, statement, or arbitrarily delimited strings of text. When manipulation is complete, a punched paper tape suitable for printout on a Flexowriter is produced. This tape may also be re-entered into the on- or off-line systems at any future time for further modification or manipulation of the data.

The off-line system, which incorporates the CDC 160A and a Burroughs 5500, allows one to specify general manipulation of the text with straightforward commands punched on paper tape by a Flexowriter or Teletype. These input paper tapes are processed to produce a fresh, cleaned-up version of the input; the output of the off-line system is both hard copy and revised paper tape. This output may, of course, subsequently be processed in either on- or off-line operations.

We come, then, to the basic and visible difference between this report and other SRI reports: With the exception of front matter, the report has been produced entirely on the on-line system that is being described. Certain features of this technique should be noted:

Statements--be they subheads, phrases, sentences, or paragraphs--are numbered and presented in hierarchical order. These statement numbers are one "handle" by which a statement may be grasped for any of the operations performed on- or off-line.

References, which appear in the Bibliography at the end of the report, are shown in the text by a mention of their statements numbers (e.g., "Ref 1b(ARMSBY2)"), rather than by the more familiar superscript notation.

Detailed study of this report requires some familiarity with the terms, concepts, computer-aid processes, and special hardware developed in this program; these are explained in the Appendices, parts of which have been extracted from the more complete "User's Guide to the Man-Machine Information System."

Under Contract NAS 1-3988 with NASA we have studied and developed the display-control techniques that represent the foundation of the on-line system. Other projects supporting the program are a recently completed project for Air Force Office of Scientific Research (Contract AF 49(638)-1024), under which the basic conceptual work was done, as well as the first off-line manipulation work; a current project for the Advanced Research Projects Agency (Contract SD-269), under which work on information structuring, basic working methodology, and the higher-level manipulation processes in the on-line system are being done; a recently completed project for Electronic Systems Division of the Air Force, (Contract AF 19(628)-4088), which studied structuring and manipulating techniques for managing information (specifically, system-program design documentation); and an internally-sponsored project at Stanford Research Institute, under which the current off-line system was developed.

1 PURPOSE OF THE PROJECT

1a The purpose of the project is to explore methods of improving a person's ability to compose and modify text with real-time computer aid, working at a computer-driven CRT display on which he views the text and sees immediate computer response to any control actions. Such a capability stems from a configuration of special hardware, human procedures, and programmed computer responses. In this report, we call such configurations "display-control schemes," or simply "schemes."

1b During task execution there is complex interplay between the human and the computer. Many actions are performed: they may be sequential, cyclical, or parallel; they may involve decisions that will affect subsequent actions; and they may very likely involve much nesting of actions within actions. In view of the speed and detailed handling characteristics of computers, there seem promised many significant possibilities for improving the speed with which this overall task activity can be performed. However, it is not immediately obvious what all these possibilities are; nor is it obvious what value (in terms of reduced operation time) would be gained by implementing any particular proposed improvement.

1c Thus our research has a dual challenge:

1c1 First, to collect, conceive, and to develop significant possibilities for improved display-control schemes.

1c2 Second, to develop systematic ways of designing, analyzing and evaluating scheme possibilities.

1d Our approach assumed a continuing project and our first year's work, as represented by this report, used the following "start up" strategy:

1d1 To select and begin to study a sub set of these possibilities. Criteria for selection are that they be relatively easy to implement so that we can immediately begin gaining practical experience in task execution and analysis; and that they include examples of several basic categories of schemes so that our initial experience is not too narrow.

1d2 To implement the most promising of these possibilities, and to use the resulting schemes to help us do our own work. An example is this report, which was prepared on our current on-line system and printed directly on the mats from computer output tapes.

1d3 To begin to develop more refined techniques for selecting which possibilities to study in detail, and for analyzing and assessing the value of those which we either implement or propose.

2 THE NOTION OF A "DISPLAY-CONTROL SCHEME"

2a A computer-aided display-control scheme provides a repertoire of joint human-computer processes for operating upon textual material and controlling the contents of the display screen accordingly. Each of these processes is represented by one command within a repertoire of commands. For each command there is a sequence of steps which the user must carry out in order to specify a computer operation to be performed on the working text (see Section III for a detailed description of one example); he must

2a1 Designate the operator (telling which command of the repertoire he wishes to call into operation), and

2a2 Supply the operands for that operator. There are two types of operand, a combination of which may be required by a particular operator:

2a2a "Literal operands," including both numerical parameters (for instance, on a "Scan Forward N Lines" command, where the user must supply the number of lines); and literal input strings of textual material (required, for instance, by an "Insert" command, where the text to be inserted must be supplied by the user in a discrete-character entry).

2a2b "Display-entity operands," such as characters or words, appearing on the display screen (required, for instance, in a command to "Delete (this) Word," where the user must then indicate which word he means by some kind of "pointing" at the screen,)

2a3 Call for execution of the command. This may be automatic when the last operand has been supplied, or it may require the user to perform a separate action.

2b The way in which the above actions are coordinated in order actually to execute a command of the repertoire is called the "command specification plan." This plan describes the procedures the user follows in specifying an operation and it includes his "escape procedures" in case he makes an error. It details the mnemonic of the command, the order in which operands are to be supplied, the command termination action, the computer responses providing feedback to the user throughout this entire process (as it guides and acknowledges his request), and the execution algorithms which process the text upon completion of a service request.

2c As an aid to describing and evaluating display-control schemes, we consider that the scheme description includes the following:

2c1 A presentation of the command repertoire

2c2 A statement of the command specification plan for each command in the repertoire

2c3 A description of the means provided for:

2c3a Designating the operator

2c3b Supplying literal input

2c3c Selecting display-entity operands.

3 ORGANIZATION OF THIS REPORT

3a Section II describes our current on-line system and the schemes available within it. Further detail is to be found in Appendix B, which is largely extracted from our "User's Guide" to the on-line system.

3b Section III reports the results of our analysis-technique study. It presents a detailed process plan for entering and executing a command, using techniques directly parallel to flow-charting to decompose the plan for this joint human-computer process. The plan shows a number of nested processes, with branching decisions and special procedures in case of contingencies and errors. A general discussion of process networks follows, with special emphasis on the probabilities associated with branching (i.e., the variations in execution sequence) and net execution time. Finally, we discuss how these probabilities can be included within the process-plan, and how this method of describing and analyzing activity-plans can be used in evaluating alternative schemes.

3c Section IV reports on the experimental evaluation we have conducted as a first step toward measuring the value associated with different possibilities for one scheme component. These experiments served to develop our techniques of experimentation and data analysis. They also gave some indications as to the strengths and weaknesses of certain components in our current set of available schemes. Before drawing any firm conclusions from this type of evaluative work, however, we will need to assess them in a more realistic "activity environment," and to coordinate the analytical evaluations from many such experiments.

3d Section V describes some plans and possibilities for future work.

3e Section VI contains summaries and conclusions.

1 INTRODUCTION

1a This section gives a description of our current on-line system, discussing the work station; the command repertoire; the command-specification plan; operator designation; literal input; and display-entity operand selection. (Further specifications of the command repertoire, and a more detailed discussion of system hardware are included in Appendix B.)

1b Various segments of the software for the on-line system were developed under different sponsorship, according to the pursuits of the respective projects.

1b1 The basic working system was developed and programmed under the sponsorship of the Advanced Research Projects Agency. This includes the routines for storing data on drum and tape; for input and output; and for executing the higher-level commands that operate on statement structures and tape files.

1b2 This project from the National Aeronautics and Space Administration developed and programmed those parts of the basic operating system that handle the core-held "current data"; the interface and interpretive routines that service the display and command-specification operations; and the basic text-editing routines.

1c The set of commands in our command repertoire represents the current stage of evolution. We began with those commands that seemed basic and relatively efficient for carrying out the composition and modification tasks that we meet commonly in our working environment.

1c1 As our experience and utilization activity grew, we added to and modified this set of commands in order to improve the system's efficiency and to accommodate new tasks.

1c2 We made no particular attempt either to limit the number of commands, or to simplify the system because we wanted to gain experience with a wide variety of commands that we thought might be useful.

1d In choosing which hardware devices to incorporate, we selected those which: would give us a representative sampling of the different "families" of display-control schemes; promised to be reasonably fast and easy in their operation; and could be obtained and implemented within the resources of the project.

2 THE WORK STATION

2a The work station of our current on-line system is shown in Figure 1. The station was designed for experimental flexibility; the configuration of work surfaces is easily changed, and all are adjustable in height. The CRT display screen can be elevated and tilted.

2b The photograph shows the work surfaces arranged in their usual configuration:

2b1 When the user is seated at the work station console, the CRT display is directly in front of him. It presents an arbitrarily specified section of the 17,000-character working text which is stored in the computer's auxiliary memory.

2b2 The typewriter-like keyboard just in front of the user allows him to enter mnemonic character sequences (called "operators") to designate controlling actions to the computer, or to enter arbitrary sequences of characters (the so-called "literal" input) to be inserted into the working text.

2b3 At the user's left is a fourteen-button control panel with which he may alternatively designate the operators for some of the most heavily used editing commands.

2b4 Within comfortable reach of the user's right hand is a device called the "mouse," which we developed for evaluation (along with others, such as light pen, Grafacon, joystick, etc.) as a means for selecting those displayed text entities upon which the commands are to operate.

2b4a As the mouse is moved over the surface of the table, its position is constantly being monitored by the computer, which displays a special tracking cross, which we call the "bug," on the screen in a position corresponding to that of the mouse on the table.

2b4b A user soon finds it very easy to keep his eyes on the screen and cause the bug to move about upon it as quickly and naturally as if he were pointing his finger (but with less fatigue).

2b4c The user selects a text entity, called an "operand," on the screen by positioning the bug near or on it and pressing the "select" button which is under his forefinger as he holds the mouse.

2b4d The Grafacon or joystick when used instead of the mouse, are operated by the user's right hand in about the

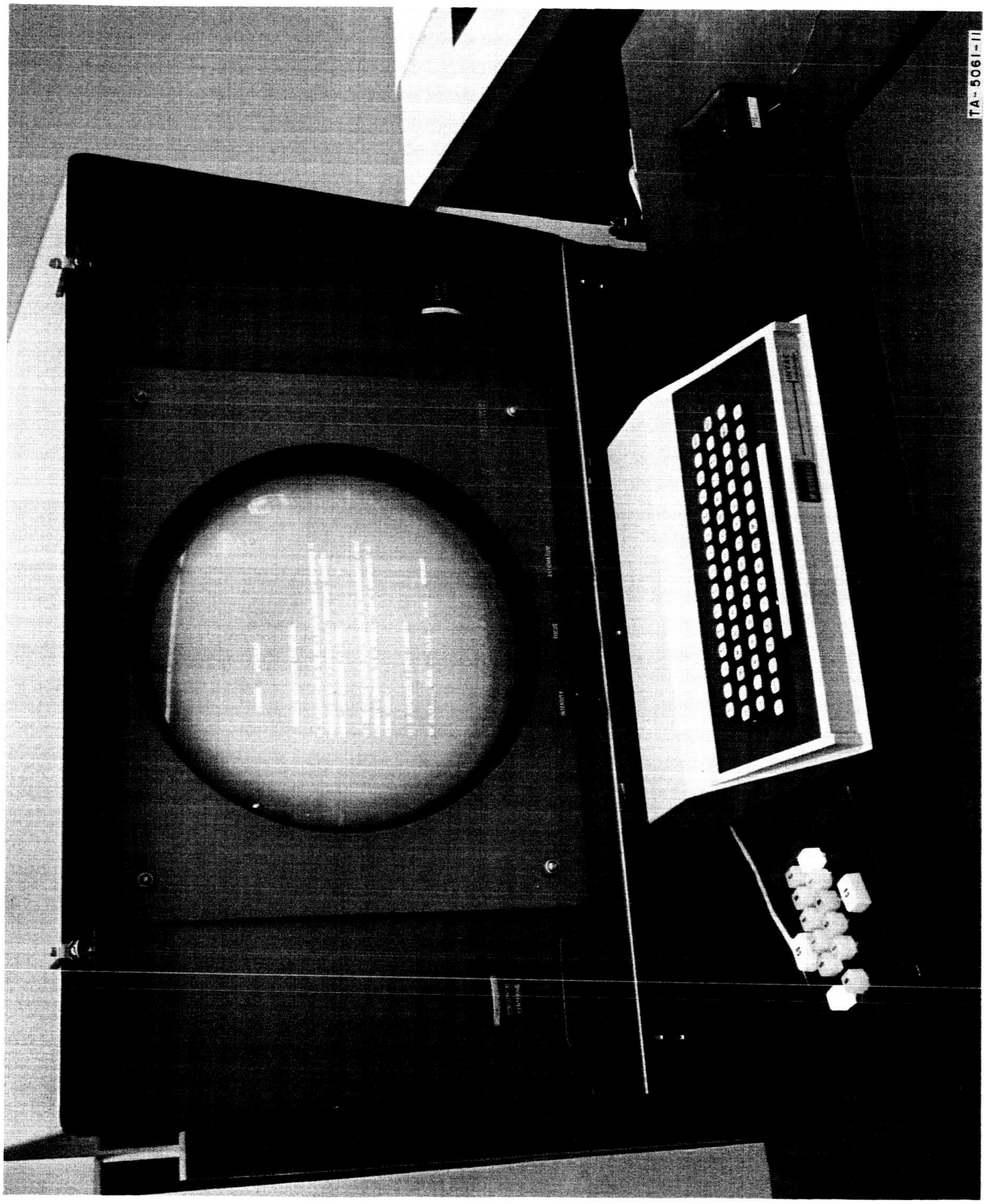


FIG. 1 THE ON-LINE SYSTEM WORK STATION SHOWING THE CRT DISPLAY, KEYBOARD, PUSHBUTTONS, AND MOUSE

same fashion. When using the light pen, the user holds the pen up to the screen and actually points at the text operands.

2b5 Further to the user's right is the on-line typewriter which is used only to convey to him certain kinds of questions and system error messages.

2c The function of these various devices becomes more apparent as they are discussed in succeeding paragraphs of this section. A more complete description of the various hardware devices is included in Appendix B of this report.

3 THE DISPLAY PRESENTATION

3a Figures 2a and 2b are photographs of the display screen, with sample text statements shown.

3a1 These photographs show a 40-character by 13-line working frame. Although we generally work with a 63-character by 16-line frame, for better detail we habitually reduce the frame count and enlarge the character when making photographs, slides and movies.

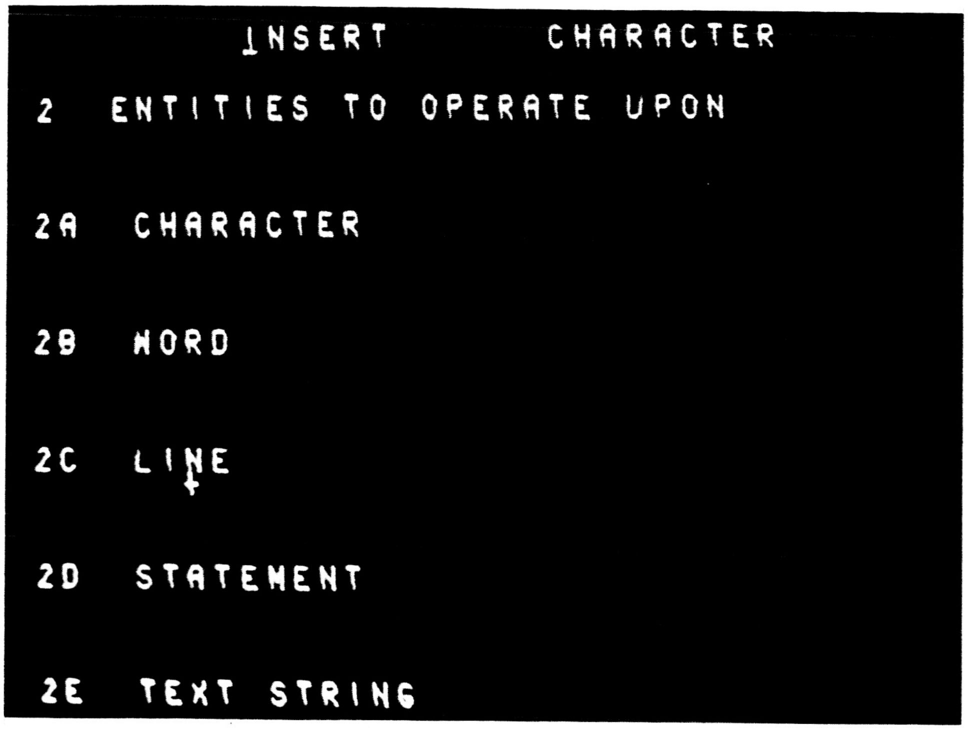
3a2 Our present display system shows only one "case" of alphabetic font on the screen. A character may be entered and held within the computer as either upper or lower case, but we use special conventions and procedures for working on-line with material (such as this report) when character case is important. New hardware and display conventions will soon provide us with directly displayed case for our alphabetic characters.

3a3 On the top line, the computer shows the user which command is now operative.

3a4 In Fig. 2a, the PLUS mark, positioned just below the "N" of Statement 2c, is the tracking spot or "bug," which the user may move freely about the working frame by operating one of the bug-positioning devices (mouse, joystick, Grafacon, etc.).

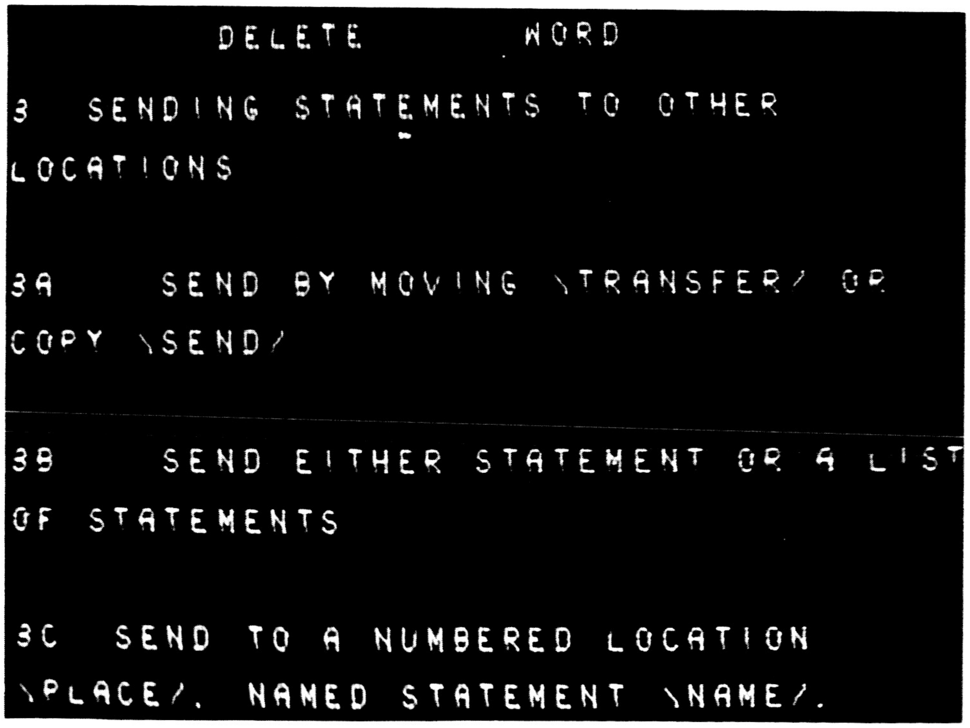
3a4a In this example (Figure 2a), the user may press the "select" button to designate the character "N" as the one behind which he wishes to insert an arbitarary string of characters which are entered from the keyboard.

3a5 In Figure 2b, the user has already made a bug selection by pushing the button when the bug was closest to the "E" character position and the computer verified this action by



```
          INSERT          CHARACTER
2  ENTITIES TO OPERATE UPON

2A  CHARACTER

2B  WORD

2C  LINE

2D  STATEMENT

2E  TEXT STRING
```

(a) INSERT CHARACTER. THE "BUG" IS UNDER THE CHARACTER "N" IN THE WORD "LINE".



```
          DELETE          WORD
3  SENDING STATEMENTS TO OTHER
LOCATIONS

3A    SEND BY MOVING \TRANSFER/ OR
COPY \SEND/

3B    SEND EITHER STATEMENT OR A LIST
OF STATEMENTS

3C  SEND TO A NUMBERED LOCATION
\PLACE/. NAMED STATEMENT \NAME/.
```

(b) DELETE WORD. THE WORD "STATEMENTS" HAS BEEN SELECTED.

FIG. 2 DISPLAY SHOWING TWO TYPES OF OPERANDS

underlining the selected character.

3a5a Pushing the "select" button a second time (or striking one of the equivalent "command accept" keys on the keyboard) will cause the word "statements" to disappear and the remaining text to close the gap; i.e., there will be room for "locations" on the end of this line, and the remainder of the working text will move up one line.

4 THE COMMAND-SPECIFICATION PLAN

4a In that a command-specification plan represents the "blueprint" for the design of a process, it can be presented in varying detail depending upon the needs.

4a1 In some of our analytic work (see Section III) we need to go into considerable detail to show the exact sequence of actions which are designed to provide a given process.

4a2 To present the essential sequence of operations to a user who wants the outline of the plan more than the details, we have a very simple form for presenting the command-specification plan.

4a2a This form of plan description is included in the command reportoire list together with the description of the commands function.

4a2b For each command, we list the mnemonic characters which are to be struck to designate the operator, and indicates the order in which they and the various types of operand are to be supplied, as a guide to specifying that command.

4b Special conventions for describing commands-specification plans in a command-repertoire list.

4b1 The specification plan for each command is presented below as a succession of character groups, each separated by a space. In this description the characters are capitalized, but in actual usage the case is not significant.

4b2 Each single letter represents the corresponding single alphabetic character, which is entered either from the keyboard or from the control panel.

4b3 "SP" represents a SPACE character, which is entered from the keyboard.

4b4 "C1, C2, ..., W1, W2, ..., L1, L2, ..., S1, S2, ..." represent the characters, words, lines, or statements specified by the user (the "display-entity operands"). Each of these entities is selected at command-specification time by using the bug-positioning device to select a single character within the entity.

4b5 "LIT" represents a literal input string. This string includes all characters which may be entered from the keyboard, including such non-printing characters as SPACE, TAB, and CARRETURN.

4b6 "NUMBER" represents a decimal integer entered from the keyboard as a numerical parameter for a command.

4b7 "CA" represents hitting the "Command Accept" key, on the keyboard or the control panel, in order to cause execution of the specified command. Alternatively, the "select" button on the bug-positioning device may be used to give a CA.

4c After operator and operands have been completely specified by the user, striking the "CA" ("Command Accept") key will cause the command to be executed immediately.

4d At any time during the specification process the user may negate the entire command by striking the "CD" ("Command Delete") key, on either the keyboard or the control panel.

4e In presenting our current on-line scheme, there are some detailed features of the command-specification plans that are useful to provide the reader, but which would be awkward to include in the "User's Guide" type of command-repertoire list. A discussion of these features follows the presentation of this list, and includes the means for providing computer feedback to the user during command designation, and the means for designating the operator, the literal-input, and the CRT-operand parts of the command.

5 THE COMMAND REPERTOIRE

5a The complete repertoire of commands available in our current on-line system includes the following general types of command:

5a1 Basic editing commands providing delete, insert, replace, move, and copy operations on characters, words, lines, statements, or text strings of arbitrary length.

5a2 Input and output commands, such as those for reading in paper tape, writing on magnetic tape, etc.

5a3 Commands for scanning forward and backward in the computer-held text.

5a4 Certain structure-related commands designed especially for working with our linked-statement structures (see Appendix A for structuring conventions); for instance, those which allow hopping to named locations, or renumbering within a list structure.

5a5 Utility commands for positioning the magnetic tape reel, clearing the working text space of the drum, and reporting system status.

5b The basic editing commands are summarized below, in our "User's Guide" form of description. These commands are the most heavily used in the task environment of primary concern so far in this project, and they serve throughout the body of the report as specific examples for the discussion.

5b1 Similar description for commands of the other four types may be found in Appendix B.

5c We find it useful to consider a command as having two main components, operator and operand.

5c1 Generally speaking, the operator specifies what is to be done, and the operands are what this is to be done to.

5c2 As our system developed, our usage of the term "operator" evolved into being a bit inconsistent with this general meaning.

5c2a In this report, one is to interpret "operator" as referring to the part of the command designation that is other than a bug selection or the entry of literal data from the keyboard (as in an "insert" operation).

5c2b Many of our "operators" contain information which qualifies the way the computer is to interpret the bug selections in determining the text operands upon which to operate. In this sense there may be operand-designation information in an "operator," which is where we have become inconsistent in our terminology.

5c2c As we learn more about the ways to convey the necessary command-designation information to the computer, we realize that the kind of techniques likely to emerge in our next system stages will force us to develop a different vocabulary to handle this and other important but (now) subtle distinctions.

5d EDITING COMMANDS AVAILABLE IN OUR CURRENT ON-LINE SYSTEM

5d1 The following are five groups of basic editing commands. For each command, the command-specification plan (see statement 4) is listed at the left, followed by a short description of the operation which is performed by the command.

5d1a Delete the designated entity, and close up the remaining text.

D T C1 C2 CA	Delete text, characters C1 through C2
D C C1 CA	Delete character C1
D W W1 CA	Delete word W1
D L L1 CA	Delete line L1
D S S1 CA	Delete statement S1

5d1b Insert LIT as indicated behind the designated entity. Rearrange prior text as required to make room.

I T C1 LIT CA	Insert LIT after character C1
I C C1 LIT CA	Insert LIT after character C1
I W W1 LIT CA	Insert SPACE LIT after last printing character of word W1
I L L1 LIT CA	Insert CARRETURN LIT after last printing character of line L1
I S S1 LIT CA	Insert CARRETURN CARRETURN LIT after last printing character of statement S1

5d1c Replace the designated entity with LIT, rearranging prior text as necessary.

R T C1 C2 LIT CA	Replace text-string characters C1 through C2 with LIT
R C C1 LIT CA	Replace character C1 with LIT
R W W1 LIT CA	Replace word W1 with LIT
R L L1 LIT CA	Replace line L1 with LIT
R S S1 LIT CA	Replace statement S1 with LIT

5d1d Move one designated entity to follow another. The moved entity is deleted from its original location. Other text is adjusted to close the deletion gap and open the corresponding insertion gap.

M T C1 C2 C3 CA	Move the text string, characters C2 through C3, to follow character C1
M C C1 C2 C3 CA	Move the text string, characters C2 through C3, to follow character C1

M W W1 W2 CA	Move word W2 to follow word W1
M L L1 L2 CA	Move line L2 to follow line L1
M S S1 S2 CA	Move statement S2 to follow statement S1

5d1e Copy one designated entity and insert it behind another. The copied entity remains unchanged. Prior text is rearranged to make room for new insertion.

C T C1 C2 C3 CA	Copy text string, characters C2 through C3, to follow character C1
C C C1 C2 C3 CA	Copy text string, characters C2 through C3, to follow character C1
C W W1 W2 CA	Copy word W2 to follow word W1
C L L1 L2 CA	Copy line L2 to follow line L1
C S S1 S2 CA	Copy statement S2 to follow statement S1

6 COMPUTER FEEDBACK

6a During the sequence of steps (including likely human errors) involved in designating a given command, the computer supplies the user with some highly useful feedback information.

6b At the top of the screen is the Computer Feedback Line (CFL), always reserved for feedback information.

6b1 On CFL is presented a description of the computer's "understanding" of the present state of operator designation.

6b1a Generally, the CFL contains a full-word description of one of the system operators, using words whose first characters are those keyed in by the user in designating the operator.

6b2 Either an "x" or a DASH is often placed by the computer under the leading character of one of the words in the CFL. This has special significance relative to the computer's state and to the next expected action of the user.

6b2a A DASH under the first character of the first word in CFL means that:

6b2a1 The computer will interpret the next keyboard-character input as an attempt to designate a new operator.

6b2a2 The last bug selection causes the DASH to disappear from CFL; it reappears only after this command is either executed (by striking CA) or negated (by striking CD).

6b2b A DASH under the first character of any other word in CFL means that:

6b2b1 The computer is currently interpreting keyboard input as designating the remainder of a partially designated command operator

6b2b2 The computer has "offered" the marked word as the most probable word next to be designated by the user

6b2b3 The computer will consider either a SPACE or a CA stroke as verification of its guess

6b2b4 Accepting the SPACE or CA is designed to save effort. One can also strike the alphabetic key corresponding to the first letter of that word (the normal designation means).

6b2c An "x" under the first character of a word in the CFL indicates that the last key struck by the user was not one which the computer could use either to verify that word or to re-designate a new one in its place.

6b2c1 The user is expected to hit an acceptable new character to designate which word he wants to see there.

7 OPERATOR DESIGNATION

7a The hardware devices for operator designation provide the user with the means for specifying which command of the repertoire he wishes to use.

7b In our current system, operators may be designated by using either of three hardware devices:

7b1 The user may strike the group of mnemonic characters at the keyboard (the case of alphabetic characters is not significant). The mnemonic sometimes includes a SPACE stroke.

7b2 He may strike the corresponding chords on an experimental 5-fingered binary keyset which, with a bit of practice, provides full alphanumeric input capability with one-hand operation. (Ref 2f(ENGELBART2))

7b3 Alternatively, he may use a special one-handed keyboard which is called as the "control panel." This panel has specially arranged pushbuttons for designating forward and backward scans, and for designating operations of delete, insert, replace, move, and copy, to operate on entities of character, word, line, statement, or arbitrarily delimited strings of text.

8 LITERAL INPUT

8a Literal input (including the numerical parameters required by certain commands) is entered from either the alphanumeric keyboard or the binary keyset. Numerical parameters may be any decimal integer; literal input strings of textual material may be of any length.

8b At the time during command specification when the computer is expecting literal input, it clears a space on the lower half of the display screen. The user sees his character-by-character input accruing in this space as he enters it from the keyboard. This literal input is terminated and the string put into the appropriate text location by a CA action.

8c During the time when literal input is being entered, hitting the BACKSPACE key will delete the last character of the literal string; and hitting the BACKSPACEWORD key (a special key provided on the keyboard) will delete the last word in the literal string. This feature provides a way of correcting errors in the input string.

8d The user does not need to designate the start of new display lines; he simply types without using the RETURN key. If the word he is entering reaches the end of a line on the display screen before its trailing SPACE is entered the computer automatically shifts that partial word to the beginning of the next line.

9 DISPLAY-ENTITY OPERAND SELECTION

9a The display-entity selection techniques provide the way of indicating which of the entities displayed on the CRT screen are to be used as operands for the current command. The types of entity operands are: character, word, line, statement, or "text" (a string of characters specified by its delimiting characters).

9b The user always selects a "character" entity (this may be a non-printing character). If an entity larger than a character is the expected operand, the computer will operate upon that entity which includes the selected character.

9c A moveable PLUS mark known as the "bug" is guided about on the display screen by means of one of the various available "bug-positioning devices." When the bug has been positioned near the desired character location, the user fixes its position by actuating the select switch on the device. The PLUS mark then becomes a DASH, under the selected character. If the wrong character was selected, the user may strike the CD key to negate the command and release the bug. (This action deletes all operands previously entered for that command.)

9d Operand-selecting devices available in our current system include the following bug-positioning devices: the "mouse"; the Grafacon; a joystick; and a knee control. A light pen has been available in an earlier version of the system, and will also be implemented in future versions. These operand-selecting devices are described in detail in Appendix B, and discussed further in Section IV.

1 INTRODUCTION

1a This section is the result of our search for organized ways of thinking about our display-control problem that might help us unearth significant possibilities for reducing execution time, and evaluate the relative advantages offered by competing possibilities.

1b To this end, our most promising approach seems to be associated with techniques for measuring, describing, and analyzing processes. We describe this approach in the following paragraphs of this section giving:

1b1 A form of primitive-process analysis

1b2 Conventions for "process plan" description

1b3 An example of a detailed plan description, for the joint man-computer process of deleting a word with our current scheme

1b4 A general treatment of nested process networks, with first-order analytic techniques for dealing with execution-time and branching probabilities

1b5 A discussion of the applicability of the techniques for aid in unearthing and evaluating improvement possibilities.

1b5a Design description

1b5b Analyses and measurement

1b5c Analyses and evaluation

1b5d Analyses and unearthing possibilities.

2 A ROUGH ANALYSIS OF SOME PRIMITIVE PROCESSES

2a Consider the following four human processes as being among the "primitive" in our display-control system:

2a1 With hands positioned on the keyboard, striking a key

2a2 With bug-control device in hand and being oriented as to relative bug position on the screen, selecting a text entity

2a3 After striking a character on the keyboard (with both hands positioned properly), grasping the bug-control device and becoming oriented as to screen position, ready to begin making bug selections

2a4 After making a bug selection on the screen, getting both hands positioned on the keyboard ready to start typing.

2b Suppose we had basic measurements of the average time for a person to execute each of these kinds of primitive processes, and we were working with the development of schemes involving the keyboard and a bug-control device.

2b1 We could apparently list the sequence of primitive processes involved in the process plan of a given command and sum their respective execution times to derive the probable execution time for the command.

2b2 In this way we could estimate the relative value of one proposed plan which represents the design of the means for designating a display-control operation compared to another plan for that operation, to decide which is more promising.

2c This approach to process-plan analysis develops threads with which is woven the pattern in this and the following two sections.

2c1 In this section, we first map out in detail the plan for one of our commands.

2c1a This requires a preliminary discussion of the particular conventions developed for this purpose.

2c1b The example reveals clearly that analyzing execution time for such plans will require more than simple summing of sequential primitive-process execution times. To accommodate the human errors and the differences in initial system state which are inevitably present in our working processes, a practical plan must involve decisions and a corresponding branching network of component processes.

2c2 We therefore develop, in a subsequent part of this section, an analytic approach for the type of complex organizations of primitive processes which represent the level of joint man-computer processes of concern to this project.

2c3 The experiments described in the following section were set up to measure the execution time for the rudimentary but critical processes of display selection, involving essentially a simple sequence of one end of the primitive processes of the first, third, and second types listed above.

3 CONVENTIONS FOR PROCESS-PLAN DESCRIPTION

3a The "plan" for a process is a description of how to execute the process. For instance, a complete set of flow-diagrams, or of source-code listings, represents a detailed plan for a computer-process.

3b The plan-description conventions which we have developed for detailed representation of joint man-computer processes are strongly indebted to earlier conventions for:

3b1 Linked-statement structuring; see (Appendix A, Section 3), as developed under ARPA sponsorship

3b2 Program-design records; see (Appendix A, Section 4), as subsequently developed under ESD sponsorship.

3c Some special tag conventions have been added, as described below.

3c1 Let the tag "*hp" designate a description statement for a human process.

3c2 Let the tag "*hsp" designate a description statement for a human sub-process--i.e., a "packaged" human process which can be called for execution from another human process, much as a closed subroutine is used in computer programming.

3c3 Let the tag "*cp" designate a description statement for a computer process.

3c4 Let the tag "*csp" designate a description statement for a computer sub-process--i.e., a closed sub-routine.

3c5 Let the tag "*cep" within a human-process description statement ST1, designate that the statements in the sublist of ST1 describe the "computer execution processes" which are called into play by the execution of the human process described in ST1.

3c5a The process described by such a sublist is considered finished either when an exit is called for in a statement, or when the last statement of the sublist is executed.

3c6 Let the tag "*c" designate a comment statement.

3c7 Let any untagged statement be considered as being of the same type as its source statement.

3d Special terminology and notation are used in our linked-statement plan description:

3d1 Upper-case letters and words are used to designate keyboard, pushbutton, or keyset actions:

3d1a A single upper-case letter represents striking that alphabetic key (or keyset code)

3d1b Non-alphabetic key strokes are designated by upper-case words: e.g. ONE, TWO,..., ASTERISK, SPACE,...,ETC.

3d1c CA represents striking one of the command-accept keys

3d1d CD represents striking the command-delete key.

3d2 Special representation for information found on CFL:

3d2a Let "txt" be used below to represent any word or string of words which might appear on CFL (but not the underwritten DASH or "x")

3d2a1 Txt1, txt2, etc. represent specific instances of txt.

3d2b Let "priorentity" be used below, in describing the contents of CFL, to designate the entity part of the operator that had been named in CFL before the last command exectuion.

3d2c Conventions for representing underwritten dash or "x" in CFL:

3d2c1 Let (-)txt indicate that the first character of txt is underwritten by a DASH.

3d2c2 Let (x)txt indicate that the first character of txt is underwritten by an "x."

3d2c3 Let (-,x)txt indicate that the first character of txt is underwritten by either a DASH or an "x."

4 (dw) *hsp DELETE A WORD (EXAMPLE OF DETAILED PLAN FOR A JOINT, HUMAN-COMPUTER PROCESS).

4a (dw) Assess current state of the process and branch to the appropriate next process. *c Branching is to (dw-clear), (dw-d), (dw-w), (dw-verify), (dw-select), and (dw-accept).

4a1 (dw) If CFL is "delete word," and a bug-fix mark is under one of the printing characters of the word to be deleted, to(dw-accept).

4a2 If CFL is "(-)delete word," to(dw-select).

4a3 If CFL is "delete (-,x)word," to(dw-verify)

4a4 If CFL is "delete (-,x)txt," to(dw-w)

4a5 If CFL is "(-)txt, to(dw-d).

4a6 Otherwise, to(dw-clear).

4b (dw-clear) Clear the system to its reference state, and return to(dw).

4b1 (dw-clear) strike CD, *cep

4b1a If CD was struck, change CFL to "(-)txtl," where "txtl" repesents the last completely designated operator shown in CFL, clear all bug-fix underline marks, put the bug on the screen, and exit.

4b1b If some other character was inadvertently struck, operate upon CFL and the working text according to the current state of the system and the character that was struck, and exit.

4b2 To(dw).

4c (dw-d) Designate "delete" operation, and return to(dw).

4c1 (dw-d) Strike D. *cep

4c1a If CFL is "(-,x)txt," if D was struck:

4c1a1 If "txt" contains "txtl," one of the operand entities "character, word, line, statement, or text," change CFL to "delete (-)txtl," and exit.

4c1a2 If "txt" does not contain one of these entities, change CFL to "delete (-)statement," and exit.

4c1b If CFL is "(-,x)txt," and if another key was (inadvertently) struck:

4c1b1 If CD, make CFL be "(-)txt," and exit.

4c1b2 If CA, fix bug mark, and exit.

4c1b3 If a valid first character for a command designation (whose first word is "txt1"), change CFL to "txt1 (-)txt2," and exit.

4c1b4 If any other character, make CFL become "(x)txt," and exit.

4c1c If CFL is not "(-,x)txt," change CFL, the bug, and the working text as appropriate to the state of the system and the character that was struck, and exit.

4c2 To(dw).

4d (dw-w) Designate "word" as operand entity, and return to(dw).

4d1 (dw-w) Strike W. *cep

4d1a If CFL is "delete (-,x)txt1," then:

4d1a1 If W was struck, change CFL to "(-)delete word," and exit.

4d1a2 If another key was inadvertently struck:

4d1a2a If CD, change CFL to "(-)txt2 txt1," where "txt2" was the operator of the last fully designated command (which would have had "txt1" as the entity part), and exit.

4d1a2b If a character acceptable for designating a valid entity (named "txt3") to be deleted by a delete command, change CFL to "(-)delete txt3," and exit.

4d1a2c If any other character, make CFL become "delete (x)txt1," and exit.

4d1b If CFL is not (delete (-,x)txt1," then change CFL, the bug, and the working text as appropriate to the system state and the key that was struck.

4d2 To(dw).

4e (dw-verify) Verify "word" as operand entity, and return to(dw).

4e1 (dw-verify) Strike CA, SPACE, or W. *cep

4e1a If CFA is "delete (-,x)word," then:

4e1a1 If CA, SPACE, or W was struck, change CFL to "(-)delete word," and exit.

4e1a2 If another key was (inadvertently) struck:

4e1a2a If CD, change CFL to "(-)txt2 word," where "txt2" was the operator of the last fully designated command, and exit.

4e1a2b If a character acceptable for designating a valid entity (named "txt3") to be deleted by a Delete command, change CFL to "(-)delete txt3," and exit.

4e1a2c If any other character, make CFL become "delete (x)word," and exit.

4e1b If CFL is not "delete (-,x)word," then change CFL, the bug, and the working text as appropriate to the state of the system and to the key which was struck.

4e2 To(dw).

4f (dw-select) Select operand word from the display, and return to (dw).

4f1 (dw-select) Position bug over one of the printing characters of the word to be deleted.

4f2 Hit CA key ("select" button is equivalent). *cep

4f2a If CFL is "(-)delete word," then:

4f2a1 If CA was hit, locate the character nearest the bug mark and underline it, remove the bug mark from the display, change CFL from "(-)delete word" to "delete word," and exit;

4f2a2 If CD was hit, do nothing but exit.

4f2a3 If some other character was struck which is acceptable for designating the first word (txt1) of a new command, change CFL to "txt1 (-)txt2" (where "txt2" is appropriate to the prior state and to the character which was struck), remove the bug from the screen, and exit.

4f2a4 If any other character was struck, change CFL to "(x)delete word," remove the bug from the screen, and exit.

4f2b If CFL is not "(-)delete word," then change CFL, the bug, and the working text according to the state of the system and the character which was struck, and exit.

4f3 Go to(dw).

4g (dw-accept) Signal the computer to accept the command as designated. -to(dw)

4g1 (dw-accept) Strike CA. *cep

4g1a If CFL is "delete word," (indicating that the bug is not on the screen and that a bug mark exists under some character in the working text), then:

4g1a1 If CA was struck, delete the marked word and exit.

4g1a1a Beginning with the first character before the marked character, search backward for the first non-printing character, C1.

4g1a1b Beginning with the first character after the marked character, search forward for the first non-printing character, C2.

4g1a1c Remove all characters between C1 and C2 and remove one SPACE if either C1 or C2 is a SPACE, then close up the resulting text-string gap.

4g1a1d Move new words to this line from its successor line, and to the successor from its successor, etc., until the line formatting is readjusted down to the first occurrence of a fixed line-start point.

4g1a1e Make CFL be "(-)delete word." clear the bug mark from the screen. put the bug back on the screen, and exit.

4g1a2 If CD was inadvertently struck, make CFL be "(-)delete word," clear the bug-select underline from the screen, put the bug back on the screen, and exit.

4g1a3 If any other character was inadvertently struck, do nothing but exit.

4g1b If CFL was not "delete word," with a bug mark under a character in working text, change CFL, the bug, and the working text according to the state of the system and the character that was struck, and exit.

4g2 If nothing was deleted from working text, to(dw).

4g3 Otherwise, exit from the Delete Word process.

5 GENERAL TREATMENT OF PROCESS NETWORKS

5a Introduction:

5a1 The following generalized treatment develops some of the necessary concepts and calculation techniques for studying and evaluating processes.

5a2 We consider a process to be composed of a set of lower-order "component processes," organized into a "process network."

5a3 These component processes may be represented as "nodes" of the network, inter-connected by directed "branches" that indicate which component processes ("nodes") may be executed before or after others.

5a4 The following discussion develops techniques for evaluating probable execution times for different process designs.

5a4a To make a process more efficient, one must determine how the exectuion time for the total process (represented by a network) depends upon the execution time of its component processes.

5a4b One must also be able to estimate execution time from analysis of the process design, as opposed to implementing and measuring.

5b Some general parameters of process networks:

5b1 System state:

5b1a The system of things involved in the processes being described and analyzed will undergo changes in various characteristics and parameters as these processes are executed.

5b1b At any point in the execution of a process, the condition of some of these characteristics and parameters is directly relevant to the processes; either affecting, or being changed by the system modifications and the sequence of the processes.

5b1c The term "system state" (or simply "state") will refer to the condition of these relevant characteristics and parameters at a given point in time or process.

5b2 Flow probability:

5b2a This term refers to a "probable flow" of control through a node or along a branch. At the point in the network where the "flow" is described, it involves both the probable number of times that control will pass this point (which can be fractional) and the probable distribution of system states..

5b2b For a given distribution of system states at the entry to a network, the expected flow at the entry of each of the component nodes ("N1," "N2," etc.) within the network will be represented by"F1," "F2." etc.

5b3 Node Branching Probability:

5b3a Some nodes will have more than one exit branch, which is the source of the major problem to the analysis of our processes.

5b3b A complete description of the process represented by such a node must specify the basis upon which a decision is made during its execution, as to which of the exit branches will be taken.

5b3c The analysis we are developing here treats this branching as a matter of "branching probability."

5b3d For a given node, this probability can be expressed meaningfully only with an entry flow having a known distribution of relevant system states.

5b3e For the general distribution of system states expected in the entry flow to a network, the probability of emerging via Branch x from Node n will be Pnx.

5b4 Process times in a branching node:

5b4a A "branching node," will generally have a different processing time for an execution instance emerging from one exit than for an instance emerging from another of its exits.

5b4b The execution time for any node is dependent upon the distribution of states in the entry flow to the node.

5b4c For a given network, when the generally expected distribution of entry-flow system states prevails, the execution time for Node n when merging from Exit x, is expressed as Tnx.

5b4d *c For a non-branching node, execution time is expressed simply as Tn.

5b4e Probable execution time of a node is calculated for an entry flow of 1 (i.e., unity probability of passage).

5b5 Process time for a network:

5b5a As for a node, the process time for a network is calculated for an entry flow of unity.

5b5b To calculate network process time, the probable flow through each entry-exit path of each node in the network must first be determined for the particular distribution of network-entry flow states assumed.

5b5c For each exit of each node, its contribution to probable network execution time is the product of: node-entry flow probability, braching probability to that exit, and probable execution time to that exit.

5c Network Analysis:

5c1 A serial-chain non-branching network (see Figure 3a):

5c1a The network entry branch coincides with the entry branch of Node 1.

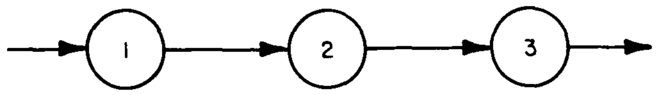
5c1b The nodes are executed in direct succession.

5c1c The exit branch of Node 3 is the exit branch of the network.

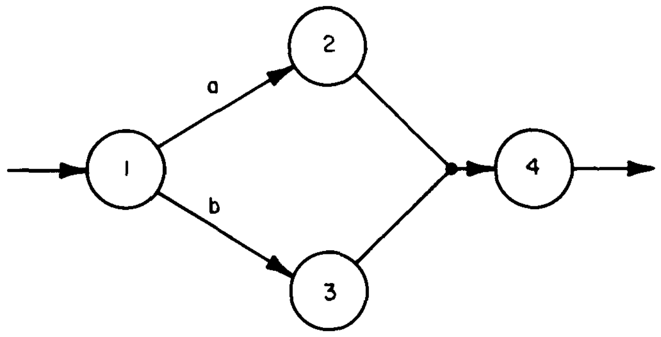
5c1d If execution times for the respective notes are T1, T2, and T3, the time for the network is Tn = T1 + T2 + T3.

5c2 Simple forward-loop network (see Figure 3b):

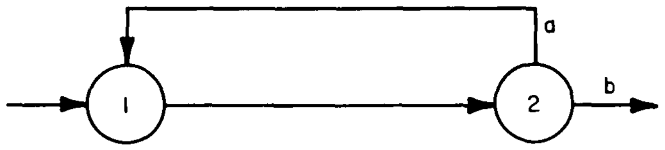
5c2a Flow distribution would be:



(a) A SERIAL-CHAIN NON-BRANCHING NETWORK



(b) A SIMPLE FOWARD-LOOP NETWORK



(c) A SIMPLE BACKWARD-LOOP NETWORK

TA-5061-13

FIG. 3 SAMPLE PROCESS NETWORK CONFIGURATIONS

5c2a1 F1 = F4 = Fn

5c2a2 F2 = P1a Fn

5c2a3 F3 = P1b Fn

5c2b The probable network time would be Tn = P1a(T1a + T2 + T4) + P1b(T1b + T3 + T4).

5c3 A simple backward-loop network (see Figure 3c):

5c3a The probable execution time for this network would be derived from the following equation:

5c3a1 Tn = F1T1 + F2P2aT2a + F2P2bT2b

5c3b To derive the network flow values:

5c3b1 For a given network entry flow, Fn, the following simultaneous equations obtain:

5c3b1a F1 = Fn + P2aF2

5c3b1b F2 = F1

5c3b2 The solution of these equtions is:

5c3b2a F1 = F2 = Fn/(1-P2a)

5c3c Probable execution time for the network, expressed in terms of execution times and branching probabilities of the individual nodes (based upon Fn = 1), becomes:

5c3c1 Tn = (T1 + P2aT2a + P2bT2b)/(1-P2a)

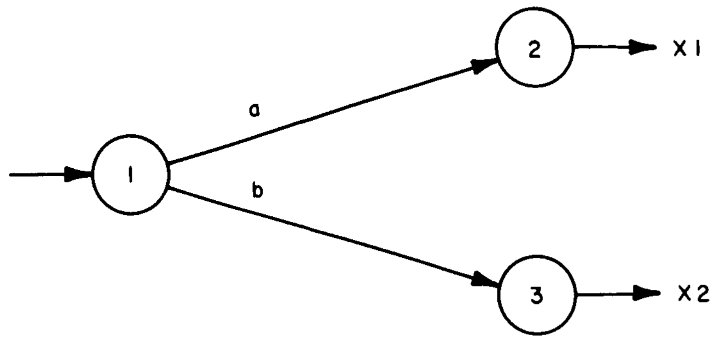
5c3d If P2a is appreciable, it has a significant effect upon the probable flow around the loop, and thus upon the network-execution time.

5c4 A simple two-exit network (see Figure 4a):

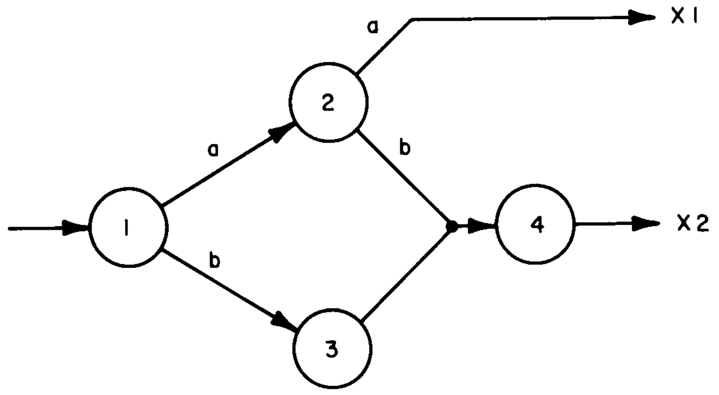
5c4a The exit probability for the network exactly corresponds to the branching probability of Node 1.

5c4b The execution time for the network is dependent upon which exit path is taken.

5c4b1 If unit flow into the network all left via Exit 1, the execution time would be Tn1 = T1a + T3, which is the probable execution time for Exit 1 of the network.



(a) A SIMPLE TWO-EXIT NETWORK



(b) A MORE COMPLEX TWO-EXIT NETWORK

TA-5061-14

FIG. 4 SAMPLE PROCESS NETWORK CONFIGURATIONS

5c4b2 For Exit 2, a probable network execution time would be Tn2 = Tnb + T2.

5c5 More complex two-exit network (see Figure 4b):

5c5a Calculation of exit probabilities:

5c5a1 Set up the basic node equations for the network:

5c5a1a F1 = Fn

5c5a1b F2 = P1aF1

5c5a1c F3 = P1bF1

5c5a1d F4 = P2bF2 + F3

5c5a2 Upon solution of these equations for F1, F2, etc., the value of P2af2 will represent the probability (Pn1) of emerging from the network via Exit 1, and F4 will represent the probability (Pn2) of emerging via Exit 2.

5c5a3 In the above network, therefore, Pn1 = P1aP2a, and Pn2 = P1b + P1aP2b.

5c5b Calculation of probable process time for each exit.

5c5b1 The probable process time for a given exit is the probable time spent within the network, for unit flow into the entry point of the network and unit flow from the given exit. (This means no flow from the other exits.)

5c5b2 Obviously, for this condition to prevail, one or more individual nodes will have different branching probabilities from the "normal" case.

5c5b2a From among the probable network-entry flow states, there will be a subset destined to cause the desired exit.

5c5b2b The expected distribution of these states will produce this "different" set of branching probabilities within the network.

5c5b3 The execution time for nodes, whether branching or not, may also be different with this different distribution of network-entry flow states.

5c5b4 We may designate that a time or a branching variable of a node is associated with the network-entry conditions giving exit only via Exit n by appending a ".n" to its normal representation, e.g., P1a.2, P1b.2, T1a.2, T1b.2 for Node 1 (see Figure 4b), if only Exit 2 is possible.

5c5b5 In the above network, for Exit 1, these probabilities obviously must be: $P1a.1 = P2a.1 = 1$, and $P1b.1 = P2b.1 = 0$.

5c5b5a The flow solutions will be: $F1 = F2 = Fn$; $F3 = F4 = 0$.

5c5b5b Probable network time to Exit 1 would be $Tn.1 = T1a.1 + T2a.1$

5c5b6 For exit 2, it is obvious that $P2a.2 = 0$ and $P2b.2 = 1$; but the process within Node 1 would have to be studied to determine the particular values of P1a.2 and P1b.2 which would exist for this new class of network-entry states.

5c5b7 The probable network time to Exit 2 would then be $Tn.2 = P1a.2(T1a.2 + T2b.2) + P1b.2(T1b.2 + T3.2) + T4.2$.

5c5c In the general case, to determine the necessary node probabilities and times, one would need to examine network representations of each node in a manner such as above -- which may lead to lower-level node analysis, etc.

6 OUR POTENTIAL UTILIZATION OF THIS TYPE OF ANALYSIS:

6a We assume that any process network we are directly concerned with will decompose, within a few levels, into primitive nodes whose time and branching characteristics we can establish experimentally.

6b We hope thus to be able to calculate probable execution time, for a display-control process, with a reasonable degree of accuracy -- at least enough accuracy to estimate relative value among proposed design variations.

6c From this type of analysis, we also hope to derive useful guidance as to the experimentation which will be really relevant to our purpose, and which can teach us the most with the least cost.

1 INTRODUCTION

1a This chapter describes the computer-aided experimentation and analysis we performed in order to begin getting empirical data helpful for comparative evaluation of alternative schemes and their components.

1a1 We chose rather simple operations to measure because we thought it best to begin learning about meaningful measuring by starting with a simpler problem, dealing with only one scheme component.

1a2 The component we chose for consideration--techniques and devices for display-entity operand selection--is a major independent component in any display-control scheme, and is readily isolated for purposes of comparative testing.

1a3 Now that our conceptual and analytical approach (see section III) can begin to benefit from experimental support, we find that we have a good set of computer aided measurement and analysis techniques to begin providing this service.

1b The tests simulated the general situation faced by a user of our on-line system when he must interpose a screen-selection operation into his on-going working operations.

1b1 He has generally been entering information on the typewriter-like keyboard.

1b2 To begin making the screen selection, his right hand leaves the keyboard and takes hold of ("accesses," in our terminology) the selection device.

1b3 By moving this control he positions an associated tracking mark on the screen over the "target" text entity.

1b4 He then actuates a pushbutton associated with the particular device, to tell the computer that he is now "pointing at" the target entity.

1b5 The computer puts a special mark under the entity which it determines as having been selected, to give the user an opportunity to see when a mistaken selection was made.

1c We designed and conducted our experiments in order to learn more about the following characteristics of the operand-selecting devices currently available in our on-line system:

1c1 The comparative speed with which they could be used to select material on the display screen. Two kinds of time period were measured:

1c1a "Access time": the time it takes for the user to move his hand from the keyboard to the operand-selecting device.

1c1b "Motion time": the time period beginning with the first movement of the bug and ending with the "select" action fixing the bug at some particular character position.

1c2 The comparative ease with which an untrained user could become reasonably proficient in using the various devices.

1c3 The comparative error rates of the various devices.

1d There were three things we were trying to learn by carrying out these experiments:

1d1 First, the experiments should indicate which of our currently available operand-selecting devices seem most satisfactory. Although we did not expect that any one device would perform more satisfactorily than all the others in every respect, we at least hoped to begin determining which devices might be best suited for special purposes.

1d2 We also hoped to learn more generally about the performance features of display-entity operand selecting devices, their comparative advantages and disadvantages, so that we could get a better idea of what would constitute an "ideal" device for effective communication between a human user and a computer. This would allow us to begin developing better devices than are currently available.

1d3 Finally, we hoped to learn more about the kinds of meaningful measurements and analysis involved in empirically evaluating this particular scheme component, and we hoped to use this new knowledge in developing a sound methodology for experimentally evaluating alternative display-control schemes.

2 DESCRIPTION OF THE EXPERIMENTS

2a The experiments were designed to test the various operand-selecting devices under conditions similar to those that the user would encounter when actually working on-line. However, certain features of the live working conditions were not closely related to the actual efficiency of the operand-selecting devices, such as the need to enter literal input from the keyboard, the need to designate commands, and the user's indecision in selecting which display-entity to select--so we tried either to eliminate these features from the experimental environment, or to fix them in some standard way throughout the experiment.

2a1 Two different kinds of display-entity "targets" were presented in the experiments: "word" targets, and "character" targets. The target patterns presented to the subject were configurations of x's rather than actual text so that the subject could recognize his target entity immediately, and to simplify design of the experiment.

2a1a A configuration simulating the "character mode" operation of the system consisted of nine x's, in a three by three array, with the array as a whole randomly placed on the display. The specific target entity was the middle x (see Figure 5a).

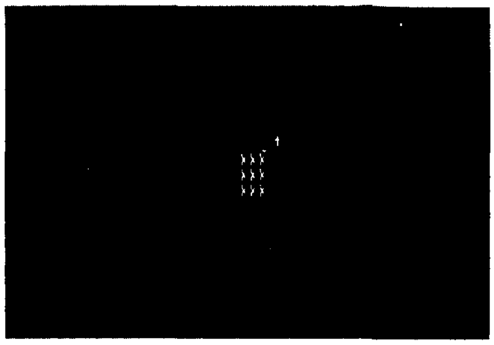
2a1b A configuration simulating the "word mode" operation of the system consisted of nine groups of five x's each, in a three by three "word" array, with the array as a whole randomly placed on the display. The target entity was any one of the five middle x's (i.e., any character in the middle "word"; see Figure 5b).

2a2 The subject was given a series of tests with each of these two types of target, and was to perform the following task sequence:

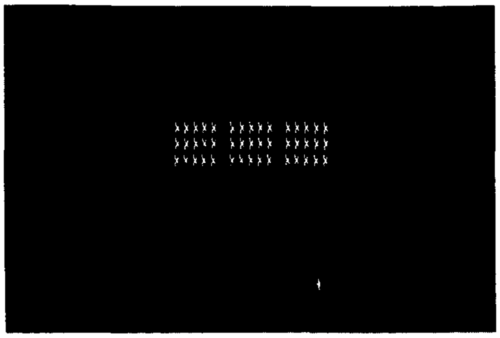
2a2a When the target appeared on the display screen, the subject was to strike the keyboard space-bar with his right hand, causing the bug to appear on the display. (Requiring that he use his right hand for both the space bar and the operand-selecting device made the experimental task more similar to the actual on-line environment, where the user would often have both hands at the keyboard before moving to the operand-selecting device. It also gave us a way of measuring the access times for the various devices.)

2a2b The subject was then to move his hand to the bug-positioning device being tested, and use it to guide the bug to the target entity on the display.

2a2c When the bug and the target coincided the subject was to "fix" the bug at that location, using the select switch of the bug-positioning device. An incorrect selection was signalled by a bell, and the incorrectly selected entity was underlined in the displayed target

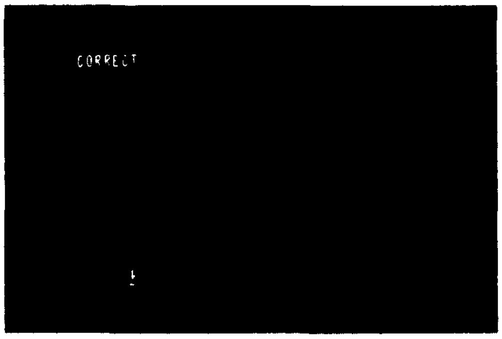


(a) "CHARACTER MODE" OPERATION SHOWING THE TARGET (MIDDLE X) AND BUG (PLUS SIGN)



(b) "WORD MODE" OPERATION. THE TARGET IS THE MIDDLE FIVE X's

(c) AN INCORRECT SELECTION IS UNDERLINED. THE CONFIGURATION OF X's AND THE BUG REMAIN ON THE DISPLAY



(d) A CORRECT SELECTION. THE POSITION OF THE TARGET IS INDICATED BY THE BUG MARK AND UNDERLINE

FIG. 5 TARGETS USED TO EXPERIMENTALLY EVALUATE THE OPERAND-LOCATING DEVICES AND RESULTS OF AN INCORRECT AND CORRECT SELECTION
Each Picture Shows Approximately Ten Percent of the Display Surface.

pattern (see Figure 5c); the subject was then to relocate the bug and reselect the target entity. A correct selection caused the target to disappear, and the word "CORRECT" to appear on the display screen (see Figure 5d). About three seconds later, the next target pattern was displayed (in some new randomly-determined position), and the process was repeated.

2a2d When the light pen rather than a bug-positioning device was used, the task sequence was much the same: after the target appeared, the subject was to strike the keyboard space bar with his right hand, then grasp the light pen and point it at the target entity (with the aid of the "finder beam," a circle of orange light projected from the end of the light pen to indicate which area the pen was currently detecting). The subject "fixed" his choice by depressing the select switch on the light pen. Correct and incorrect selections were signaled in the same way as with the bug-positioning devices.

2a3 There were two groups of subjects: eight "experienced" subjects who were already somewhat familiar with the on-line system, and three "inexperienced" subjects who had never before used either the system or the particular devices being tested. The experienced group were given experiments to test the devices after a reasonable amount of practice. The inexperienced group were tested to see how quickly and how well they learned to use the devices without previous practice.

2a3a For the experienced subjects, the entire testing procedure was broken into two time periods and it proceeded as follows:

2a3a1 The subject was given a brief explanation of the experiment and the target patterns.

2a3a2 He was then given his first device and allowed to practice with it for about two minutes.

2a3a3 Next he was tested using this first device, in both the "word" mode and the "character" mode of selection. Thirty-two targets of each type were presented.

2a3a4 After a two-minute rest period, the subject was given his second device and allowed to practice with it for about two minutes. He was then tested with this device; again, with 32 targets of each type.

2a3a5 This same sequence of rest, practice, and testing was carried out for each of the devices being tested. This constituted the first time period of the experiment.

2a3a6 During the second time period, the subject proceeded backward through the list of devices, begining with the last device he had used in the previous time period, then using the next-to-last device, and so on.

2a3a7 Each subject began with a different device and was presented with devices in a different order.

2a3b For inexperienced subjects, the experimental procedure was somewhat different:

2a3b1 The subject was given an explanation of the experiment, the target patterns, and the way the particular operand-selecting device worked. He was allowed to get the feel of the device, but was not given a practice period. He was then presented with ten sequences of eight target-patterns each, in the "character" mode. With each of these ten test sequences, of course, the subject gained a little more practice with that particular device; so each test sequence was taken as establishing a point on the subject's "learning curve" for that particular device.

2a3b2 This procedure was followed for each of the devices being tested.

2a3b3 Each subject began with a different device, and was given a different order of devices to work with.

2a4 The computer was used extensively in conducting these experiments; for presenting target patterns, signalling of correct and incorrect selections, determining the (random) position of the next target pattern, determining the short time-delays between a correct selection and the presentation of the next target, etc. In addition, for each presentation-selection event, the computer recorded the following information on magnetic tape for later analysis:

2a4a The position of the bug (in relation to the target entity) was recorded each 10 milliseconds.

2a4b The time the subject hit the space bar, and the times he made either a correct or an incorrect entity selection, were recorded and appropriately tagged to aid in identifying these significant points in the later data analysis.

2a5 The length of the experimental runs; the rest periods allowed between runs; the order in which the various devices were tested; and the modes of operation ("character" or "word" targets) were controlled by the person conducting the experiments.

2b The types of operand-selecting devices tested varied between the experienced subjects and the inexperienced. A total of five different hardware devices were tested, one of which operated in either of two modes.

2b1 The devices included the light pen, and four types of bug-positioning devices: a joystick, a "mouse", a Grafacon, and a knee control. All these devices are described in Appendix B of this report.

2b1a For these tests we devised a simple magnetic mounting for the light pen to hold it at the edge of the display screen when not in use. With a little practice, a subject soon could pick the pen off this magnetic "hook" as his hand moved from the keyboard to the screen, or deposit the pen there on the way back, with relatively little delay or probability of fumbling.

2b1b The joystick was used in two modes of operation (making it, in effect, two different bug-positioning devices): an "absolute" mode, in which the bug position is proportional to the stick's deflection from center, and a "rate" mode in which the bug velocity is proportional to the stick's deflection from center.

2b2 Our reason for giving a different set of test devices to the experienced and the inexperienced groups respectively, is that we wanted the experienced subjects to test only those devices with which they were familiar; i.e., those devices which either were a part of the current on-line system or had been incorporated in some earlier version of that system. This inluded the joystick (absolute mode); mouse; Grafacon; and light pen (although the light pen is not implemented in the current system, it was available in an earlier version of the system). The knee control, though available in the system now, had not been incorporated at the time the experiments were begun. All five devices, plus the rate-mode operation of the joystick, were presented to the inexperienced group of subjects.

3 DESCRIPTION OF THE DATA ANALYSIS

3a The analysis software was designed to allow flexibility in studying individual performance curves and results. This software provided operator commands for scanning the recorded data on the magnetic tape, selectively printing out results, producing CRT-displayed curves of each subject's performance, and calculating certain averages over a block of tests.

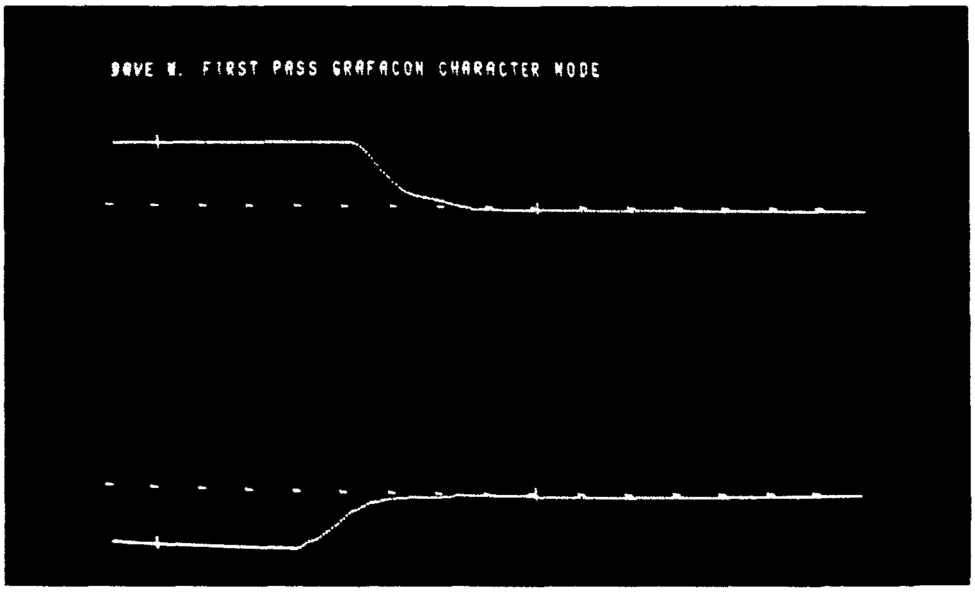
3a1 Tape-handling operations, controlled by commands from the on-line keyboard, facilitate searching through the data recorded on the magnetic tapes. These commands allowed one to scan forward or backward by one 32-target block of tests (or, an 8-target block, in the records for inexperienced subjects); and, within that block, to scan forward or backward one target (i.e. one presentation-selection event) at a time.

3a2 For each target-fix, the CRT could display a graph showing the bug's distance from its target entity as a function of time. This was displayed as two curves (see Figure 6), one showing variation with time of horizontal distance, and the other of vertical distance. The time-count was begun when the target appeared on the display. Vertical lines on the curves mark the time at which the space bar was struck and the time at which the target was correctly selected. Incorrect selections are shown as x's on the curve.

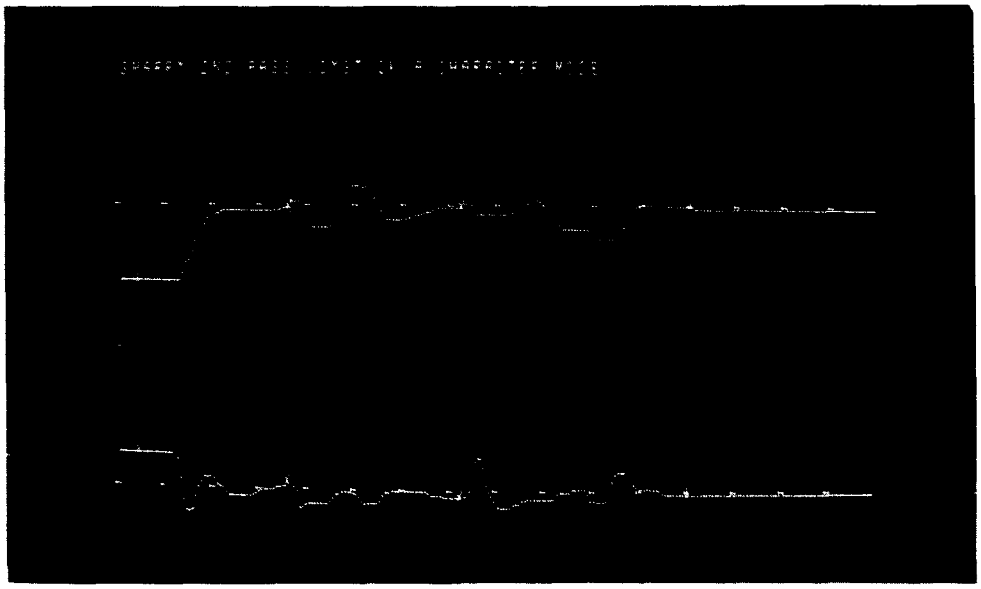
3a2a Figure 6 presents two examples of these curves. Figure 6a shows a typical performance curve for the Grafacon; Figure 6b shows an example of joystick performance in which the subject made several errors before selecting the correct target entity.

3a2b When viewed on-line on the CRT display, the scale of these curves can be changed by keyboard-entered commands which independently change either the distance or the time scale. This time scale change feature was included because of the radical variations in the times, among various devices and various subjects. The distance scale change allows detail examination of performance when the bug is near the target.

3a3 When studying a given target-fix event, the experimenter can, if he wishes, initiate output (to the on-line typewriter) of performance data: the time at which the space bar was struck, the time at which the bug movement began, the time at which the target was correctly selected, and the number of errors (incorrect selections) made. This software also computed and printed out the following incremental



(a) SPACE BAR AND CORRECT SELECTION TIMES ARE INDICATED BY VERTICAL DASHES.
HORIZONTAL SCALE: 2.5 SECONDS FULL SCALE
VERTICLE SCALE: NORMAL (DISTANCE BETWEEN THE AXES REPRESENTS TWICE THE SCREEN WIDTH)



(b) THE X's MARK INCORRECT SELECTIONS. THIS PHOTOGRAPH EMPHASIZES THE DIFFICULTY IN CONTROLLING THE JOYSTICK IN "CHARACTER MODE" OPERATION.
HORIZONTAL SCALE: 10 SECONDS FULL SCALE
VERTICAL SCALE: 4 TIMES NORMAL SIZE

FIG. 6 ANALYSIS CURVES OF THE EXPERIMENTS. The Top Curve in Each Set Shows the Vertical Distance of the "Bug" from the Target as a Function of Time. The Bottom Curve in Each Set Shows the Corresponding Horizontal Distance.

times; The access time (from the time the space bar was struck until the time the bug movement began, measuring how long it took the subject to move his hand from the keyboard to the device); the motion time (from the time the bug began moving until the time the target was correctly selected); and total time (from the time the space bar was struck until the time the target was correctly selected--i.e., the sum of access time plus otion time).

3a4 Finally, there is another command which causes the computer to search through a 32-target block of target fixes and compute (for output to the on-line typewriter) the average incremental times, and total number of errors, for that block.

3b The CRT curves of distance-vs.-time could be scanned with the on-line system, in order to determine where the subjects spent most of their time; how much time they spent in actually selecting the target entity after the bug was already positioned correctly; whether the errors seemed more predominant in one direction than in another (horizontally or vertically); and other such detailed information relating to individual performances.

3c The numerical averages computed with the help of the rest of the analysis software were collected and summarized as experimental results, presented in the following description.

4 EXPERIMENTAL RESULTS

4a Summary data: Figures 7 through 9 contain the bar charts comparing the various operand-selecting devices with respect to the time required for a correct selection.

4a1 Figures 7 and 8 are taken from the results of the eight experienced subjects, some of whom were very familiar with the on-line system and had used the devices often. Figure 7a shows the average total time (for all experienced subjects) required for a correct selection of the "character" target, with no penalty for errors; Figure 7b shows the results of the same tests with a 30% penalty for errors. Figures 8a and 8b, respectively, show the same for the "word" target.

4a1a The 30% error penalty is an approximate figure arrived at by the following argument; if a user wished to correct an incorrectly selected operand, he would need to strike the "Command Delete" key with his other hand before re-attempting a correct operand selection. This would take about as long as the time required to strike the space bar when the target first appeared. From the

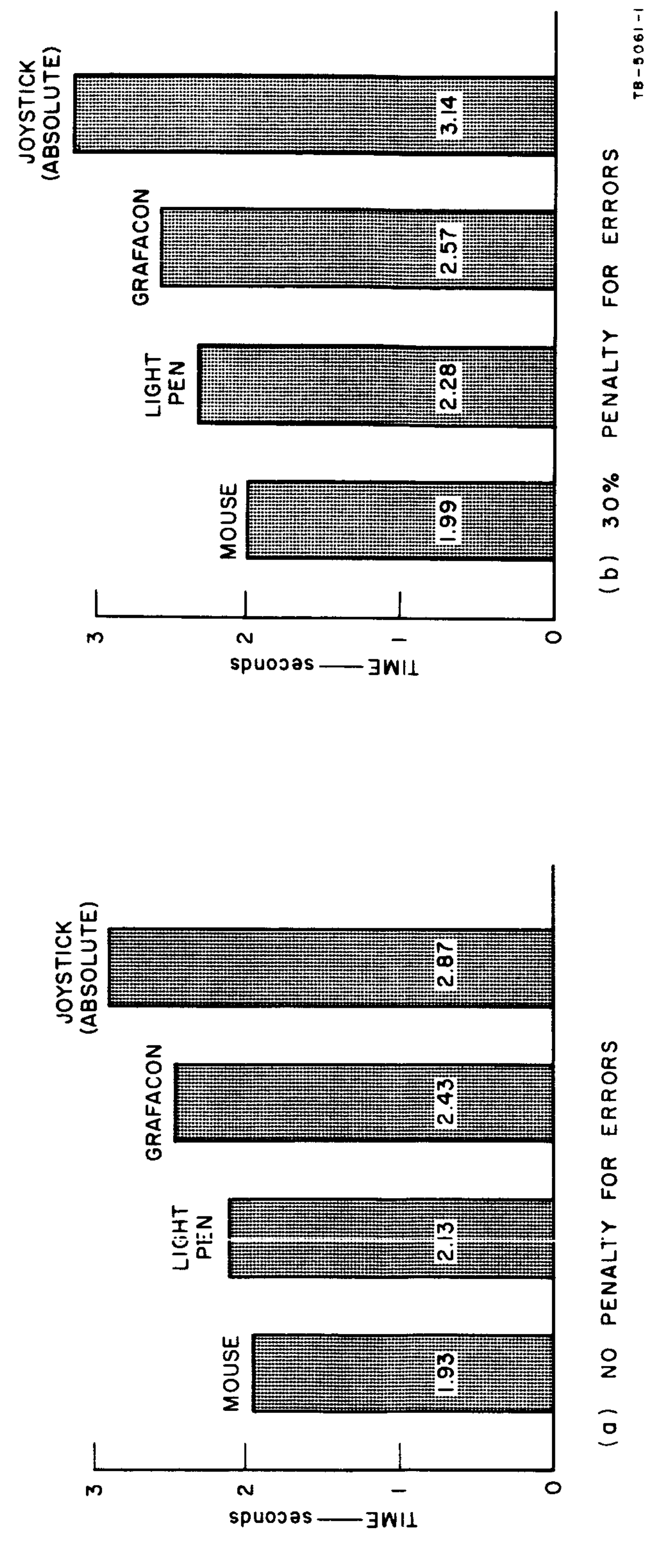


FIG. 7 COMPARISON OF THE OPERAND-LOCATING DEVICES FOR "EXPERIENCED" SUBJECTS, "CHARACTER MODE" OPERATION. Vertical Scales Indicate the Total Average Time (Seconds) from Space Bar to Correct Selection.

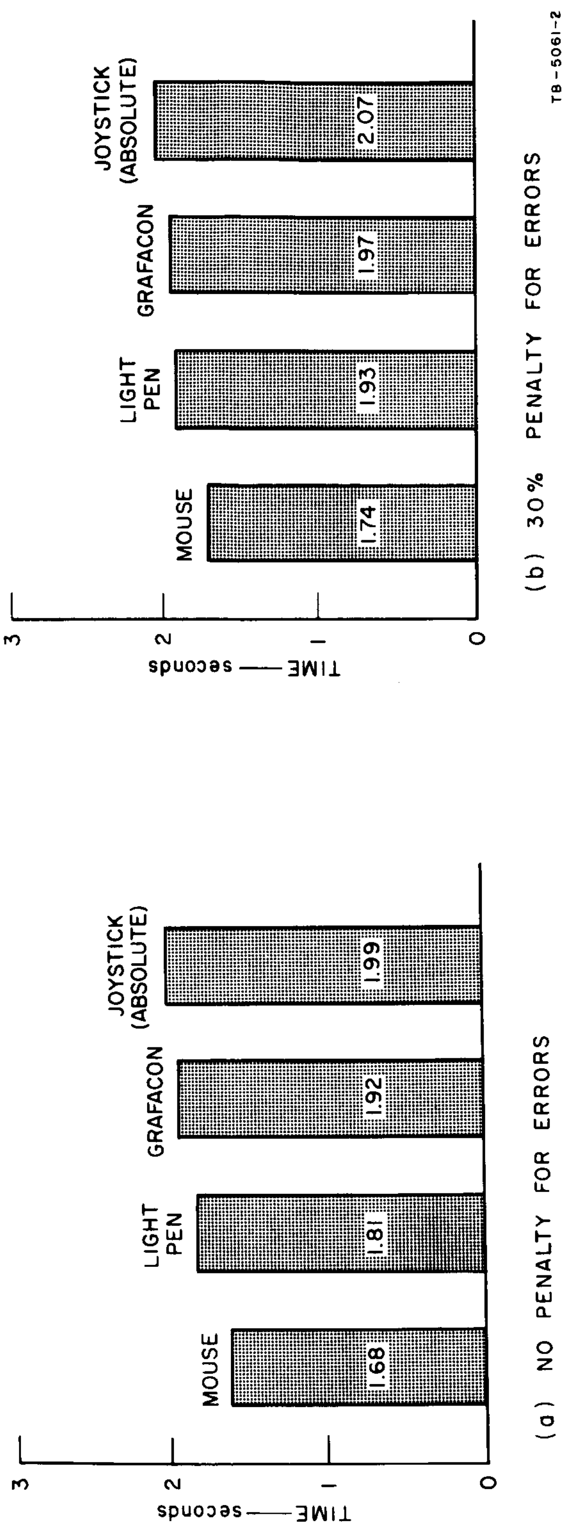


FIG. 8 COMPARISON OF THE OPERAND-LOCATING DEVICES FOR "EXPERIENCED" SUBJECTS, "WORD MODE" OPERATION. Vertical Scales Indicate the Total Average Time (Seconds) from Space Bar to Correct Selection.

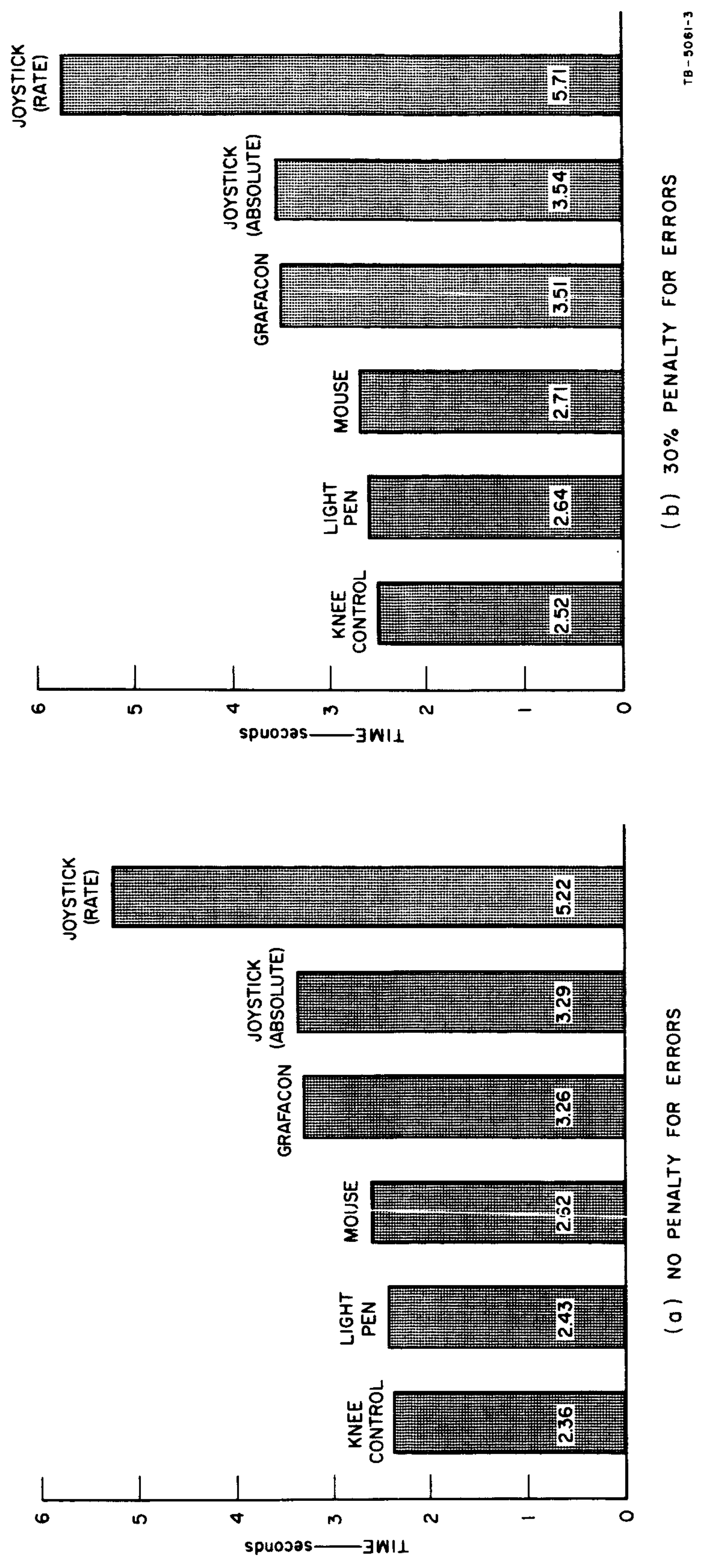


FIG. 9 COMPARISON OF THE OPERAND-LOCATING DEVICES FOR "INEXPERIENCED" SUBJECTS, "CHARACTER MODE" OPERATION. Vertical Scales Indicate the Total Average Time (Seconds) from Space Bar to Correct Selection.

experiments we found that the time required to strike the space bar accounted for about 30% of the total time. Thus we computed the time for the error-penalty graphs by multiplying the subject's error rate on that device times 30% of his average time, and adding that figure to the total time.

4a2 Figure 9 shows the results from the tests of subjects who had had no previous experiences with the devices. Figure 9a imposes no penalty for errors. Figure 9b imposes a 30% penalty for errors, as explained above.

4a3 Figures 10 and 11 compare the error rates for the various devices. Figure 10 shows the results for the "character" and "word" tests, as performed by experienced subjects (using four different operand-locating devices); Figure 11a shows the results of the "character" tests for inexperienced subjects (using six different operand-locating devices).

4b These results indicate that for the more experienced subjects the "mouse" was both faster and more accurate than any other device--including the light pen. Inexperienced subjects, however, tended to perform better with both the light pen and the knee control than with the mouse.

4b1 As mentioned above, the knee control was not developed soon enough to include it in the tests for the experienced subjects (where we included only devices that had been available for some time, in order to avoid bias). We did, however, perform a few individual check tests with experienced subjects, using the knee control; in these tests the knee control appeared both slower and less accurate than the light pen and mouse.

4b2 Inexperienced subjects found the knee control was the fastest device. Undoubtedly the main reason for this was that the knee control, unlike all the others, has no access time. (If the access time is subtracted from the total times measured for the other devices, the knee control no longer shows up so favorably.)

4b3 Inexperienced subjects also found the light pen faster than the mouse. A reason for this may be that the light pen exploits one's inherent tendency to select something by straightforwardly "pointing" at it rather than by guiding a bug across a screen toward it from a remote control. This means that an inexperienced subject can become reasonably proficient in using a light pen with relatively little practice.

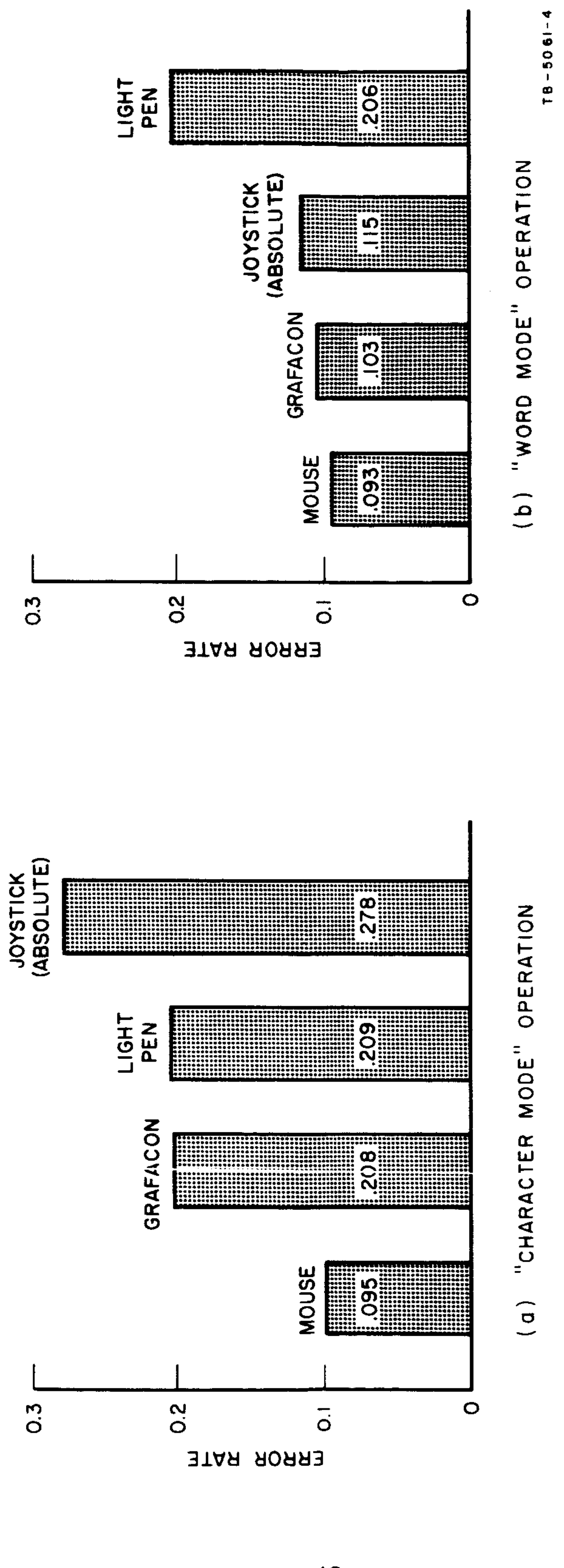


FIG. 10 ERROR RATES FOR "EXPERIENCED" SUBJECTS

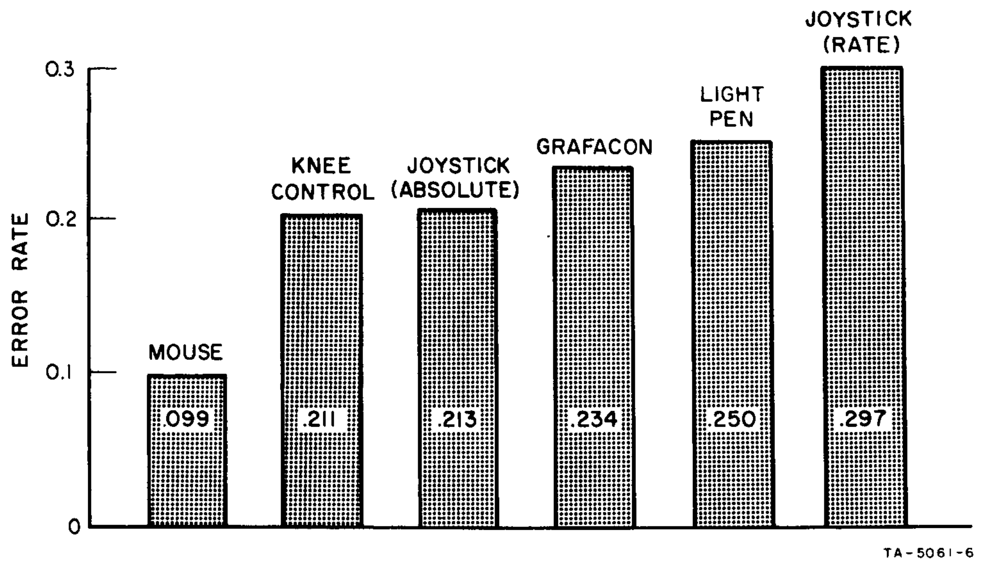


FIG. 11 ERROR RATE FOR "INEXPERIENCED" SUBJECTS, "CHARACTER MODE" OPERATION

4b4 The joystick proved to be both the slowest and the least accurate of the devices we tested, in both modes of its operation ("absolute" and "rate"), and among both the experienced and inexperienced subjects.

4b5 It is interesting to note, however, that both the joystick and the Grafacon showed up more favorably (relative to the other devices) when used to select word entities rather than character entities. These two devices seem to perform better where fine control is less critical; they can move into range quickly at the grosser level.

4c There were some obvious defects is the particular devices which we tested. For this reason and because of the very limited nature of the tests we should be careful not to apply these results to the class of device used, but only to the particular examples being tested.

4c1 Both the Grafacon and joystick suffer from a lack of independence in the actions required to actuate the select switch and to move the bug. Particularly with the joystick, it is very difficult to operate the switch without disturbing the bug position. By contrast, the mouse is moved by an action of the entire hand, while the switch is easily operated by one finger and does not tend to cause bug motion.

4c2 The scale factor between bug motion and device motion is almost the same for the mouse and Grafacon (about 2:1 as we have been using them). With the joystick considerably less stick motion is involved (about 4:1 for a normal finger position on the stick); which contributes to the lack of fine control (high error rate) when it is used.

4c3 The rate mode with the joystick is very poor, partly because of the software implementation. We used a non-linear relationship between deflection and rate of bug motion (approximating a square law), and left too much dead space around the center position of the stick. This made large bug motions very easy, but too much stick motion was involved in changing directions. In the experiments one reason for the very high error rate in this mode is that the subjects tried to "catch" the target on the way past, to avoid changing direction.

4d Figure 12 shows the composite learning curves for inexperienced subjects learning to use the various devices. Points on the curves represent the time (averaged for the nth 8-target test of all subjects) required for correct selection of

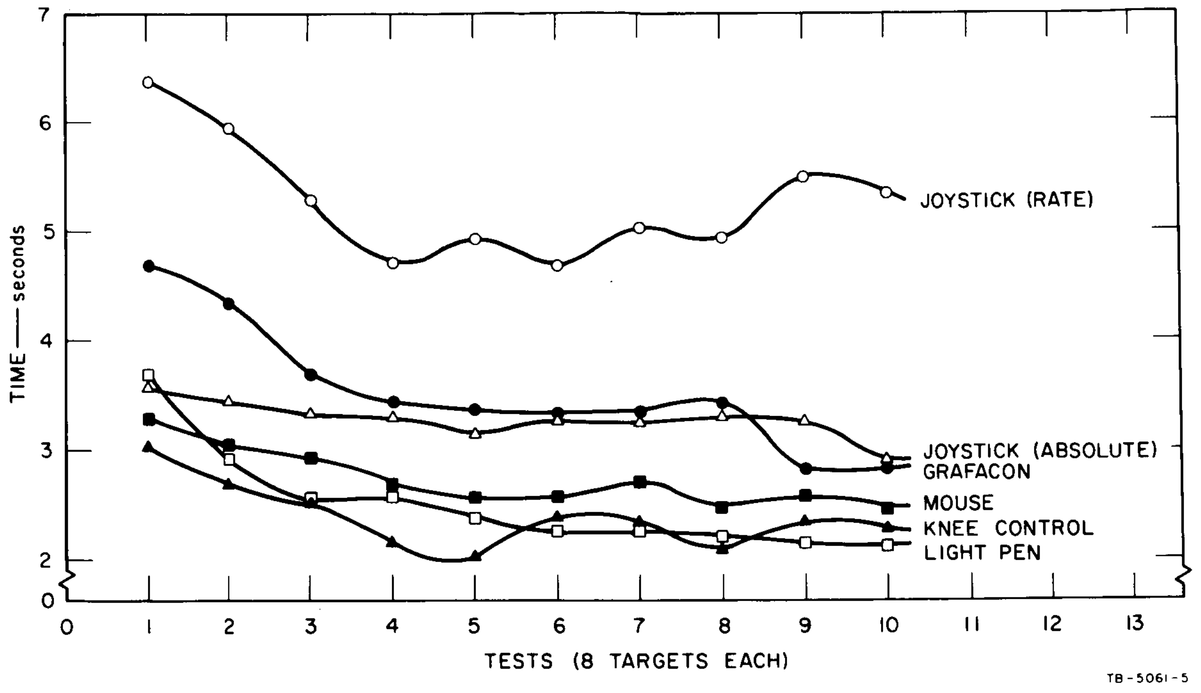


FIG. 12 COMPOSITE LEARNING CURVES

the target entity; this is the "total time," from the striking of the spacebar until the fixing of the bug (without penalty for errors). These curves must be interpreted very cautiously, as they represent only a small number of experimental runs, with only three subjects.

4d1 The curves seem to indicate that the joystick, in its absolute mode, takes the least practice to attain a reasonable level of proficiency, and that this level is, in addition, somewhat better than for the other devices.

4d2 The light pen seemed to require less practice than the mouse, leveling off after about three practice periods (of eight selections each). Performance with the mouse, by contrast, seemed to continue improving throughout the entire series of ten practice periods. These results might be taken as mildly confiming our specualtions above, about the "natural" character of pointing with a light pen (as contrasted with the less natural technique of guiding a bug). But certainly any conclusive confirmation would require more extensive testing than we have as yet performed.

4e We initially expected to find with bug-positioning devices that the starting distance between the bug and its target entity on the face of the display would significantly affect the motion time required for selecting the target. However, the results compiled and plotted to test this hypothesis did not show any significant correlation. An examination of the CRT-displayed performance curves suggests that this may be because the time required to move the bug close to the target is relatively small compared to the average access time, or to the average time required for selecting the target after the bug has already been moved close to the target.

4f Examination of the CRT-displayed curves (distance from target as a function of time) allows several other observations as well:

4f1 In using the Grafacon and the joystick (rate mode), the subjects tended to overshoot the target to lose a significant amount of time in changing the bug's direction and bringing it back into position for a select action.

4f2 While our experiments did not provide a measure of access time for the light pen, we found (from observing the subjects) that a good deal of time was consumed in reaching from the keyboard to grasp the light pen.

4f2a The mounting of the pen was somewhat clumsy and the subject had to reposition the pen on this mounting after each target selection, returning to the keyboard in time for the next target presentation. This tended to cause hurried motions, and may have resulted in many of the incorrect selections made with the light pen.

4f2b A second reason for the higher error rate is that for some tests the intensity of the displayed targets was too high, making it easy for the pen to pick up light from an adjoining character. This difficulty could be overcome, and the overall performance of the light pen improved, if computer feedback were provided, to indicate to the subject which character the pen was actually detecting.

4f2c If these speculations are correct, the light pen might show up considerably better if it were provided with an improved mounting and computer feedback.

4f3 Though the knee control showed up well in its performance as compared with the other devices, an examination of its CRT-displayed curves shows that its operation is relatively unsmooth; the bug tends to move erratically, and it appears to be difficult to move the bug vertically on the display.

4g Our other source of "data"--gained by asking the subjects how they liked the various devices--reveals that the light pen, while operating in a natural way, does tend to be fatiguing; and that the mouse--though it requires some practice--seems to be a satisfying device to use (accurate, and non-fatiguing).

5 CONCLUSIONS

5a Some specific conclusions about our particular devices.

5a1 The operand-selecting devices which showed up well in our tests were the mouse; the knee control; and the light pen. These three were generally both faster and more accurate than the other devices tested.

5a2 Inexperienced subjects did not perform quite as well with the mouse as with the light pen and knee control, but experienced subjects found it the "best" of the devices tested, and both groups of subjects found that it was satisfying to use and caused little fatigue.

5a3 The select switches on both the Grafacon and joystick tended to move the bug and cause an incorrect fix. These two devices could probably be improved by redesigning their select switch mechanisms.

5a4 Although the knee control was only primitively developed at the time it was tested, it ranked high in both speed and accuracy, and seems very promising. It offers the major advantage that it leaves both hands free to work at the keyboard.

5a5 The major advantage of the light pen appeared to be its psychological "naturalness" of operation in pointing at the item to be selected. This means that an untrained user can quickly understand it and gain enough proficiency to do useful work.

5a5a Weighed against this, however, is the disadvantage that the pen must be held in the air while it is being used. Many subjects expressed feelings of fatigue while using it for a prolonged time. To some extent, this disadvantage might be alleviated by a carefully designed mounting for the pen.

1 INTRODUCTION

1a The dominating theme for our future work is the integration of our analytic and experimental techniques into a first-stage scheme-analysis approach.

1a1 This is expected to provide a continuing source of orientation, focus, and stimulus, and to be a promising way to develop skill in designing and analysing on-line systems.

1a2 In this section we discuss the direct effects of concentrating on this unifying theme more than has been possible in the first year when analysis and experiment were each getting established as basic techniques.

1b In addition to discussing the implications of this "theme" in our work, we also describe a number of specific possibilities which have occurred to us for improving various parts of our schemes.

2 ANALYTIC TECHNIQUES

2a There should be an early effort made to describe and study in detail some of the typical operations involved in the lower levels of our text-manipulation activity.

2a1 This would be similar to the example in Section III for the "delete word" operation in form of description, with variation in the type of operation, the level, and the types of scheme components involved.

2b We should pay special attention to isolating a set of "primitive processes" which:

2b1 are necessary and sufficient to build up the higher-level processes of our system, and

2b2 can have their time and branching probabilities determined experimentally.

2c We should choose some typical medium-level processes for which we have made detailed descriptions (and thus have nested-network representations), and utilize the measurements of primitives to obtain calculated process-time and branching-probability figures.

2c1 These figures then should be checked experimentally (i.e. the performance over these medium-level processes should be measured for the same subjects as for the primitive processes), and the discrepancies used to lead to better analytic or experimental techniques.

2d The higher-level processes involved in our text-manipulation activity (such as writing this report, or writing and modifying computer programs), should be studied with the following purpose in mind:

2d1 We need some "typical" operations or processes whose execution times by a skilled user, with a given display-control scheme, can be used as the measure of "value" of that scheme in our system.

2d2 Such test processes should be of a high-enough level that they make use of all of the commands in a scheme repertoire with a frequency and a probability of succession that is typical of the general usage of the system.

2d3 But they will have to be of a low-enough level that we can easily handle them analytically and experimentally.

2e Then there will be detailed work of the sort:

2e1 Finding explanations within our analytic framework of the differences observed in user performance with different devices, operation sequences, tasks, motivation, practice, etc.

2e2 Where satisfactory explanations aren't forthcoming from the present analytic framework, rework it so that they are.

2f To treat many processes to the detailed description and analysis implied by the above activities will impose quite a burden on the experimenter. We plan to develop as much computer aid to these activities as possible.

2f1 For the process-description activity, the conventions and techniques are very close to those of computer-program description, and the on-line aids being developed by other projects in our research program to improve our speed at this sort of activity will directly help our efforts in scheme-plan description. Composing, studying, and modifying such structured text is directly in line with our major program activity.

2f2 Determining network characteristics from node characteristics will be a very frequent task in our studies.

2f2a Consider a process plan which is described in depth with the linked-statement conventions, as is the example in Section III.

2f2b There is enough information already available to the computer for it to determine the topology of each of the nested processes.

2f2c In such a description, straightforward extension of the tagging feature will permit attaching to each branch in a network its process-time and branching-probability values, in such a manner that both the computer and the user can "see" and make use of them.

2f2d For any such description in which at least the lowest-level processes have these values assigned by the user, the computer can automatically determine the general network flow solution.

2f2e In calculating network-exit processing times, it seem promising that treatment of both the altered network flow states and the resulting alteration in internal execution times and branching probabilities could also be done or at least materially aided by the computer.

2f2f In studying a network with this kind of computer help, the user should be able to examine the linked-statement description on the display, freely making changes in the organization of the network or the ascribed values of some of the basic node values, and then request a solution of some network parameter for given flow-state conditions. The computer would immediately do the calculation and produce the answers; probably by putting tagged values on the appropriate points in the description text.

2g EXPERIMENTAL TECHNIQUES

2g1 The execution-time and branching-probability characteristics of the primitive processes, as isolated analytically above, need measuring. This can probably be done with relatively little extension of our present technques.

2g2 The same sort of measurements need to be obtained for higher-level processes, in coordination with the developments outlined above for the analytic techniques. This is likely to require a new level of instrumentation and data analysis.

2g3 Even further extension is likely to be useful, probably along similar lines, to allow monitoring of a user's actual working activity in such a way as to derive meaningful measurements for different processes at different levels.

2g3a Both of these above extensions will involve programming our (new-computer) next system to have the necessary monitoring and recording features. It may be necessary to develop some special equipment and signalling techniques so that user actions which are not otherwise discernible to the computer may be recorded.

2g3b Extensions of our current computer analysis and display techniques need be developed to handle these new types of data.

2g3c It is hoped that data forthcoming from such monitoring and analysis would include probabilities of branching, as well as distributions of flow states.

3 NEW COMPUTER SYSTEM

3a In mid-July we are scheduled to receive a CDC 3100 with the following characteristics:

3a1 Sixteen thousand words (24 bits) of 1.75 usec core storage.

3a2 Three I/O channels, one to be compatible with the interface presently used on some of our equipment for coupling to the CDC 160A.

3a3 Paper tape I/O.

3a4 Two magnetic tape transports.

3a5 One IBM 1311 disk file (2,000,000 character capacity).

3a6 One 150-line/minute printer.

3b We also expect to have our new character generator operating with our display system, giving us an expanded character set (63 characters).

3b1 We have straightforward conventions established for differentiating between upper- and lower-case alphabetic characters.

3b2 The character-writing rate will be approximately four times what we now have--i.e. at a sixty-cycle repetition rate, we will be able to display about 4,000 characters

3c The display, keyboard, pushbuttons, etc. will all be compabitle for connection to the 3100.

3d We are implementing a new assembler and assembly language for the 3100, into which will be built an on-line debugging system. This will provide for examining and altering registers by symbolic address and content, installing breakpoints and getting trial runs over prescribed segments, automatically setting up patches, etc.

3e We are also implementing a translator for a slightly modified form of SNOBOL3, the string-manipulation language developed at Bell Telephone Laboratories. We intend to try developing as much of our on-line system as possible using this language; making use of machine-coded functions in places where the processing time is too critical for SNOBOL3. It promises to make much more flexible our experimentation with variations in schemes, or our development of computer aids for instrumenting and analysing our experiments.

3f We plan to keep the 160A system running until the middle of September, upon which we can continue to do useful work and experimentation while our new system is being developed. We are setting up means for the two computers to communicate, so that on-line work with the present 160A-based system can be interacting with programs on the 3100 (especially, so that we can modify our symbolic source code and send it straight to the assembler-debugger).

4 SOME POSSIBILITIES FOR IMPROVED SCHEMES

4a The work station

4a1 Explore the potential of an auxiliary CRT display at the work station; to display often used text such as outlines or as a flexible control panel.

4a2 Investigate the use of a typewriter-like printer at the work station.

4a3 Provide means for lighting and holding printed reference material, without interfering with display visibility.

4b Command repertoire

4b1 Define a set of basic commands which can be used to generate user-specified commands or "macros" tailored to the user's particular tasks.

4c Improved scanning

4c1 Investigate means for implementing "scroll scanning" to eliminate the need for interspersing scan commands with editing commands.

4c1a Scroll scanning should allow the user to move freely over the working text independent of any command which he may be in the process of specifying.

4c1b This would allow editing operations (such as move and delete) to operate over the entire working space.

4c1c A separate console device (such as a centered lever) could be used for scanning, or an added button on the bug-positioning device could be used to give it control of scanning.

4d CRT operand selection

4d1 There is perhaps more room for improvement in operand selecting devices than in any other area of system hardware, especially in bug-positioning devices that leave both hands free. Two promising possibilities for this are:

4d1a The knee control which was developed in an experimental model during the present project. A preliminary experimental evaluation of this device is presented in Section IV, and a more detailed description of it is included in Appendix B.

4d1b Another interesting possibility would be an "accelerometer platform," a small platform containing two orthogonally mounted accelerometers. This platform could be mounted on any convenient part of the body to let us experiment with the user being able to "point" at CRT entities by moving different body members in different ways.

4d1b1 The output of each accelerometer would be integrated twice by analog integrators, yielding the absolute position of the platform. The output of the second integration would then go to the analog-to-digital converter, as with the present bug-positioning devices. There would have to be provision for referencing the bug position on the screen, relative to some given absolute position of the platform.

4d2 The use of scale changing algorithms for bug-positioning devices should be investigated, particularly for use with the "no hands" devices which may be lacking in fine control. The scale between device motion and bug motion could be under user control or could be changed dynamically (proportional to rate for example) with device motion.

1 This study is considered as a first stage in developing improved means for a human to control a computer-driven CRT display in his composing and modifying of text. Such a study is treated as a design problem in a particular system area, where good design is considered to stem from:

1a A sound conceptual framework within which to consider and discuss the parts of the problem and their interrelationship. This would include the background deriving criteria for success, etc..

1b Good techniques for measurement.

1c Good analytic concepts, principles, and procedures.

1d Well-developed intuition and judgement--which come only with the seasoning of working and struggling with design and analysis in a good environment of conceptual framework, measurement technique, and analytic methods.

1e Creative imagination.

2 Our approach has set out to develop the first four of these ingredients of a design environment.

2a We began with a conceptual framework which considered:

2a1 The display-control operations as the basic ones upon which all of the higher-level operations in an on-line system would be built;

2a2 That to develop new means for providing better operations at a higher level, one should establish a working system with realistic dependence of higher-level operations upon lower-level operations, and pursue the possibilities for improving the lower-order operations by studying them in this environment.

2a3 That, in the man-computer system, evolution of improved operations would involve not only improved hardware and software, but significant associated changes in the concepts, terminology, and working methods of the human.

2b In pursuit of this project, and in coordination with several other projects, we immediately set up an on-line system, based upon prior experience, intuition, and expediency. This was a text-manipulation system with which we actually undertook to do a portion of our research work.

2b1 This provided the multi-level system environment, and gave us a continuing base of experience in which intuition and judgement could develop.

2b3 This system has evolved, as our understanding grew, and with concentration upon making it more efficient wherever we could see the opportunity, to the point where we now do a quite significant portion of our research work on it. A good deal of this report was composed on line, all of it was held on magnetic tape and updated recurrently during the growth of the report, and the final mats were printed directly from computer output.

2b4 It is now the best on-line text-manipulation system that we know of (see the description in Chapter II and Appendix B). However, it is regarded by us as but a good start toward what such a system could provide in speed and flexibility.

2c When we began to feel that we were understanding the basic factors involved in the problem, we developed some computer-driven experiments to measure the speed with which users could select typical text entities from the CRT display. These experiments compared performance using different of the display-selection devices which we had implemented for our working system. Measurement and analysis techniques, data, and specific conclusions are given in Chapter IV. Following are the general conclusions drawn from the experiments:

2c1 The principal value of our experimental work to date was in developing the techniques of experiment and analysis, and in isolating some of the factors in the design of display-selection means which are important to fast operation.

2c2 Any comparative evaluation of the different types of devices must be qualified to such an extent that it is not significantly useful in a direct sense toward choosing from among the types of devices.

2c2a For instance, one could couple a "gadget" just the shape of our mouse to either a Graphacon or a below-the-table joystick (with a handle long enough to give the five inches or so free motion), in such a way that the user would find essentially the same "feel" with each of these three devices. There is no reason to believe, from our experiments and experience, that performance in the above experiments with three such devices would show any difference between them.

2c2b In other words, what is of importance is not the particular way in which desk-top horizontal motion of the hand is actually converted into computer-entered coordinates for control of the bug--i.e. it doesn't matter whether coupling to the computer is through displacement in spherical coordinates for the joy stick, in polar coordinates for the Graphacon, or in rectangular coordinates for the mouse.

2c3 What is important to fast, efficient display selection is the particular feel to the user of the thing he grasps and moves--e.g.:

2c3a Where he reaches to grasp it;

2c3b How it fits his grasp;

2c3c How the scale of horizontal displacement is related to bug motion on the screen;

2c3d How he actuates the select switch;

2c3e How much mass he moves;

2c3f How the large-motion capability of arm and wrist can coordinate with fine-motion capability of the fingers;

2c3g How he can rest his arm, hand and wrist (or how much weight does he have to support);

2c3h And whether, when he removes his hand, the thing stays put, returns to a standard position, drifts away (as our modified form of the Grafacon did), falls down on the table, or has to be put down or hung on something.

2c4 To make final judgements between display-selection devices, more must be learned about the desireable way to adjust and coordinate each of these factors. Then it must be seen which basic-device approach can best provide this.

2c5 Comparative comments of a general sort can follow these observations:

2c5a For the light pen, there is enough less freedom to vary the above-listed design factors than there is for the other devices, that its probability of being the best candidate diminishes appreciably.

2c5b Any final, significant differences between best designs for joy stick, Graphacon and mouse are not discernible now.

2c5c The fact that a no-hands bug-control device can allow both hands to remain on the keyboard is an important factor in their consideration. Even if its selection speed and resolution could not be developed to match that of a good hand-controlled device, what we are learning about the importance of smooth coordination between the different primitive operations would make it a strong candidate.

2c6 An important, general conclusion from our tests is that the relative value of different schemes cannot be judged on the basis of their appeal to inexperienced users

2d Our analytic development began with attempts to do task analysis and scheme categorization, and to develop a rationale for what schemes to select for implementation, for what experiments to perform, and for what measures to use for judging the different scheme components.

2d1 From these attemtps grew a unifying concept of "process," and the realization that detailed manageable descriptions of the processes performed conjointly by man and computer provided an approach to the analytic needs expressed above.

2d2 We made use of the special linked-statement conventions which we had developed for describing our computer programs, added a few new conventions, and developed a way to describe multi-level man-computer processes to any detail desired. An example is given in Chapter III, describing the relatively low-level process of an on-line user deleting a word from the text displayed on the screen

2d3 The example clearly shows the multi-level nature of processes--i.e. that larger processes are composed of smaller processes.

2d4 It also shows the relatively large amount of branching involved, where flexible provisions must be made in the plan of the process for changing the sequence in which lower-order processes are executed. This is necessary so that execution of the larger process can adapt to human errors or to variations in the state of the data or of the system.

2d5 The detailed design plan for a process is an important part of the implementation "scheme" which provides a given on-line display-control capability. The measure (for us) of the value of a scheme is the execution time of typical text-manipulation processes whose execution capability was provided by the scheme. (our ultimate valuation would be based upon higher-level processes than "delete a word").

2d6 We need an analytic approach with which we can calculate the execution time for a process from the detailed description of its design--without having to implement and measure for each variation of design. Faced with the complexity of multi-level, branching processes as being "real life" for operations at any significant level in the system, we developed the "network" approach (see Section III) to deal with branching probabilities, process flow states, probable execution times, etc..

2d7 The calculating techniques allow us, in principle, to derive the branching probability, flow states, and execution times for a higher-level process from the characteristics for the "bottom-level" processes in the network (i.e. in the detailed design description of the higher-level process plan).

3 We have yet to apply these analytic techniques toward analyzing our scheme possibilities and guiding our experimental activity. But this coordination of analysis and experiment, with the concurrent evolution of techniques for both, is the dominating theme of our plans for continued work.

4 The net conclusions drawn from our work to date seem disappointingly nonspecific--but therein lies one of the most important lessons we have learned.

4a This "lesson" would be expressed as follows:

4a1 Display-control activity is important because it provides basic processes whose speed and flexibility promise to affect strongly the speed and flexibility which can be developed for higher-level processes. And it is this latter speed and flexibility, for a human to execute tasks at a meaningful intellectual level, which is the goal of research in on-line working systems.

4a2 Thus, the display-control processes whose design is our direct concern are really important only in the way that they serve as components in larger processes.

4a3 The value of our design effort then must be measured in the improvement it thus provides in higher-level performance.

4a4 To pursue this kind of development and evaluation work, it is necessary to consider the interaction of higher-level considerations with those of direct involvment with the low-level processes where concern is likely to focus.

4b As an example of the evaluative position which this realization estabilshes, one cannot simply say that the light pen (or the RAND Tablet, etc.) is the best display-selection device for on-line work.

4b1 Irrespective of the speeds with which one can make successive display selections with a given device, the tradeoffs for the characteristics of fatigue, quick transfer to and from a keyboard, etc. will heavily weigh the choice among the devices. And these tradeoffs, and the possibilities for designing around them, aren't apparent until after a good deal of design and analysis has been done for the rest of the system.

5 We conclude generally that we are on a promising track--promising not only for the pursuit of this project, but also for establishing design, analysis and experimentation techniques applicable for user-system design over most of the domain of real-time computer-aided human work.

6 In closing, we feel it not unreasonable to set as a goal for the next year the development of an on-line system which would provide a trained user with the capability for executing meaningful test tasks (in our text-manipulation) at least twice as fast as our current system would.

1 INTRODUCTION

1a Within our man-computer research program, we have developed the following special conventions to help harness more computer aid within our everyday working life.

1a1 It was part of our initial program conception that special structuring of one's working information would be important, and below are our current developments in this direction.

1b First is presented the conventions for the basic form in which all of our working text (as, for example, this report) is organized and manipulated.

1b1 It is basically a hierarchical (or "outline") form into which may be organized the individual basic "working modules" of text--our "statements."

1b2 The ability to name and tag individual statements, and to form arbitrary "cross-reference" links between any two statements, when added to the basic hierarchical form, yields a very flexible and useful set of conventions.

1b3 These conventions were developed under ARPA support.

1c We next present the added conventions developed to allow representation of flow-diagram like computer-program designs.

1c1 We are developing these conventions to allow us to use our computer aids effectively for working with the design records of our computer programs.

1c2 These a3d4b conventions were developed under ESD support, Ref(ESD2).

1c3 The direct similarity between joint man-computer processes of interest in the NASA project, and the computer-only processes for which these conventions were developed led us to apply them to "plan description" usage as in Section III above.

2 Terminology and conventions used below:

2a The composition of a string of characters is often represented by a sequence of upper-case characters or words.

2a1 A single upper-case character represents the occurrence of that character in the string.

2a2 An upper-case word represents the occurrence of either a single character or a special substring of characters:

2a2a ASTERISK, SPACE, CARRETURN, etc. represent corresponding "single-key-stroke" characters.

2a2b PRINTCHAR represents the occurrence of any one printing character--and n-PRINTCHARS, of an unbroken string of n successive printing characters.

2a2c SPACINGAP represents the occurrence of an arbitrarily long, unbroken string of successive non-printing characters--i.e., an arbitrary succession of instances of SPACE, TAB and CARRETURN

3 LINKED-STATEMENT STRUCTURING CONVENTIONS AND TERMINOLOGY

3a Statements:

3a1 Any appearance of the sequence CARRETURN CARRETURN NUMERIC is assumed to signal the beginning of a new statement, with the NUMERIC as the first character of the first "word."

3a2 The length of a statement is arbitrary; so is its composition, except for the special requirements for "location numbers," "names," "tags," and "links," which are described below.

3a3 Location numbers:

3a3a The first word of a statement is its "location number"; the first character of this location number is a digit. The location number consists of a string of digits and alphabetic characters, with no spacingaps included.

3a3b A "field" in a location number is a continuous string of alphabetic characters or a continuous string of numeric characters, broken possibly by a period or comma. The characters in a given field indicate the ordering on a unique list in the structure of statements.

3a3c The location number represents the unique location of its statement within the larger structure of statements.

3a4 Names:

3a4a A name may be associated with any given statement. This name is enclosed in parentheses, and is the first printing string to appear after the location number.

3a4b The choice and sequence of printing characters composing a name is arbitrary, but no spacingaps may be included between the parentheses. The length of a name is limited to 16 characters.

3a5 Tags:

3a5a Special words called "tags" may be included within statements to serve as descriptors, etc. As many tags as desired may be embedded within the same statement. They may be located anywhere after the location number and name.

3a5b Each tag is identified by the sequence SPACINGAP ASTERISK n-PRINTCHARS SPACINGAP. There is no restriction on "n," or on the composition of a tag -- except that no spacingaps may be included.

3a6 Links:

3a6a Special words called "links" may be included within statements, to establish cross-references to other statements. As many links as desired may be included in any statement. They may be located anywhere after the location number and name.

3a6b Each link is identified by the sequence SPACINGAP n-PRINTCHARS OPENPAREN m-PRINTCHARS CLOSEPAREN SPACINGAP-or-PUNCTUATION, where the parens enclose the name of some statement.

3a6c The PRINTCHARS preceding the OPENPAREN represent the "link type" code string. This string may be of arbitrary length and composition -- except that no spacingaps may be included.

3b Lists of statements:

3b1 Any statement may have a "list successor," which is another statement. The sequential string of statements formed by the successor of a statement, by its successor, etc., until finally a statement is reached that has no list successor, is called a "list of statements."

3b2 The first statement on such a sequential list of statements is called the "head statement" of the list; the last statement on such a list is called the "tail statement."

3b3 A list may contain an arbitrary number of statements, but must have at least one statement.

3b4 For each statement in a given list, the last field of the location number indicates the statement's location in that list. Interpolative breaks may appear in a field of the location number; in this case the numbers indicate only the relative location number. A list that is in "clear ordinal state" will have no interpolative breaks in its last field; the last field then indicates the true ordinal location on the list.

3c List structures:

3c1 Various structural relations are (implicitly) provided for by the conventions described above: the sequential association of statements within a list, and inter-statement linkages between any two statements.

3c2 In addition there is "hierarchical" structuring of lists.

3c2a Each list of statements may be a sublist of one (and only one) statement; that statement is known as the "source statement" of that list. The location number of each statement on such a list will differ from that of its source statement only by the addition of one more field.

3c2b Any statement in a sublist may be the source statement for another sublist of its own, etc., to arbitrary depth. The sublist of a statement, plus the sublists of the sublist statements, etc., form the "substructure" of the given statement.

3c2c A statement ST2 is said to be a "logical successor" of a statement ST3 if there could exist a hierarchical structure of statements such that, by their location numbers, ST2 could succeed ST3 in the text. For instance, following a statement "2b3" one could logically accept only "2b3a," "2b4," "2c," or "3." The presence of any other location number than these on the next statement establishes a "logical break" in the text.

3d Terminology Conventions:

3d1 Basic Entities:

3d1a Let ST1, ST2, etc., refer to arbitrary statements. (The integers carry no implications as to the structural relationship between the statements.)

3d1b Let LN1, LN2, etc., be used to represent arbitrary location numbers.

3d1c Let 1F1, 1F2, etc., refer to the first, second, etc., fields of LN1; and 2F1, 2F2, etc., to the first, second, etc., fields of LN2.

3d1d Let NM1, NM2, etc., refer to arbitrary statement names.

3d1e Let LS1, LS2, etc., represent arbitrary lists of statements.

3d2 Operations (where an operation on one entity represents another entity):

3d2a General:

3d2a1 Let LCN ST1, LCN ST2, etc., represent the location numbers of statements ST1, ST2, etc.

3d2a2 Let STM LN1, STM LN2, etc., represent the statements whose location numbers are LN1, LN2, etc.

3d2a3 Let STM NM1, STM NM2, etc., represent the statements whose names are NM1, NM2, etc.

3d2a4 Let NAM ST1, NAM ST2, etc., represent the names of statements ST1, ST2, etc. (Let NAM ST1 be ZERO if ST1 has no name.)

3d2b Fields within a location number:

3d2b1 Let FL1 LN1, FL2 LN1, etc., represent the first, second, etc., fields of location number LN1.

3d2b2 Let FL(expression) LN1 represent the nth field of LN1, where "n" is the numeric obtained by evaluating the expression.
3d2b3 Let FLi LN1, FLj LN1, etc., refer to the ith, jth, etc., fields of LN1.

3d2b4 Let FLT LN1 represent the last (tail) field of LN1.

3d2c The depth of a statement--the level down from the top of the structure at which it lies--is an integer. The topmost level (location numbers of 1, 2, etc.) has a depth of 1; the next level down (location numbers of 1b, 4d, etc.) has a depth of 2, etc.

3d2c1 Let DPT ST1, DPT ST2, etc., represent the depths of ST1, ST2, etc.

3d2c2 Let DPT LN1, DPT LN2, etc., represent the depths of STM LN1, STM LN2, etc.; these should always be equal to the number of fields in LN1 LN2, etc.

3d2d To represent a statement having a particular structural relationship to another statement:

3d2d1 SCS ST1, successor of ST1 (list successor).

3d2d2 PRD ST1, predecessor of ST1 (list predecessor).

3d2d3 HED ST1, head of the list containing ST1.

3d2d4 TAL ST1, tail of the list containing ST1.

3d2d5 SBH ST1, sublist head of ST1--the head statement of the sublist of ST1.

3d2d6 SBT ST1, sublist tail of ST1--the tail statement of the sublist of ST1.

3d2d7 SRC ST1, source of ST1--the source statement of ST1.

3d2e To represent a list having a particular structural relationship to a statement:

3d2e1 LSC ST1, list containing ST1--the entire list of statements.

3d2e2 LSF ST1, list from ST1--the list of statements including ST1, SCS ST1, etc., down to and including TAL ST1.

3d2e3 LSB ST1 ST2, list between ST1 and ST2--a binary operation, representing the list that begins with ST1 and ends with ST2. (ST1 and ST2 must be in the same list.)

3d2e4 LST ST1, list to ST1--the list of statements from HED ST1 through PRD ST1.

3d2e5 SBL ST1, sublist of ST1--the entire list.

3d2e6 SRL ST1, source list of ST1--the list containing SRC ST1.

3d2f To represent a statement having a particular relationship to a list:

3d2f1 HED LS1, head of LS1.

3d2f2 TAL LS1, tail of LS1.

3d2f3 SRC LS1, source of LS1.

3d2g Relating a list to a list:

3d2g1 SRL LS1, source list of LS1--the list containing SRC LS1.

3d3 Concatenated operations:

3d3a An operator may operate upon an entity that is represented as the product of another operation.

3d3b Two successive operator terms separated by a spacingap indicate that the entity represented by the rightmost operation is to be operated upon by the preceding operator term. (Obviously, the product of the rightmost operation must be an entity upon which the preceding operator can validly operate.)

3d3c An integer "n," or an expression representing such an integer, appearing between parentheses after an operator, designates n successive applications of that operator. Any other printing character or characters appearing between two operations indicates that they are not to be concatenated.

3d4 Special entities and relationships:

3d4a The "source chain" of ST1 is composed of ST1, SRC ST1, SRC(2) ST1,..., SRC(DPT ST1) ST1.

3d4b The "branch chain" from ST1 is composed of LST ST1, tied onto the end of LST SRC ST1, tied onto the end of LST SRC(2) ST1, etc., to the head of the top-level list of the structure.

3d4c ST1 is said to be "structurally above" ST2 if ST1 is a member of the branch chain from ST2, and is said to be "structurally below" ST2 if ST2 is a member of the branch chain of ST1.

3d4d ST1 is said to be "branch related" to ST2 if either statement is a member of the other's branch chain, and is said to be "branch independent" of ST1 if neither statement is a member of the other's branch chain (i.e., if they are not branch related).

3d4e ST1 is said to be the "branch node" between statements ST2 and ST3 if it lies in the branch chains of both ST2 and ST3, and if it is below every other statement that does so.

3d4e1 The branch chains from any two statements in the same structure will always meet to produce such a node.

3d4e2 The branch node between two branch-related statements will be the "upper" of the two statements--i.e., the one which is structurally above the other.

3d4e3 Let BRN ST2 ST3 be a symmetrical, binary (two-parameter) operator whose result represents the branch-node statement (e.g., ST1 = BRN ST2 ST3 = BRN ST3 ST2.

3d4f The "bridge chain" from ST1 and ST2 is the concatenation of the section of the branch chain of ST1 from ST1 to BRN ST1 ST2, with the section of branch chain of ST2 from BRN ST1 ST2 to ST2.

4 BASIC CONVENTIONS FOR PROGRAM-DESIGN RECORDS

4a The purpose of the techniques described below is to provide a complete and consistent way of representing, in a linked-statement form, all the important facts, considerations, and relations that could usefully be entered into the working record of a program design. The discussion uses the terminology and definitions from the preceding section. In addition:

4a1 Let "PRC ST1" ("process of ST1") represent the actual process represented and described by ST1.

4b The design description of a computer program contains several distinct types of statements: those which

4b1 Describe an initial specification, requirement, or constraint.

4b2 Describe the purpose and usage of the finished program, for instance, to someone who wants to use that program.

4b3 State a convention, rule, or definition to be used within the design document in order to facilitate description.

4b4 Describe the data structure.

4b5 Represent and describe an actual program process: an actual object-code statement for the computer; a source-code statement, for a translator program; or a higher-level statement, in whose substructure all the lowest-level statements are of either of the above types.

4b6 Describe special tricks or tactics in design.

4b7 Describe some aspect of a particular processing state.

4c These types of statements can be distinguished in several ways: by the content of the statement; by the kind of name given the statement; by a special tag within the statement; or even by being untagged (in which case, the type is assumed to be the same as that of the first higher source statement that is explicitly tagged).

4d In the following discussion we deal only with the data-description and process-description types of statement; these represent the greatest possibility for immediately improving program documentation.

4e Special conventions for process descriptions;

4e1 A process-structure tag appearing in a statement ST1 has the following significance:

4e1a *p (for "process"): ST1 represents and describes a process.

4e1b *c (for "comment"): used two ways:

4e1b1 Appearing at the head of ST1, after location number and name (if any), *c designates that ST1 and its substructure are comment rather than process statements.

4e1b2 Appearing in the body of ST1, after some relevant process designation, *c indicates that the remaining text of ST1 (or, up to an *o tag) is to be treated as comment information. ST1 and its substructure are still treated as process statements.

4e1c *d (for "data"): ST1 represents and describes data that are to be stored in the computer, as opposed to processes to be stored and executed.

4e1d *sr (for "subroutine"): ST1 represents a closed subroutine (and must therefore be named).

4e1e *o (for "OSAS"): The remaining text in ST1, between the *o tag and the end of the statement, is composed of lines of OSAS code, formatted as for the assembler. (Other source-code languages will have their corresponding unique tags.)

4e1f *i (for "incomplete"): The sublist SBL-ST1 is incomplete--i.e., it does not describe PRC ST1 completely.

4e1g *ib (for "incomplete below"): At least one statement in SBL ST1 has either an *i tag or an *ib tag, or both. (Use not mandatory.)

4e2 The normal control sequence (i.e., process flow when not directed by a TO or CALL link) is from one statement, ST1, to its list successor, SCC ST1. Control bypasses any non-process (e.g., *c-tagged) statement. Control may not pass (by any means) to a statement having a *d tag, and may pass to an *sr-tagged statement only by means of a "CALL" link.

4e3 Branching operations: A link "TO(NM1)" appearing in a statement indicates transfer of control to the statement named NM1, under whatever conditions are specified in the preceding text of that statement. If no condition is specified in the preceding text, transfer is unconditional. If the specified conditions are not met, the link is ignored and control passes on through the rest of the statement.

4e4 Subroutine calls: A link "CALL(NM1)" appearing in a statement indicates a jump-return subroutine call to the statement named NM1, under whatever conditions are specified in the previous text of the statement. If no conditions are specified, the jump is unconditional. If the specified conditions are not met, the link is ignored, and (as when control returns after subroutine execution) control passes on through the rest of the statement.

4e5 Sublists of process statements: If ST1 is a process-description statement, its sublist (SBL ST1) represents a complete description of PRC ST1 as a set of lower-order processes, each represented by a statement of the sublist.

4e5a The first process statement of SBL ST1 to which control will pass is:

4e5a1 The first process statement on the list, if ST1 has no name.

4e5a2 The process statement bearing the same name as does ST1, if ST1 has a name.

4e5a3 *c If control can arrive at ST1 by passing through the previous statement (i.e., not via a TO(NAM ST1) link), then control must pass first to the first process statement of SBL ST1.

4e5b Any nonprocess statement in SBL ST1 must be explicitly tagged; process control will then bypass it.

4e5c If process control passes SBT ST1 (in other words, to try to go to its (nonexistent) list successor), this is an implicit designation that the process PRC ST1 is finished, and that control is to pass from ST1 to its successor, SCS ST1.

4e5d Designation of control transfer from ST1 to SCS ST1, from within SBL ST1, may be accomplished by means of a TO(NAM SCS ST1) link in any (or several) of the process statments of SBL ST2. In SBL ST1, designation of control transfer to statements other than SCS ST1 must be made with TO(NM1) links.

4e6 Multiple instances of identical TO(NM2) links may represent a given program-control branching path. These must appear at each successive level below the highest-level instance, to represent the same branching operation in ever-more detailed descriptive context. In a properly formulated program description, the statement STM NM2 will always be in the same list as the highest-level instance of the TO(NM2) link.

4e7 Multiple names, and link following, adhere to these conventions:

4e7a Under certain conditions, a number of specially related statements may have the same name. If ST1 is the lowest-level statement of a group of statements thus having the same name, then the others must lie on the source chain of ST1 (i.e., they are either SRC ST1; or, SRC(2) ST1; or, etc.).

4e7b Statements bearing a common name represent the same process point, as found at different levels of description. It thus makes no difference, in any sense of correct process execution, to which such statement one assumes control to transfer via a link to that name. But to one studying the process structure and wanting to follow a link referring to a multiply-used name, it does make a difference. He should transfer his attention according to the following rules:

4e7b1 Assume that ST1 contains a link to N1; that NM1 is the name of statements ST2, ST3..., ST4; and that ST2 is the lowest and ST4 the highest of these statements (on the source chain from ST2).

4e7b2 The single general rule: Choose the first of these statements encountered in following the bridge chain from ST1 to ST2. If this is a "reentrant link" the statement thus chosen will be the bridge node between ST1 and ST2. Otherwise, the chosen STM NM1 will be ST4, the highest-level of the chain of NM1-named statements.

4e7b3 If it is a TO(NM1) link in a properly composed program description, then (besides the foregoing) the chosen STM NM1 will also always lie in the same list as the branch node between ST1 and ST2 (and will often be the branch node).

4e7c If ST1 contains a TO(NAM ST2) link, the following rules affect the allowable value of LCN ST2:

4e7c1 DPT LCN ST2 = D2 must be equal to or less than DPT LCN ST1; and FLi LCN ST2 = FLi LCN ST1 for i from 1 to D2-1. For a reentrant branch, equality also will exist when i=D2.

4e7c2 In other words, LCN ST2 can differ only in its last field (and may be equal there) from the string of fields that is derived by truncating LDN ST1 to a depth D2. Equal last fields imply a reentrant branch. For example, if LCN ST1 = 3b4d5, then some of the allowable values for LCN ST2 are 3b4d2, 3b4g, 3b3, 3d, and 6; and some disallowed values are 3d4d2a, 3d4g2, 3b3f, 3d4 and 6b.

4e8 Converse links exist; if statement ST1 links to statement ST2 with link XXX(NAM ST2), this may be explicitly noted in statement ST2 by the converse link -XXX(NAM ST1).

This is a complete and standard link in its own right.

4f Each list or sublist may be thought of as equivalent to a flow chart, and therefore must provide a process description that is complete at its particular level of detail. In such a representation, every point where two or more process-control paths may converge must be associated with the start of a new (named) statement.

4g Parameter-state designation, showing parameter PR1 to have value VL1 at a given point in the process, may be done by writing PR1:VL1, with no spacing on either side of the colon; either punctuation or spacing must appear at the end of the character string designating VL1. The designation of VL1 may be abbreivated or not, according to preference, but using one unbroken character string may avoid ambiguities of statement content.

1 INTRODUCTION

1a Section 1 of this appendix describes the commands available in the on-line system which were not covered in chapter II.

1b Section 2 describes the computer facility and the special peripheral equipment which is used with the system.

2 SUMMARY OF COMMANDS:

2a Input/Output Commands:

2a1 Enter text from designated source into working space on drum.

E P CA	Enter from paper tape.
E M CA	Enter from currently positioned file on mag tape.
E K CA LIT CA	Enter from keyboard--automatically positions display at end of drum's working text, and adds keyboard entry (LIT) character by character to the end.

2a1a This new data is added to the end of the existing working data on the drum.

2a1b The "enter" process will halt when drum is near full, and the typewriter will print appropriate notice. This allows for some free space (about 2000 characters) for copying and inserting. Reinitiating the "enter" command will load until working space is full.

2a1c When entering from a mag-tape file, the tape will remain positioned where the "enter" process stopped, and unless disturbed by an intervening tape-file command, a subsequent E M command will continue reading in that file from that point.

2a2 Output part or all of the working text to the designated device. The working text remains undisturbed. Three characters are required for operation designation.

O P A CA	Output to punch all working text.
O T A CA	Output to typewriter all working text.
O M A CA	Output to currently positioned mag-tape file all working text, replacing prior contents of that file.
O P S S1 S2 CA	Output to punch statements S1 through S2 (S1 may equal S2 for one-statement output).

PRECEDING PAGE BLANK NOT FILMED.

O T S S1 S2 CA	Output to typewriter, statements S1 through S2.
O P P C1 C2 CA	Output to punch partial, characters C1 through C2.
O T P C1 C2 CA	Output to typewriter partial, characters C1 through C2.

2b Scanning Commands.

F S S1 CA	Move forward so as to position statement S1 at top of screen.
F S NUMBER SP	Move forward NUMBER statements.
F L L1 CA	Move forward so as to position line L1 at top of screen.
F L NUMBER SP	Move forward NUMBER lines.
F A CA	Move forward all the way to end of text.
B S S1 CA	Move backward so as to position statement S1 at bottom of screen.
B S NUMBER SP	Move backward NUMBER statements.
B L L1 CA	Move backward so as to position line L1 three lines from bottom of screen.
B L NUMBER SP	Move backward NUMBER lines.
B A CA	Move backward all the way to the beginning of text.

2c Commands relating to linked-statement structures.

2c1 Position display frame on working text of drum.

H N CA LIT CA	Hop to put statement named LIT at top of screen.
H P CA LIT CA	Hop to put statement numbered LIT at top of screen.
H L W1 CA	W1 a link word, i.e., of form TT..T(LL..L); hop to put statement named LL..L at top of screen.
F B S1 CA	Move forward to next logical break in numbering sequence starting from indicated statement.
B B S1 CA	Move backward to next logical break in statement-numbering sequence starting from indicated statement.

2c2 Renumber successive statements in the working text.

N S1 LIT CA	Give statement S1 the new number LIT, and give successive statements correspondingly appropriate new numbers until a statement

ST2 is reached such that either ST2 is of a higher level than S1, or ST2 is not a "logical successor" to the statement preceding it. Display view ends with the predecessor of ST2 at the top of the screen.

2c3 Move or copy statements selected from the display and insert them just before a specified statement somewhere else in the drum-held working text. These operations require a three-character designation.

T S N S1 LIT CA	Transmit (move) S1 to the statement named LIT.
T S P S1 LIT CA	Transmit S1 to the place (statement numbered) LIT.
T L N S1 S2 LIT CA	Transmit the list of statements S1 through S2 to the statement named LIT.
T L P S1 S2 LIT CA	Transmit the list of statements S1 through S2 to the place (statement numbered) LIT.
S S N S1 LIT CA	Copy S1 to statement named LIT.
S S P S1 LIT CA	Copy S1 to place numbered LIT.
S L N S1 S2 LIT CA	Copy list, S1 to S2, to statement named LIT.
S L P S1 S2 LIT CA	Copy list, S1 to S2, to place numbered LIT.

2d Utility Commands.

2d1 Locate and examine tape-file items. Each fixed-length item space can hold a full drum load of working text, and the items are referenced by decimal-integer serial number corresponding to their order on the tape. Any "look" operation displays the first frameful of text from the tape without either disturbing the drum data or losing the position on tape.

L H CA	Look here, i.e., at text just beyond current position on tape.
L I NUMBER CA	Look at item numbered NUMBER--positions tape at head of the item and provides a look.
L N CA	Look at next item--the one just beyond the current position.
L P CA	Look at prior item--the one just ahead of the current position.

2d1a Trying to look beyond the last item, either with L I NUMBER for too large a NUMBER, or with a L N from the very last item of the file, will produce the displayed message, "Beyond last item."

2d1b An O M command at this point will create a new item on the end of the file.

2d2 Clear the working space on the drum of its present contents.

Z W S	Zero work space.

2d3 Type out system-status data.

O S CA	Output system status, causes typing in the form: x channels left, item y last read in, tape positioned to item z.

3 ON-LINE COMPUTER EQUIPMENT

3a The computer: CDC 160A.

3a1 Memory: The cycle time is 6.4 usec. There are two memory banks with 4,096 12-bit words each, directly addressable. Each bank has independent access circuitry. Bank control is set by the program, for four categories of access.

3a2 Instruction repertoire: full complement of add, subtract, conditional branch, transfer, logical product, selective complement, shift, input-output, and selective stop and jump (responding to console switches). Since 12 bits can just exactly address 4096 words, instructions requiring operand specification over a complete bank require two successive words (one for operation specification, one for operand specification). A significant proportion of the instructions require only one word, however, and use 6 bits of operand specification in one of five special addressing modes. Variations in the op-codes of nearly all the instructions indicate which way the operand is to be obtained for that instruction.

3a3 Interrupt feature: any of four independent sources (two internal, two external) may cause an interrupt. An interrupt signal causes the contents of the program counter to be saved in a special cell; the computer then gets its next instruction from the succeeding cell. The interrupts may be locked out or enabled by program.

3a4 Input-ouput provision: there are two input-output channels that can operate independently (the "normal" and "buffer" channels). Selecting an input or output device causes all subsequent input or output operations to use that device, until a different one is selected. There are a family of single-word transfer instructions (sending or receiving one word per instruction), as well as a family of block-transfer instructions (sending or receiving arbitrary-length blocks to or from consecutive cells of memory, at the rate determined by the external device).

3b Peripheral Equipment:

3b1 Paper tape reader: The reader is a photo-electric device that can read at an asynchronous rate up to a maximum of 320 frames per second. It will accept 6-, 7-, or 8-level tape, and is always connected to the normal channel.

3b2 Paper tape punch: The punch is a Teletype product, punching 8-level oiled tape, at an asynchronous rate up to to a maximum of about 120 frames per second. It is always connected to the normal channel.

3b3 On-line typewriter: This is an IBM typewriter, with a CDC interface. It can be connected to either the normal or the buffer channel.

3b4 Magnetic tape transport: The tape unit is a CDC Type 603, compatible with IBM equipment. The programmer can write records of arbitrary length with the transport automatically leaving inter-record gaps. There is an "end-of-file" code that can be put on under program control. The unit will read forward one record at a time, or back up one record at a time, from a single instruction.

3b5 Drum: The drum is a 32,000-word, fixed-head auxiliary storage device with a speed of about 30 revolutions per second. It can make access only to records. There are 32 tracks, with 2 records per track, each containing 512 12-bit words.

3b6 Interface logic: The interface logic unit provides the logic for gating information between special equipment used with the on-line system and the CDC 160A computer.

3b6a Special devices that communicate with the computer by means of the interface are: bell, pushbuttons, pushbutton lights, interrupt circuits. analog-to-digital converter, and light pen, (The keyboard has a separate interface discussed in the "keyboard" section below.)

3b7 Operand-locating devices: Operand entities displayed on the screen of the CRT display are selected by selecting a character within the operand entity (word, line, or statement). The character is selected with either a light pen or a bug-positioning device. The light pen or bug is first located near the desired character, then the SELECT switch on the device is depressed (or alternatively, the CA button on the control panel or the keyboard may be struck).

3b7a Bug-positioning Devices: All the bug-positioning devices are of the resistive voltage divider type. The outputs of the voltage dividers are fed into a Dynamic Systems Electronics Model ADC-2C-4M analog-to-digital converter. This converter has four analog input channels which are automatically sampled in sequence, and is capable of performing a conversion every 400 microseconds. The converter produces ten bits (nine of which are used as computer inputs) plus sign, with an input range of plus or minus 1.0225 volts and a resolution of one millivolt. (We decided to use voltage dividers and an analog-to-digital converter (as opposed to some less expensive scheme, such as digital shaft position encoders) because of the flexibility offered by the converter. By changing voltage supply settings, such factors as zero position or device sensitivity are easily adjusted, requiring no change in the software.) The following bug-positioning devices provide the analog inputs to the analog-to-digital converter (one horizontal and one vertical, each to two inputs):

3b7a1 Grafacon (see Figure 13): The Grafacon (Ref 2g(FLETCHER1)) was manufactured by Data Equipment Company as a graphical input device for curve tracing. (The device that we have is no longer available. Data Equipment Company now markets the Rand Tablet under the name "Grafacon.")

3b7a1a The Grafacon consists of an extensible arm connected to a linear potentiometer. The housing for the linear potentiometer, in turn, is pivoted on an angular potentiometer. The angular range is plus or minus 50 degrees from center, and the range of extension is 10 inches.

3b7a1b The voltage outputs from the Grafacon represent polar coordinates about the pivot point, but are interpreted by the system exactly as the outputs from the "mouse" or joystick, which represent rectangular coordinates. This means that to trace a straight line across the screen with the bug, the user must actually move his hand in a slight arc.

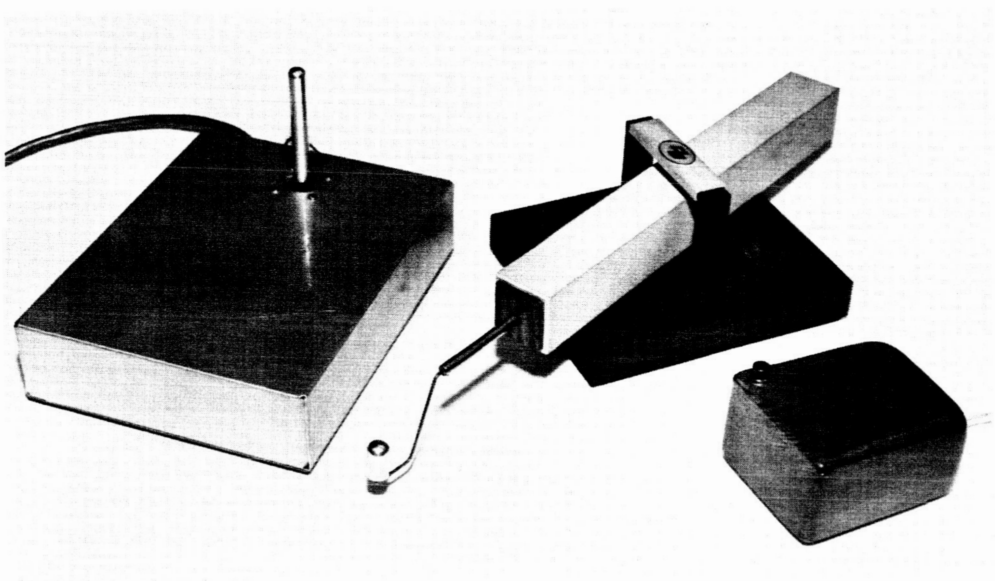


FIG. 13 BUG-POSITIONING DEVICES. From Left to Right: Joystick, Grafacon, and Mouse

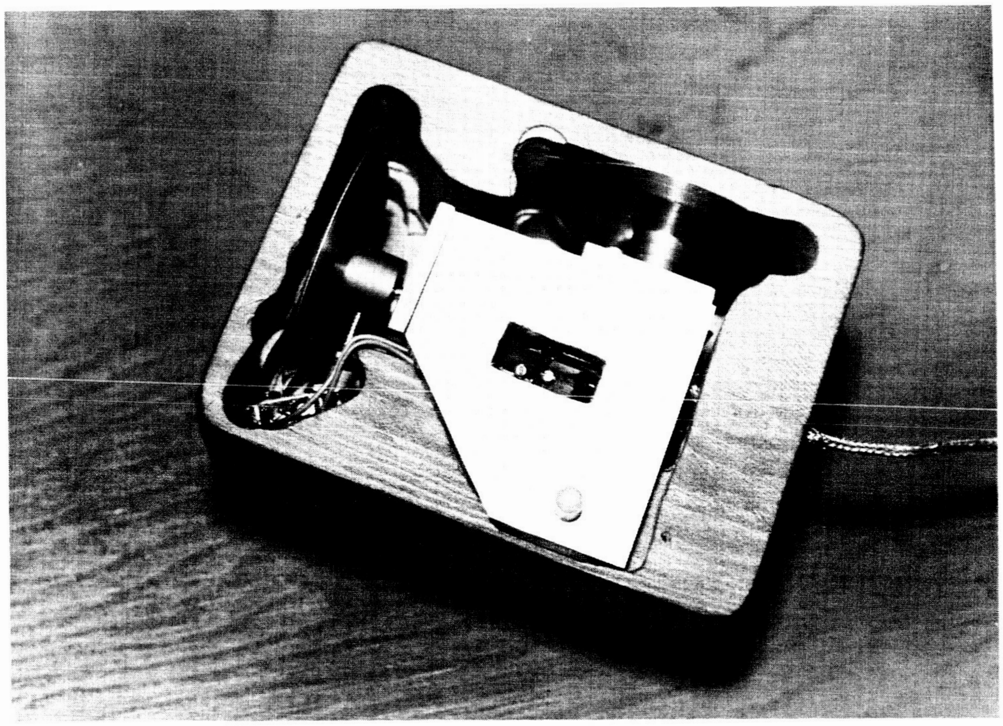


FIG. 14 BOTTOM SIDE OF MOUSE, SHOWING MECHANICAL DETAILS

3b7a1c The Grafacon as originally obtained was equipped with a ball point pen mounted in a linkage which permitted the handle of the pen to move about while the point remained fixed relative to the potentiometers. This mounting did not operate smoothly and was not really needed for our purposes, so the pen was replaced with a fixed knob. This knob is moved about by the user, and is depressed to activate the select switch (added by SRI) associated with the Grafacon.

3b7a2 Joystick (see Figure 13): The joystick is manufactured by Bowmar Associates, Model X-2438,

3b7a2a It is constructed from two potentiometers, mounted perpendicularly and coupled to a vertical stick in such a way that they resolve the motion of the stick into two components. One output is used for information about vertical position, and the other for horizontal.

3b7a2b Two modes of operation with the joystick were implemented: An "absolute" mode, in which the bug's position on the screen corresponds to the position of the joystick handle; and a "rate" mode, in which the bug's direction of motion is determined by the direction of joystick handle deflection, and the bug's rate of motion is determined by th amount of joystick deflection.

3b7a2c The original stick was 1 1/2 inches long; a 3 inch extension to the shaft, housing a switch was added by SRI. The switch is actuated by pressing down on the stick itself. A maximum stick deflection of 28 degrees in any direction from its spring-loaded center position is possible.

3b7a3 Mouse (see Figure 13): The "mouse" was developed by this project. It is constructed from two potentiometers, mounted orthogonally, each of which has a wheel attached to its shaft (see Figure 14). The mounting frame for the potentiometers is enclosed in a 2" x 3" x 4" (HWD) wooden case. As the case is moved over a surface, the wheels ride on the surface and turn the potentiometer shafts. The motion is resolved into two components. A travel of about five inches is required for full edge-to-edge or top-to-bottom coverage of the CRT screen. A switch mounted on the case is used for the select function.

3b7a4 Knee Control (see Figure 15): The knee control, a preliminary model made for the project, consists of two potentiometers and associated linkage plus a knee lever. The linkage is spring-loaded to the right and gravity-loaded downward. The user pushes the lever with his knee; a side-to-side motion of the knee moves the bug edge-to-edge, while the top-to-bottom bug movement is controlled by an up-and-down motion of the knee (i.e., a rocking motion on the ball of the foot). The horizontal range of motion is 60 degrees; the vertical range is 20 degrees, for full edge-to-edge and top-to-bottom deflections respectively.

3b7b Light Pen (see Figure 16): The light pen is manufactured by Sanders Associates of Nashua, New Hampshire; it is their Model EO-CH.

3b7b1 The unit consists of a hand-held pen and a detector electronics package. These two are connected by a flexible cable that contains a fiber optic bundle as well as wires. A photo-multiplier tube in the elctronics package receives light through the fiber optic bundle from the hand-held pen, which contains a lens that focuses light on the bundle. When a light pulse with a suitably fast rise time is detected an electrical pulse is generated in the electronics package. The switch on the body of the pen unit gates this pulse to the interface logic.

3b7b2 Only a single pulse is transmitted to the interface unit after the pushbutton is depressed. Thus, the first character "seen" by the light pen after the button is depressed causes a pulse to be transmitted.

3b7b3 When the logic interface receives a pulse from the light-pen control unit, an interrupt is sent to the computer and the six most significant bits of the last computer output word are stored. (These six bits represent the horizontal position of the display character that produced the light pulse.)

3b7b4 A circle of orange light is projected from the pen unit as a locating aid. This circle, indicating the field of view of the lens system, is transmitted from a source in the electronics package to the hand-held pen, through a group of fibers in the same fiber optic bundle that is used to transmit light from the pen to the photo-multiplier in the electronics package.

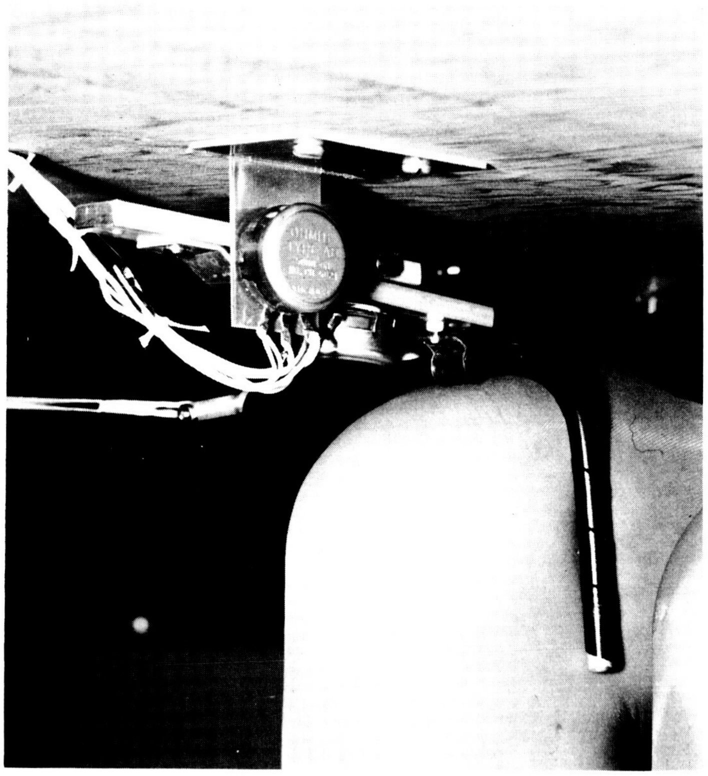


FIG. 15 KNEE CONTROL BUG-POSITIONING DEVICE

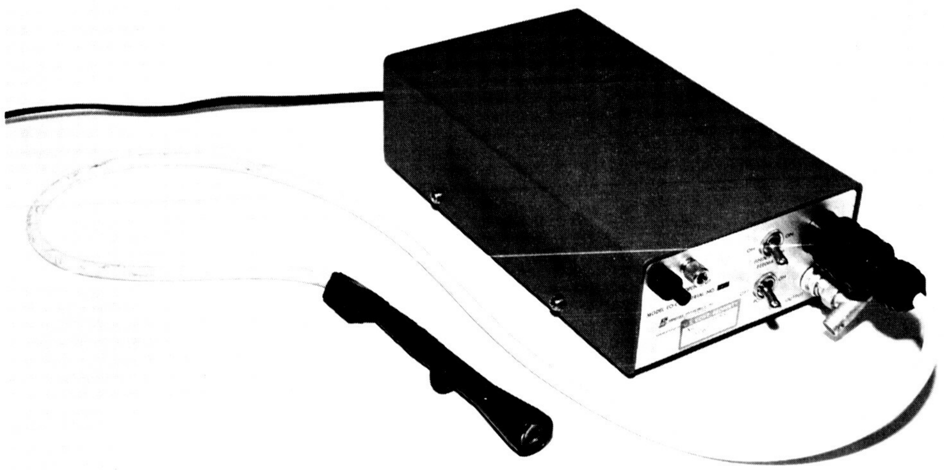


FIG. 16 LIGHT PEN

3b8 Bell: A signalling bell is mounted near the CRT display and is rung under computer control.

3b9 Control panel: A pushbutton control panel, built by the project, is available for entering commands. The layout of this panel is shown in Figure 17. The top bar ("CD") is for the "Command Delete" function, and the bottom bar ("CA") for the "Command Accept" function. The two buttons at the left-hand side of the panel are used for entering the direction of scan. Basic editing commands are entered by depressing one of the right-hand group of five buttons in the first row and one in the second row. For example, the "Delete Word" command would be entered by depressing the "D" and "W" buttons.

> 3b9a The twelve pushbuttons in the top and bottom rows are of the illuminated type. The pushbutton lights are under computer control from a light gating circuit in the interface unit, and are in no way connected to the buttons. These lights indicate the present command, even if the command was entered from the keyboard rather than from the control panel.
>
> 3b9b The pushbuttons are encoded by a diode matrix so that the top row of six buttons and the "CD" bar are gated by the interface logic onto three computer lines, and the bottom row of six buttons plus the "CA" bar are gated onto another group of three computer lines. The output of the interface gating circuits may be connected to any input line by means of a patch panel on the interface rack.
>
> 3b9c It should be noted that the encoded pushbuttons are gated into the computer without "bounce protection," and that protection must therefore be provided by the computer program.

3b10 Keyboard: The experimental control console uses a 64-key photoelectric keyboard (Model PK-164) manufactured by the Invac Corporation, with keys and codes as specified by SRI. The keyboard has an SRI-constructed interface for gating keyboard outputs into the computer. The layout of the keyboard (see Figure 17) is designed to be similar to that of the Teletype machines used in other phases of the SRI Man-Machine program. In addition to the usual alphanumeric and punctuation keys, there are keys for the special control functions "Command Accept," "Command Delete," and "Backspace Word" (the "Command Accept" appears at each side of the keyboard, for easy operation). The keyboard was modified by SRI so that the shift keys do not generate a strobe pulse.

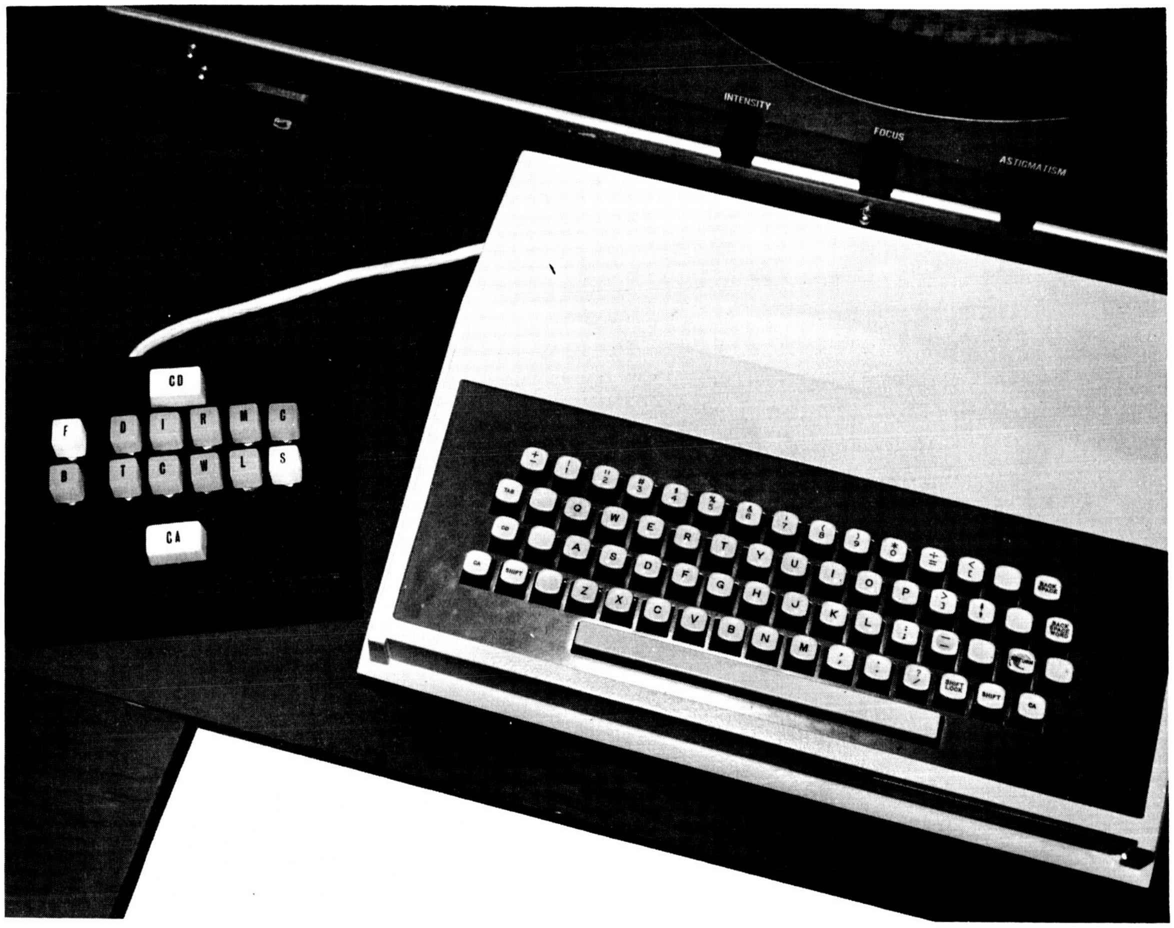


FIG. 17 PUSH-BUTTON PANEL (With Forward Statement Operator Indicated by Push-Button Lights) AND ON-LINE KEYBOARD

These keys are not mechanically interlocked with other keys on the keyboard and, when pressed simultaneously, with some other key, add the seventh bit to the code produced by the other key.

3b10a Because of the photoelectric operation it is relatively easy to change the coding for any given character, or to alter the placement of the keys in the keyboard.

3b10b The touch is light, and an adjustable servo-assist provides the user with mechanical feedback.

3b10c The keyboard is equipped with a key interlock mechanism that prevents the actuation of more than one key at a time. If desired, the interlock can be controlled remotely to inhibit the actuation of the keys (with the exception of the shift key).

3b11 CRT display and character generator: The 16-inch cathode ray tube display used in the experimental control console is an electrostatic unit maunfactured by Data Display Incorporated. It is used in conjunction with a CDC Model 220 character generator (a prototype no longer available), which provides the display with unblank and deflection signals, and serves as a computer interface.

3b11a The character repertoire consists of the 36 alphanumeric characters, plus PERIOD, DASH, EQUALS, RIGHTSLASH, LEFTSLASH, PLUS, and CENTERDOT.

3b11b The present system uses 16 lines of display plus the computer feedback line with a maximum of 64 characters per line. Character writing time is about six microseconds, but the character rate is limited to the maximum computer output rate of one word every 15.5 microseconds.

1 Publications involving task analysis and classification of tasks. These papers generally comment on the problems involved in defining and classifying human tasks as related to man-machine interactions.

1a (ARMSBY1) Armsby, Donald H. and Charles E. Zeleny, "Design Standards for Man-Machine Tasks in Signal Corps Systems," Quarterly Progress Report No. 5, Applied Psychology Corp., (1 July - 1 October 1960).

1b (ARMSBY2) Armsby, Donald H., "Task Demands Analysis," Human Factors, pp. 381-387, (December 1962).

1c (CHENZOFF1) Chenzoff, Andrew P., "A Review of the Literature on Task Analysis Methods," Technical Report: NAVTRADEVCEN 1218-3 Applied Science Associates, Inc., Valencia, Pennsylvania, prepared for U. S. Naval Training Device Center, Port Washington, New York, Contract N61339-1218, (June 22, 1964).

1d (COTTERMAN1) Cotterman, T. E., "Task Classification: An Approach to Partially Ordering Information on Human Learning," WADC Technical Note 58-374 (January 1959) ASTIA 210 716.

1e (FOLLEY1) Folley, John D., "Development of an Improved Method of Task Analysis and Beginnings of a Theory of Training," Technical Report: NAVTRADEVCEN 1218-1, Applied Science Associates, Inc., Valencia, Pennsylvania, prepared for U. S. Naval Training Device Center, Port Washington, New York, Contract N61339-1218, (June 22, 1964).

1f (FOLLEY2) Folley, John D., "Guidelines for Task Analysis," Technical Report: NAVTRADEVCEN 1218-2, Applied Science Associates, Inc. Valencia, Pennsylvania, prepared for the U. S. Naval Training Device Center, Port Washington, New York, Contract N61339-1218, (June 22, 1964).

1g (ZELENY1) Zeleny, Charles E., "Design Standards for Man-Machine Tasks in Signal Corps Systems," Applied Psychology Corp., Quarterly Progress Report No. 6, (1 October - 1 January 1961).

1h (ZELENY2) Zeleny, Charles E. and Donald H. Armsby, "Design Standards for Man-Machine Tasks in Signal Corps Systems," Fifth Quarterly Progress Report, (1 July 1960 - 1 October 1960).

1i (ZELENY3) Zeleny, C. E., F. J. Mc Grane, and H. D. Lerner, "Design Standards for Man-Machine Tasks in Signal Corps Systems," 7th Quarterly Report, Army SRDL, Fort Monmouth, New Jersey, (January - April 1961).

2 Papers on devices for graphical or textual input/output to a man-machine system and schemes for human control and communication.

2a (BACON1) Bacon, C. R. T., "Text Editing Display: Project Upgrade," University of Pittsburgh, (April 8, 1965).

2b (COONS1) Coons, S. A., "Notes on Graphical Input Methods," Memo 8436-M-17, Massachusetts Institute of Technology, Department of Mechanical Engineering, (May 4, 1960).

2c (DAVIS1) Davis, M. R. and T. O. Ellis, "The RAND Tablet: A Man-Machine Graphical Communication Device," AFIPS Conference Proceedings, Vol. 26, Spartan Books, Inc. (1964).

2d (DEININGER1) Deininger, Richard L., "Desirable Push-Button Characteristics," IRE Transactions of the Prof. Group on Human Factors in Electronics, Vol. HFE-1, No. 1, pp. 24-30, (March 1960).

2e (ENGELBART1) Engelbart, D. C., "Augmenting Human Intellect: Experiments, Concepts, and Possibilities," Summary Report, Contract AF 49(638)-1024, SRI Project 3578, Stanford Research Institute, Menlo Park, California (March 1965).

2f (ENGELBART2) Engelbart, D. C. and P. H. Sorensen, "Explorations in the Automation of Sensorimotor Skill Training," Technical Report: NAVTRADEVCEN 1517-1, Stanford Research Institute, Menlo Park, Calif., (1965).

2g (FLETCHER1) Fletcher, William E., "On-Line Input of Graphical Data," paper presented to Digital Euipment Computer Users Society Meeting, 18, 19 Nov., 1963, Livermore, Calif., (1963).

2h (HUFFORD1) Hufford, L. E., and R. Coburn, "Operator Performance on Miniaturized Decimal-Entry Keysets," U. S. Navy Electronics Laboratory, San Diego, California, Research Report 1083, (December 4, 1961).

2i (INFORONICS1) Inforonics, Inc., "Study of the Use of Man-Machine Consoles for Text Processing," Final Report of work performed from May 20 to Dec. 6, 1963, under P.O. No. RX39409, (December 6, 1963).

2j (LEVINE1) Levine, S., "Present Types and Future Trends in Input/Output Equipment for Information-Handling Systems," Technical Paper presented at the S.I.T.A. Symposium, Brussels, (1963).

2k (MINOR1) Minor, Frank J. and Stanley L. Revesman, "Evaluation of Input Devices for a Data Setting Task," Journal of Applied Psychology, Vol. 46, No. 5, pp. 332-336, (1962).

2l (POLLUCK1) Polluck, William T. and Gilbert G. Gildner, "Study of Computer Manual Input Devices," Technical Documentary Report No. ESD-TDR 63-545, Project 9678, Task 967801, prepared under Contract No. AF19(628)-435 by Bendix Systems Division, Ann Arbor, Michigan, Decision Sciences Laboratory Electronic Systems Division, Air Force Systems Command, United States Air Force, L. G. Hanscom Field, Bedford, Massachusetts, (September 1963).

2m (RAND1) RAND, Description of the RAND-ARPA Graphic I/O Project Display System, unofficial working paper.

2n (RUTCHKA1) Rutchka, Alexander, "Graphical Output Device," MAC-M-120, (December 21, 1964).

2o (SAMPSON1) Sampson, Philip B. et al, "The Feasibility of Using the Eye as a Source of Control Signals in Tracking," Institute for Applied Experimental Psychology Report, ASTIA, AD No. 231516, Tufts University, Medford, Mssachusetts, DE-15435, (December, 1959).

3 Publications related to CRT displays and evaluation of display systems for man-machine communication.

3a (BOWERS1) Bowers, W. J., "A Survey of Character Display Devices," Electronic Packaging and Production, Vol. 3, No. 10, pp 10-16, (October 1963).

3b (BUCKLAND1) Buckland, L. F., "The Use of a Cathode Ray Tube Display Console for Editing Textual Information," Automation and Scientific Communications, Topic 9: Document Storage, Retrieval and Display, American Documentation Institute, 26th Annual Meeting, Chicago, Ill., pp. 179:180, (October, 1963).

3c (DAMMANN1) Dammann, J. E., E. J. Skiko, and E. V. Weber, "A Data Display Subsystem," IBM Journal of Research and Development, Vol. 7, No. 4, pp. 325-333, (October 1963).

3d (LAWSON1) Lawson, Curtis G., "An Integrated Display and Control System for Man-Machine Communication." U. S. Naval Postgraduate School, (1962).

3e (MITCHELL1) Mitchell, J., "Specification for Display Console, Project 500," Working Paper from the Mitre Corporation, (February 7, 1962).

3f (MITCHELL2) Mitchell, J., "Project 500 Display Console Acceptance Tests," Working Paper from the Mitre Corporation, (August 31, 1962).

3g (MITCHELL3) Mitchell, J., "Project 500 Display Console Acceptance Tests," Working Paper from the Mitre Corporation (January 8, 1963).

3h (NIELSEN1) Nielsen, R. S., "Specification for Display Console (Project 500)," Working Paper from the Mitre Corporation, (August 28, 1961).

3i (RISING1) Rising, H. K., "Studies of Display Symbol Legibility: The Effects of Line Construction, Exposure Time, and Stroke Width," Project 703, Technical Memorandum TM-3515, Mitre Corporation, (December 4, 1962).

3j (SMITH1) Smith, Sidney L., "Color-Coded Displays for Data Processing Systems," Electro-Technology, pp. 63-69, (April 1963).

4 Publications concerned with the significance of man-machine systems for editing and design.

4a (CULLER1) Culler, Glen J. and B. D. Fried, "The TRW Two-Station On-Line Scientific Computer," Computer Augmentation of Human Reasoning, Spartan Books, Inc., Washington, (1965).

4b (JOHNSON1) Johnson, D. L. and A. L. Kobler, "Man-Computer Interface Study," Contract No. AF49(638)-1070, Dept. of Elec. Engineers, University of Washington, Seattle, Washington, (June, 1962).

4c (JOHNSON2) Johnson, D. L. and A. L. Kobler, "Man-Computer Interface Study," Grant No. AF-SR-62-366, Dept. of Electrical Engineering, University of Washington, supported by AFOSR, (June, 1963).

4d (LICKLIDER1) Licklider, J. C. R. and Clark, W. E., "On-Line Man-Computer Communication," Proceedings Spring Joint Computer Conference, Vol. 21, pp. 113-128, National Press, Palo Alto, Calif., (May, 1962).

4e (NUGENT1) Nugent, W., "The Automated Editing of Text Via Computer and Off-line Devices," Automation and Scientific Communication, Topic 2: Recording of Information, American Documentation Institute, 26th Annual Meeting, Chicago, Ill., pp.

125-126, (October, 1963).

4f (SUTHERLAND1) Sutherland, I. E., "Constraint Satisfaction for Computer-Aided Design," (proposal for thesis research), MIT Research Lab of Electronics, Cambridge, Massachusetts, (January 10, 1962).

4g (THOMAS1) Thomas, Ralph E. et al, "The Effect of Various Levels of Automation of Human Operators Performance in Man-Machine Systems," Batelle Memorial Institute, Wright Air Development Division, WADD Tech. Report 60-618, (February 1961).

5 Papers about systems for computer-aided graphical and textual design.

5a (COONS2) Coons, S. A., "An Outline of the Requirements for a Computer-Aided Design System," from Collected Papers of the Sessions on Computer-Aided Design, Spring Joint Computer Conference, Detroit, May 23, 1963, MIT, Cambridge, Massachusetts, (March, 1963).

5b (ENGELBART3) Engelbart, D. C. and Bonnie Huddart, "Research on Computer-Augmented Information Management," Technical Report No. ESD-TDR-65-168, Contract AF 19(628)-4088, Stanford Research Institute, Menlo Park, Calif., (March 1965).

5c (ITEK1) Itek Corporation, "Operating Instructions, Digigraphic System," Lexington, Massachusetts, (March, 1963).

5d (JACKS1) Jacks, Edwin L., "A Laboratory for the Study of Graphical Man-Machine Communication," paper found in AFIPS Conference Proceedings, Vol. 26, Spartan Books, (June, 1964).

5e (MIT1) MIT, "Investigations in Computer-Aided Design," Interim Engineering Report 8436-IR-1-MIT Project No. 8477, Air Force Contact No. AF-33(600)-40604, Massachusetts Institute of Technology (December 1, 1959 - May 30, 1960).

5f (STRUBLE1) Struble, William, "A Computer-Aided Design System," The Technology Review, pp. 27-28, (March, 1963).

5g (SUTHERLAND2) Sutherland, I. E., "Sketchpad: A Man-Machine Graphical Communication System," from Collected Papers of the Session on Computer-Aided Design, SJCC, Detroit, May 23, 1963, MIT, Cambridge, Massachusetts, (March 1963).

6 Reports from Project MAC pertinent to their present system and its related software and command structure.

6a (BAYLES1) Bayles, Richard U., "New Operating System for the E/S/L Display Console," Memorandum 9443-M-118, Memorandum MAC-M-201, (December 10, 1964).

6b (FANO1) Fano, R. N., "The MAC System: A Progress Report," Computer Augmentation of Human Reasoning, Spartan Books, Inc., Washington, (1965).

6c (KLIMAN1) Kliman, E. M., "Abstracts of CTSS Console Commands," MAC-M-174-2, CC-239-2, (January 4, 1965).

6d (MIT2) MIT, "Editing Systems at Project MAC," (December 22, 1964).

6e (SALTZER1) Saltzer, J. H., "TYPESET and RUNOFF, Memorandum Editor and Type-Out Commands," CC-244-2, MAC-M-193-2, (January 11, 1965).

7 Papers pertinent to software for future Man-Machine systems at SRI.

7a (DEUTSCH1) Deutsch, L. P. and B. W. Lampson, "DDT Time-Sharing Debugging System Reference Manual, Internal Memo No. 30.40.10, Berkeley Remote Station Research Project, University of California, Berkeley, Calif. (March 25, 1965).

7b (FARBER1) Farber, D. J., R. E. Griswold, and I. P. Polonsky, "SNOBOL, A String Manipulation Language," Journal of the Association for Computing Machinery, Volume 11, No. 2 (January, 1964)."