ESD-TDR-65-168

(FINAL REPORT)

RESEARCH ON COMPUTER-AUGMENTED INFORMATION MANAGEMENT

TECHNICAL DOCUMENTARY REPORT NO. ESD-TDR-65-168

MARCH 1965

D. C. Engelbart
Bonnie Huddart

DIRECTORATE OF COMPUTERS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

(Prepared under Contract No. AF 19 (628)-4088 by Stanford Research Institute, Menlo Park, California.)

ESRC

AD622520

ESD-TDR-65-168

(FINAL REPORT)

RESEARCH ON COMPUTER-AUGMENTED INFORMATION MANAGEMENT

TECHNICAL DOCUMENTARY REPORT NO. ESD-TDR-65-168

MARCH 1965

D. C. Engelbart
Bonnie Huddart

DIRECTORATE OF COMPUTERS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

(Prepared under Contract No. AF 19 (628)-4088 by Stanford Research Institute, Menlo Park, California.)

This report presents results of a research and experimental project in computer-aided information management. The report is itself a product of the project: with the exception of "front matter," the entire report was composed, edited, and produced with on-line and off-line computer aids.

For this project, the techniques of computer aids were applied to two areas: task monitoring and program design. The processes and techniques developed offer a promising beginning to computer-aided programming design extending from initial specification to final debugging in a unified design record that grows and evolves to complete final documentation. The processes and techniques also offer promise in increasing the productivity of individuals and groups of programmers.

Future work envisioned for information-management systems such as that used in this study include program design records, external-reference documentation, and user reference manuals.

REVIEW AND APPROVAL

This technical documentary report has been reviewed and is approved.

Frank E. Herin, Jr.
FRANK E. HERIN, JR.
1st Lt., USAF
Project Officer

C. K. Wilcox, Lt Colonel, USAF
for PAUL G. GALENTINE, JR.
Col., USAF
Director of Computers
Deputy for Engineering & Technology

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 Documentary Reports issued by Electronic Systems Divisions and its contractors. A glance at the pages of this report 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--oftentimes 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 manipulation 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 investigating a man-machine "system" 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 other symbols e.g., graphics, mathematics, chemical formulas--(which, for the most part, are actually quite isolated in the context of our total working system).

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

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

The on-line system includes the following facilities:

- CDC 160A computer, with storage for 8,000 12-bit words of core storage, 6.5 μsec access time, auxiliary storage provided by a 32,000-word drum and one magnetic tape transport; paper tape input/output facilities.
- CDC 220 character generator and DDI 16-inch monitoring scope to provide on-line display.
- Invac keyboard, Saunders Associates light pen, and other various graphical input devices for on-line operator input.

Using this system, about 18,000 characters of working data can be written on the 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, move, copy, replace, delete, and insert, can be performed on entities of character, word, line, or statement. 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. Using the off-line system, substructures of text of any size can be deleted or moved with a few simple commands, new statements and substructures can be inserted as desired, and existing statements can be modified. Presently, turn-around time for the off-line system is a half day or more. This makes it more limited in its applications than the on-line system; however, some tasks--such as updating operations--are easier to perform off-line than on-line.

We come, then, to the basic and visible difference between this report and other ESD Technical Documentary Reports: With the exception of this Foreword and other front matter, this report is produced entirely on the on-line and off-line systems that are 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 at the end of the report, are shown in the text by their computer mnemonic (e.g., Ref (SRI 1), rather than by the more familiar superscript notation.

Detailed study of this report requires familiarity with the terms, concepts, and computer-aid processes developed in this program; these are contained in Ref (SRI 1), a copy of which is printed as the appendix to this report.

Under Contract AF 19(628)-4088, ESD has sponsored study of structuring and manipulating techniques for management of information--specifically, the system-program design documentation. Other projects supporting the program are a recently completed project for Air Force Office of Systems 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 current project with National Aeronautical and Space Administration (Contract NAS 1-3988), under which display-control techniques that represent the foundation of the on-line manipulative system are being studied and developed; and an internally-sponsored project at Stanford Research Institute, under which the current off-line system was developed.

1 BASIC ROLE OF PROJECT WITHIN THE PROGRAM

1a To explore the possibilities of using closely coupled computer aids for performing significant information-management tasks, by developing and experimenting with improved information-management techniques for our own everyday use within the program.

2 THE FORMULATION OF OBJECTIVES IN THE ORIGINAL PROPOSAL

2a The specific objectives of the proposed study are to develop systems of hardware, software, concepts, and methods that will permit the on-line operator to:

2a1 Analyze and structure information in a quantity and variety that significantly exceed the capability of a human not aided in this fashion.

2a2 Update the information structure in response to more rapid changes in information or user need than he could previously have accommodated.

2a3 Retrieve and compile significant information from the structure more quickly and comprehensively.

2b There has been one apparent qualification of this original formulation.

2b1 Our work has been to harness computer aids for the type of information needs sketched above; but we have not restricted this to on-line aids--we have also explored and gained working experience with off-line man-computer cooperation. The program-design documentation study discussed in Section IV is one example of this; this report itself is another.

2b2 The Information-Management project has been particularly stimulated and aided by the potentials opened up by the operation of our off-line text-processing system. Our formulation of objectives must now take shape within this new set of needs and potentials.

2c The objectives we pursue in this Information-Management project are best conceived as a particular kind of user-system research, in the sense described in Ref(OSR2).

2c1 The total context is "the many coordinated aspects of human intellectual effectiveness."

2c2 The particular aspects we explore are "the coordinated set of concepts, conventions, methods, and skills" which enable a human problem-solver to harness computer aids in managing his working information.

2c3 This includes schemes for structuring information, articulating it in special ways to bring out its various kinds of significance (e.g., see Section III); techniques for modifying, updating, and consulting this body of structured information; plus the human procedures, methodology, and skills that knit these together into an effective user system.

3 METHOD OF APPROACH

3a The overall basic method of approach throughout this project has been:

3a1 To take real, live information management problems from our own working environment.

3a2 To derive tentative solutions that utilize the hardware and software products of the other projects in the program.

3a3 To implement these solutions in rough, preliminary, experimental versions, and try them out, in order to gain working experience as a basis for evaluating their functional weaknesses and potentials.

3a4 To continue from this point, modifying and adding to the system to evolve continually better solutions and to expand the scope of problems being handled.

3b This basic method of approach has two unusual characteristics:

3b1 We must largely follow where the problem leads. This is exploratory research, without a predetermined itinerary; the needs brought out in our changing environment influence our course.

3b2 We must coordinate closely with the other projects within the program, by developing, applying, and testing products they can use, and using the products they provide. The changing possibilities of our working environment influence our course.

3c The initial, more specific formulation of this general method of approach, governing our work in the earlier stages of the project, had two aspects.

3c1 The project was to assume responsibility for specifying and monitoring, in an overall way, an information-managing scheme for the working information involved throughout all the projects in the program. This would include specifying the structuring conventions, the terminology, and the procedures of information management to be followed by all our projects.

3c2 Within some smaller "focal" area (representative in its dimensionality, but more manageably delimited in its scope), we would as rapidly as possible specify, design, implement, and gain working experience with an actual information-management subsystem incorporating our computer aids.

3c2a This subsystem was to be conceived as a balanced, coordinated set of information formats and structuring conventions, terminology and notations, and procedures for entering information and maintaining useful up-to-date records that could be quickly and flexibly consulted.

3c2b We would select an area where the quantity, complexity, and variety of information, and the functional requirements, were small enough so that we could develop useful models and evaluative techniques.

4 EVOLUTION OF OUR WORK

4a The specific area initially selected for the "focal" study was a body of status information about the programming work in progress. A trial scheme of task definition and status reporting was implemented and operated for several months (see Section IV).

4a1 We found that for this ever to become a really useful body of working information we would need far more detailed task descriptions, and easier ways of modifying them.

4a2 Tasks are hierarchical in nature--to give a detailed description of a task usually involves isolating its subtasks, together with the resources, constraints, method of approach, etc.

4a3 The "linked-statement" structuring conventions (which had meanwhile been developed within the ARPA, NASA, and internally sponsored projects) adapt very naturally to representing these types of relationships. For instance, the linked-statement structuring of a task-description allows analysis to whatever depth might be relevant or useful in the particular case.

4a4 We soon found that for purposes of analyzing status the best programming-task description was the description of the current state of the program design. But obviously this type of evolving record would be useful for other purposes than as an input to a task-monitoring system.

4a5 It seemed likely, for instance, that these methods of depicting the design records could prove very powerful for documentation of our (and others') programming-system development work.

4a6 The fast and efficient computer processes for modifying such evolving structures promise to make updating these records easy and quick enough so that the system designer or programmer can actually do his designing work (including his "scratch work") this way.

4b Our more recent activity has investigated this area in specific detail, as well as re-examined our overall information-management system in the light of the computer aids which other projects had made available to us. These new structuring conventions and processing abilities proved to be well suited for describing computer program structures. (Our multilevel program-design explorations are described in Section III.)

4b1 System-program documentation has offered a good workout for the new capabilities; it provides variety and complexity enough to test the conventions and processes.

4b2 We have developed and (to some extent) refined a reasonably adequate and useable set of descriptive techniques for recording complex program structures, embedding the relevant kinds of supplementary information and commentary at the appropriate points. The information is formatted, tagged, and linked in special ways to make this a usefully articulated record, and give aid in comprehension.

4b3 We have also begun to explore how these same structure and processing conventions could be used for developing a program description while the program is being designed and written--to incorporate descriptive material about data structures; record design considerations and decisions; explain special coding devices; and so on. We hope to develop a programming methodology incorporating these aids throughout the entire design process, providing an evolving up-to-date record of the work in progress. With very little reworking, the design record would then become the final documentation of the finished program--an unusually complete and useful documentation.

4c The new structure and processing aids will also be valuable to us in service of other information-management needs:

4c1 They provide a far more flexible and useful framework for our group documentation than the "file folder" descriptors we worked out earlier in the project; a framework for exchanging and merging information, and for maintaining an up-to-date central file of "reference" documents (such as our supplementary reference, "SRI1," printed as an Appendix to this report).

4c2 They can be applied within our external citation files and documents, in order to search and classify the contents of those files, and to compile materials for special purposes.

4c3 They provide the required tools for devising a realistic and mobile scheme of status-reporting and task-definition, which would allow both a more effective coordinating of group activity and a more accurate (and less burdensome) monitoring of individual progress.

1 The dominant features of the work reported here are that the work itself is part of an experiment; within this experimental environment, the work was coordinated with several other projects; and there was a common aspect of "bootstrapping" involved in their coordinated approach.

1a We are experimenting with computer-aided working techniques as a way of exploring their potential value. Thus, our main product is a report of experiences with the aids we have developed and an assessment of their potentials.

1b This project is coordinated with others, each of which is developing aids for some aspect of our working system, meanwhile using and evaluating its own developments together with those of the other projects.

1c This use and evaluation takes place by applying the developed tools to our everyday work. Thus, the products of our work are used by us to improve our ability to do our work (i.e., we are "bootstrapping").

1c1 This report is an example of both coordination and bootstrapping.

1c2 The report was composed and modified by means of computer aids and produced directly on the mat from computer output.

1c3 The linked-statement form (the "outline" appearance), which is one aspect of our development, is integral with our way of working; we do all of our writing this way.

2 For this project, two particular applications of these techniques were taken up: task monitoring and computer-program design.

2a The task-monitoring activity was aimed at providing a supervisor with information about task description and status that would enable him to assess the state of a developing system.

2a1 In this early activity, computer aids did not enter into the collection of this information--filled-out forms and clerical procedures were used.

2a2 The computer aids were to be involved in the analysis of this information, mostly to be done by Information-Management researchers or by the programming supervisors.

2a3 More complete descriptions of the tasks was needed, which led to the development of techniques for programming-task description that turned out to be very promising in their own right for providing comprehensive design records.

2a4 Consequently, the task-monitoring activity became overshadowed by its offspring--by the burgeoning possibilities that emerged in connection with the program-design activity--and is likely to remain dormant until the more promising possibilities of the program-design activity have been developed.

2a5 When we turn our attention back to the task-monitoring problem, the kinds of structuring and processing of design records that are developing in the present design activity should provide an almost ideal data base and techniques with which to derive task description and status information.

2b The program-design activity is aimed toward developing the forms for the design records and the processes for manipulating them. These are to provide a coordinated means for recording all relevant design information, and an associated means for effective computer-aided modification of these records.

2b1 We have developed two types of computer aids for manipulating these design records: an on-line system that uses a cathode-ray-tube display for instantaneous study and modification of records, and an off-line system that provides hard-copy printer output of a modified record after a normal job shop turn-around delay.

2b1a Special conventions for naming, linking, and tagging accommodate the particular aspects and relationships involved in a program-design record.

2b1b For example, a list of statements may represent a complete flow diagram of a process; each subprocess is represented by a statement. Branching and subroutine calls are handled by special types of inter-statement links.

2b1c Use of this form is independent of the programming language used; any such language may be embedded within this form with equal advantage.

2b1d We find that within this one consistently-structured design record we are able to accommodate any of the information that is commonly found on program listings, flowcharts, data-format tables, and written specifications and constraints.

2b1d1 There can be a particular place in the record for every particular kind of relevant information.

2b1d2 The structure is arbitrarily expandable, serving well the disorderly, cut-and-try process of design.

2b1e The form is particularly amenable to computer manipulation; it also provides natural concepts and operations for a human to use in designating such manipulation.

2b2 The processes for working on this integrated form allow the designer to add or modify with such speed and flexibility that such a record really could keep up with the cut-and-try design processes, always representing the current state of the design.

2b2a The on-line system is fast and flexible enough to represent a promising beginning of effective computer-aided programming design through all the stages, from initial specification to final debugging. The unified design record would grow and evolve to become the complete final documentation at the end of the process. This approach can integrate with any of the emerging developments in on-line compiling and debugging.

2b2b The off-line system offers many of the same advantages. In addition, it can be used on any conventional (job-shop type) computer system. The basic techniques of form and manipulation for program-design records are thus available to almost any programmer.

2b3 Provocative possibilities for on-line aids in debugging emerge in connection with this form of design record:

2b3a Quickly and comprehensively scanning and studying the record--e.g., scanning at any desired level of detail, automatically locating special points of interest by context, easily following cross-reference links.

2b3b Easily designating trial execution of process blocks of any level, with flexible, comprehensive features for tracing and trapping and for portraying the results.

2b3c Keeping track of hypotheses, and of evidence needed and evidence obtained.

2b3d Deducing the source of a bug from the gathered clues.

2b3e Quickly looking up relevant reference information--such as system conventions, equipment characteristics, etc.

3 From our experience to date, we conclude that these design-record techniques offer promising possibilities in the following ways:

3a The individual programmer's productivity can be increased if his way of working can usefully incorporate an efficient record-keeping system, especially if these are used in conjunction with computer aids for design and debugging.

3b The productivity of a cooperating group of programmers may be increased if each makes good use of the unified record-keeping system. The working exchanges of information among them and with their supervisor can achieve both the uniformity provided by standardization, and the speed and flexibility provided by computer aids applicable comprehensively over the gamut of relevant recorded design information.

3b1 Such a group inevitably changes its task specification during the design process. The new techniques promise to increase the speed and flexibility with which such changes are accommodated.

3c This working methodology offers a form of "self-documenting system development."

3c1 The unified design record, embodying all the relevant specifications, considerations, etc., will evolve through all the stages of the design process, becoming the complete final documentation of the system.

3d In subsequently changing a system that has been designed and documented in this way, these same techniques allow new design possibilities to be evaluated or implemented quickly and completely--with "self documentation" obtaining for the system modification as well.

4 Our work to date brings us to the following conclusions about our general approach:

4a As an exploratory tactic, bootstrapping is simultaneously provocative, frustrating, and well worthwhile.

4a1 Depending upon our newly-developed techniques in our own work injects a down-to-earth realism into the needs and possibilities with which we concern ourselves.

4a2 While the total form of the new working method is being developed, the many imperfections and inconsistencies are a continual source of frustration, even though they provide the necessary realism, orientation, and stimulation.

4b An important hypothesis upon which the experiment is based is that the changes in working methodology and language (the form of one's working record), required for effectively harnessing closely-coupled computer services, would prove at least as important and worthy of design attention as would the development of those computer processes themselves.

4c The linked-statement form is only a primitive first step in structuring our working records. But its impact upon our ways of thinking and working, upon the computer processes we have developed and the wealth of future possibilities that these stimulate, leads us to feel that this "methodology and language" hypothesis has been verified.

4d There are promising possibilities for future exploration in connection with program-design records, external-reference documentation, and user reference manuals. We hope to pursue these applications in future work within the program.

1 INTRODUCTION

1a The purpose of the techniques described below is to provide complete and consistent means for representing all of the important facts, considerations, relationships, etc., that could usefully be entered into the working record of a program design. The rules are not intended to force the user into rigid, formalized ways of recording his work; we introduced conventions and formalisms only where we felt that there was a definite advantage to the user.

1b The discussion uses the following definitions and terminology:

1b1 The entire set from Ref(SRI1) is assumed.

1b2 Let "PRC ST1" ("process of ST1") represent the actual process represented and described by ST1.

2 BASIC RULES

2a All description is written in structured-statement form.

2b A design description of a computer program contains several distinct types of statements:

2b1 Describing an initial specification, requirement, or constraint.

2b2 Describing the purpose and usage of the finished program, for instance to someone who wants to use the program.

2b3 Describing a convention, rule, or definition to be used within the design document to facilitate description.

2b4 Describing the data structure.

2b5 Representing and describing an actual program process:

2b5a An actual object-code statement for the computer (rare).

2b5b A source-code statement, for a translator program.

2b5c A higher-level statement, in whose substructure all the lowest-level statements are of either of the above types.

2b6 Describing special tricks or tactics in design.

2b7 Describing some aspect of a particular processing state.

2c These types of statements can be distinguished in several ways:

2c1 By the text content of the statement.

2c2 By the nature of the name given the statement.

2c3 By a special tag in the statement.

2c4 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.

2d We deal below with only the data- and process-description types, which represent the greatest need and possibility for improving documentation of programs.

2e Special conventions for process description are as follows:

2e1 The standard conventions from Ref(SRI1) are assumed.

2e2 Tags for process structures--if the given tag appears in ST1, it has the associated significance:

2e2a *p (for process): ST1 represents and describes a process.

2e2b *c (for comment): used two ways:

2e2b1 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.

2e2b2 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.

2e2c *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.

2e2d *sr (for subroutine): ST1 represents a closed subroutine (and must therefore be named).

2e2e *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.

2e2f *i (for incomplete): The sublist SBL-ST1 is incomplete-- i.e., it does not describe PRC ST1 completely.

2e2g *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.)

2e3 The normal control sequence (i.e., process flow when not directed by a TO or CALL link) is as follows:

2e3a Process control normally passes from one statement, ST1, to its list successor, SCC ST1.

2e3b Control bypasses any non-process (e.g., *c-tagged) statement.

2e3c Control may not pass (by any means) to a statement having a *d tag.

2e3d Control may never pass to an *sr-tagged statement by any other means than a CALL link.

2e4 Branching operations are as follows:

2e4a 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.

2e4b If no condition is specified in the preceding text, transfer is unconditional.

2e4c If the specified conditions are not met, the link is ignored and control passes on through the rest of the statement.

2e5 Subroutine calls are treated as follows:

2e5a 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.

2e5b If no conditions are specified, the jump is unconditional.

2e5c 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.

2e6 Sublists of process statements are treated as follows:

2e6a 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.

2e6b The first process statement of SBL ST1 to which control will pass is:

2e6b1 The first process statement on the list, if ST1 has no name.

2e6b2 (X) The process statement bearing the same name as does ST1, if ST1 has a name.

2e6b3 *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.

2e6c Any nonprocess statement in SBL ST1 must be explicitly tagged; process control will then bypass it.

2e6d 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.

2e6e Also, designation in SBL ST1 of control transfer from ST1 to SCS ST1 may be accomplished by means of a TO(NAM SCS ST1) link in any (or several) of the process statements of SBL ST2.

2e6f In SBL ST1, designation of control transfer to statements other than SCS ST1 must be made with TO(NM1) links.

2e7 Multiple instances of identical TO(NM2) links may represent a given program-control branching path:

2e7a These must appear at each successive level below the highest-level instance, to represent the same branching operation in ever-more detailed descriptive context.

2e7b 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.

2e8 Multiple names, and link following, adhere to these conventions:

2e8a Under certain conditions, a number of specially related statements may have the same name.

2e8a1 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.). See(X) in the discussion of process sublists above.

2e8b Statements bearing a common name represent the same process point, as found at different levels of description.

2e8c 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.

2e8d 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:

2e8e Assume that ST1 contains a link to NM1; 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).

2e8f The single general rule: Choose the first of these statements encountered in following the bridge chain from ST1 to ST2.

2e8g If this is a "reentrant link" (i.e., a branch from within a process back to the beginning of the process, or a recursive self-calling from within a closed subroutine), the statement thus chosen will be the bridge node between ST1 and ST2.

2e8h If it is not a reentrant link, then the chosen STM NM1 will be ST4, the highest-level of the chain of NM1-named statements.

2e8i 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 often will be the branch node).

2e8j If ST1 contains a TO(NAM ST2) link, the following rules affect the allowable value of LCN ST2:

2e8j1 DPT LCN ST2 = D2 must be equal to or less than DPT LCN ST1.

2e8j2 FLI LCN ST2 = FLI LCN ST1 for i from 1 to D2-1. For a reentrant branch, equality also will exist when i=D2.

2e8j3 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.

2e8j4 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.

2e9 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.

2f Discussion of process structures:

2f1 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.

2f2 Concise and consistent form are important in synthesizing, composing, modifying, and studying the program description.

2f2a This applies to form at all structural levels:

2f2a1 A several-character term within a statement, its significance and coding.

2f2a2 The layout and terminology of statements representing often-occurring types of processes or descriptions.

2f2a3 The roles, role-marking, and ordering of statements in lists having common types of purpose.

2f2a4 The roles, role-marking, and structuring of statements and lists in structures having common types of purpose.

2f2a5 The types of links used, and the codes that designate these types.

2f2a6 The types of tags used, their encoding, and their placement within statements.

2g Suggestions:

2g1 Tag all nonprocess statements with *c (for comment) initially. We can supply other tags later to differentiate between significant categories of such statements.

2g2 Locate subroutine descriptions wherever it seems most appropriate.

2g2a Subroutines can be categorized and grouped, with several levels above the *sr-tagged statements, to possible advantage.

2g2b This should not be taken as a rule for all subroutines--e.g., a subroutine used only within one process might better be described under an *sr statement within the list.

2g3 Parameter-state designation, showing parameter PR1 to have value VL1 at a given point in the process, may be done by writing PR1:VL1.

2g3a Use no spacing on either side of the colon.

2g3b Either punctuation or spacing must appear at the end of the character string designating VL1.

2g3c The designation of VL1 may be abbreviated or not according to preference, but using one unbroken character string may avoid ambiguities of statement content.

2g3d Reserve "a" to mean "contents of accumulator," when used as PR1.

2g3e Examples: index:3, Flag:neg, a:nonzero, etc., where the first of each pair is an already-defined parameter.

1 In this part we show that computer programs, commonly represented in flowchart form, can be equally well represented in linked-statement structure, using the basic rules presented in Part A above.

1a We demonstrate this by presenting graphic flowchart and linked-statement structure representations of our on-line system.

1b We start with an overall view of the on-line system; we subsequently examine segments of this system.

2 Overall On-Line System

2a The overview of the on-line system is represented in graphic form in Figure 1. The conventions used in this and succeeding flowcharts are essentially those presented in Ref(ACM1).

2a1 △ or ▽ represents a jump in the logic to a named location. The direction of the arrow indicates where this name may be found on the flow charts.

2a2 ⬭ is the terminal symbol for subroutine entrances and exits.

2b The overview of the on-line system is represented in linked-statement form in Figure 2.

2c The statement numbers and names from Figure 2 are repeated outside their corresponding flowchart symbols in Figure 1.

3 The Main Executive routine is shown in graphic and linked-statement forms in Figures 3 and 4, respectively.

4 The Display Frame Image subroutine, called from within the Main Executive routine, is shown in graphic and linked-statement forms in Figures 5 and 6, respectively.

5 The routine that displays the frame image one line at a time, which is part of the Display Frame Image subroutine, is shown in graphic and linked-statement forms in Figures 7 and 8, respectively.

6 The routine that samples the external devices and formats any inputs, which is part of the Display Frame Image subroutine, is shown in graphic and linked-statement forms in

Figures 9 and 10, respectively.

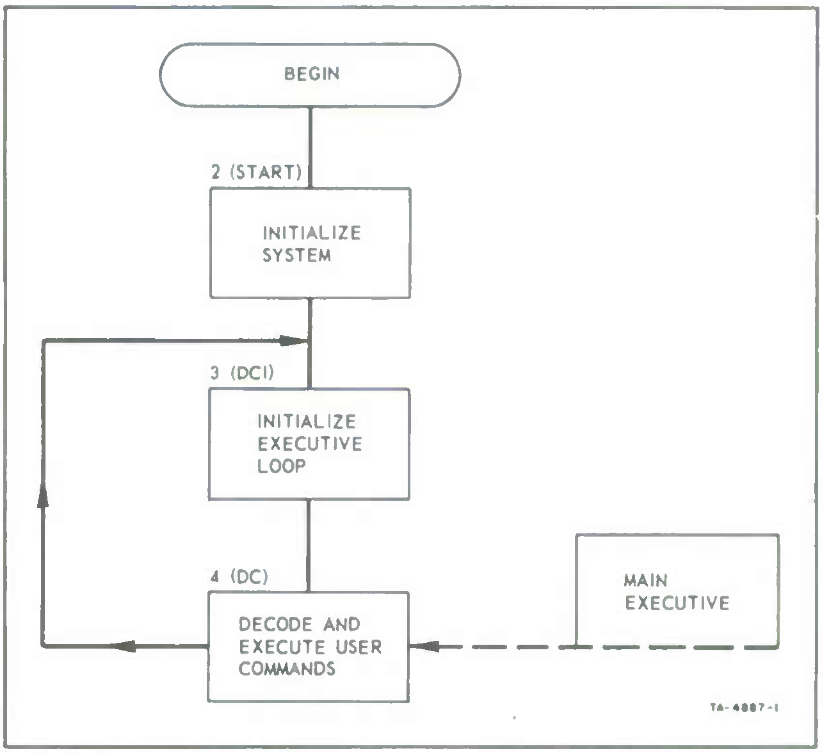


FIG. 1 GRAPHICAL REPRESENTATION: OVERALL ON-LINE SYSTEM

```
0  *p  Online System

1  (ABBREVIATIONS)  *c  Abbreviations used in this writeup:

        1a  "char" means "character code"

        1b  "FWA" means "first-word address"

        1c  "LWA" means "last-word address"

2  (START)  Initialize system.

3  (DCI)  Initialize executive loop.

4  (DC)  (Main executive loop)  Decode and execute user commands.
To(DCI).
```

FIG. 2 LINKED-STATEMENT REPRESENTATION: OVERALL ON-LINE SYSTEM

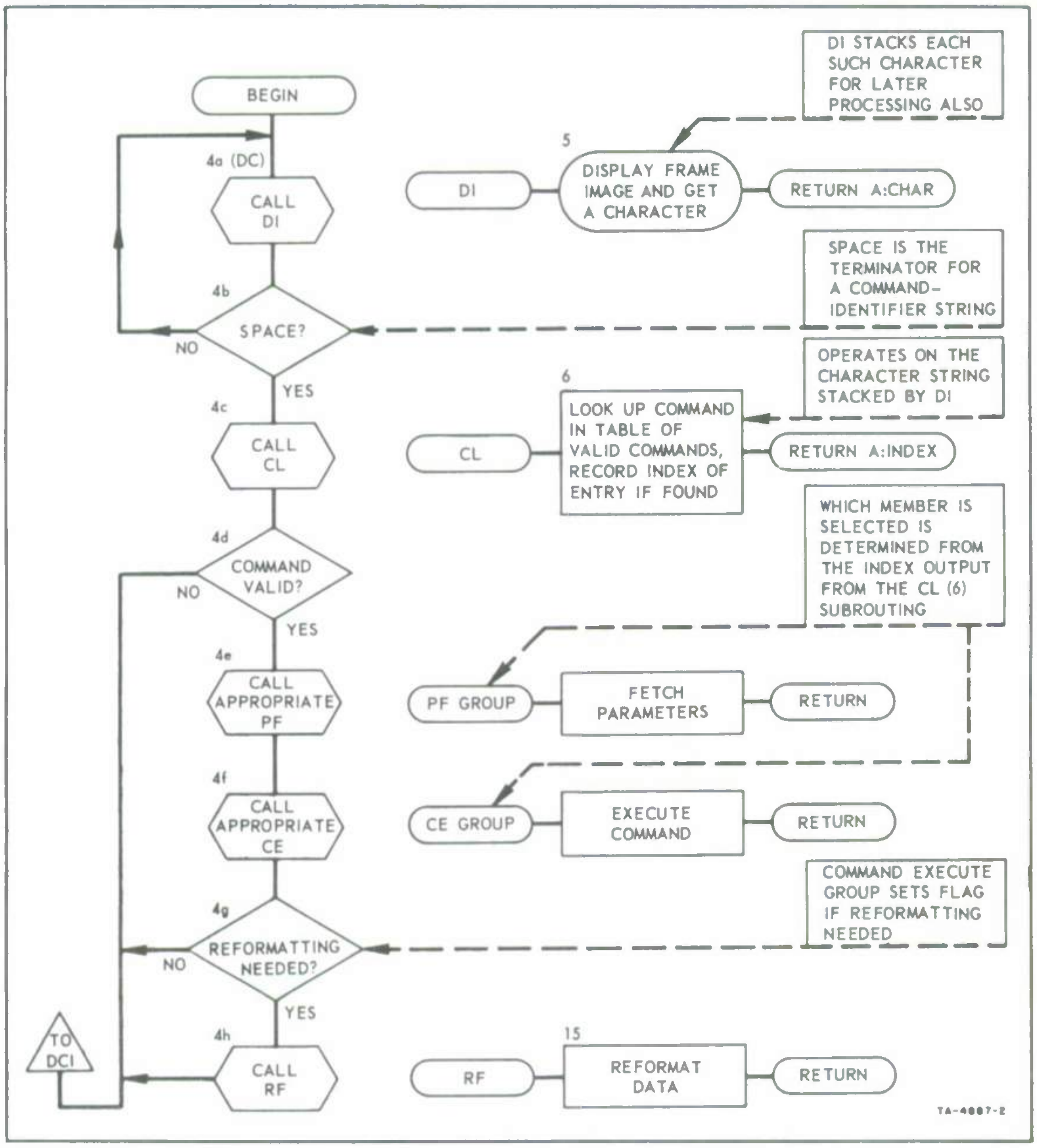

BEGIN
4a (DC)
CALL DI
4b
SPACE?
NO
YES
4c
CALL CL
4d
COMMAND VALID?
NO
YES
4e
CALL APPROPRIATE PF
4f
CALL APPROPRIATE CE
4g
REFORMATTING NEEDED?
NO
YES
4h
CALL RF
TO DCI
DI STACKS EACH SUCH CHARACTER FOR LATER PROCESSING ALSO
5
DI
DISPLAY FRAME IMAGE AND GET A CHARACTER
RETURN A:CHAR
SPACE IS THE TERMINATOR FOR A COMMAND-IDENTIFIER STRING
OPERATES ON THE CHARACTER STRING STACKED BY DI
6
CL
LOOK UP COMMAND IN TABLE OF VALID COMMANDS, RECORD INDEX OF ENTRY IF FOUND
RETURN A:INDEX
WHICH MEMBER IS SELECTED IS DETERMINED FROM THE INDEX OUTPUT FROM THE CL (6) SUBROUTING
PF GROUP
FETCH PARAMETERS
RETURN
CE GROUP
EXECUTE COMMAND
RETURN
COMMAND EXECUTE GROUP SETS FLAG IF REFORMATTING NEEDED
15
RF
REFORMAT DATA
RETURN
TA-4007-2

FIG. 3 GRAPHICAL REPRESENTATION: MAIN EXECUTIVE ROUTINE

```
4  (DC)  (Main executive)  Decode and execute user commands.  To(DCI).

        4a  (DC)  Call(DI), to display the frame image and get a
        character from the external devices.  *c  DI stacks each
        such character for later processing, and also returns
        with it in the accumulator.

        4b  If the character is not a space, to(DC).  *c  Space
        is the terminator for a command-identifier string.

        4c  Call(CL), to identify command and obtain parameters.
        *c  Operates on the character string stacked by DI.

        4d  If not a valid command, to(DCI).

        4e  Call the appropriate subroutine (from "parameter fetch"
        group) to obtain the command's parameters.

        4f  Call the appropriate subroutine (from "command execute"
        group) to execute the command.  *c  The "command execute"
        subroutine sets a flag if reformatting is needed.

        4g  If data does not need reformatting, to(DCI).

        4h  Call(RF), to reformat the data.  To(DCI).

5  (DI)  *sr  Display frame image.  Periodically sample the external
devices, and format any inputs.  Exit a:char when a character is found.

6  (CL)  *sr  Look up command in table of valid commands.  Record
the index of the entry if found.  Set a:o if not found.
.
.
.
15  (RF)  *sr  Reformat data.
```

FIG. 4 LINKED-STATEMENT REPRESENTATION: MAIN EXECUTIVE ROUTINE

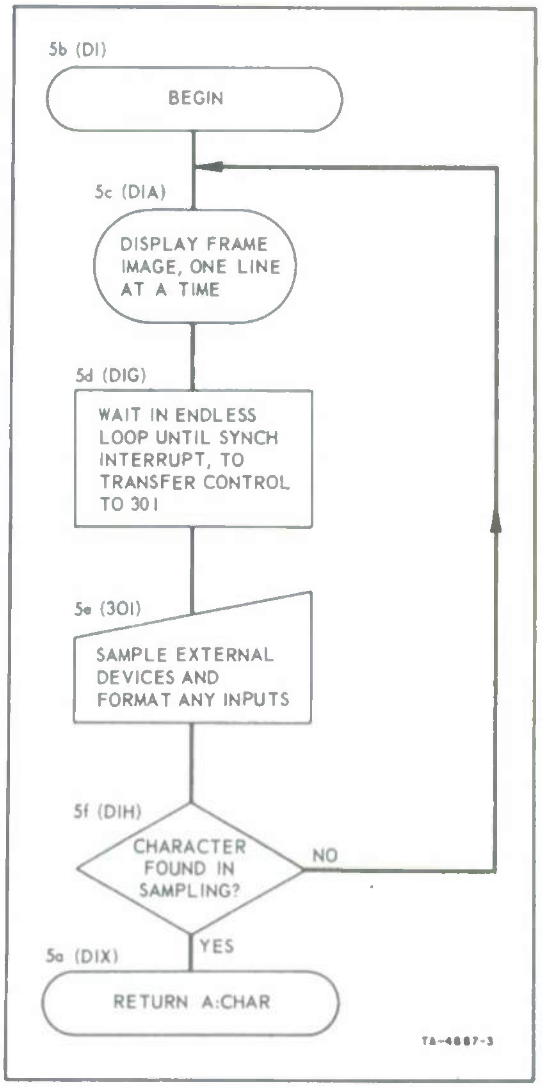

5b (DI)
BEGIN
5c (DIA)
DISPLAY FRAME IMAGE, ONE LINE AT A TIME
5d (DIG)
WAIT IN ENDLESS LOOP UNTIL SYNCH INTERRUPT, TO TRANSFER CONTROL TO 301
5e (301)
SAMPLE EXTERNAL DEVICES AND FORMAT ANY INPUTS
5f (DIH)
CHARACTER FOUND IN SAMPLING?
NO
YES
5a (DIX)
RETURN A:CHAR
TA-4887-3

FIG. 5 GRAPHICAL REPRESENTATION: DISPLAY FRAME IMAGE SUBROUTINE

```
5  (DI)  *sr  Display frame image.  Periodically sample the external
devices, and format any inputs.  EXIT a:char when a character is found.

         5a  (DIX)  Exit *o
DIX      JFI     1

         5b  (DI)  Entry *o
DI               0

         5c  (DIA)  Display frame image, one line at a time.

         5d  (DIG)  Wait in endless loop until synch interrupt occurs,
         to transfer control to(30I).  *o
DIG      CIL
         LDN     0
         ZJR     DIG
         CON     31
         JFI     1
                 30I
         PRG

         5e  (30I)  Sample external devices, and format any inputs.

         5f  (DIH)  If a character was found in sampling, EXIT a:char.
         Otherwise, to(DIA).  *o
DIH      ZJR     DIA
         NZR     DIX
```

FIG. 6 LINKED-STATEMENT REPRESENTATION: DISPLAY FRAME IMAGE SUBROUTINE

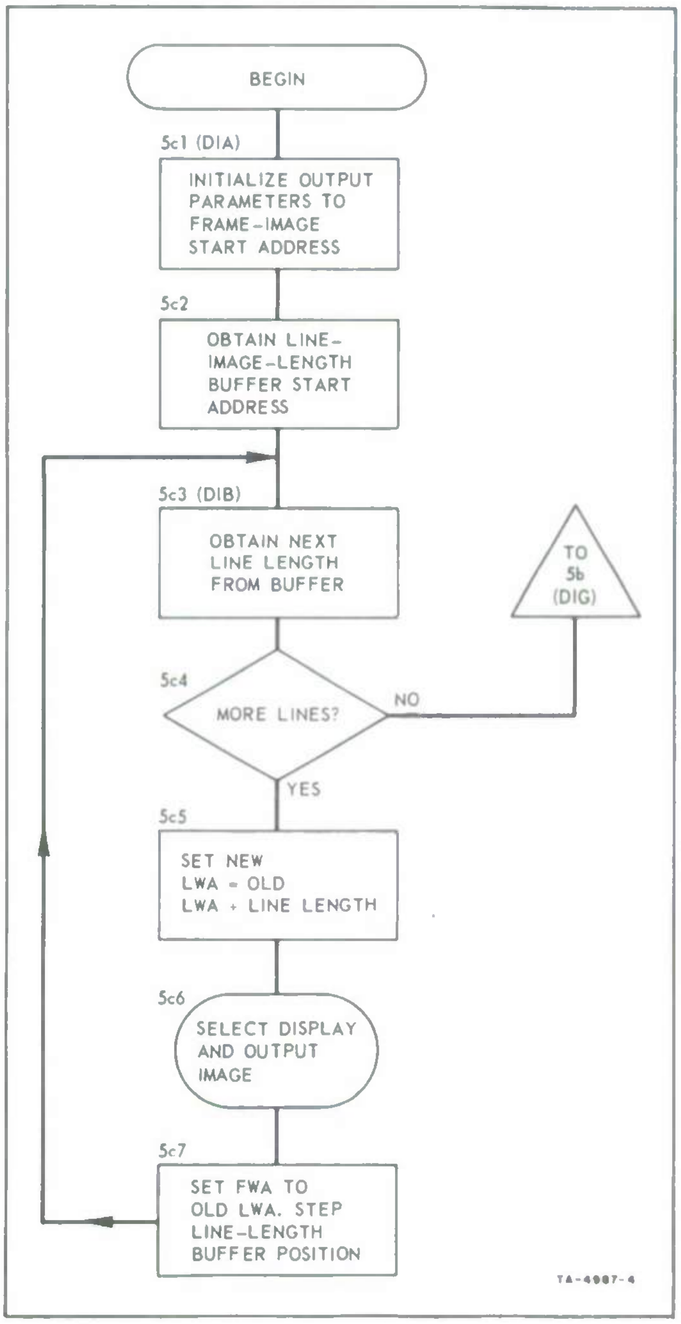

BEGIN
5c1 (DIA)
INITIALIZE OUTPUT PARAMETERS TO FRAME-IMAGE START ADDRESS
5c2
OBTAIN LINE-IMAGE-LENGTH BUFFER START ADDRESS
5c3 (DIB)
OBTAIN NEXT LINE LENGTH FROM BUFFER
TO 5b (DIG)
5c4
MORE LINES?
NO
YES
5c5
SET NEW LWA = OLD LWA + LINE LENGTH
5c6
SELECT DISPLAY AND OUTPUT IMAGE
5c7
SET FWA TO OLD LWA. STEP LINE-LENGTH BUFFER POSITION
TA-4987-4

FIG. 7 GRAPHICAL REPRESENTATION: DISPLAY FRAME IMAGE ONE LINE AT A TIME
(Part of Display Frame Image Subroutine)

```
          5c  (DIA)  Display frame image, one line at a time.

                   5c1  (DIA)  Initialize output parameters to
                   frame-image start address.  *o
DIA       LDD      CDB3                DISPLAY IMAGE START ADDRESS
          STF      DIE                 FWA
          STF      DIF                 LWA PLUS 1

                   5c2  Obtain line-image-length buffer start address.  *o
          LDD      CDB2                LINE IMAGE LENGTH BUFFER START
          STF      DIC

                   5c3  (DIB)  Obtain next line length from buffer.  *o
DIB       LDI
DIC                0

                   5c4  If no more lines (0 line length), to(DIG).  *o
          ZJF      DIG

                   5c5  Set new LWA to old LWA plus line length.  *o
          RAF      DIE

                   5c6  Select display, and output image.  *o
DID       EXC      EXCSWR
          OUT      DIF
DIE                0                   LWA OF IMAGE

                   5c7  Set FWA to current LWA.  Step line-length buffer
                   position. To(DIB), to display next line.  *o
          STF      DIF
          AOD      DIC
          NZB      DIB
          ZJB      DIB
DIF                                    FWA
```

FIG. 8 LINKED-STATEMENT REPRESENTATION DISPLAY FRAME IMAGE ONE LINE AT A TIME (Part of Display Frame Image Subroutine)

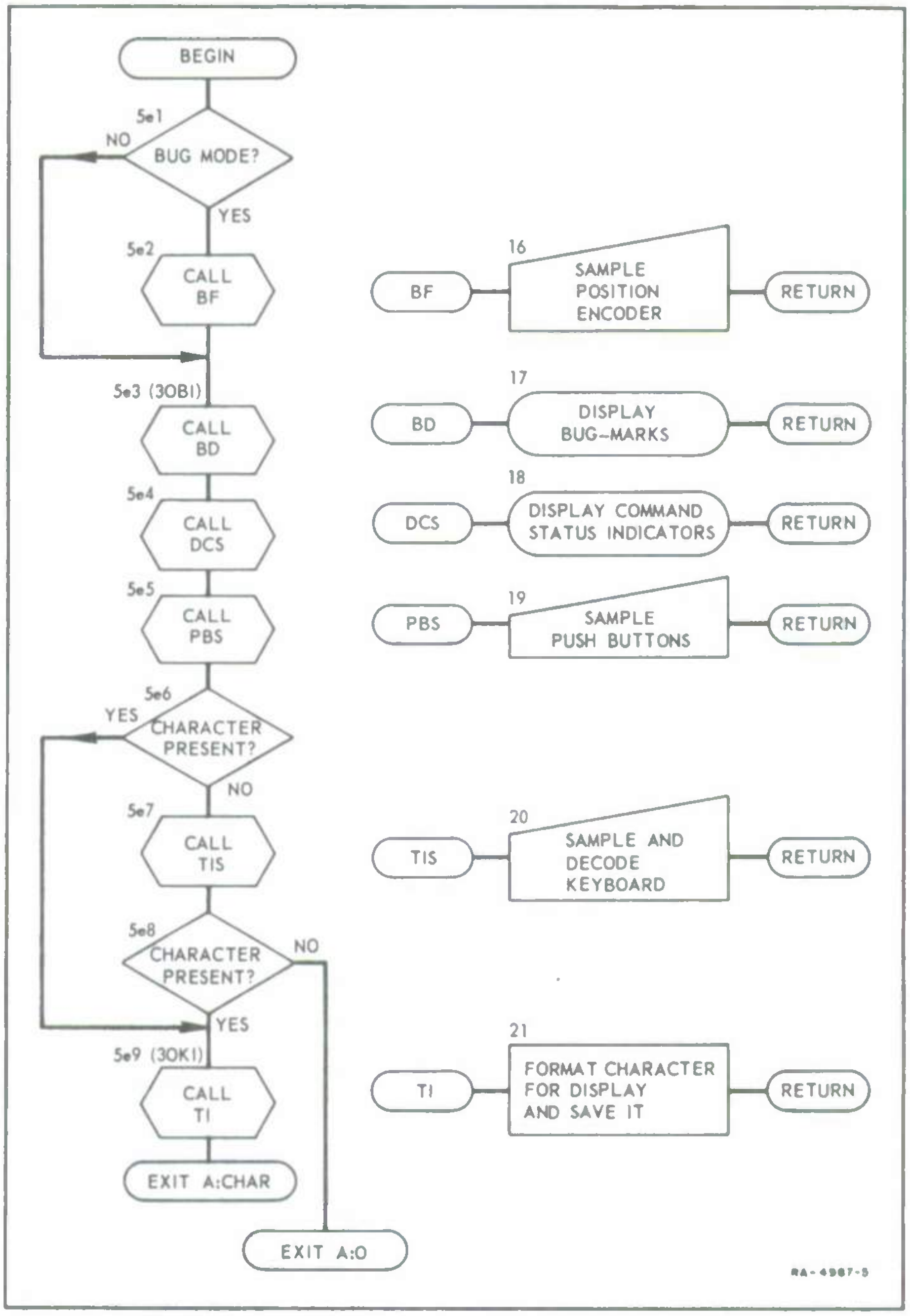


FIG. 9 GRAPHICAL REPRESENTATION: SAMPLE EXTERNAL DEVICES AND FORMAT ANY INPUTS (Part of Display Frame Image Subroutine)

```
               5e  (30I)  Sample external devices, and format any inputs.

                      5e1  (30I)  If system is not in "bug" mode,
                      to(30BI).  *o
30I     LDD           ATDMOD
        PJR           30BI

                      5e2  Call(BF), to sample the position-encoder and
                      convert input (bug-mark position) to internal
                      coordinates.  *o
        JPR           BF

                      5e3  (30BI)  Call(BD), to display current bug marks.  *o
30BI    JPR           BD

                      5e4  Call(DCS), to display current command status
                      indicators.  *o
        JPR           DCS

                      5e5  Call(PBS), to sample external pushbuttons and
                      encode any input present.  *o
        JPR           PBS

                      5e6  If input is present, to(30KI).  *o
        NZR           30KI

                      5e7  Call(TIS), to sample keyboard and encode any
                      input present.  *o
        JPR           TIS

                      5e8  If no input present, EXIT a:o  *o
        NZR           30KI
        JFI           1
                      DIH

                      5e9  (30KI)  Call(TI), to format character for
                      display and save it. EXIT a:char.  *o
30KI    JPR           TI
        JFI           1
                      DIH
.
.
.
16  (BF) *sr  Sample the position-encoder, convert input (bug-mark
position) to internal coordinates, and update the current bug-mark data.

17  (BD) *sr  Display current bug-marks.
.
.
.
21  (TI) *sr  Format character for display, and save it.
```

FIG. 10 LINKED-STATEMENT REPRESENTATION: SAMPLE EXTERNAL DEVICES AND FORMAT ANY INPUTS (Part of Display Frame Image Subroutine)

1 SCOPE

1a Both our off-line and our on-line systems may be used to compose and modify the linked-statement structures (see Ref(SRI1) for detailed descriptions of these two systems).

2 SUMMARY OF OFF-LINE SYSTEM USAGE AND FEATURES

2a Typed text, recorded on punched paper tape, is processed off line by a program that recognizes instructions embedded in the text. These direct the modification in structure or content of any of the prior text.

2a1 The typist may introduce such instructions as needed during the input typing.

2a2 Some of these instructions may modify or delete other instructions.

2a3 The conventions for designating the instructions are such that, from the printed copy, one can determine unambiguously what is expected after computer processing.

2a4 After processing a cleaned-up hard copy, a printout is provided, as well as a punched paper tape representation.

2a5 The user may prepare a new input, referencing both the previously processed material (in its final printout state) and the earlier typing of this current input material, to make modifications of either.

2a6 The paper tape from both this current typing and the previous computer output can be fed back through the processor, to obtain a next cycle of updated printout and paper tape records.

2a7 In developing a body of material, cycling of this kind can be done repeatedly.

2b The user has a variety of instructions that he can employ:

2b1 Insertion of a new statement anywhere in the previous structure can be specified merely by giving it the appropriate location number.

2b1a No matter where a statement occurs in the input text, the processor will put it into its proper position as designated by its location number.

2b1b Interpolative designations for the fields of location numbers are permitted.

2b1c For example, giving a statement a location number 2a3.5 would designate that it is to be inserted between Statements 2a3 and 2a4 of the existing structure.

2b2 Simple statements may specify that any prior statement is to be moved to a new insertion point.

2b3 Similarly, one may specify the deletion of any prior statement (including one that represents an instruction).

2b4 The complete substructure of any statement that is deleted or moved will automatically be deleted or moved along with that statement.

2b5 Renumbering is automatically done by the processor, so that the statements, as newly located within the structure, have proper location numbers without interpolations.

2b6 One can designate new input to be appended to any prior statement, and in this new input embed directions for the modification of that statement.

2b6a This uses the Z-code conventions described in Ref(OSR2), allowing arbitrary insertion, deletion, or replacement from freshly typed material or material that has been cycled through the off-line system) may be loaded onto the drum.

3 SUMMARY OF ON-LINE SYSTEM USAGE AND FEATURES

3a The user sits at the CRT console with the on-line program operating and in control of the computer.

3a1 The paper tape record of any material (either freshly typed material, or material that has been cycled through the off-line system) may be loaded onto the drum.

3a2 The data thus stored in the drum can be scanned and manipulated on the CRT display.

3a3 After such manipulation, the contents of the drum may be punched out on paper tape for off-line printout (Flexowriter) and for later input to either the on-line or off-line system.

3a4 The drum full of data may also be transferred to a storage block on magnetic tape.

3a4a An arbitrary number of such blocks may be kept on magnetic tape.

3b There are two types of processes available to the on-line worker.

3b1 Within the structure contained in any given drum load of data, he can do the following:

3b1a Hop to any designated location number or named statement.

3b1b Scan up or down the lists of statements.

3b1c Perform any of the basic operations of inserting, deleting, replacing, moving, or copying on any one of (or string of) the entities: character, word, line, or statement.

3b1d Send any statement ST1 to be inserted in front of any other statement ST2 in the structure, as specified by either the location number or name of ST2.

3b1e Specify a new location number for a given statement, and have the following statements renumbered automatically.

3b2 Within the file of drum-load data blocks on magnetic tape, the user can do the following (the blocks are filed by decimal serial number):

3b2a Go to to any block, by specifying the desired block number.

3b2b Read the block into the drum.

3b2c Rewrite the block with the current drum contents.

3b2d Go to the end of the file and write the current drum contents on the end of the file, as an added last block.

4 RELATIVE MERITS OF OUR CURRENT ON-LINE AND OFF-LINE SYSTEMS

4a Either of the two text-manipulation systems can be used exclusively, but there are special advantages to each in the present states of development.

4b Straightforward modification of an existing structure is more simply designated by the off-line techniques.

4b1 One reason for this is the limitation in scanning in the current on-line system.

4b1a It is harder, when working over a large structure, to keep oneself oriented.

4b2 When scanning some hard copy and recognizing a change that is desired, it is simple to designate the changes right on the spot, for the off-line system to process.

4b3 A straightforward modification as designated by off-line techniques is simple to specify--secretaries, clerks, and machine operators can do the rest of the work.

4b4 In contrast, to make such a modification with the on-line system currently requires signing up for the machine, loading the material, and trying to remember the changes that were to be made.

4c When making extensive modifications with the off-line system, it often becomes very difficult to picture the structure as it has been newly specified so that further additions and changes can be made. By contrast, when working on-line, one may always view the structure in its immediate, up-to-date state.

4d Which system one can use to best advantage generally depends upon the state of one's work.

4d1 Using the services of the off-line system during first rough composition helps get the statement-by-statement formulation generated in clean form.

4d2 Local manipulations within a list and within statements are better done on line--during the development of one's thinking, when many changes are being made.

4d3 If changes are straightforward and a new view of the modified structure is not needed immediately, the off-line system serves best.

4e The availability and turn-around times for these systems establish how "current" one's working records may be.

4e1 At one extreme, constant availability of an on-line system would permit all design work, including the moment-by-moment "scratch-paper trials," to be in the general structured-statement form.

4e2 At the other extreme, a long turn-around time with the off-line system would limit the utilization of computer aids largely to an "after the fact" documentation of detailed design work.

4e3 Even with a one-day turn-around for the off-line system, it seems feasible to keep the major share of our system design records in a structured-statement form, and to keep the records essentially up to date--with a one-day lag in the availability of hard copy.

1 BASIC CONCEPTS

1a The two main components to program-design techniques are the form in which the design is recorded, and the computer-aided processes for operating on that record.

1b The particular form of the record is developed from the basic list, name, link, and tag features of our linked-statement conventions. The record is arbitrarily expandable.

1b1 There is a place for, or a way of tying in, every kind of relevant information--process steps, comments, data, definitions, specifications, etc.

1b2 Any character-string language can be used at any level, including any formal (i.e., machine-translatable) programming language. At higher levels in the structure, above the programming language, free English or any formally-defined language can be used.

1b3 The form can be produced with a standard character set on a printer or CRT display.

1b4 The form itself is adaptable to future needs; the way lists, names, links, and tags are used may be varied for a wide range of structural forms.

1b5 The nature of the form lends itself to manipulation.

1b5a The computer processes may be neatly organized and implemented.

1b5b The processes of the human user in conceiving and designating appropriate manipulation operations are also helped by the form.

1b5c With the stripping, translating, and debugging improvements (discussed in Section V), this basic form will be suitable for a designer to use for the whole cycle of work from initial conceptualization through final debugging.

1b5d The output from on-line processing is compatible with the off-line system.

1c The processes for human-directed manipulation of the form may be either on-line or off-line.

1c1 On-line processes are fast enough so that the user can keep within his unified design record all of the notes and tentative design trials--moving, deleting, and appending so that the record reflects his minute-by-minute progress.

1c2 Off-line manipulation, although less immediately responsive to the needs of the user, has the advantage of being available to many more people than our real-time work stations and manipulating processes. The output of the off-line system is compatible for use with the on-line system.

1c3 A computation center giving one to two runs per day would allow updating processes that could keep much of the design record in "current" state. The on-line system would surpass this most dramatically mainly in the aids it would provide to the minute-by-minute type of work.

2 ADVANTAGES OF PROGRAM-DESIGN TECHNIQUES

2a The individual programmer is given a new design methodology for keeping notes, records, etc., in one uniform structure, and for keeping these constantly in updated "current" condition.

2a1 The programmer can work to depth in any one aspect; when this aspect is under control, he can shift to some other aspect and some other level without fear of losing track of the state of his progress.

2a2 Temporary notes can be entered into the record and deleted from it as needed, without either getting in the way or getting lost.

2a3 A new way of thinking is opened with this new freedom to cut and try at any level or any stage of the design.

2a3a Uniform ways of thinking and working are augmented for every conceptual level in the design problem. In the same way that the use of formal program languages encourages more orderly thinking at that level of the design, the conventions of form and procedure throughout the rest of the design-record structure encourage more general development of orderly thinking.

2b A cooperating group of programmers gain similarly from those techniques.

2b1 Assume that each programmer is utilizing these techniques and thus benefiting in his own work as discussed above.

2b2 Communications between individuals are much improved if the working record of each has the completeness and uniformity offered by these techniques.

2b3 The supervisor of such a group can use a completely compatible record form and set of manipulation processes for the design work at his level.

2b4 Under the supervisor's record form, the individual record structures of each individual (which completely describe his contribution) may be integrated within a single comprehensive, uniform record.

2b5 This integration may be carried on up through an arbitrary number of levels of supervisory control to accommodate very large coordinated programming-system designs.

2c The system, as a whole, gains a new form of documentation.

2c1 A form of "self-documenting" system is realized; the working records of the individuals and groups provide both the in-process documentation for their own use, and a post-development documentation for others to use.

2c1a With appropriate conventions and procedures for maintaining the records during a design process, little or no additional work should be required to produce extremely good post-development documentation.

2c2 Subsequent maintenance or modification of the system by others would be facilitated.

2c2a The record should be complete in every relevant detail.

2c2b The organization and tagging of the record would make it easy to locate necessary information and to gain the necessary comprehension required for troubleshooting, or for evaluating modification possibilities.

2c2c The manipulation processes allow flexible modification for either minute or extensive changes.

3 COMPARISON OF THESE PROGRAM-DESIGN TECHNIQUES WITH FLOW-CHARTING TECHNIQUES

3a A definite advantage to flow charts is the quicker perception they provide of the "topology" of the process flow. This advantage, however, must be weighed against the following advantages of the linked-statement form:

3a1 The linked-statement form is easier to store and manipulate in the computer and to portray on a display or printer.

3a2 The linked-statement form does not provide any recomposition problem as do flow charts when changes must be made.

3a2a If the computer were asked to handle such rearrangements in the flow chart, deriving and implementing the processes for automatic arrangement of a flow chart for easy comprehension would be challenging.

3a2b An easy solution of this, of course, would be to order the boxes of a flow chart in linear fashion with arrows running up and down the row; but this is essentially the linked-statement form, with drawn-in links (a possibility with which we may soon experiment).

3a3 In a linked-statement record, the length of the given statement may be arbitrary; whereas in a flow chart the text within a box must often be overly abbreviated to comply with geometric constraints.

3a4 A linked-statement record gives a more natural inclusion of non-process information--e.g., specifications, usage pointers, data structure, comments, parameter states, and design tricks.

3a5 In particular the many separate pieces of the record will not tend to get misplaced or get in the way. For instance, there are no separate flow charts, separate fragments of trial code, bits of data-structure, symbol-assignment notes, subroutine-identification notes, etc.

1 INTRODUCTION

1a An independent study conducted by our Systems Engineering Laboratory, working closely with Information-Management personnel, examined our program's aims and information needs in an attempt to identify specific payoff areas for computer-aided information management. Among the promising areas identified were:

1a1 Problem statement detailed in document form, including (where appropriate) an explicit coding specification for programming to be done.

1a2 Possibilities for algorithmic flowcharting.

1a3 A complete system-features description, including operating instructions and user guides, maintained in an up-to-date form.

1a4 Ways of increasing the usefulness of our external documentation citation files and references.

1a5 Ways of obtaining and handling information about currently assigned tasks, their progress and problems--"status information."

1b The last of these was selected to serve as vehicle for an intensive and detailed study leading to computer-aided processing of status information. It was felt that this was an acute need of our own program's information system, and should be of interest to a broader community as well.

1b1 We planned to implement a manual system of forms and procedures for status information and study this clsely, seeing where computer aids could most usefully be incorporated, and then implementing them in an on-line system as soon as possible.

1b2 This activity finally issued in two such schemes, largely complementary in their functions, which were conducted jointly over a period of several months. These are described in the following sections.

2 FIRST STATUS-REPORTING SCHEME

2a Rationale

2a1 The passage from a contemplated or planned task, into an assigned task on which work would begin, was marked by issuing a memo known as the "task description." (Task-descriptions were issued at whatever time this particular stage had been reached--they represented a phase-cut in the process.)

2a2 Durng the implementation, "status reports" marking the progress against the defined tasks were issued at regular time intervals. (Status reports represented time-cuts in the process--whatever stage had been reached.)

2a2a Stages of progress could be checked: e.g., design, coding, checkout, and final documentation, (in the case of a programming task).

2a2b A given task might be either "active" or "inactive" during a particular reporting period.

2a2c The reporting included an "estimated time to completion," which could be revised weekly if necessary.

2a2d There was provision for entering extra commentary.

2a3 The completion of (for instance) a programming task to the point where a new system feature had become operational was announced by an "Op" memo ("new feature operational"); like the task description, this was a phase-cut. This memo was issued even before final documentation had been registered (though documentation was considered a part of the assigned task).

2a4 With the "phase-cuts" of 2al and 2a3, plus the "time-cuts" of 2a2, we hoped to get an adequate cross-sectioning of the process, which would reflect its progress and temporal structure.

2b Implementation of the Scheme: Forms and Procedures

2b1 The forms used in status-information recording were memos extracted from our group-documentation files. Headers were specially preprinted; information content was closely specified; and the documents were usually highly formatted.

2b1a The "Task Description" memo told who had assigned the task; which project within the program was being charged; how long the task would probably take; and the major subtasks involved in completing the task. Method of approach and any extra commentary could also be recorded on this form. Thought and planning, as well as write-up, were required in issuing this document; it was not a simple checklist operation.

2b1b The "Status Report" memo, for registering progress against defined and assigned tasks, was issued to the reporter each week in an updated form. Filling out this form usually required only entering a number or letter, or checking a box, in order to record progress to a new phase or subtask or to revise a time estimate. If status information had not been changed from the previous week's report, no action was needed--except to return the form. There was provision for adding any extra commentary.

2b1c The "Op" memo was extremely brief and highly formatted--there was virtually nothing to write in, except initials and date. The header was prepared at the time the task description was entered; at the appropriate time--task completion to an operational stage--this memo was initialed and turned in. Provision was made for adding comments.

2b2 Issuance and distribution of these forms was procedurally controlled:

2b2a Task Descriptions were to be entered before any work was begun on the task; copies went to all program members, and to the master file.

2b2b Updated Status Report forms were distributed and collected weekly. Copies were distributed only to the Information-Management project personnel: the originals were filed (available to any member of the program), and used in preparing the next week's status forms.

2b2c Op Memos were distributed immediately to all program members, as well as being filed with the corresponding task-description memo in the master file.

2c Operation of the Scheme

2c1 This system of status-information reporting was instituted on a weekly basis, and operated for a period of 27 weeks.

2c2 Task descriptions were issued by each member of the program at the time the scheme was initiated. During most of the period of operation, two people participated in the weekly reporting--though not always the same two.

2c3 Most of the reporting concerned programming and system-design tasks, i.e., implementing system software features. This yielded well-defined, naturally delimited tasks. It also restricted the weekly reporting to just a few individuals (we wanted to try these ideas with a very small number of participants at first), and gave us status information in an area where a real need was felt.

2d Results of Trial Operation

2d1 The most serious problem was that the information conveyed by the status reporting proved to be of little value. We attribute this to the fact that it was not possible to formulate a task description realistically in enough detail to make it a useful basis against which to register one's progress. As a problem in managing information, this took two forms:

2d1a First, we needed ways of incorporating more detail into the task descriptions; representing more realistically the subtasks involved, and their complex interrelations; and displaying the relations to tasks which others in the program might be working on concurrently. This was a problem in representing and structuring information usefully.

2d1b Secondly, we needed easier and more flexible ways of changing that task description--as the task definition itself evolved into modified forms, and as progress was made against it. This was a problem in processing information usefully, and one which called for computer help.

2d2 If our current structuring conventions and off-line computer aids had been available at the time we began the status reporting, we could have handled this problem more satisfactorily; for they give us ways of representing complex hierarchically-organized information, tagging and labeling and linking it to bring out its qualitative significance; and using computer processes to operate on this organized information, modifying it and updating it. These are just the capabilities that were needed for a more realistic scheme of reporting status information.

2d3 These off-line aids are just now getting to the trustworthy stage. If we work up a second attempt at status reporting, we have available a far more useful set of tools, well adapted to the kind of information problems we uncovered there. A logical first move would be to try framing complex task descriptions, tagged and linked in ways that bring out the most significant interdependencies; then to use the associated off-line computer processes to operate upon these and carry out the modifications on them--modifications due both to the changing nature of the task definition (for instance, as new constraints come into view), and to the progress marked up against the defined task.

3 SECOND STATUS-REPORTING SCHEME

3a Rationale

3a1 The second status-reporting scheme was conceived as real-time reporting, with a "sign-on" when one sat down to work and a "sign-off" when one completed it, left, or was interrupted. The reporter would state, in his own words, what he planned to do when he started working and note what impediments (if any) were in the way of his completing it when he left.

3a2 The format was designed specifically for on-line use, as detailed below in 3b1; the same format was also used as the basis for a manually operated system , as described in 3b3.

3a3 The goal was to make the whole process of reporting as automatic and natural as possible. Thus the report was flexible in both its content and its timing; it was entered whenever appropriate, with quick feedback of information to supervisory personnel.

3a4 In particular, this scheme was to be a flag-setting scheme, notifying of impediments or potential problems.

3b Implementation of the Scheme: Formats and Procedures

3b1 The automatic status-reporting format was intended for on-line use as follows:

3b1a When the system had been started up and the on-line program had been loaded, the word "OPERATOR" would appear on the screen. The operator would then type in the required literal string (ID information), followed by the delimiter.

required literal string (ID information), followed by the delimiter.

3b1b This delimiter would activate the command, entering the literal string and bringing up the next heading to the screen: "DATE."

3b1c Each entry would bring up the next in this way, until the sign-on part of the reporting had been completed. (See the attached form, Figure 11.)

3b1d When his work on line was completed, the operator would type in a code for "sign-off" and the words: "TIME OFF" would appear on the display. Typing in the time plus the literal string delimiter would then bring up the next item: "I ACCOMPLISHED," and finally, the item: "REMARKS," completing the sign-off information.

3b2 This status data would be routed to the appropriate parties:

3b2a The complete report, including both the sign-on and sign-off information, would be treated as a memo to the project leader, program manager, and/or records clerk, as appropriate.

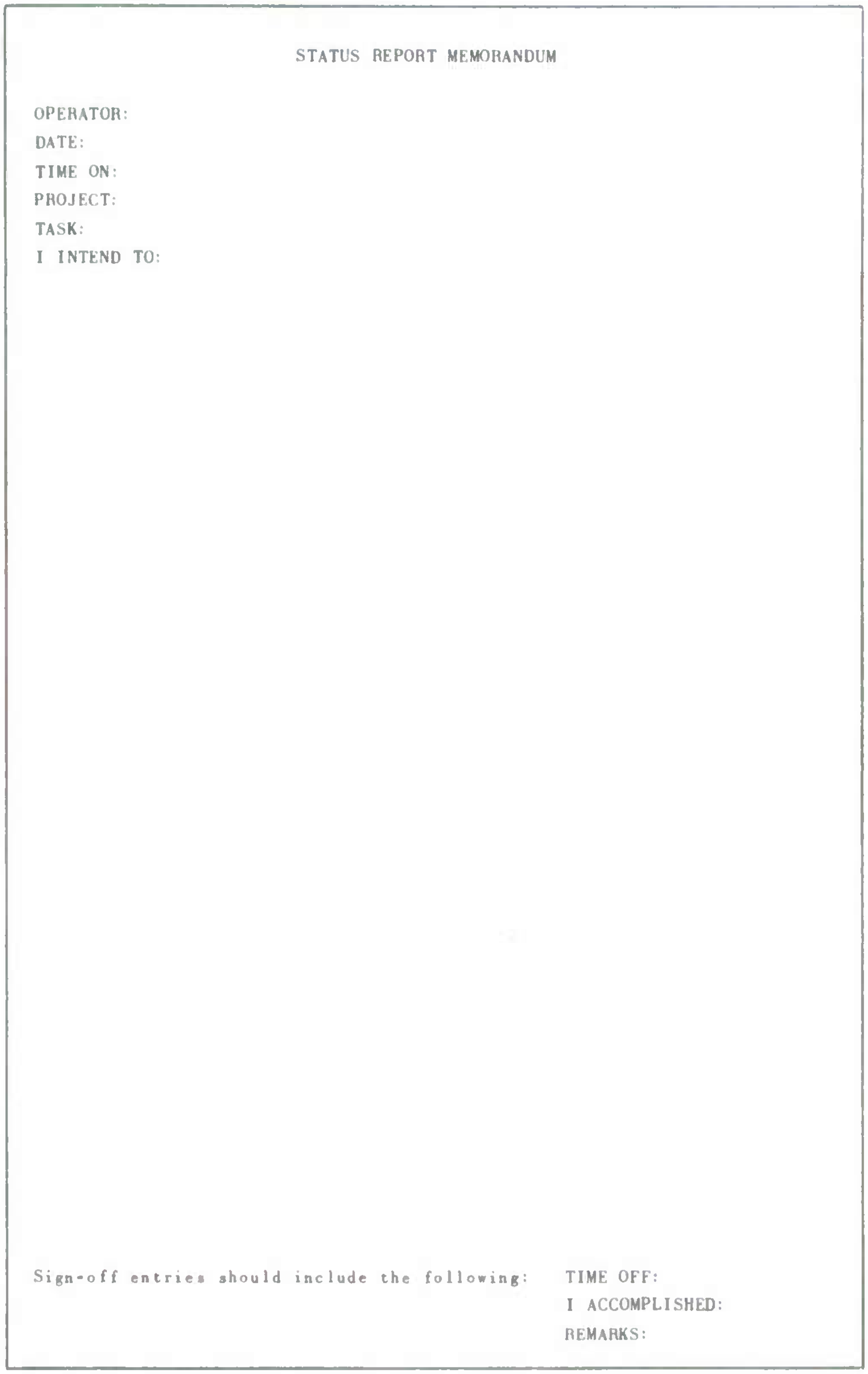
3b2b Copies of the report would be routed to people named or referenced by initials in the "REMARKS" section; or alternatively, a "COPIES TO" entry could be added to the sign-off format, for designating others not normally included in the status report distribution.

3b3 For manual use, blank forms with the appropriate headings were distributed to group members.

3b3a The forms were kept very simple, to minimize the chore of filling them out and maximize the probability that this would be done conscientiously.

3b3a1 Unnecessary entries were omitted completely.

3b3a2 Ample space and leeway were provided for the researcher to include comments in his own words, chosen without system constraint.



STATUS REPORT MEMORANDUM

OPERATOR:
DATE:
TIME ON:
PROJECT:
TASK:
I INTEND TO:

Sign-off entries should include the following: TIME OFF:
I ACCOMPLISHED:
REMARKS:

FIG. 11 SAMPLE OF STATUS REPORT MEMORANDUM FORM

3b3a3 The "I ACCOMPLISHED" or "REMARKS" section of the report could contain reference to any system malfunction or limitation bearing upon completion of the task, as well as to any organizational problems.

3b3b Each person was to fill out the "sign-on" section of the report as he sat down intending to put in a significant amount of time on a given task.

3b3c When he completed his objective, or before then if he was interrupted, he would fill out the "sign-off" section of the report.

3b3d Filling out of the forms was not to be postponed and done ex post facto, for one objective was to simulate the on-line situation and obtain some feedback useful for on-line instrumentation.

3c Operation of the Scheme

3c1 This second scheme of status information reporting was instituted in its manual (or off-line) form, and conducted concurrently with the first scheme for a period of about twelve weeks.

3c2 One member of the program, whose use of the forms was very faithful, found that the timing varied from several sign-on and sign-off periods within the same day to several days on the same reported segment of work.

3c3 The filled-out forms were given to Information-Management project personnel; they were examined, and occasionally brought to the attention of the program manager before being filed. No problems were flagged down by this means, however.

3d Results of Trial Operation

3d1 On-line implementation of this scheme would have made a significant difference in its operation and results; in fact, no fair and realistic evaluation of the scheme can be made without on-line experience, for which it was tailored. This is especially true in its method of entering status information, and of routing it appropriately.

3d2 Because this type of report serves mainly a "flagging" function, it does not rely explicitly upon an adequate "task description," as the first scheme did.

That is, it does not need to face the problem of framing an adequately representative and flexible task description, yet it does presuppose existence of such a description in the background in order to be maximally effective and yield significant information. When the problem of task definition has been more satisfactorily resolved, this second scheme of extracting status information looks very promising, and probably should receive on-line implementation and testing.

4 CONCLUSION

4a Our initial model for obtaining status information about a programming process distinguished three points where written information should be entered into the record: the transition into the active phase, marked by a "Task Description"; the "Status" reports at time intervals during the implementation; and an "after-the-fact" final documentation.

4b One very serious oversimplification in this model is that it overlooks the temporal interleaving of these phases; they do not occur in any simple temporal sequence. Much of the "final documentation" is actually done during the implementation stage; in fact, even the task definition is a useful part of the complete documentation that might finally be retained in the record.

4c Ideally, program documentation and status information about the programming should "fall out" as a natural product of the methodology followed throughout the design-and-programming process. This is one of the issues at stake in the linked-statement design records described in Section III.

4d We would like (ideally) to manage status information and documentation in ways that made them virtually indistinguishable from the substantive work of design and programming--so interwoven in the programmer's methodology that the documentation would be the very framework within which he builds his work. This would not be a documentation scheme which would "reflect" his substantive work; but rather, a documentation which would "display" the current (and evolving) state of his substantive work.

1 INTRODUCTION

1a A new on-line computer system has been ordered. This will have a general effect upon the course and scope of our work.

1a1 Delivery: 1 July 65.

1a2 Central processor: CDC 3100; 8k by 24-bit, 1.75 usec core memory with three I/O channels.

1a3 Peripheral equipment:

1a3a Both paper tape and punched card I/O.

1a3b IBM 1311 disk file (2,000,000 characters).

1a3c Two magnetic tape transports.

1a3d Line printer.

1a3e Straza character generator (arriving 1 April 65)--display scope and on-line input terminals will be the same as at present.

1b We plan to incorporate both a list-processing and a string-processing language facility to work compatibly within common programs.

1c The on-line printer will make a difference in both our on-line and off-line text-manipulation systems.

1c1 Immediate availability of selective printout for the on-line worker offers interesting possibilities for aiding and expanding his working methodology.

1c2 We plan to implement the off-line text-manipulation program, probably as the first program in the new system. The improved accessibility will give us much quicker recycling of our working text.

1d We plan to evolve time sharing in easy stages. The first application would probably be to allow our off-line text-manipulation processes to go on as interleaved background work while the on-line system is being used.

2 PROGRAMMING METHODS

2a General Evolution from the Present State.

2a1 Our intended usage of the techniques described in this report will stimulate a steady stream of new needs and possibilities affecting conventions, style, processes, and working methods.

2a2 We feel that such evolution could actively and profitably be pursued for many months.

2a3 For instance, we have only begun to explore the use of links and tags. At a given point in a process design, specifications, resources, constraints, etc., described in other portions of the record, influence the design; it would probably be valuable to install appropriate types of links, to provide convenient records of the influence of these factors.

2b Stripper-Translator

2b1 Embedding the actual source code within the design record provides homogeneity in both the documentation and the design process.

2b2 It would be wasteful to require a keypunch operator to transcribe source code from this design record, so we plan to develop a "stripper" program to pull out the source language in a form suitable for input to the translator.

2b3 (P1) *p Stripping Processor--A recursive subroutine to strip out source code from the design-record structure is relatively straightforward.

2b3a *c Directed at a process statement, this subroutine will strip out, in appropriate order, all of the source code in the list and its substructure.

2b3a1 We assume OSAS source code in the description below.

2b3a2 ST1 is the head of the list and is given as a parameter to this process.

2b3a3 Let ST2 be the current statement being examined at any time by this processor.

2b3a4 Terminology as from Ref(SRI1): SBH ST2 is the head statement of the sublist of statement ST2; TAL ST1 is the tail statement of the list containing statement ST1; and SCS ST2 is the list-successor of statement ST2.

2b3b (P1) Setup.

2b3b1 ST2=ST1.

2b3c (P11) If ST2 has a *c tag, TO(P12).

2b3d If ST2 has an *o tag, TO(P13).

2b3e CALL(P1) for SBH ST2.

2b3f (P13) Strip OSAS from asterisk to end of statement.

2b3g (P12) If ST2=TAL ST1, exit.

2b3h ST2=SCS ST2, TO(P11).

2c Cross Referencing between Object-Code Listing and Design Record

2c1 Information contained in the design record would often be valuable during debugging.

2c2 The listing and the design record will contain the same reference names.

2c3 The computer can then aid in on-line cross referencing as follows:

2c3a A modification of our on-line system would enable the user to scan and manipulate text in the format of the object-code listing.

2c3b In this format, a machine address would become the equivalent of a location number; a symbolic name or label would become the equivalent of a statement name.

2c3c Scan and hop commands on location and name would thus be available over the object-code listing.

2c3d Cross-record processes could be developed that would allow hopping from one record to a named statement in another record, to provide effective cross referencing.

2c4 More natural cross reference could be obtained by a slightly different arrangement--integrating the translator output back into the design record.

2c4a A special link type (e.g., "*listing") could be used to link from any statement containing an *o tag (or other source-code tag) to the corresponding point in the object-code listing.

2c4b Normal scanning of the record would show the source code embedded in the lowest levels of the design record structure.

2c4c A special scanning mode would not show the source code, but instead would automatically follow the "*listing" links to locate and show, as the lowest level, the translated object code.

2d On-Line Executing and Debugging

2d1 For completeness, these cross-reference aids should be accompanied by the ability for the user to execute and modify operating programs on line.

2d2 A natural operating system for this would allow the user, when viewing the design record, to select a process statement (at any level), stipulate the necessary entry parameters, and have the appropriate section of code executed.

2d2a Special aids would be available to help the user establish the desired entry parameters.

2d2b If execution were normal, the resulting parameter states could be displayed and (if desired) used as the input parameters for the next process step.

2d2c If the process did not execute properly, the user could drop down to the sublist of this process statement and begin executing these statements, one at a time, to isolate the trouble.

2d3 Special processes to aid on-line debugging, such as help in establishing program patches, would be a natural addition to the above aids.

2d4 *c Both of the above feature (statement execution and on-line patching) have previously been developed and used in DDT--the on-line (typewriter) debugging aid for the PDP series of computers.

2d5 On-line stripping, translating, and reintegration of the object code are also important features to plan for.

2e We plan to develop our on-line system further along these general lines.

3 Indexing and retrieval for our external documents.

3a In our original task breakdown, we had divided the information-management problems in our program into three types:

3a1 A personal documentation system (PDOC) in which an individual could get help in managing his own working information.

3a2 A group documentation system (GDOC) for managing the information representing interpersonal and group working records.

3a3 The external documentation system (XDOC) for managing the information that the group collects from the outside world.

3b The activity in the status reporting and program documentation areas all lie in the categories of PDOC and GDOC system work.

3c Over the past five years, we have collected (mostly under AFOSR sponsorship) some 1800 bibliographical items of external reference data relevant to computer-aided problem solving.

3c1 These have all been entered into our system in a standard format.

3c2 They have all been punched on paper tape through the years as the collection grew.

3c3 At present, they are filed only by chronological accession number and by a (manual) author card file.

3d The information structuring conventions and computer-aided manipulation processes already described in this report and Ref(SRI1) would be basically adequate for organizing these items into a structured file.

3d1 The format of the individual entries is completely compatible with the conventions of both our off-line and on-line system, for each to be handled as a separate item.

3d2 The present file is a one-level structure (composed of just one long list).

3d3 A simple computer program could give each item a name (in our special sense of the term); this would be its current accession number.

3d4 An initial categorization structure could be developed.

> 3d4a Such a structure could be one level deep in some areas, many levels deep in other areas.
>
> 3d4b It need be only tentative, for our techniques would allow us later to modify the structure very easily.

3d5 Our manipulation techniques would let us scan the list and send each statement to some designated position in the structure.

> 3d5a For the first few passes, where items may be moved within a very large file (in terms of our current work data capacity), we would likely have to use our system in special, tricky ways.
>
> 3d5b Once the statements became fitted into the structure so that individual category lists were manageable within our working file size, there would be little problem in moving statements among lists.

3d6 Our present on-line system could accommodate a substructure of the overall file containing up to 60 or 70 bibliographical items.

3d7 In our on-line system, any number of such blocks of data, representing substructures of the large structure, can be stored on the magnetic tape. This would provide access at tape-scan speeds, allowing easy study, extraction, or modification of any block.

3d8 The first blocks on mag tape could contain the high-level structuring for the total file; thus they could represent a category index for the blocks that contained the data substructures.

3e On-line searching, updating, and restructuring.

3e1 The on-line techniques for structure scanning can give fast search through the index or other structured blocks.

3e1a Structure scanning allows scanning down any given list (at any desired level), and being able instantly to drop to a level below any designated statement seen on the current list, Ref(SRI1).

3e1b This immediately gives us the basic "tree selection" technique.

3e2 Our conventions for using tags provide a ready-made technique for attaching descriptors to any item or statement at any level in the structure, and the planned general tag-search process would give flexible retrieval on a descriptor search basis.

3e3 Our conventions for naming and linking, and the processes for link hopping on-line (i.e., for automatically hopping to the statement referred to by a link), would provide yet another search technique.

3e3a This provides directly for the "associative trail" techniques prescribed by Bush in his classic "Memex" paper, Ref(Bush1), Ref(OSR1).

3e3b The use of different link-type tags (i.e., the printing characters preceding the open paren of a link term) would allow us to give individual trails separate identity, and to categorize the types of trails.

3e3c Following a trail within the 70-item current working file limit would be simple. One would point the light pen at a link term and strike the hop code. There would then be a wait of no more than a second or two to view the linked-to statement in its resident location within the structure.

4 REFERENCE MANUALS

4a Introduction

4a1 In our program, we have many equipments, programs, and user systems whose features are changing.

4a2 Our usage of equipment, software, and user systems shifts; this means our requirements and dependence upon reference information are continuously shifting also.

4a3 A reasonable area of information-management focus with our program would thus be conventions and procedures for structuring, updating, and referencing special materials.

4a4 There would be two main types of users:

4a4a The "off-line" user, who would refer to hard copy.

4a4b The "on-line" user, who would hop and scan within computer-held reference material.

4a5 The work of organizing, structuring, and modifying the records (for either the off-line or on-line user) can be aided by both our on-line and off-line systems.

4b Organizing and Structuring the Reference Material

4b1 The hierarchical structuring, the cross-reference linking, and the tagging of statements all offer extensive possibilities for organizing, relating, and identifying the various types of information needed in a reference document.

4b2 For initial experimentation with these features, the processes and procedures already described for our on-line and off-line manipulation of structured text are quite adequate.

4b3 Existing reference documents--e.g., programming or equipment manuals:

4b3a One approach to these would be first to transcribe the document directly into machine code, in its original format.

4b3b Once in machine code, either our off-line or on-line system could be effectively used to structure the information, reorganizing it and establishing reference links and tags as desired.

4b4 The features of both systems provide considerable facility for modifying and updating reference records rapidly and efficiently.

4c Hard Copy Reference Records

4c1 Selective-depth printout

4c1a The printout processes that operate upon a structured record will soon include the facilities for printing out only statements above a specified depth.

4c1b A two- or three-level printout would thus provide an effective table of contents (indeed, to make it more like a table of contents, the page number at which each statement's substructure is to be found could automatically be attached to each lower-level statement in the limited-level printout).

4c1c This operation of "limited-level table of contents printout" might well be nested:

4c1c1 When one turned to the referenced location, he would find a limited-level printout of the substructure--i.e., the table of contents of that substructure.

4c1c2 This could be continued for subsequent levels.

4c1d This type of printout should be explored for providing new ways to study or reference a record.

4c2 Automatic Cross-Reference Updating

4c2a Cross-reference links, when printed out, could automatically be supplied with the page number of the linked-to statement.

4c2b This would encourage liberal use of cross references within a record, since their updating (after the record had been modified) would be entirely automatic.

4c3 Tags would be some help (although not nearly as much as for the on-line user).

4d On-Line Reference Facility

4d1 Basic Operations:

4d1a Structured scanning, which allows scanning to limited depth, with immediate selective changing to new depth limits at any point.

4d1b Chain scanning, which lets one scan over the successive statements of specified chains--e.g., the source chain of a given statement, or the chain that follows a given type of linkage through the structure.

4d1c Link hopping and location-number hopping, which allow instantaneous jumps to other portions of the record.

4d1d Tag searching, which provides for hopping to successive occurrences of given tagging configurations; this gives descriptor-search facility.

4d1e Symbol-string searching, which provides for hopping to successive occurrences of a specified symbol string.

4d2 Factors to Explore:

4d2a Organizing and Structuring:

4d2a1 The ways to organize reference data into hierarchical structure.

4d2a2 The different types of links and the ways to use them.

4d2a3 The different types of tags and the ways to use them.

4d2b Reference, Study, and Search.

4d2b1 The methods of thinking and working that most effectively harness the above features of structure and operation.

4d2b2 The processor features that best utilize the clues provided by structure, linkage, and tagging.

4d2b3 The best set of operations (to be initiated by specific commands) to provide an efficient facility for following the procedures and methods.

4e Our approach will be a natural exploration of the several avenues in parallel as we try to use our developing techniques to best advantage in our own work.

1 (ACM1) Rossheim, Robert J, "Report on Proposed American Standard Flowchart Symbols for Information Processing," Communications of the ACM, Vol. 6, No. 10 (October 1963), pp. 599-604.

2 (BUSH1) Bush, Vannevar, "As We May Think," The Atlantic Monthly, Vol. 176, No. 1 (July 1945), pp. 101-108.

3 (ESD1) Bourne, Charles P., "Research on Computer Augmented Information Management," Final Report, Contract AF 19(628)-2914, SRI Project 4506, Stanford Research Institute, Menlo Park, California (November 1963).

4 (OSR1) Engelbart, D. C., "Augmenting Human Intellect: A Conceptual Framework," Contract AF 49(638)-1024, SRI Project 3578, Stanford Research Institute, Menlo Park, California (October 1962), AD 289 565.

5 (OSR2) 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).

6 (SRI1) "User's Guide, Man-Machine Information System," Stanford Research Institute, Menlo Park, California (April 1964).

Appendix A

USER'S GUIDE
MAN-MACHINE INFORMATION SYSTEM
(Revised June 1965)

1 The Man-Machine Information System is aimed at improving the work performance of a programmer by the use of computer aids, many of them real-time. Although a programmer is the main target for the work, many of the processes have a wider applicability.

2 This manual describes the current state of the system, which is in continuous development. The manual is published in two forms--in a looseleaf notebook and in a fixed binding.

2a The looseleaf form is intended for those who will use the system.

2a1 Such users will receive, when appropriate, modifications and additions to this manual to keep their copy updated with the status of the system.

2b The fixed binding version of this manual is for information purposes only. New versions will be issued from time to time as significant additions are incorporated into the system.

2c Requests for further copies of this manual in either version should be made to Mr. W. K. English, Building 314b.

3 The stimulus for the design of the system has been the Institute's research program on "Augmented Human Intellect."

3a The initial conceptual framework for the Augmented Human Intellect Study was supported jointly by the Air Force Office of Scientific Research (AF 49(638)-1024) and Stanford Research Institute over the period from 1961 onwards.

4 In this manual, the system is broken down into several components that have been developed under various contracts to form a coordinated whole.

4a This particular version of the User's Guide is assembled specifically to accompany the March 19, 1965 report of the ESD project.

4b The sections describing the conventions and procedures for program-design documentation are missing in favor of the more complete writeup in the report itself.

5 The contents are arranged in the following categories:

5a The conventions, concepts, and definitions for the linked-statement structure form.

5a1 Essentially all of our text is now composed and manipulated in this form; thus our computer-aided processes are oriented specifically toward manipulating this form of text (although they will also handle other forms).

5b The procedures and processes available on our on-line system for manipulating our working text, and the equipment comprising our on-line facility.

5c The procedures and processes available on our off-line system for manipulating our working text.

6 Currently both the off-line and on-line systems work with paper-tape input and output.

6a The paper-tape output is the result of the operations done upon the paper-tape input text and can be printed on the Flexowriter to obtain corresponding hard copy.

6b The paper-tape output of either the on-line or the off-line system is compatible as input to either system for a next stage of manipulation.

1 These conventions and terminology for linked-statement structuring were developed under the sponsorship of the Advanced Research Projects Agency.

2 Statements.

2a 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."

2b The length of a statement is arbitrary.

2c The composition of a statement is arbitrary, with the following explicit exceptions:

2c1 Special requirements for Location Numbers see(LOCNUMDF), Names see(NAMDEF), Tags see(TAGDEF), and Links see(LINKDEF) are described below.

2d (LOCNUMDF) Location numbers.

2d1 The first word of a statement is its location number; its first character is a digit.

2d2 The location number is composed of a string of digits and alphabetics, with no spacing gaps included.

2d2a A "field" in the location number is a continuous string of alphabetic characters, or a continuous string of numeric characters, broken possibly by a period or a comma.

2d2b The characters in a given field indicate the ordering on a unique list in the structure of statements see(STRUCDEF).

2d3 The location number represents the unique location of its statement within the structure of statements.

2e (NAMDEF) Names.

2e1 A name may be associated with any given statement.

2e2 The name is enclosed in parentheses and is the first printing string after the location number.

2e3 If an open paren is the first printing character after the location number, it is assumed to signal the presence of a name.

2e4 The name may contain no spacing gaps--i.e., there will be no spacing gaps between the parentheses.

2e5 The choice and sequence of printing characters composing a name is arbitrary.

2e6 The length of a name is limited to 16 characters (printing or non-printing). This is an arbitrary and tentative limit.

2f (TAGDEF) Special words called "tags" may be included within a statement; they may serve as descriptors, etc.

2f1 As many tags as desired may be included within a statement.

2f2 They may be located anywhere after the location number and name.

2f3 Each is identified by the sequence SPACINGAP ASTERISK n-PRINTCHARS SPACINGAP.

2f4 There is no restriction on "n," or on the composition of a tag--except that no spacing gaps may be included.

2g (LINKDEF) Special words called "links" may be included within a statement; they serve to establish cross-reference linkages to other statements.

2g1 As many links as desired may be included within a statement.

2g2 They may be located anywhere after the location number and name.

2g3 Each is identified by the sequence SPACINGAP n-PRINTCHARS OPENPAREN m-PRINTCHARS CLOSEPAREN SPACINGAP-OR-PUNCTUATION.

2g4 The parens enclose the name of some statement, see(NAMDEF).

2g5 The PRINTCHARS preceding the OPENPAREN represent the "link type" code string; this string may be of arbitrary length and composition-- except that no spacing gaps may be included.

3 (LISTDEF) Lists of Statements.

3a Any statement STi may have a "list successor," which is another statement.

3b 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."

3c The first statement on such a sequential list of statements is called the "head statement" of the list.

3d The last statement on such a sequential list of statements is called the "tail statement" of the list.

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

3f For each statement in a given list, the last field of the location number indicates the statement's location in that list.

3f1 Interpolative breaks (e.g., 2f1.5) may appear in a field of the location number; in this case the numbers indicate only the relative location.

3f1a A special interpolation convention is needed in order to insert something before the head statement of a list.

3f1b Let a COMMA, when used as an interpolative break in a field, designate that the interpolation is to come before (rather than after) the statement indicated by the field characters up to the interpolative break.

3f1c Example: 3b1,5 (or: 2a,e) would belong in front of, and at the same list level as, 3b1 (or: 2a).

3f2 A list in which the location numbers are in "clear ordinal" state will have no interpolative breaks in the last field; this field will then indicate the true ordinal location in the list.

4 (STRUCDEF) List Structures of Statements.

4a Various structural relations are already implied:

4a1 Sequential association within a list.

4a2 Inter-statement links, see(LINKDEF).

4a2a Any statement may be linked to any other in this manner.

4b Besides this, there is hierarchical structuring.

4b1 Each list of statements may be a sublist of one (and only one) statement.

4b2 That statement is known as the "source statement" of that list.

4b3 The location number of every statement on such a list will differ from that of its source statement only by the addition of one more field.

4b4 Any statement in that list may be the source statement for a sublist of its own, etc., to arbitrary depth.

4b5 The sublist of a statement, and the sublists of the sublist statements, etc., form the "substructure" of the given statement.

5 Terminology Conventions.

5a About the choice of mnemonics: each entity described below has a name that is generally accepted and usually easy to remember. The three-character mnemonic term for designating an entity is derived from this name by means of the following rules:

5a1 The case of any alphabetic character within a mnemonic is not significant.

5a2 For a one-word name, take the first three non-repeated, non-silent consonants.

5a3 If there are not enough consonants, include the first phonetic vowels, ordered with the consonants as they appear in the word.

5a4 For a two-word name, take two characters from the first word, and a third character from the second word, according to the two rules above.

5a5 For a word and a number, take two characters of the word (as above) and append the number--even if the number is several characters.

5a6 For two words and a number, take one character from each word, and append the number--even if the number is several characters.

5a7 If two names would produce the same 3-character mnemonic, use this mnemonic for the name which precedes alphabetically. For the other mnemonic, try rejecting its second character and picking another character, for a new second or third character, according to the selection rules above.

5b Basic Entities.

5b1 Let ST1, ST2, etc., refer to arbitrary statements.

5b1a The integers carry no implications as to the structural relationship between the statements.

5b2 Let LN1, LN2, etc., be used to represent arbitrary location numbers.

5b3 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.

5b4 Let NM1, NM2, etc., refer to arbitrary statement names.

5b5 Let LS1, LS2, etc., represent arbitrary lists of statements.

5c Operations--where an operation on one entity represents another entity.

5c1 General:

5c1a Let LCN ST1, LCN ST2, etc., represent the location numbers of statements ST1, ST2, etc.

5c1b Let STM LN1, STM LN2, etc., represent the statements whose location numbers are LN1, LN2, etc.

5c1c Let STM NM1, STM NM2, etc., represent the statements whose names are NM1, NM2, etc.

5c1d Let NAM ST1, NAM ST2, etc., represent the names of statements ST1, ST2, etc.

5c1d1 Let NAM ST1 be ZERO if ST1 has no name.

5c2 Fields within a location number:

5c2a Let FL1 LN1, FL2 LN1, etc., represent the first, second, etc., fields of location number LN1.

5c2b Let FL(expression) LN1 represent the nth field of LN1, where n is the numeric obtained by evaluating the expression.

5c2c Let FLi LN1, FLj LN1, etc., refer to the ith, jth, etc., fields of LN1.

5c2d Let FLT LN1 represent the last (tail) field of LN1.

5c3 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.

5c3a Let DPT ST1, DPT ST2, etc., represent the depths of ST1, ST2, etc.

5c3b 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.

5c4 To represent a statement having a particular structural relationship to another statement:

5c4a SCS ST1, successor of ST1 (list successor).

5c4b PRD ST1, predecessor of ST1 (list predecessor).

5c4c HED ST1, head of the list containing ST1.

5c4d TAL ST1, tail of the list containing ST1.

5c4e SBH ST1, sublist head of ST1--the head statement of the sublist of ST1.

5c4f SBT ST1, sublist tail of ST1--the tail statement of the sublist of ST1.

5c4g SRC ST1, source of ST1--the source statement of

ST1.

5c5 To represent a list having a particular structural relationship to a statement:

5c5a LSC ST1, list containing ST1--the entire list of statements.

5c5b LSF ST1, list from ST1--the list of statements including ST1, SCS ST1, etc., down to and including TAL ST1.

5c5c 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.)

5c5d LST ST1, list to ST1--the list of statements from HED ST1 through PRD ST1.

5c5e SBL ST1, sublist of ST1--the entire list.

5c5f SRL ST1, source list of ST1--the list containing SRC ST1.

5c6 To represent a statement having a particular relationship to a list:

5c6a HED LS1, head of LS1.

5c6b TAL LS1, tail of LS1.

5c6c SRC LS1, source of LS1.

5c7 Relating a list to a list:

5c7a SRL LS1, source list of LS1--the list containing SRC LS1.

5d Concatenated operations.

5d1 Notation:

5d1a An operator may operate upon an entity that is represented as the product of another operation.

5d1b 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.

5d1c Obviously, the product of the rightmost operation must be an entity upon which the preceding operator can validly operate.

5d1d An integer n, or an expression representing such an integer, appearing between parentheses after an operator, designates n successive applications of that operator.

5d1e Any other printing character or characters appearing between two operations indicates that they are not to be concatenated.

5d1f Some reasons for this notation:

5d1f1 Spacingaps between concatenated terms are desirable so that long chains can be conveniently broken by line spacing without any complications.

5d1f2 Prefix Polish notation offers a good precedent. (So does suffix notation--we arbitrarily selected prefix.)

5d2 Examples:

5d2a LCN TAL SRC ST1 is the location number of the tail statement of the list containing the source statement of ST1.

5d2b SCL ST1 = LSC SRC ST1.

5d2c SBL ST1 = LSF SBH ST1.

5d2d LST ST1 = LSB HED ST1 PRD ST2.

5d2e FL(DPH LCN ST2) LCN ST1 is the field of LCN ST1 at a depth corresponding to the last field of LCN ST2.

5e Special entities and relationships:

5e1 The "source chain" of ST1 is composed of ST1, SRC ST1, SRC(2) ST1,..., SRC(DPT ST1) ST1.

5e2 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.

5e3 ST1 is said to be "structurally above" ST2 if ST1 is a member of the branch chain from ST2.

5e4 ST1 is said to be "structurally below" ST2 if ST2 is a member of the branch chain of ST1.

5e5 ST1 is said to be "branch related" to ST2 if either statement is a member of the other's branch chain.

5e6 ST1 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).

5e7 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.

> 5e7a The branch chains from any two statements in the same structure will always meet to produce such a node.
>
> 5e7b 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.
>
> 5e7c 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.

5e8 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.

1 Various segments of this on-line system have been developed under different sponsorship, according to the pursuits of the respective projects.

1a 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 inputting and outputting; and for executing the higher-level commands that operate on statement structures and tape files.

1b A 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-designation operations; and the basic editing routines.

2 With this system, one can load an arbitrary number of working records (each up to 18,000 characters in length) onto magnetic tape by typing at the on-line keyboard, or by reading in paper tape from any of our paper-tape-punching typewriters or from the output of our off-line system.

2a The system will handle a variety of text forms (including the normal sentence-paragraph form), but a number of its special features are specifically designed for the linked-statement form.

3 With the CRT display as a very mobile "window" to scan a record, and with the computer to maneuver the window and alter the record in instantaneous response to his directions, the user can study and/or modify any such record with great facility.

4 He may access any of his working records, for study and modification; or make an internal copy, for independent storage and alteration as a new record; or extract from a number of such records, merging them to form a new record.

5 At any time he may punch a record onto paper tape, to be kept permanently if desired. At any later time he may then use this tape to re-enter this information back into the on-line system; to type a printed version on the Flexowriter; or as an input to the off-line system.

6 Once the equipment has been turned on, and the on-line program has been loaded and initiated at the computer, the user

directs all further system actions from the work station (the CRT display, keyboard, etc.) by means of successively designated commands.

6a Each command is executed immediately.

6b The function of the commands, individually and collectively, has been designed to be maximally useful in the task environment of working with the linked-statement structures that represent our working records of plans, specifications, computer-program design records, system-reference documents, external-document reference files, report drafts, etc.

6c Each command is designated by a simple, convenient combination of keyboard-character strokes and screen-selection actions (with light pen or table cursor).

6d *c A sizeable portion of our research effort continually goes toward improving the repertoire and designation means of these commands.

1 Various segments of the on-line system have been developed under different sponsorship, according to the pursuits of the respective projects.

1a 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 inputting and outputting; and for executing the higher-level commands that operate on statement structures and tape files.

1b A 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-designation operations; and the basic editing routines.

2 The two basic components of a command--the operator and the operands (or parameters).

2a The operator--specifying which command of the repertoire is to be executed.

2a1 Generally designated by several mnemonic alphabetic characters (with perhaps a SPACE stroke) struck by the user on the keyboard. Case of alphabetics is unimportant.

2a2 Or, a special one-handed keyboard may be used, leaving the other hand free for light-pen or cursor use. This has specially arranged keys for designating forward or backward scan, and for delete, insert, replace, move, and copy operations on text, character, word, line, and statement entities.

2a3 Full name for operator appears on top line of display immediately after the operation is thus designated.

2a4 After command execution, operation name remains displayed; successive executions do not require re-designating the operation.

2a5 Generally, input characters will be interpreted as command-operation designation only after: a command has just been executed (by striking the CA key), a command has just been aborted (by striking the CD key), or the system has just been started up.

2b The operands and parameters--three types:

2b1 A numerical parameter, e.g., for designating how many lines to scan or which type-file item to access. Entered at appropriate time (see below) from the alphanumeric keyboard.

2b2 Operand entities displayed on the screen.

2b2a Selected by locating the light pen or cursor near a character or printing space and hitting the associated SELECT button.

2b2b User actually selects a character (which can be a non-print character); if a larger entity (i.e., word, line, or statement) is called for as an operand, the computer takes that entity which includes the selected character.

2b3 Literal input, a string of characters entered at the appropriate time on the alphanumeric keyboard.

2b3a Always terminate LIT with a CA.

2b3b At the time during a command designation that LIT is expected by the computer, a space is cleared on the display and the user sees the character-by-character accrual of his keyboard input--to be put in the specified text location by the final CA action.

2b3c During LIT input, a BACKSPACE deletes the last character of the LIT string.

2b3d Similarly, a BACKSPACEWORD (a special key) deletes the last word.

2b3e The user need not be concerned with new-line designation; if a word is being entered and the end of the line is reached before a SPACE is entered, the computer automatically shifts the partial word to the start of the next line.

3 Executing or aborting a command.

3a After designating appropriately the operation, parameters, and operands, striking the CA key (there is one on each side of the keyboard) will cause the command to be executed.

3a1 On the commands not involving a literal input, a SPACE key (generally easier to strike) may be used optionally in place of the CA key.

3a2 A bug-select actuator on a cursor has exactly the same effect as the CA key, and may be used in its stead at any time.

3b At any point in designating a command, striking the CD (command-delete) key will abort the command.

3b1 The operator designation in the top line of the display will remain as it was before hitting the CD key.

4 Many commands change the contents of statement; the new formatting is automatically done by the computer.

4a In general all the text of a statement is cut into new line assignments. A given line is terminated (by a new-line start) at the inter-word gap which comes nearest to filling out a stipulated length of the line.

4b The exception: if a line contains a TAB in it, then its line-start text position remains fixed.

4c On type-out or punch-out, leading SPACE and TAB codes are inserted to indent each line of a statement 3d spaces, where "d" is an integer one less than the structural depth of that statement.

5 Command-description conventions.

5a A description of the way a given command is designated is presented below as a succession of (upper-case) character groups, each separated by a SPACE.

5b The single letters each represent the corresponding single alphabetic character to be entered. (Case is unimportant in actual usage.)

5c SP represents a SPACE character.

5d C1,C2,..., W1,W2,..., L1,L2,..., S1,S2,..., represent user-designated characters, words, lines, or statements--each specified at command-designation time by selection of any single character within the entity.

5e LIT represents a literal-input string and includes all characters entered, even SPACE, TAB, and CARRETURN.

5f NUMBER represents any decimal integer entered on the alphanumeric keyboard.

5g CA represents hitting the CA (command-accept) key.

6 Commands currently available:

6a Summary list:

6a1 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

6a1a This new data is added to the end of the existing working data on the drum.

6a1b The "enter" process will halt when drum is near full, and the typewriter will print appropriate notice. This allows for some free space for copying and inserting. Reinitiating the "enter" command will load until working space is full.

6a1c 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.

6a2 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 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
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

6a2a See 6a4a for definition of "logical break."

6a3 Modify text seen in display frame.

6a3a 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

6a3b 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

6a3c 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

6a3d 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, character C2 through C3 to follow character C1
M C C1 C2 C3 CA	Move the text string, character 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 L1
M S S1 S2 CA	Move statement S2 to follow statement S1

6a3e 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

6a4 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 up with the predecessor of ST2 at the top of the frame.

6a4a ST2 is said to be the logical successor to ST3 if there could exist an actual hierarchical structure such that (by their location numbers) ST2 could succeed ST3 in the text. For instance, following 2b3 one could logically accept only 2b3a, 2b4, 2c or 3.

Presence of any other number on the next statement establishes a "logical break" at this point in the text.

6a5 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

6a6 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 (not yet implemented)
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)
O T S S1 S2 CA	Output to typewriter, statement
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

6a7 Clear the working space on the drum of its present contents.

Z W S	Zero work space

6a8 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

6a8a 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."

6a8b An O M command at this point will create a new item on the end of the file

6a9 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. "Channels" refer to the 512-character modules of drum working space, of which there are a total of 36.

1 This section contains brief descriptions of the computer and associated peripheral equipment currently used by our on-line text manipulation system.

2 THE COMPUTER (CDC 160A)

2a Memory:

2a1 6.5 usec cycle time.

2a2 12-bit word.

2a3 4,096 words per bank, directly addressable.

2a4 Two banks on our machine--programmer must set up bank controls to shunt his access requests, independently for four categories of access, to the appropriate bank.

2a5 Each bank has independent access circuitry.

2b Instruction repertoire:

2b1 No built-in multiply, divide, square root, etc.

2b2 Full complement of add, subtract, conditional branch, transfer, logic (logical product, selective complement), shifting, input-output, and selective stop and jump (responding to switches on console).

2b3 Since 12 bits can just exactly address 4096 words, all instructions requiring operand specification over a complete bank require two successive words--one for operation specification and one for operand specification.

2b4 A significant proportion of instructions require but one word, and operate with 6 bits of operand specification in one of the following modes:

2b4a Relative forward--addressing one of the 64 words following the cell in which the single-word instruction was located.

2b4b Relative backward--addressing one of the 64 words preceding the cell in which the single-word instruction was located.

2b4c Direct--addressing one of the first 64 words in a bank specified by the direct-bank bank-control setting.

2b4d Indirect--telling the computer (with a one-word instruction) to go to the specified one of 64 direct-bank words, take the 12-bit contents as the full-bank address of the operand, and look for the operand in the bank specified by the indirect-bank bank control.

2b4e No address--a 6-bit operand is to be found in the lower six bits of the instruction word.

2b5 Variations in the operation code of nearly all the commands indicate which way the operand is to be obtained for that instruction. For example, the add instruction will have the following variations:

2b5a Add no address (adn), add the lower six bits of the instruction word to the accumulator.

2b5b Add direct (add), add to the accumulator the contents of the direct-bank cell specified by the lower six bits of the instruction word.

2b5c Add memory (adm), add to the contents of the accumulator the contents of the memory-bank cell specified by the 12 bits of the word following the instruction word (then get the next instruction from the word following that one). Which bank to use for operand accessing is specified by the setting of the memory-bank control.

2b5d Add indirect (adi), add to the contents of the accumulator the contents of cell in indirect bank that is specified by the contents of the cell in direct bank whose address is the lower six bits of the instruction word.

2b5e Add constant (adc), add to the contents of the accumulator the contents of the cell following the instruction--and get the next instruction from the cell following that.

2b5f Add forward (adf), add to the contents of the accumulator the contents of the cell that is forward of the instruction cell by the six-bit number found in the lower half of the instruction word.

2b5g Add backward (adb), add to the contents of the accumulator the contents of the cell that is backward from the instruction cell by the six-bit number found in the lower half of the instruction word.

2c Interrupt feature:

2c1 Four independent sources, two internal and two external, may cause an interrupt of what the computer is currently doing.

2c2 Interrupt signal causes contents of accumulator to be put into special cell, and the computer to get its next instruction from the succeeding cell.

2c3 The special cells, for the four sources, are cells 10, 20, 30, and 40--hence the sources are generally called the interrupt-10, interrupt-20, interrupc-30, and interrupt-40 sources.

2c4 Programmer can lock out these interrupt inputs programmatically.

2c5 If interrupts are not locked out, interruption occurs at completion of current instruction.

2d Input-output provision:

2d1 Two input-output channels that can operate independently--termed "normal" and "buffer."

2d2 Normal works as one expects--give a command to input or output and the computer waits until the job is done before it goes on to do further work.

2d3 Buffer works independently of the normal instruction cycles. Give an instruction for a buffer in or out and the main sequence of operations will continue while this input or output is being carried out. Every time the buffer channel needs access to the memory it steals a cycle from the main program sequence without otherwise bothering it. At the end of the buffer operation, an interrupt-20 automatically occurs--and the programmer has had to be ready with the appropriate instructions starting at cell 21 to take care of this.

2d4 After a device has been selected, all subsequent input (or output, if selection was for output) instructions operate with that device.

2d5 There is a family of single-word transfer commands that send or receive one word per instruction.

2d6 There is family of block-transfer commands that will send or receive an arbitrary-length block to or from consecutive cells of memory, at the rate determined by the external device.

3 PERIPHERAL EQUIPMENT:

3a For any device, transfer to and from the computer (on either channel) can be accomplished by single-word-at-a-time commands, or by block-transfer commands.

3b Paper tape reader. Photo-electric. Can read at asynchronous rate up to maximum of 320 frames/sec. Will accept 6-, 7-, or 8-level tape. Always on normal channel.

3c Paper-tape punch. A Teletype product, punching 8-level oiled tape. Can punch asynchronously up to maximum of about 120 frames/sec. Always on normal channel.

3d On-line typewriter. IBM typewriter, with CDC interface. Can couple to either channel.

3e Character generator.

3e1 Several modes of operation, in which it interprets differently the words sent from the computer.

3e2 The mode is determined by the program code used to select the character generator for coupling to the output channel.

3e3 The mode we use for text interprets the words following the select instruction as follows:

3e3a The first word specifies vertical position (nine bits) and the least-significant three bits of horizontal position.

3e3b All succeeding words (until another select instruction) specify a character to be displayed (with 6 bits) at the vertical location already designated, and the most-significant six bits of the horizontal position.

3e3c This allows a whole line of characters to be outputted as a block following a select instruction which specifies the vertical position of the line.

3e4 There is a repertoire of 43 characters to select from.

3e5 Characters are generated in an asynchronous operation that takes a maximum of 6 microseconds--but the output channel cannot deliver words to the output in less than about 17 microsecond intervals--so we have a maximum generation rate limited by this factor of a little less than 60,000 characters per second.

3e6 We display about 1,000 characters maximum on our screen, and run it at a rep rate of 60 frames/sec.

3e7 The new character generator being installed soon will interpret output words as specifying two characters per word, and will double our displayable capacity.

3f Mag tape. A CDC Type 603, compatible with IBM and Burroughs B5500. Programmer can write records of arbitrary length--transport automatically leaves inter-record gaps after stop sending it data. Has end-of-file code that can be put on programatically. Will read forward one record at a time, or back up one record at a time, from a single instruction.

3g Drum--a 32,000-word, fixed-head auxiliary storage device.

3g1 Speed, about 30 rev/sec.

3g2 Can only make access to records--two records per track, 32 tracks.

3g3 Each record holds 512 12-bit words.

3h Special interface and associated devices used by the on-line console.

3h1 Light pen, manufactured by Sanders Associates of Nashua, New Hampshire.

3h1a A photomultiplier tube in the control unit receives light by means of a fiber-optic bundle from a hand-held pen containing a lens which focuses light on the bundle.

3h1b A circle of orange light is projected from the pen to aid in aiming. The source for this light is in the control unit, and light is again transmitted by a fiber bundle.

3h1c When a light pulse of suitably fast rise time is detected an electrical pulse is generated in the control unit. A switch on the body of the pen unit gates this pulse to the interface logic.

3h1d In the single-pulse mode of operation, only one pulse is produced each time the pen button is depressed and the finder beam goes out to indicate a successful detection.

3h1e In the continuous mode of operation, a pulse is sent to the interface each time a light pulse is detected, as long as the pen button is held down.

3h1f The pulse mode is set by means of a switch on the control box (to which the 34-inch fiber-optic bundle attaches).

3h1g When the 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 character on the display which produced the light pulse.)

3h2 An analog-to-digital converter, manufactured by Dynamic System Electronics, allows the digitizing, selecting, and inputting to the computer of four different analog input channels. The converter produces nine bits plus sign, with a settling time of 400 microseconds. The converter is used to input positional information from the following operand locating devices:

3h2a A joystick, manufactured by Bowmar Associates, has two potentiometers coupled to a vertical stick. The potentiometers are used as voltage dividers, and produce voltages proportional to the X and Y deflection of the stick from its central location. A switch, actuated by pressing down on the stick, may be used as an input to the computer--to mark operand locations, for example.

3h2b The Grafacon, manufactured by Data Equipment Corporation, consists of a linear potentiometer mounted in a frame which is pivoted on an angular potentiometer. The voltage outputs from the two potentiometers represent polar coordinates about the pivot point. A ball or a pen, mounted on the end of the linear potentiometer shaft, is moved about by the operator and is depressed to actuate a switch which may be used as a computer input.

3h2c The mouse, made by SRI, consists of two potentiometers mounted in a frame with their shafts orthogonal and a wheel on each shaft. As the frame is moved about a surface the potentiometers resolve the motion into two coordinates. A switch mounted on the frame may be used as a computer input.

3h2d A footpedal, made by SRI, consists of a potentiometer coupled to a pedal which is pivoted at its center. Rocking the foot forward and backward operates the potentiometer; a switch operated by the other foot chooses horizontal or vertical input for the output of this potentiometer.

3h3 The interface provides for input to the computer of external contact closures. The switch circuits are arranged in three groups; a group of 15 are encoded to 4 computer input lines, a group of 7 are encoded to 3 input lines, and a group of 5 are input directly to 5 input lines. Actual input lines are selected by means of a patch-panel to provide flexible assignment of bits in the input word.

3h4 A bell mounted in the on-line console may be rung by a select code from the computer.

3h5 An interrupt for timing purposes may be sent to the computer at a selected rate. A multivibrator in the interface covers an interrupt rate range of approximately 30 to 150 cycles. An external input will accept a rate up to about 5000 cycles.

3h6 All interrupts from the interface may be locked out by a select code from the computer, and enabled by another select code.

SECTION IV -- OFF-LINE TEXT MANIPULATION SYSTEM
PART A -- INTRODUCTION

1 Implementation of the Off-Line System has been funded in part as an in-house project and in part by the Air Force Office of Scientific Research.

1a Development of statement-manipulation techniques and programming on the B5500 were supported by Stanford Research Institute as an Institute Sponsored Research project. Included in this effort was the 160A programming required to translate between tapewriter codes and Burroughs code.

1b Z-Code editing features incorporated into this system were developed and programmed on the 160A under the sponsorship of the Air Force Office of Scientific Research.

2 The Off-Line System was implemented to make available machine-aided text editing and updating on a fast-turn-around basis to a larger community than can be served by the current On-Line System.

3 The Off-Line System makes use of the combined facilities of the CDC 160A computer in the Systems Engineering Laboratory and the Burroughs B5500 computer operated by the Mathematical Sciences Department.

3a Since paper tape provides a convenient medium for entering text, and since the B5500 is not equipped for paper-tape input, the 160A is used to translate paper tape input in Flexowriter or Teletype code to Burroughs code on a magnetic tape.

3b The larger core and drum memories of the B5500 are utilized for rapid access to statements anywhere within a fairly long document to combine text from separate input tapes and/or to restructure the contents of a given document according to commands specified in one or more of the input tapes. Statements are inserted, moved, or replaced essentially by successive modifications of statement-to-statement links defining a path through the document. Separate documents may be spliced end-to-end or merged such that their statements are intermingled. Additional text may be appended to existing statements by means of similar links. The B5500 produces an output magnetic tape in which the document is restructured as specified, with its statements renumbered according to a standard format.

3c The 160A converts Burroughs code on the output magnetic tape to Flexowriter code and executes Z-Code editing commands embedded in statements or appended to them during the statement-manipulation process on the B5500. The 160A produces a paper tape that may be listed on the Flexowriter to produce hard copy or entered as input to the On-Line System. The output tape may, of course, also be used as input to a later pass through the Off-Line System for updating or further editing or restructuring.

4 The ability to append Z-Code editing commands (which can reach any point within a statement) during the restructuring process permitted separation of the gross restructuring process from the detailed editing process. Since the latter had been previously programmed on the 160A, this organization minimized the programming effort required to implement the system.

5 Any number of paper tapes may be merged to produce a single document.

6 Any number of documents may be processed in a single batch, up to the capacity of a single magnetic tape (with high-density recording).

7 Statement-manipulating procedures and Z-Code editing functions have been so designed that everything about the eventual output from the Off-Line process can be unambiguously determined by examining the tapewriter input.

7a This principle assures the user that he can edit or otherwise manipulate text material according to the way it appears on the hard-copy listing without risk of error due to non-printing keyboard actions or phantom characters that would throw line, word, or character counts off.

7b Adherence to this principle has made it possible to take "old" documents produced with early versions of the text-editing processes and rework them using later techniques without being trapped by some forgotten (hidden) feature of their machine coding.

1 Implementation of the Off-Line System has been funded in part as an in-house project and in part by the Air Force Office of Scientific Research.

1a Development of statement-manipulation techniques and programming on the B5500 were supported by Stanford Research Institute as an Institute Sponsored Research project. Included in this effort was the 160A programming required to translate between tapewriter codes and Burroughs code.

1b Z-Code editing features incorporated into this system were developed and programmed on the 160A under the sponsorship of the Air Force Office of Scientific Research.

2 Input is via paper tape prepared on Flexowriter or Teletype machines.

2a "Notes for Orientation of Personnel Preparing Copy for the Off-Line System" is a useful reference for the first-time user.

2b "User Guide to Statement Manipulation in the Off-Line System" is a concise reference for the experienced user.

2c "Z-Code Reference Summary" is a reference document describing editing operations within statements.

2d "Capitalization and Underlining on the Model 33ASR Teletypewriter" is a guide to the use of this machine for the preparation of input material.

3 All tapes should carry the source data in man-readable form, i.e., initials of originator and date in white pencil or gummed label on the tape leader.

3a Tapes to be merged should carry identical source data, i.e., the source data of the original memo.

3b Tapes for different jobs carrying the same initials and date must be identified by serial numbers following the date, i.e., ART 15 FEB 65-1 and ART 15 FEB 65-2.

4 All tapes should be labelled as to the machine code: FLX if prepared on Flexowriter, TTY if prepared on Teletype machine, FL if output from a previous pass through the off-line (FL) system, and NL if output from the on-line system.

4a Output from the off-line and on-line systems will normally be in FLX code.

5 A single original tape or the primary tape to which others are to be merged need not carry additional information.

6 Two types of merge operation are available. In labelling tapes for processing, "merge," in the narrow sense, will be used to refer to tapes carrying data to be interleaved with a primary tape. "Follow" will refer to tapes carrying statement lists to be tacked onto the end of a primary tape. The latter mode permits separate memos repeating some of the same statement numbers to be spliced in sequence to form a longer memo.

6a Tapes to be merged with a primary tape should carry the word "merge" and a number indicating the order of merging; thus "merge #1" would be merged with the primary tape before the tape labelled "merge #2."

6b Tapes to follow a primary tape should carry the word "follow" and a number (which must be a multiple of 10) to be prefixed to each statement number of the following memo. This number must be larger than the highest principal-statement or heading number of the memo it follows, and it must be distinct from the prefix used for any other "follow" tape to be combined with the same memo. Operation of the prefix is that of placing 10. in front of each statement number in the following memo, if 10 is the prefix designated.

7 A brief form on a 3-by-5 card, available at the collection point, must be filled out for each job. This form is self-explanatory.

8 The tapes for each job should be stacked on top of the 3-by-5 card at the collection point.

9 Normal hard-copy outputs are (1) a B5500 listing, with the Z-Code commands not yet executed, and (2) a Flexowriter listing of the output paper tape, produced after Z-Code execution. The paper tape in FLX code is the machine-readable output.

9a At times, the Flexowriter may be a bottleneck in the system. At such times, faster turn-around may be achieved by working with the B5500 listing and not waiting for the Flexowriter listing. Care must be exercised, however, since the Z-Code processing will result in reformatting within statements, so that format on the B5500 listing may not be the same as that on the paper tape.

9b If Z-Code commands are used only to modify immediately adjacent text, i.e., text within the entered statement in which they occur, Z-Code processing can be performed prior to B5500 processing, and the B5500 listing will be "clean." This will not work, of course, for Z-Code commands in APPEND statements that reach into text entered in a previous statement or on another tape.

10 Until the format of the 3-by-5 cards is modified to include a specific place for this information, please write on the card either "Z-Code FIRST" or "Z-Code LAST."

1 Implementation of the Off-Line System has been funded in part as an in-house project and in part by the Air Force Office of Scientific Research.

1a Development of statement-manipulation techniques and programming on the B5500 were supported by Stanford Research Institute as an Institute Sponsored Research project. Included in this effort was the 160A programming required to translate between tapewriter codes and Burroughs code.

1b Z-Code editing features incorporated into this system were developed and programmed on the 160A under the sponsorship of the Air Force Office of Scientific Research.

2 A STATEMENT is a segment of text headed by a statement number preceded by two carriage returns (or, on the Teletype, two line feeds).

2a All elements of the text, including the Source, Title, Abstract, etc., must be in statement format; that is, they must be preceded by two carriage returns (or two line feeds) and appropriate statement numbers.

2b The first characters entered on any tape must be preceded by two carriage returns (or line feeds).

3 A STATEMENT NUMBER is an alternating sequence of numbers (one or more digits) and letters (doubled, tripled, etc. if necessary).

3a The first symbol of a statement number must be a numerical digit.

3b Literal elements of statement numbers, a, b, c, etc., must be lower case. Slashes (/) and plus signs (+) within or preceding a statement number will invalidate the number.

3c Statements headed by numbers alone designate the highest level in the text structure, either the major headings or the principal lead statements.

3d Statement numbers of the form 2a, 2b, 2c, etc. designate elements of a statement list, or substructure, subordinate to the head statement designated by the number 2 alone.

3d1 If the number of items in a statement list carrying a letter as its last character exceeds 26, letters are doubled up according to the following convention: 2x, 2y, 2z, 2aa, 2ab, 2ac, . . . 2az, 2ba, etc.

3e Statement numbers of the form 2b1, 2b2, 2b3, etc. designate elements of a statement list, or substructure, subordinate to the head statement designated by the statement number 2b.

3e1 Numerical sequences may be as long as required: 2b8, 2b9, 2b10, 2b11, . . . 2b99, 2b100, etc.

3f Regardless of their order in the input text, the B5500 will output statements in the order determined by their statement numbers.

3f1 In the reordering of statements according to statement number, the substructure under each statement will be outputted directly following that statement, and this rule will govern down to the lowest level of the structure, as in this document.

4 Statements may be interpolated into an existing list by utilizing the following conventions:

4a A statement to be inserted between major headings 2 and 3 and of equal rank with them may be assigned the statement number 2.5 (the 5 could be any digit or decimal number). The B5500 will renumber this inserted statement 3, change the former 3 to 4, etc. all the way to the end of the list. Furthermore, it will make the same changes to the first numbers of all subordinate statements, so that each heading statement will retain its own substructure.

4a1 If several statements are to be interpolated between two existing statements, they may be numbered 2.3, 2.4, 2.5, 2.52, 2.6, etc., and they will be inserted in order of their decimal values; that is, 2.52 would come after 2.5 and before 2.6 in the B5500 output, regardless of their order in the input text.

4a2 If an inserted statement should carry a substructure of subordinate statements, they may be designated as follows: 2.52a, 2.52b, 2.52c, etc. When the 2.52 is changed to a whole integer in renumbering, the subordinate statement numbers will be altered to agree, so that the substructure will follow the referenced statement.

4b A statement to be inserted between 2b and 2c and of equal rank with them may be assigned statement number 2b.m (the m could be any letter of the alphabet or string of letters.). The B5500 will renumber this inserted statement 2c, change the former 2c to 2d, etc. all the way to the end of the substructure list under heading 2. Furthermore, it will make the same changes to the corresponding letters in the numbers of all subordinate statements involved, so that each statement will retain its own substructure.

4b1 If several statements are to be interpolated between two existing statements with final literals in their statement numbers, they may be designated as follows: 2b.a, 2b.c, 2b.m, 2b.mb, 2b.n, etc., and they will be inserted in alphabetical order, treating second letters as interpolations between first-letter designations; that is, 2b.mb would come after 2b.m and before 2b.n in the B5500 output, regardless of their order in the input text. (This amounts to a decimal interpretation of the literal string, consistent with the interpretation of the numerical string.)

4b2 If an inserted statement that will carry a final literal in its statement number should carry a substructure of subordinate statements, they may be designated as follows: 2b.mb1, 2b.mb2, 2b.mb3, etc. When the 2b.mb is changed to a number followed by a simple literal in renumbering, the subordinate statement numbers will be altered to agree, so that the substructure will follow the referenced statement.

4c The conventions described above may be utilized at all levels of the text structure. If the level in which interpolation is to take place is designated by statement numbers with final numerical symbols, the interpolation string is numerical. If the level in which interpolation is to take place is designated by statement numbers with final alphabetical symbols, the interpolation string is alphabetical.
4d In cases where it becomes necessary to insert a statement before the first item of a list or sublist, the following convention is useful: 1,5 will be renumbered 1, with all subsequent numbers increased, so that the list is pushed down. 2a,m will be renumbered 2a, with all subsequent second literals in the list advanced one letter, thus pushing down this sublist. All other conventions discussed in 3a thru 3c hold when the period (.) is replaced by the comma (,). Interpolation now takes place before the statement whose number precedes the comma, rather than after the statement whose number precedes the period. The relative order of multiple insertions is governed by the same decimal interpretation as when the period is used; i.e., the comma does not reverse the sense of the

interpolation, it merely designates interpolation into the preceding rather than the following interval.

4e If two statements should be inadvertently entered with the same number, the statement entered last will follow the first, and they will be renumbered consecutively.

5 DELETE, REPLACE, MOVE, and APPEND Operations are achieved by utilizing statements with coded instructions tacked onto their statement numbers.

5a Command codes are literal elements, d, dt, dl, r, m, and a, following a colon (:). These literal elements must be lower case. Slashes (/) and plus signs (+) within the command structure will invalidate the command.

5b Each of the following commands must be entered as a separate statement; that is, the coded statement number must follow a double carriage return (or double line feed).

5c The coded statement number 2b1:d will delete statement 2b1 wherever it exists, either in the original copy or in the correction copy.

5c1 Deletion of a statement automatically deletes all of the substructure under that statement; thus the command 2b1:d will delete not only statement 2b1 but all statements with 2b1 followed by any combination of letters and numbers. It will remove 2b1a, 2b1a1, 2b1b, etc.

5c2 When a statement, with its substructure, is deleted, the B5500 will renumber the remaining elements of the list and the substructure statements under them, so that there will be no discontinuity in the number designations.

5c3 The delete code may be used to delete a statement that is itself a delete command. For instance, if the delete command 2b1:d has been entered anywhere in text as a statement, the statement 2b1:d:d will remove the delete command, and the original statement 2b1 will stand.

5d The coded statement number 2b1;1a:dt will delete statement 2b1 and any and all statements following it in the input text up to and including statement 1a. (In this example, it is assumed that text is being entered out of order and that there is a statement 2b1, followed later on by a statement 1a, with any number of intervening statements.) This :dt code is used to remove statements from the text material on the tape currently being prepared on the tapewriter, whether new material or correction copy. The :dt code will not reach material on any previously processed tape with which the currently prepared text is to be merged. Neither will it reach beyond the segment of input text bounded by the referenced statements. A statement numbered 2b1a, for instance, would be deleted along with its heading statement 2b1 only if statement 2b1a lay between 2b1 and 1a in the input text; otherwise it would remain.

5e The coded statement number 2b1;2d:dl (final character is letter "l") will delete statement 2b1 and any and all statements following it in structured order, up to and including statement 2d and all of the substructure under 2d. The statements between 2b1 and 2d, and the substructure of 2d, may have been entered on separate tapes, intermixed with any other statements, etc. Wherever they exist in the structured or unstructured text, items headed by statement numbers beginning with 2b1, 2b2, 2b3, . . . 2c, and 2d will be deleted. Statements with numbers out of this range will not be deleted, even though they may be intermixed in the text.

5e1 When a group of statements, with their substructure, is deleted, the B5500 will renumber the remaining elements of the list and the substructure statements under them, so that there will be no discontinuity in the number designations.

5f The coded statement number 2b1:m 2b3.5 will renumber the statement numbered 2b1 with the number 2b3.5 and thus cause it to be moved to a position in the structure between the statements previously numbered 2b3 and 2b4. Since the original 2b1 is now removed, however, all of these numbers may be changed. The single space following the code letter "m" is required. The second referenced statement number, 2b3.5, need not be an interpolation number; it could be 2b6, 2c, 3f, or any other.

5g The coded statement number and following literal string 2b1:r Now is the time for all good men . . will replace the previous text of the statement numbered 2b1 with the text "Now is the time for all good men . ." The replacement code :r will not affect any other statement except the referenced one. Replacement is complete, and cannot be partial; that is, the whole of the statement is removed and replaced by the literal string following the coded statement number.

5h The coded statement number and following literal string

2bl:a Now is the time for all good men . . will append the words "Now is the time for all good men . ." to the end of statement 2bl. All of the former statement remains intact, and the addition will be made only at the end. A single space following the code letter "a" is required. Any additional spaces preceding the literal string will appear as a spacing gap between the end of the former text and the beginning of the addition. If one desires to leave two spaces before an added sentence, the first letter of the sentence should be separated from the code letter "a" by three spaces.

5hl Since, in the current system, Z-Code processing will follow statement processing on the B5500, and since the range of Z-Code commands will be limited to one statement, the :a operation may be used to append Z-Code commands to statements, providing for deletion and insertion of text within selected statements. Note that the Z-Code INSERTION command must be followed by a spacing character, and that this spacing character will be deleted when the command is executed. In order to avoid deletion of one of the required carriage returns at the end of the statement, one should follow the insertion command by one or more spaces. In the event that a carriage return is inadvertently entered immediately following an insertion command, follow it with at least two more carriage returns.

6 Since Z-Code processing within statements will follow statement processing on the B5500, Z-Code commands cannot be used to delete, modify, or insert statement numbers or coded statement numbers constituting commands to the statement-manipulating system. Therefore, the following conventions have been implemented to permit modification of statement numbers (either uncoded, or coded with command symbols):

6a If an error is recognized while typing a statement number, and only the last few symbols are in error, the PERCENT (%) sign may be typed. Each %-sign will delete one character backward in the statement number. Thus lb2a%b will be corrected to read lb2b, and lb2a%%3a will be corrected to read lb3a. Command symbols may be similarly corrected; for instance, lb2;lc:dl%t will be corrected to read lb2;lc:dt.

6b If an error is recognized while typing a statement number and it would be just as well to start over from scratch, the DOLLAR ($) sign may be typed. The $-sign deletes all that has been typed of the statement number (and command code), back to the double carriage return that preceded it. Thus 1b2a$1b3a will be corrected to read 1b3a, and 1b2;1e:d1$1c2;1e:dt will be corrected to read 1c2;1e:dt.

6c The %-sign and $-sign delete commands must be made within the statement number or command, and thus depend on catching the error before going past it by too many symbols. If the incorrect statement number or command has been completed and a spacing character typed, it may be corrected or deleted by a later command constituting a separate statement.

6c1 A MOVE command may be used to correct a statement number. For instance, if a statement has been entered with an incorrect number, 1b3, a later statement consisting of the move command 1b3:m 1c3 will have the effect of correcting the statement number to read 1c3.

6c2 An incorrect command code or an incorrect statement number in a command can best be corrected by deletion and re-entry. Deletion is accomplished by repeating the incorrect command followed by the symbols :d as a separate statement. In such cases, the correct command will then have to be typed as another separate statement.

1 Implementation of the Off-Line System has been funded in part as an in-house project and in part by the Air Force Office of Scientific Research.

1a Development of statement-manipulation techniques and programming on the B5500 were supported by Stanford Research Institute as an Institute Sponsored Research project. Included in this effort was the 160A programming required to translate between tapewriter codes and Burroughs code.

1b Z-Code editing features incorporated into this system were developed and programmed on the 160A under the sponsorship of the Air Force Office of Scientific Research.

2 Processing conventions:

2a The desired operation is completely specified by the first word of each entry statement--generally the standard location number or some variant on this.

2b When a statement is deleted, its substructure is deleted.
2c When a statement is moved, its substructure is moved with it.

3 User processes:

3a Insertion:

3a1 If, for a given statement, its location number has only alphanumerics, periods, or commas in it, and is followed normally, i.e., by a spacing gap, then that statement is to be inserted as a new statement in the location implied by the location number.

3a1a If several statements are thus assigned the same location numbers, their ordering in the eventual structure will be the order of their entry. They will be given consecutive location numbers in the final renumbering.

3a2 Allow use of interpolative numbering in location numbers to designate eventual location of statements being referenced.

3a2a Let 2a4.5 (or 4a.d) designate a location number coming after 2a4 (or 4a) in eventual interpolative order.

3a2b Let 2a1,2 (or 4a,g) designate a location number that comes before 2a1 (or 4a) in eventual interpolation order.

3a2b1 Interpret the characters after the comma as though they were positive-ordered designators that started from the predecessor location number (even though there may be no predecessor statement--i.e., the statement referenced is a head statement).

3a2b2 Assume that 2a1,3 (or 3a,c) would come before 2a1,5 (or 3a,e).

3a2c Compound interpolation is allowed: e.g., 2a4.5.2 (4a.d.b) designates a location number which would be between 2a4.5 (4a.d) and 2a4.6 (4a.e).

3a2d Multiple-character fields are not to be confused with interpolation designation: e.g., 2a3.12 and 2a3.25 represent the twelfth and twenty-fifth interpolative positions between 2a3 and 2a4--and are not the second and fifth positions between 2a3.1 and 2a3.2, or 2a3.2 and 2a3.3.

3b Appending to and modifying a prior statement:

3b1 Let a statement beginning with LN1:A designate that the rest of this statement will be appended immediately after the last printing character of STM LN1.

3b1a A SPACINGAP must appear after the "A" in the append command.

3b1b The processor removes this SPACINGAP during the append operation (before the Z-code processes are executed).

3b2 Any Z-Codes included in the appended string will be executed, treating the new composite statement as a whole, after all of the inserting, appending, deleting, and moving of statements has been done.

3c Replacement:

3c1 Let LN1:R designate that the entire text of STM LN1 is to be replaced by the text following the R.

3c2 The new STM LN1 will have the same location number (LN1), with new text.

3d Deletion:

3d1 Let LN1:D designate that STM LN1 be deleted.

3d2 Let LN1;LN2:DT designate that the input text string including and between STM LN1 and STM LN2 is to be deleted.

3d2a This deletes all statements, of any kind and level, in this string.

3d3 A delete command can operate upon a prior delete-command statement by using as the reference location number the entire compound word heading that statement.

3d3a For example, STM 2a4b is deleted by a statement headed 2a4b:d. But this delete command can itself be deleted by a statement headed 2a4b:d:d.

3e Moving statements and structure sections:

3e1 Let LN2:M LN1 designate that, to the structure location specified by LN1, the statement STM LN2 and its entire substructure is to be moved.

3e1a The statement STM LN2, its substructure, and the lists and substructures displaced by this move, will all be renumbered after the deleting, inserting, and moving operations are done.

3f Correcting statement-manipulation commands:

3f1 Let $ in a location number (in the op-code part of our statement-manipulation) designate that the $ and all characters up to it, are to be deleted by the B5500 processor before the command is interpreted.

3f2 Let % in the location number designate that both the % and the character just preceding it are to be deleted by the B5500 processor before the command is interpreted.

3f3 Before interpeting any command statement, the processor will begin at the left end of the location number and proceed to the right, character by character, looking for $ or % characters, and executing them immediately.

3f3a This means that n successive % characters will delete the n preceding characters.

3g General considerations:

3g1 The new numbers, appearing on the subsequent printout, will have no interpolation numbers.

3g2 The user may consider that the actual moving is not done until the very last of the processing for the whole job. Thus, for instance, after a LN1:M LN2 command, he can refer to STM ln1 or any statements of its substructure by their location numbers as seen in the "original" hard copy.

3g3 It may help if the user thinks of these commands as establishing new structural linkages (i.e., to list-successor and sublist-head statements) between existing statements, with renumbering to be done after all such new linkages are established.

3g4 The compound location numbers that effect relocation and deletion of other statements are to be the heads of empty statements.

3g5 It is useful to remember that the processor makes two passes through the entire input text.

3g5a First pass, backwards, executing only delete commands.

3g5b Second pass, forward, executing all other commands.

3h Things to be careful about:

3h1 Use no Z-codes in the location number (or command)--the % and $ signs are the only acceptable ways to make corrections in the location number.

3h2 For statements that are given the same location number (not a forbidden event--they will be inserted in order and given new numbers), the processor will hang up if one tries to refer to that location number for a move, delete, append, or replace.

3h3 Tabs appearing at the beginning of the line (i.e., immediately after a carriage return) will be removed.

4 Special Features:

4a Merging of two records:

4a1 Assume that the location numbers of the two records are independent of one another and that for each record they began with 1.

4a2 One may designate to the operator to load the second tape with a prefix integer, N.

4a3 Upon loading the second tape, the operator keys this integer in as a special parameter, and all statements in that record will have a prefix attached to the front of their location numbers composed of the integer N followed by a PERIOD.

4a4 The user would then write a new third tape to specify the manner in which the contents of the second tape are to be integrated with those of the first tape.

4a4a When referencing statements of the second record, the user must be careful to designate their location numbers with the appropriate prefix which he specified.

4b Multiple sequence input entry:

4b1 A user sitting at his own tape-punching typewriter preparing material dealing with a number of independent records, often finds that new thoughts occur for the modification of one record while he is typing on the modification for another.

4b2 The feature here described allows him in such a situation to interrupt the sequence being composed for the one record and introduce, on the same paper tape input, new statements for the sequence referring to the other record.

4b3 To use this feature, one designates an integer job number for each of these independent input sequences which he wishes to use. (He will communicate to the operator which paper tape records each of these corresponds to.)

4b4 When typing his input, the user may insert at any point a statement beginning with a # character followed immediately by an integer and then a SPACINGAP.

4b4a The integer designates to which record the following statements are to refer.

4b4b In this sequence-break statement, any comment-type text may follow the SPACINGAP, and will be ignored by the processor.

4b5 The operator will insert the necessary parameters at load time so that for each of the independent input records, the processor will scan the input tape and extract the statements referring to that record.

1 Implementation of the Off-Line System has been funded in part as an in-house project and in part by the Air Force Office of Scientific Research.

1a Development of statement-manipulation techniques and programming on the B5500 were supported by Stanford Research Institute as an Institute Sponsored Research project. Included in this effort was the 160A programming required to translate between tapewriter codes and Burroughs code.

1b Z-Code editing features incorporated into this system were developed and programmed on the 160A under the Air Force Office of Scientific Research.

2 In the current version of the off-line system the following steps are taken to process information:

2a Tapes (either Teletype or Flexowriter) are first converted to magnetic tape using the CONVERT program for the 160a.

2b Statement manipulation commands are then executed on the B5500.

2c The MAG-TAPE ZCODE program is used to execute Z-code commands on the statements and output a Flexowriter paper tape, which can be listed and recycled through the system.

3 Use of the 160a for converting paper tapes to magnetic tapes:

3a Turn on power.

3b Master clear.

3c Turn on paper tape reader.

3d Put a magnetic tape with a write ring on the tape unit; put magnetic tape unit on 0; and connect tape to normal channel.

3e (LOAD) Load CONVERT program paper tape at 0000.

3f Master clear.

3g (RUN) Put data tape in reader and run.

3h STOPS: The computer will halt at one of the following locations:

3h1 0727: System has failed to clear magnetic tape parity error. Do not master clear. Reset run switch.

3h2 0254 or 0754: The data tape has been processed. Do not master clear. Clear all jump switches. Do one of the following:

3h2a If the next tape is to be "merged" (same job), put 0000 in A and go to(RUN).

3h2b If the next tape is to "follow" (same job), enter prefix in A and go to(RUN).

3h2c If the next tape is not to be merged (different job), put 0001 in A and go to(RUN).

3h2d If there are no more tapes to be processed, put 0002 in A and run.

3h3 0153: Tape did not start with a carriage return. If the paper tape is a Teletype tape, reset run switch. Do not master clear. If the paper tape is a Flex tape, set selective jump switch 2 and reset run switch. Do not master clear.

3h4 0321: Normal completion of processing.

3h5 Any other stops: Computer error. Go to(LOAD) and start processing over.

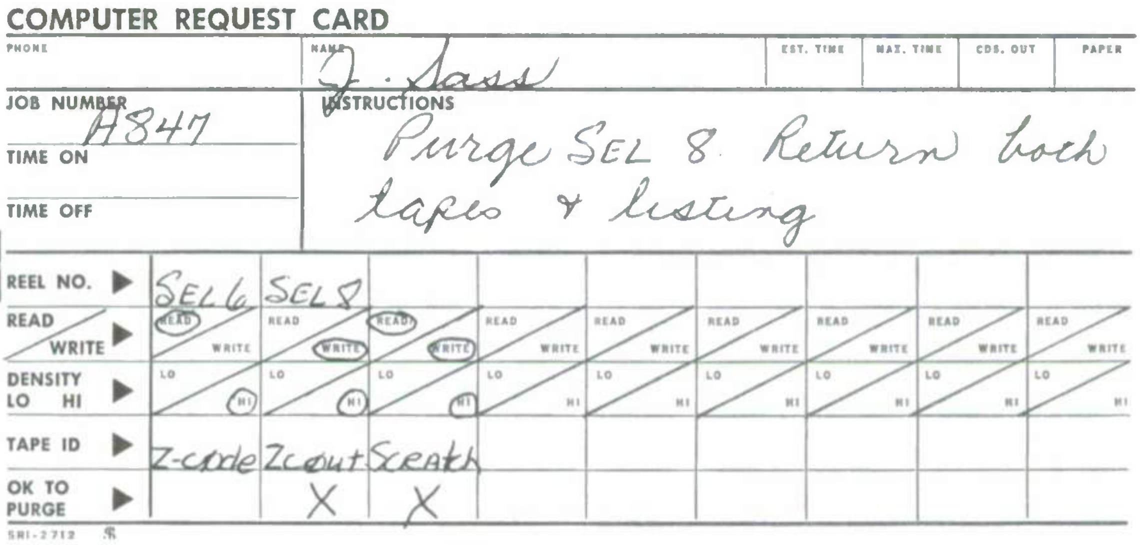
4 Use of the B5500 for statement manipulation:

4a Take write ring off tape (input tape of B5500).

4b Put write ring on another tape (to be output tape of the B5500).

4c Carry both magnetic tapes to the computation center.

4d Fill out an operator card as follows:



COMPUTER REQUEST CARD

PHONE	NAME J. Sass	EST. TIME	MAX. TIME	CDS. OUT	PAPER
JOB NUMBER A847	INSTRUCTIONS Purge SEL 8. Return both tapes & listing				
TIME ON					
TIME OFF					

REEL NO. ▶	SEL 6	SEL 8							
READ / WRITE ▶	[✓ READ] / WRITE	READ / [✓ WRITE]	[✓ READ] / [✓ WRITE]	READ / WRITE	READ / WRITE	READ / WRITE	READ / WRITE	READ / WRITE	READ / WRITE
DENSITY LO HI ▶	LO / [✓ HI]	LO / [✓ HI]	LO / [✓ HI]	LO / HI	LO / HI	LO / HI	LO / HI	LO / HI	LO / HI
TAPE ID ▶	Z-code	Zcout	Scratch						
OK TO PURGE ▶		X	X						

SRI-2712

4e Put card deck, both tapes, and instruction card on the table for B5500 input jobs.

4f Pick up both tapes, output, and card deck after processing is complete on the B5500. The processed data is on the magnetic tape with the write ring.

5 Use of the 160a for final processing:

5a Turn on power.

5b Master clear.

5c Turn on paper tape punch.

5d Turn on magnetic tape unit 1, load data magnetic tape in the magnetic tape unit, and connect tape on buffer channel.

5e Load MAG-TAPE ZCODE program paper tape at 0000.

5f (NEXT) Master clear.

5g Put run switch in run position.

5h Normal stop. Computer will come to a normal stop after the entire data on magnetic tape has been processed and the computer has punched a Flexowriter paper tape. If another magnetic tape is to be processed, go to(NEXT).

5i If computer stops before processing is complete, reset run switch. Do not master clear.

1 The Z-Code editing features of the off-line system were developed and programmed for the 160A under the sponsorship of the Air Force Office of Scientific Research.

2 DEFINITIONS

2a Printing Character = any symbol that prints out on the tapewriter (alphanumeric, punctuation, mathematical symbol, etc.).

2b Non-Printing Character = any command function that records on tape but does not print out on the tapewriter (space, carriage return, tab, backspace, etc.).

2c Word = printing character or unbroken string of printing characters isolated by non-printing characters.

2d Gap = non-printing character or unbroken string of non-printing characters bounded by printing characters.

2e Line = character string initiated by a carriage return. A line may be empty (two carriage returns in sequence).

2f Statement = segment of text within reach of Z-Code editing commands. The statement delimiter consists of two carriage returns (or line feeds on the Teletype) and a statement number.

3 CONVENTIONS USED IN THIS WRITEUP

3a All printing characters used in Z-Code descriptions stand for themselves except the letter N and N followed by an integer.

3b The letter N will denote a general integer whose value will specify a number of lines, words, characters, or tab stops in a Z-Code control string.

3c The letter N followed by an integer, N1, N2, N3, etc., will denote a subscripted N, that is, a general integer. Subscripted N will be used in expressions or discussions involving two or more integers that can take on independent values.

4 STRUCTURE OF A Z-CODE EDITING COMMAND

4a The computer recognizes a Z-Code editing command by the occurrence of a letter Z followed by an integer.

4b Initiation and termination of specific Z-Code commands is either explicit, involving specified symbols, or implicit (e.g., terminated by completion of a control string).

4c Point of editing within the current statement is designated by a control string of general form N1LN2WN3C specifying a count of N1 lines (L), N2 words (W), and N3 characters (C).

4d An editing command may contain a data string (text and/or other characters) delimited by parentheses (xxx...xxx).

5 EXECUTION OF Z-CODE EDITING COMMANDS

5a The computer searches backward through a statement, finding and executing Z-Code commands on a "last entered, first executed" basis.

5b Z-Code commands are treated as normal text words when they occur at the point of editing; hence, later commands can delete or modify earlier commands.

5c After execution of all Z-Code commands in a statement, text is "closed up" by replacing line-initiation commands with spaces or spaces with line-initiation commands as required to justify text to left margin and fill out complete lines.

6 DELETION: ZNL, ZNW, ZNC, ZN1LN2WN3C

6a ZNL deletes N lines backward in text, counting as the first line the one in which the Z-Code command occurs.

6a1 Line deletion is executed by deleting backward in text until N carriage returns have been removed.

6a2 Deletion of a carriage return automatically removes any tabs and/or spaces preceding the carriage return.

6a3 Point of reentry after line deletion is immediately following the last printing character on the preceding line. If that line is empty, the reentry point will follow the carriage return.

6b ZNW deletes N words backward in text, counting as the first word the Z-Code command or any unbroken string of printing characters of which it is a part.

6b1 Word deletion is executed by deleting backward in text until N gaps have been removed.

6b2 Deletion of a word thus removes the gap preceding that word. The deleted gap may contain any number of carriage returns.

6b3 Point of reentry after word deletion is immediately following the last printing character of the preceding word.

6c ZNC deletes the Z-Code command and N characters immediately preceding the Z-Code.

6c1 Both printing and non-printing characters are counted, including spaces and carriage returns. Each space introduced by a tab is counted as a separate character.

6c2 Deletion of a carriage return automatically removes any tabs and/or spaces preceding the carriage return; thus after deletion of a carriage return, the next character counted will be the last printing character on the preceding line. If that line is empty, the next character counted will be its carriage return.

6c3 Point of reentry after character deletion is immediately following the last surviving character, which may be either a printing or non-printing character.

6d ZN1LN2WN3C deletes N1 lines, N2 words, and N3 characters backward in text according to the conventions described above for the separate commands.

6d1 Order of execution is line deletion, followed by word deletion, followed by character deletion, regardless of order within the control string of the command.

6d2 Line counting begins with the line that includes the Z-Code command. Word counting begins with the last word of the last surviving line. Character counting begins with the last printing character of the last surviving word.

6d3 Point of reentry after compound deletion is immediately following the last surviving character.

6e The deletion command is implicitly terminated. The next character immediately following the control string will appear at the point of reentry following execution of the deletion command. This may be either a printing or non-printing character. Only characters of the form NL, NW, or NC must be excluded, since they would be interpreted as additions or amendments to the control string.

6f Since text is scanned backward toward the beginning during execution of deletion commands without interpreting deleted words, earlier editing commands may be deleted before execution, and thus will never be executed.

7 INSERTION: Z.IN1LN2WN3C(xxx...xxx)Z.2I gap

7a *c The periods (.) inserted in the above example and in similar examples to follow are to be ignored. Their sole purpose is to "spoil" the Z-Code command so that the example will remain in text and not be interpreted as a valid editing command, since this memo is being prepared using Z-Code editing techniques.

7b Insertion commands are explicitly initiated by the character string Z.1I followed immediately by a control string.

7c The control string N1LN2WN3C is of the same form as that of a deletion command, but its interpretation is different:

7c1 Non-zero line, word, and/or character counts in the control string key on the beginning of the statement, line, and/or word, and counting proceeds forward in text during execution.

7c1a If none of the integers in the control string N1LN2WN3C is zero, the point of insertion will be immediately following the N3-th character of the N2-th word of the N1-th line of the current statement.

7c1b A zero character count (0C) in the above control string would place the point of insertion before the first character of the N2-th word of the N1-th line of the statement, i.e., following the gap that precedes the N2-th word of that line.

7c2 Omitted line, word, and/or character counts in the control string designate the last line of a statement, last word of a line, and/or last character of a word.

7c2a Omitted or zero line count specifies insertion within the line containing the Z-Code command.

7c2b Omitted or zero word count specifies insertion within or adjacent to the last word of the designated line.

7c2c Omitted character count specifies insertion immediately following the last character of a designated word.

7c2d Note that omitted character count and zero character count (OC) are distinct and produce different results.

7d An insertion string, enclosed in parentheses (xxx...xxx), follows the control string.

7d1 The insertion may be any string of characters, including text, punctuation, control characters, and Z-Code deletion commands, but excluding Z-Code insertion commands.

7d2 A deletion command within an insertion string will be inserted at the specified point in text, to be executed later when the translator has scanned backward to that point.

7d3 If spaces are required to separate an insertion from adjacent text, they must be included in the parentheses.

7e Insertion commands are explicitly terminated by the character string Z.2I, followed immediately by a gap or the control string of an additional insertion command.

7e1 Multiple insertion commands are formed by following the character string Z.2I by the control string, insertion string, and terminating string of each successive command, without repeating the Z.1I initiating string.

7e2 A gap must follow the final Z.2I of a multiple insertion command.

7e3 The first non-printing character following an insertion command will be deleted with the command statement when it is executed; hence, this should be a space or extra carriage return not required in the ultimate formatting during close-up of text.

7f Restrictions on the formation of insertion commands:

7f1 Deletion commands embedded in the insertion command, other than within the insertion string, will invalidate the insertion command.

7f2 Carriage returns embedded in the insertion command, other than within the insertion string, will invalidate the insertion command if thy follow non-alphabetic characters but will be ignored if they follow alphabetic characters.

8 CONTROL STRING DETAILS COMMON TO DELETION AND INSERTION:

8a Any unbroken string consisting solely of integers alternating with any of the letters L, W, and C that begins with an integer and ends with one of the letters is a semantically valid control string.

8b Line, word, and character counts specified by a control string are the integers just preceding the last occurrence of the letters L, W, and C, respectively. Prior entries of a repeated specification are ignored.

8c Order of occurrence of L, W, and C in a control string may be completely arbitrary. Execution will be the same, regardless of the order in which the final specifications are made.

8d Amendment of specifications during construction of a control string may thus be achieved by merely appending revised specifications to the end of the string.

9 TABULATION

9a Tab stops are "set" in the software package as being at every eighth character position from the left margin.

9b Occurrence of a tab character will insert spaces as required so that the following character will occupy the character position designated by the next tab stop.

9c LEFT margin control: ZNT

9c1 The Z-Code command ZNT, where N is an integer, establishes a "normal" left-hand margin at the N-th tab stop. The command itself will be removed from text in the editing process. The "normal" left margin established by this command controls formatting of all following text until this formatting specification is revised or removed by another ZNT command, where N is another integer or zero.

9c2 The effect of this formatting command is to establish a "normal" line-initiation string consisting of a carriage return followed by N tab characters.

9c3 Carriage returns embedded in running text must be followed by N tab operations if ZNT has been specified and it is intended that edited text be justified to this "normal" left margin.

9c4 When text is "closed up" following execution of all Z-Code commands, "normal" line-initiation strings may be replaced by spaces and spaces by "normal" line-initiation strings as required to justify text to the "normal" left margin and fill out complete lines.

9c5 Line-initiation strings consisting of carriage returns alone or carriage returns followed by other than N tab characters will not be deleted or altered in the "close-up" process.

1 The operations for specifying capitalization and underlining, when preparing copy on the Model 33ASR Teletypewriter, were developed and programmed for the 160A under the sponsorship of the Air Force Office of Scientific Research.

2 CAPITALIZATION: /, +

2a The slash (/) preceding an alphabetic character will capitalize that character unless the slash is immediately preceded by an alphanumeric character.

2b The PLUS sign (+) preceding an alphanumeric character string will capitalize all the alphabetic characters in the string unless the PLUS sign is immediately preceded by an alphanumeric character.

2b1 String capitalization will be terminated by the first non-printing or non-alphanumeric character encountered following the command.

3 UNDERLINING: < , >

3a The LESS-THAN sign (<) preceding an alphabetic string will underline that string unless the LESS-THAN sign is immediately preceded by an alphabetic character.

3a1 Underlining will be terminated by the first non-alphabetic character encountered following the command.

3b The GREATER-THAN sign (>) preceding a non-alphabetic string will underline that string unless the GREATER-THAN sign is immediately preceded by a non-alphabetic printing character.

3b1 Underlining will be terminated by the first alphabetic or non-printing character encountered following the command.

4 CAPITALIZATION AND UNDERLINING LIMITATIONS:

4a Arbitrary mixing of upper- and lower-case alphabetic characters within one word cannot be achieved.

4b Capitalization and underlining cannot be specified for the same characters.

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Stanford Research Institute, Menlo Park, California	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP n/a

3. REPORT TITLE

Research on Computer-Augmented Information Management

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Final Report

5. AUTHOR(S) *(Last name, first name, initial)*

Engelbart, D. C. and Huddart, Bonnie

6. REPORT DATE March 1965	7a. TOTAL NO. OF PAGES 144	7b. NO. OF REFS 6

8a. CONTRACT OR GRANT NO. AF 19 (628)-4088	9a. ORIGINATOR'S REPORT NUMBER(S) ESD-TDR-65-168
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* Final Report, SRI Project 4987
d.	

10. AVAILABILITY/LIMITATION NOTICES

Copies available from the Defense Documentation Center. (DDC)
DDC release to CFSTI is authorized.

11. SUPPLEMENTARY NOTES None	12. SPONSORING MILITARY ACTIVITY Directorate of Computers, Electronic Systems Div., Air Force Systems Command, USAF, L. G. Hanscom Field, Bedford, Massachusetts, 01731

13. ABSTRACT

This report presents results of a research and experimental project in computer-augmented information management. The report is, in itself a product of the project: With the exception of "front matter," the entire report was composed, edited, and produced with on-line and off-line computer aids. For this project, the techniques of computer aids were applied to two areas: task monitoring and program design. The processes and techniques developed offer a promising beginning to computer-aided programming design extending from initial specification to final debugging in a unified design record that grows and evolves to complete final documentation. The processes and techniques also offer promise in increasing the productivity of individuals and groups of programmers. Future work envisioned for information-management systems such as that used in this study include program design records, external reference documentation, and user reference manuals.

DD FORM 1 JAN 64 1473

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Information Management Systems						
Design						
Computers						
Data Storage Systems						
Data Processing Systems						
Data Retrieval Systems						
Documentation						
Programming						
Computer Aids (on line-off line)						
User Manual						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2*a*. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2*b*. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7*a*. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7*b*. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8*a*. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8*b*, 8*c*, & 8*d*. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9*a*. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9*b*. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through
_______________________________________."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through
_______________________________________."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through
_______________________________________."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional

File With ESTI Copy Of Report

EDITOR'S CHECKLIST FOR TECHNICAL REPORT FORMAT	ESD TDR NUMBER 65-168 (Contr. AF 19(628)-4088)
	DATE 12 Apr 65 (Stanford Research Institute)

FRONT COVER

NO DISCREPANCIES COVER OMITTED CLASSIFICATION MARKING OMITTED ESD IDENTIFICATION OMITTED IMPROPER COVER STOCK IMPROPER FORMAT	[ ] DATE OMITTED [X] OTHER (Mock-up cover attached) (see item #1, reverse)

SPECIAL NOTICES (Inside Front Cover)

NO DISCREPANCIES ALL OFFICIAL NOTICES OMITTED AVAILABILITY NOTICE OMITTED DISSEMINATION NOTICE OMITTED DISPOSITION NOTICE OMITTED	[ ] LEGAL NOTICE OMITTED [X] OTHER Notices that apply to this report have been included on reverse of mock-up

TITLE PAGE (First right-hand page)

NO DISCREPANCIES AUTOMATIC DOWNGRADING NOTICE OMITTED ESPIONAGE NOTICE OMITTED CLASSIFICATION NOTICE OMITTED TITLE PAGE OMITTED	[ ] TITLE OMITTED [X] OTHER Include a Title Page in final copy.

FOREWORD (~~[illegible]~~)

NO DISCREPANCIES FOREWORD PAGE OMITTED SYSTEM, PROJECT OR TASK NUMBER OMITTED CONTRACTOR'S NAME AND ADDRESS OMITTED DISTRIBUTION LIMITATION REASON OMITTED CONTRACT NUMBER OMITTED	[ ] DATE OMITTED [ ] CAPTION OMITTED [X] OTHER (See item # 2, reverse)

APPROVAL STATEMENT (~~Last part of Foreword~~)

NO DISCREPANCIES APPROVAL STATEMENT OMITTED	[X] OTHER (See item #3, reverse)

ABSTRACT (~~Next right-hand page~~)

X NO DISCREPANCIES ABSTRACT OMITTED EXCESSIVE LENGTH	[ ] OTHER

(Continued on reverse side)