NASA CONTRACTOR REPORT

NASA CR-

c. 1

NASA CR-1827

ADVANCED INTELLECT-AUGMENTATION TECHNIQUES

by D. C. Engelbart and
Staff of Augmentation Research Center

Prepared by
STANFORD RESEARCH INSTITUTE
Menlo Park, Calif. 94025
for Langley Research Center

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • FEBRUARY 1972

1. Report No. NASA CR-1827	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle ADVANCED INTELLECT-AUGMENTATION TECHNIQUES		5. Report Date February 1972
		6. Performing Organization Code
7. Author(s) D. C. Engelbart and Staff of Augmentation Research Center		8. Performing Organization Report No. SRI Project 7079
9. Performing Organization Name and Address Stanford Research Institute Menlo Park, California 94025		10. Work Unit No.
		11. Contract or Grant No. NAS1-7897
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D. C. 20546		13. Type of Report and Period Covered Contractor Report
		14. Sponsoring Agency Code

15. Supplementary Notes

This report covers the extension of research previously reported in NASA CR-1270.

16. Abstract

This report covers a two-year project, at the eleventh year of a growing, multiproject program that is exploring the value of computer aids to augmenting human intellectual capability.

Outlined briefly are the background and the "bootstrapping" nature of the program, its resources, and the activities it has undertaken in pursuit of its goals.

User experience in applying our augmentation tools and techniques to various normal working tasks within our Center is described so as to convey a subjective impression of what it is like to work in an augmented environment.

It is concluded that working-support, computer-aid systems for augmenting individuals and teams, of the general sort we have been experimenting with, are undoubtedly going to be widely developed and used.

A very special role in this development is seen for multi-access computer networks: they will become special marketplaces where a new kind of competitive evolution will take place, not only in hardware, software, and special services as "bought" from a "utility," but also in roles, skills, working methods, and employment dynamics for the intellectual workers at the terminals.

17. Key Words (Suggested by Author(s)) Intellect augmentation; Man-Machine communication; On-line interaction; ARPA network; Text manipulation; Computer displays	18. Distribution Statement Unclassified - Unlimited

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 201	22. Price* $3.00

*For sale by the National Technical Information Service, Springfield, Virginia 22151

1. Man machine systems
2. On line data processing
3. Computer systems programs

24 mar 72

PREFACE

The research described in this report represents conceptual, design, and development work by a large number of people; the program has been active as a coordinated team effort since 1963. The research reported here was a cooperative team effort involving the entire ARC staff. The following is an alphabetical listing of the current ARC staff:

Geoffrey H. Ball, Walter L. Bass, Vernon R. Baughman, Mary G. Caldwell, Roberta A. Carillon, David Casseres, Mary S. Church, William S. Duvall, Douglas C. Engelbart, William K. English, Ann R. Geoffrion, Martin L. Hardy, Jared M. Harris, J. David Hopper, Charles H. Irby, L. Stephen Leonard, John T. Melvin, N. Dean Meyer, James C. Norton, Bruce L. Parsley, William H. Paxton, Jake Ratliff, Barbara E. Row, Martha E. Trundy, Edward K. Van de Riet, John M. Yarborough.

The following former ARC staff members also contributed to the research:

Donald I. Andrews, Roger D. Bates, David A. Evans, Stephen R. Levine, Stephen H. Paavola, Helen H. Prince, Jons F. Rulifson, Elmer B. Shapiro, F. K. Tomlin.

CONTENTS

ILLUSTRATIONS

I INTRODUCTION

A. General

The Augmentation Research Center (ARC) is a community of researchers, supported by several contract sponsors. It is dedicated to exploring the possibilities for augmenting the intellectual activities of people working in complex problem-solving situations.

By "augmentation" we mean increasing the capability of a person or organization to approach complex situations and identify problems present there, to gain comprehension of the nature and context of these problems, and to derive solutions satisfying given constraints.

Such increased capability may be reflected in any of the following ways: faster and better comprehension, the possibility of gaining a useful degree of comprehension in situations that were previously too complex, faster and better solutions, and the possibility of finding solutions to problems that previously seemed insoluble.

B. Research Approach and Strategy

ARC's orientation is based on the fact that modern man is confronted with problems of increasing complexity and urgency, and the assumption that in attacking these problems the best long-term payoffs will more likely come through the development of more powerful problem-solving tools of a general nature than through direct, piecemeal attacks on specific problems of immediate urgency.

This orientation was spelled out in 1962 in a planning document (Ref. 2) that provided the conceptual framework under which ARC has been evolving. (Note: reference numbers are not consecutive in this report, but refer to the chronological bibliography.) Our approach to augmentation research has two essential aspects: the externalization of intellectual structures in symbolic form, making use of highly interactive computer systems, and the application of a bootstrapping strategy in the research program for developing augmentation systems.

The purpose of externalization via computer systems is to make it possible for people to work with intellectual structures (such as computer programs or highly interconnected bodies of textual information) of much greater size and complexity than can be effectually handled with traditional techniques. The aim in designing the computer systems has been to provide people with increasingly flexible and powerful ways of using structures of symbols to represent intellectual structures and of viewing and manipulating these symbolic structures.

The bootstrapping strategy has been used in order to ensure the tightest possible feedback in the process of developing augmentation aids and in order to catalyze the research program by bringing higher and higher levels of augmentation to the researchers themselves. All augmentation aids designed by ARC are intended for actual, practical use in ARC itself, and once implemented they are, for the most part, used heavily on a day-to-day basis.

C. Principal Concerns During the Contract Period

During the period covered by this contract, three ongoing processes reflected the principal concerns of ARC: the following sections describe these processes.

1. NLS Development

At the end of the previous contract period (see Ref. 12), the multi-user On-Line System, NLS (which had been developed from previous single-user systems), had been carried from the early design stages through almost complete implementation. During the subsequent two years, NLS has developed into an experimental operational system that is now in heavy routine use by the ARC staff.

ARC resources have been heavily committed to increasing the operational speed and reliability of the many components of the system, improving the interaction of these components within the total system, and improving and extending the user features of the system.

2. Evolution of Goals

One usage trend that has become evident during this period is a tendency for staff members who are working on a common problem to gather around an NLS console so as to have on-line access as a group to the working files that they are using in common. Even when working individually, group members frequently sit at neighboring consoles so as to be able to converse about related tasks in progress.

This trend has been reflected, through bootstrapping, as an evolution of ARC goals from the augmentation of individuals to the augmentation of task-oriented teams.

Much thought has been put into deciding which areas of research seem to offer the greatest promise for improving the ability of augmented individuals to cooperate on a common problem, and during the next few years one of the major activities at ARC will be the development of team-augmentation facilities and techniques. Some of the developments currently being considered are discussed in Section V.

3. ARPA Computer Network Participation

During the contract period we completed plans for connecting ARC's computer facility into the experimental ARPA Network, which eventually will link the computer facilities of about 14 computer-science research centers. We feel that the Network is a significant step in the evolution of augmentation technology, and we are working with the other Network participants in order to ensure the success of this experiment.

We have agreed to provide a Network Information Center (NIC) for the Network, and over the past two years we have been committing an increasing portion of our resources to the planning and development of NIC services. We expect that the success of the Network experiment will be significantly affected by the quality of services that the NIC can provide.

D. Structure of This Report

The remainder of this report is divided into four major sections and a supporting appendix. We have attempted to describe what we have done and learned during the past two years, without elaborating either on the technical details of our hardware/software system or on the past history of our research program.

These latter topics are covered in the references cited in the Bibliography of this report. In particular, the following may be of interest:

The 1962 planning document (Ref. 2) provides the conceptual background for our approach to augmentation research,

Section I
INTRODUCTION

The 1968 FJCC paper (Ref. 14) gives an overview of our Augmentation Research Center,

The 1970 final report for another sponsor (Ref. 17) discusses many technical details of our system implementation.

Section II of this report describes the resources -- human, organizational, hardware, and software -- that ARC has available for its research and discusses ARC's ongoing activities. This material is supplemented by Appendix A, which briefly describes several of the primary elements of our augmentation system.

Section III is a collection of subjective descriptions of our experience as augmented workers in several of ARC's central activities. Two of the descriptions (for software engineering and management research) are supported by illustrated scenarios showing actual work in progress.

Section IV contains a description of the ARPA Network from a user's standpoint and a discussion of some probable, early uses of the Network. We point out some of the services that Network participants will need in order to make effective use of the Network and indicate how we are attempting to satisfy those needs through the Network Information Center.

In Section V we discuss some of the conclusions drawn from our research and indicate the directions in which we plan to develop our research in the future.

II RESOURCES AND ACTIVITIES

A. Introduction

In this section we describe the resources that have been available to us during the past two years and the activities we have undertaken to apply these resources to the pursuit of our augmentation goals.

We use the term "resources" in the broad sense to include all the assets we possess as a team, including the following:

(1) Hardware facilities

(2) System software

(3) User systems

(4) Personnel (human resources)

(5) Organizational relationships.

Our major organizational relationships are to SRI and to the ARPA Network.

The Augmentation Research Center operates at the group level within the SRI Information Sciences Laboratory and, consequently, has access to the extensive and varied resources available through a large, diversified research organization such as SRI.

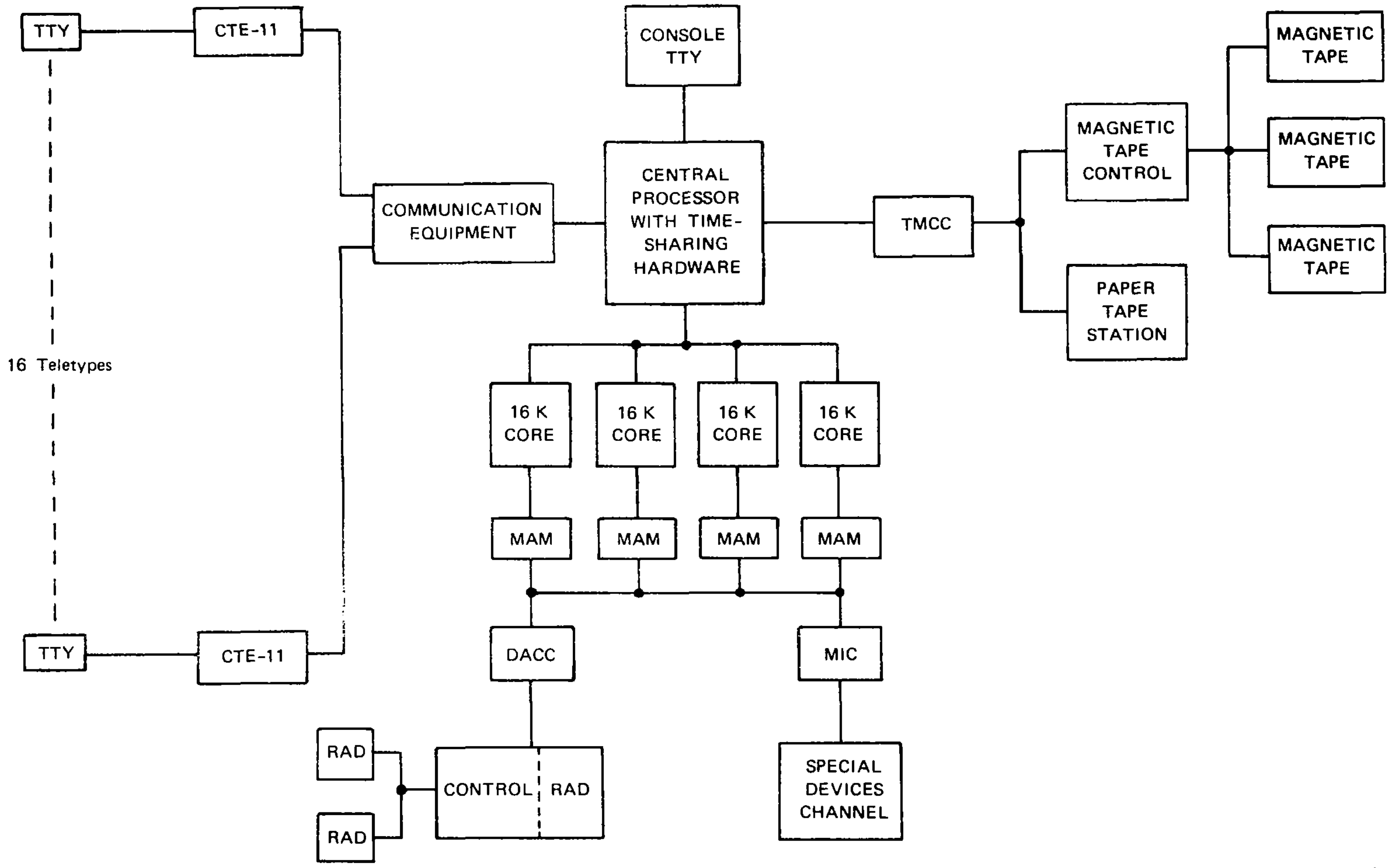
As an initial participant in the ARPA Network, we will have access to all services that eventually become available through the Network. The implications of this are explored in Sections IV and V of this report.

Our other resources are discussed in some detail below, along with brief descriptions of our major on-going activities.

B. Resources

1. The Computer Facility

At the beginning of the contract period the current ARC computer facility was almost complete, and the basic configuration has been relatively stable over the past two years. This Section briefly describes this facility (diagrammed in Figures II-1 and II-2) and discusses some of the changes and additions made during the contract period.



TA-5919-3R

FIGURE II-1 **XDS940** COMPUTER FACILITY

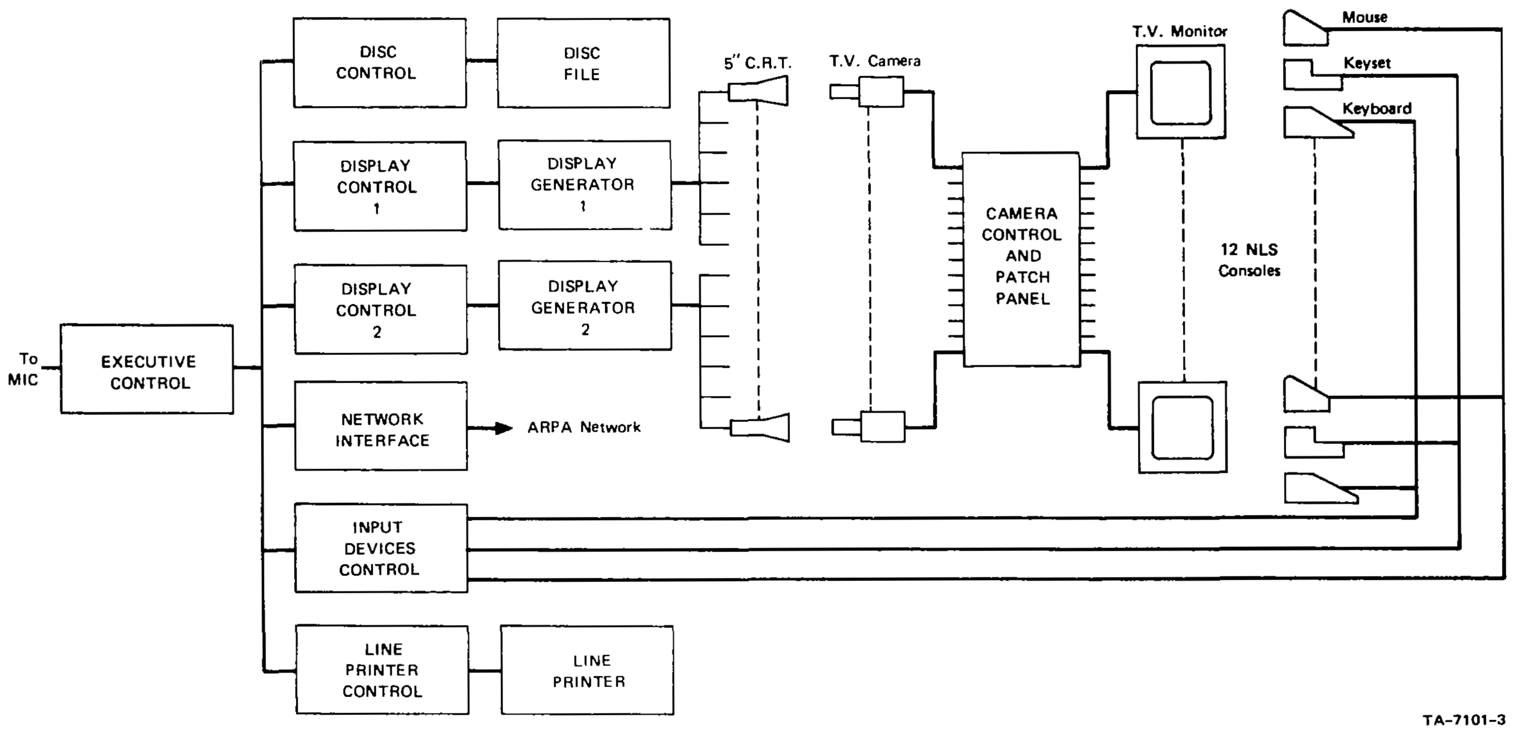


FIGURE II-2 SPECIAL DEVICES CHANNEL

The most siginificant additions have been the ARPA Network interface and an external core system.

A more complete description of the facility is contained in Refs. 11 and 17.

a. The Leased Computer

Figure II-1 is a block diagram of the facility as leased from XDS.

A central processor with timesharing hardware operates from a 64K memory in four banks with 24-bit words and a cycle time of 1.8 microseconds.

On channels sharing memory access with the CPU are three magnetic-tape drives, a paper-tape station, and communication equipment for sixteen typewriter terminals.

A second memory buss provides direct access to memory for the RADs (Rapid Access Devices -- e.g., drums) and the non-XDS portion of the facility, designated "Special Devices Channel" in Figure II-1.

There are three drums on the system, operating from a common controller and accessing memory through an XDS device called a Direct Access Commmunications Channel (DACC). Each drum has a capacity of 500,000 24-bit words, a transfer rate of 120,000 words per second, and an average latency of 17 milliseconds.

b. Special Devices Channel

Figure II-2 is a block diagram of the portion of the facility that has been put together by ARC. The following sections describe the major components.

(1) Executive Control

The executive control provides an interface to the 940 through the Memory Interface Connection (MIC). It acts as a multiplexer that allows asynchronous access to core by any of the six devices connected to it.

It includes extensive debugging and monitoring aids, allowing the monitoring of data and addresses for any selected device and permitting "off-line" operation of any of the devices.

(2) Disc File System

The disc file system was put in operation in August, 1968. It consists of a Bryant Model 4061 disc file and associated controller. The system has a capacity of 32 million words, an average access time of 185 milliseconds, and a data transfer rate of 43,000 words per second.

The disc controller was designed and built by Bryant to interface with the executive control. Specifications for the controller were developed jointly by Bryant, Project GENIE at UC Berkeley, and ARC.

(3) Display System

The display system consists of two identical subsystems, each with a display controller, a display generator, and six high-resolution 5-inch CRTs. A closed-circuit television system carries display images from the CRTs to television monitors in the working area.

The display controllers were designed and built by ARC. They access and process "command tables" that are resident in 940 core.

A command is roughly associated with a user and points to a "display list" in the user's core space. The display list in turn points to buffers containing actual display instructions (commands to the display generator to produce images).

The display controller handles all core accessing, including memory mapping for the user's core space. It passes the display instructions along to the display generator.

The display generators and CRTs were purchased from Tasker Instruments to ARC's specifications. They have general character and vector capabilities.

Presentations for each of the six CRTs are generated sequentially, and unblank signals from the display controllers select one or more of the CRTs at a given time.

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

(4) Input Device Control

In addition to the television monitor, each console has a keyboard, a binary keyset, and a mouse. Appendix A describes the use of these devices.

The state of these input devices is read by the input device controller at a preset interval (about 30 milliseconds) and written into a fixed table in 940 core.

Bits are added to information from the keyboards, keysets, and mouse switches to indicate when a new character has been received or when a switch has changed state during the sample period. A new character or switch change causes an interrupt to be issued at the end of the sample period.

Mouse coordinates are digitized by an A/D converter and formatted by the input device controller as beam-position instructions to the display generator. A user program may include the mouse coordinates, as written by the input device controller, as part of a display list. This allows the mouse position to be continually displayed with no attention from the CPU and no perceptible delay to the user.

(5) Line Printer

The line printer is a 96-character drum printer leased from Data Products Corporation (Model M600-11A). With the 96 characters, printing speed is 340 lines per minute.

The line printer controller processes print buffers of arbitrary length that have been set up in core by a controlling program (single-line buffers are normally used).

(6) Network Interface

The network interface provides communication between the 940 and an Interface Message Processor (IMP) on the ARPA Computer Network. The interface operates from message buffers in 940 core. Messages to the Network are read by the interface from these buffers and transmitted to the IMP. Similarly, messages received from the IMP are written into buffer space in 940 core. Instructions from the 940 enable the system for receiving messages and control the sending of messages. A "linked-buffer" scheme permits flexible memory allocation.

The interface message processor and its communications protocol are discussed in detail in Ref. 18.

c. Typewriter Terminals

At the begining of the project the only terminals in use (other than the display consoles) were Model 33 Teletypes. Since then we have been experimenting with different types of terminals, looking for improvements in speed, print quality, and portability.

The newer terminals now in use, and some of their features, are briefly described in this section.

It should be remembered that the only terminals we have considered are those with upper- and lower-case print and full-duplex operation.

(1) The Model 37 Teletype was one of the first terminals added, and three are now in use.

It operates at 15 characters per second (as compared to 10 characters per second for the Model 33) and has excellent print quality and reliability.

It has a high noise level and is large and heavy.

(2) We have four GE Terminet-300 terminals in use.

These have switch-selectable rates of 10, 15, and 30 characters per second. They have a chain printing mechanism that is relatively quiet and have good print quality when in adjustment.

These terminals seem to require more frequent maintenance than any others in use, but we have early models and they may improve in later production.

(3) The best terminal we have found for portability is the Execuport, manufactured by Computer Transceiver Systems.

These terminals are used by our staff for remote operations, usually in a staff member's own home, and come in a portable case complete with acoustic coupler and weighing only 26 pounds. They have switch-selectable rates of 10, 15, and 30 characters per second.

The thermal print mechanism is very quiet in operation but print quality is poor; characters are made from a 5 x 7 dot matrix.

d. Modifications in Progress

Two modifications to the facility that will provide significant improvement in service are now being implemented. These are an external core system and faster drums. In addition, an accurate clock system is being added.

(1) External Core System

An external core system has been completed and will be integrated into the facility in the near future. It currently consists of a single 32,000-word bank with access switching to allow access by up to eight devices. Provisions are included in the design for expansion to 16 devices and two core banks of 64,000 words each. The core cycle time is 1.5 microseconds and the word length is 24 bits.

The primary purpose of this core system is to provide storage for display buffers, the network interface, and the line printer. These are the devices that need constant buffers for relatively long periods of time and therefore require "frozen pages" when operating from 940 core -- a significant factor in limiting system response, since thay take up space that could otherwise be used for swapping.

(2) Faster Drums

From system response studies (Ref. 17) it is apparent that a primary factor in response is the swapping bandwidth. To improve response (and add more users), we are in the process of replacing the XDS drums with Univac FH-432 drums.

These drums rotate at 7200 RPM, giving a transfer rate of 365,000 words per second (as compared to 120,000 for the present drums) and an average access time of about 4 milliseconds.

In addition, we are formatting the new drums in a way that will allow a page transfer to begin at any position on the drum. Since a 2048-word page fills two-thirds of a band, this will give an average page transfer time of about 8 milliseconds.

The interface for the drums will be designed and built by ARC. It will connect to the 940 through a second Memory Interface Connection (MIC), replacing the current RAD-DACC combination shown in Figure II-1.

(3) Clock System

An accurate clock system is being added to assist us in system measurements.

This clock system provides two types of time information -- absolute and relative -- that are written into fixed locations in 940 core at regular intervals. The long-term drift on the clock will be less than 1 second in 250 days.

2. Software Systems

The central focus of software activity at ARC is the evolutionary development of the On-Line System (NLS). This takes place within a rich environment of software systems, many of which were created specifically to aid in its development. The following is a brief description of the major systems.

a. The Timesharing System (TSS)

Most basic to the operation of NLS, as well as all our other software systems, is the timesharing system (TSS) running on the XDS940.

TSS was originally developed by Project GENIE at UC Berkeley, but responsibility for maintenance of the ARC version currently lies with the Center itself. NLS runs as a subsystem under TSS, and users also have access to other subsystems such as the NARP assembler, KDF file storage system, and DDT debugging system.

The support of new hardware and improved response to the NLS user are the major areas of improvement in TSS over the past two years.

b. Software Architecture

(1) Introduction

The development of NLS has been facilitated greatly through the use of a powerful complement of languages and compilers, most of which were designed at ARC.

The languages used range in generality from the NARP assembly language through a collection of special-purpose languages (SPLs) unique to NLS implementation. Their major features are discussed briefly below.

(2) NARP

A few parts of NLS can be most conveniently coded in assembly language (e.g., the data page and the display-buffers), and for these the NARP assembly language is used.

Also, for historical reasons, the timesharing system (TSS) and most of its subsystems (e.g., KDF and DDT) are coded in NARP.

(3) MOL940

MOL940 (or simply MOL) is a machine-oriented language for the XDS940 and was created by ARC to aid in the programming of NLS.

MOL combines the flexibility of assembly language with the algorithmic clarity of higher-level procedure-oriented languages. Much of NLS is coded in MOL.

During the contract period MOL has been substantially rewritten to improve its performance and provide new programming features. The current MOL compiler was produced using the new version of Tree Meta (described below); consequently, the MOL compiler now generates binary machine code directly rather than producing assembly-language code as an intermediary.

(4) Special-Purpose Languages (SPLs)

Many of the higher-level operations of NLS are carried out by programs written in one of a set of special-purpose languages (SPLs). Each of these languages is translated into machine code by a custom-made compiler produced with the Tree Meta system.

Each SPL represents an attempt to formalize a particular function of NLS, aiming at a syntax appropriate to the data base and operations required for NLS, while at the same time embodying the potential and peculiarities of the XDS940 computer.

The four SPLs currently in use are the input-feedback language, the structure-manipulation language, the content-analysis language, and the string-construction language.

Detailed descriptions of the SPLs will be found in Appendix D of Ref. 17.

Although extensive changes in the SPLs are planned for the near future, no basic conceptual changes were made during the contract period.

(5) Tree Meta

Tree Meta is a compiler-compiler developed at ARC; it is used to produce compilers for MOL, for all the special-purpose languages, and for itself as well.

Section IV-C-2 of Ref. 17 contains a brief overview of the current version of Tree Meta; a more detailed description is in preparation for release as a separate report.

c. NUTILITY

To aid our software engineers in maintaining the approximately 150 symbolic and binary files that make up NLS, a special subsystem called NUTILITY has been developed.

This subsystem can archive or retrieve any of these files, compile or produce listings for any of the source-code files, and load the entire NLS system or any part of it, requiring programmer intervention only in the event of errors that NUTILITY cannot handle itself.

3. User Systems

This section briefly describes our principal on-line user subsystems; more complete descriptions are contained in Appendix A.

a. On-Line System (NLS)

The On-Line System, NLS, as currently implemented, is a highly sophisticated system oriented toward the on-line construction, editing, and viewing of files.

NLS is a subsystem of the timesharing system described above. Its size is currently about thirty thousand machine instructions, of which about half make up the most frequently used portions. The source languages used are NARP, MOL940, and the SPLs.

A complete description of NLS, including program documentation, is in Ref. 17.

b. Typewriter-Oriented Documentation-Aid System (TODAS)

In response to growing pressures to make access to our on-line files available to Network participants, as well as to members of our own staff working at remote locations, we developed a new subsystem called TODAS (Typewriter-Oriented Documentation-Aid System).

TODAS uses the same structure for working files as NLS and provides most of the same editing and view-specification capabilities, along with a few special features of its own. Thus, users can freely move from one type of terminal to another without losing access to any of their on-line working records.

TODAS offers our on-line users the possibility of trading off speed of operation for freedom of location. It also makes it possible for us to increase the number of on-line users that can be serviced simultaneously, since a TODAS user places less heavy demands on our computational resources than does an NLS user.

c. Output Processor (PASS4)

The Output Processor (also called "PASS4" for historical reasons) is a program for formatting on-line text files for output through a variety of devices including line printers, paper-tape-driven typewriters, and computer output to microform (COM) devices.

4. Personnel

The final, critical element of ARC's resources is its staff of professional researchers and support personnel. The experimental nature of our work requires high-caliber individuals who can function effectively in our team, learn quickly to work with our systems and methodologies, and progress creatively along with the rest of the team and its products.

Our staff members need to be able to adapt to the rapidly changing ARC environment and to persevere through very disturbing periods of system service level fluctuation.

Our selection of people for specific tasks is influenced not only by skills already developed, but also by potentials for further development.

Effective utilization of our human resources requires a careful balance between selecting those people most likely to get critical tasks done well and those whose developed capabilities will be strained by the tasks they undertake (and hopefully extended in the process).

In ARC's earlier years our primary need was for individuals skilled in tool building, with only secondary importance assigned to methodology development skills. Thus, most of our earlier researchers were system-oriented software and hardware engineers.

As the systems developed became more generally useful, providing aids and service levels beyond the system engineers' basic needs, we entered a maturing phase of facing the challenge to use the system for a broader range of tasks.

People have recently been added whose interests focus on the study of the user system itself, on use of the system for management purposes, and on its role in supporting the Network Information Center.

These people have brought different needs and perspectives into the group, directly aiding the design of many system improvements. Interaction at the day-to-day system development level has provided a rich learning experience for most people, particularly in technical areas they might not otherwise have learned much about.

This diffusion of knowledge in areas such as system design, system building, system analysis, and management adds new perspective to each person's approach to problems in his own areas of specialization.

Section II
RESOURCES AND ACTIVITIES

At present we have a full-time staff of 25, constituted as follows:

Professional		18
Supervisory	1	
Software	11	
Hardware	4	
Other	2	
Non-professional		7
Technical	3	
Clerical	4	

C. Activities

1. Introduction

This section outlines the nature and purposes of the major activities carried on during the past two years of our project. It was the pursuit of these activities that produced the developments, experiences, and conclusions discussed in the other sections of this report.

The greatest portion of our resources were allocated to the basic ARC bootstrapping pursuit, which consists of these major activities:

(1) Development and operation of our service system

(2) Development of methodologies for harnessing the user features offered by the service system

(3) Application of these augmentation tools and techniques to the pursuit of all our activities

(4) Assessment of our overall augmentation system to guide its further evolution.

Two other major activities received considerable attention and resources:

(5) Connection of our Center into the ARPA Network

(6) Development of an Information Center for the Network.

2. Development and Operation of Our Service System

The coordinated development of hardware and software aspects of our augmentation system, which has been under way since 1963, was continued with particular emphasis on activities aimed at preparing us for our role in the ARPA Network.

We have committed a substantial portion of our resources to improving the reliability and capacity of the service system and will continue to do so, a major milestone being our planned transfer to a PDP-10 computer this fall.

Another major effort has been preparation for expanded remote-worker capabilities, both to facilitate use of our system by other Network participants and to continue our efforts to increase the flexibility of working arrangements for members of our own research community.

There is already one member of our team (a software engineer) who works remotely from his home in Sonoma County (about 100 miles from our Center), and we have plans to extend this capability to other members as conditions permit.

3. Development of Methodologies for Harnessing the User Features Offered by the Service System

It has long been part of our plans to apply toward this activity resources comparable to those for the service system development activity, but unforeseen difficulties in the latter area have forced us to divert more of our energies to it than was originally envisioned.

However, many methodological developments have been evolving more or less naturally as individuals adapt the features of the service system to their particular needs. The strongest such evolution has taken place within the area of software engineering, which is the focus of consistently heavy activity by many of our people.

In one specific area -- management systems -- we have engaged in an explicit effort to develop improved methodology. This has been funded separately by a contract with the Rome Air Development Center, which has enabled us to apply one or two full-time people toward this Management Systems Research activity for the past two years.

4. Application of Augmentation Tools and Techniques to the Pursuit of All Our Activities

Our augmentation system is in daily use by most members of our staff. Detailed discussions of several examples of this use are given in Section III.

5. Assessment of Our Overall Augmentation System to Guide Its Further Evolution

One of the basic requirements for carrying out a bootstrapping operation is to maintain constant awareness of the status of that operation so as to be able to make rational decisions about what developments should be undertaken next.

We have made some attempts to quantify this process through the taking of system performance measurements and the simulation of various existing and proposed system configurations. However, most of our work in this area still has a very intuitive character, and we have been hampered in our efforts to improve on this by the necessity for committing so much of our resources to basic service-system development activities.

6. Connection of our Center into the ARPA Network

To make it possible for ARC to participate in the ARPA Network (see Section IV), we had to design, build, and install a device to interface our computer facility to a Network Interface Message Processor (IMP). This Network interface is described in Section II-B-1-f.

7. Development of a Network Information Center (NIC)

In preparing to fill our role as Information Center for the ARPA Network, we have been involved in developing the tools and methodologies that we will need in order to provide an increasingly on-line, special-purpose library for a widely distributed clientele. The high technological base of this Network experiment demands that we try to harness as much as possible of this technology for the Network Information Center so that this service will be likely to enrich the whole experiment.

In addition to investing our resources in special-purpose capabilities for the NIC, we have felt compelled to increase our investment of resources in basic service-system development so as to be able to supply a better level of service to Network participants.

We had to get our system architecture, our compiler languages, our documentation, and our hardware configuration into a state where we would be able to expand our user-carrying capacity rapidly. This led to a number of major efforts:

(1) A series of significant architectural and compiler changes within our software systems

(2) Several rather intensive trial designs for alternate expanded-capacity system configurations

(3) A concerted effort in developing computer aids for analyzing performance characteristics of proposed systems.

These efforts culminated in a decision to move our systems onto a PDP-10 computer late this fall, and we are currently involved in the leasing, purchasing, contracting, engineering, fabricating, and programming activities necessary to accomplish this transfer.

III APPLICATIONS AND EXPERIENCE

A. Introduction

This section consists of relatively informal accounts describing the application of our augmentation systems to several areas of our own work while attempting to convey some feeling for what it is like to work within an augmented total system.

Section III-B was written by Charles H. Irby, an ARC software engineer. It describes in considerable detail the ways in which an augmented software engineer can exploit our computer systems to help him in the continuing development of these same systems.

In addition, Section III-B serves as an introduction to our On-Line System, NLS. It is systematically illustrated with photographs of an NLS display, showing actual sequences of displays that the software engineer would see as he worked. (Readers who are not familiar with NLS will find a description of its user features in Appendix A, which includes a glossary of special NLS terminology.)

Section III-C deals with "augmented management" and was written by James C. Norton, who heads our Management Systems Research Activity and is also involved with carrying out much of our operational project management.

This section, illustrated with display photographs like those of Section III-B, shows how some of the most sophisticated features of NLS can be put to use in a field outside of software engineering. Specific applications to project cost accounting, cost estimation, work planning, and other areas are covered.

The discussion develops a total-system concept of the meanings of "management" and "management augmentation" and reveals how augmentation interacts with various aspects of ARC.

Section III-D describes our use of augmentation systems in writing, editing, and producing reports. The author of this section, David Casseres, is ARC's technical writer and has been centrally involved in ARC report writing since the inception of this contract.

Here we see the use of augmentation systems in a field where it is difficult to apply them, because the problems of technical writing are so many and so dependent on individuals.

It appears that although the benefits of augmentation in writing, editing, and producing reports are considerable, the most significant benefits come from the use of a very close team-collaboration method, which would be virtually impossible without augmentation.

Section III-E was written by Douglas C. Engelbart and covers the augmentation of communication between a speaker and his audience.

This kind of communication, which we have used for explaining aspects of our work to audiences ranging from a single person to thousands, is potentially one of the most exciting areas of application for augmentation technology.

B. The Augmented Software Engineer

One of the central objects of our augmentation research has been to develop special tools and working methods for augmenting the design and implementation of our system software.

Section III-B-1 below outlines the general augmentation needs of a software engineer and the aids provided at ARC to meet these needs.

Section III-B-2 describes the ways in which our software engineers actually use the systems we have been developing in their daily work.

We pay particular attention to the use of the system-guide file, SYSGD, which is central to the software-engineering augmentation system.

The use of SYSGD is illustrated with photographs of an ARC interactive display, showing the views seen by the software engineer as he works to implement a new NLS feature, using SYSGD as a reference.

1. General Considerations

The augmented software engineer needs the following minimal capabilities:

(1) The ability to rapidly access, understand, and manipulate the source code

(2) The ability to easily compile the source code

(3) Easy access to appropriate loading and debugging capabilities.

The NLS, TODAS, NUTILTY, DDT, and disc archive subsystems together with the SYSGD source-code file directory provide the ARC software engineers with these capabilities.

To the ARC software engineers, the aspects of NLS that are perhaps of most importance are the ability to find a particular place in the code or documentation quickly and then to change it easily. Our code and documentation files are normal NLS text files, and the languages that we use have been specially developed to be compatible with the NLS system.

These languages are all string-based, and translation is independent of the hierarchical statement structure. This permits the programmer to use structural conventions to make his source-code text easier to understand and manipulate.

They also allow NLS links to be used for identifiers, thus permitting one to use the "jump" commands to follow subroutine calls in the source code. In addition, level clipping, line truncation, and so forth may be used to control the depth of detail of the code and associated documentation that one sees. These are very important factors in quickly finding a specific location (or series of locations) in a file.

Content-analysis filtering is often used to locate references to variables or subroutines or to locate a particular piece of code (for example, code containing a syntax error reported by the compiler). The content analyzer is also used to locate changes that have been made to a file by a specific programmer or during a specified period of time by scanning the automatically maintained "statement signatures." These signatures contain the date of most recent modification for each statement and the initials of the user who made the modification and may also be used without the content analyzer to see who wrote (or last modified) each statement of code or documentation and when he did it.

When viewing a segment of a file, the software engineer must be able to easily modify the text of the file. The extensive editing power of NLS makes this possible.

The ability to easily effect changes to various textual and structural entities and to make multiple string substitutions over structural entities while controlling the selection of statements for the substitution via level clipping, content analysis, keyword reordering, and so forth gives one great flexibility and power while editing.

In addition to being able to modify the code and associated documentation quickly, the software engineer may compile (or cross-reference) his code files directly from NLS. Once again, he may use the statement-selection mechanisms of NLS to control what the compiler gets as input. This might be done, for example, to limit the compilation to just one branch of a file (one specified statement and all its substatements), thus making it possible to have in a single file source code written in several different languages.

We also have available as another subsystem the Project GENIE on-line loader and symbolic debugger, DDT, for testing and debugging the results of our changes to the code files. Although DDT works at the assembly-language level, it is a powerful debugger with such features as multiple breakpoints, symbolic addressing, single-instruction execution, conditional breakpoints, and so forth.

To aid the software engineers in maintaining their approximately one-hundred-fifty or so symbolic and binary files, a special subsystem called NUTILITY was developed. This subsystem is able to archive, or retrieve from archive, any of the files, to compile or produce listings for any of the source files, and to load the entire system, requiring human intervention only in case of errors.

Central to our collection of source-code files is an on-line file called SYSGD. It contains the following:

(1) A schematic diagram of the overlay structure of the NLS system. The overlays represent (as far as practicable) functional areas of the system, e.g., parameter specification, structure manipulation, and so forth.

(2) Links to other files that describe the architecture and logic of the system.

(3) An overlay index listing all overlays, with the following information about each overlay:

(a) A link to the file containing the source code for the overlay.

(b) Information about where it runs in core, how large it is, and where it is in the archive.

(c) A list of all the procedures in the overlay, with one-line functional descriptions and with links to the code and documentation in the source-code file.

(4) A procedure index with a categorical listing of procedures according to function. The NLS keyword system can be used on this index to retrieve a list of procedures (with links to the source code and documentation) when a set of categories has been selected.

(5) A section where one may document bugs found in the system.

(6) A section where one may record ideas for improving the system.

Because of present file-space problems, it is not possible to keep the SYSGD file, the associated documentation files, and the source-code files on line at all times (they are generally kept in a disc archive). This situation renders the SYSGD file less useful than it would be if the files were always readily accessible.

The combination of the capabilities discussed above gives the ARC software engineer the ability to easily manipulate the NLS system to fit the needs of ARC experimentation, thus greatly increasing his own abilities as a software engineer. These capabilities would be greatly enhanced by the addition to NLS of an interactive incremental language. This would eliminate the time now spent on compiling parts of the entire system and loading them in order to test changes made in the source code of the system, and would allow one to debug at the source-code level.

2. The Augmented Software Engineer in Action

a. Introduction

This section is in the form of a scenario, describing how a software engineer works to implement a new NLS feature by manipulating files of text (both documentation and actual code). The reader is encouraged to pay special attention to the following:

(1) The ease with which one can move within and among files

(2) The languages that have been developed for particular purposes and the way in which they fit into the NLS framework

(3) The ease with which one can locate desired code and modify it

(4) The design of the NLS system-guide file and the ways in which it can help the software engineer

(5) The use of the NUTILITY subsystem to automatically archive, compile, print, and cross-reference files as well as to load new versions of the system

(6) The use of the DDT debugging program to test and debug additions to the system at the machine-language level.

Please note also that one moves within and among files by means of "Jump" commands. At certain times when jumping, the user has an opportunity to change the parameters ("VIEWSPECS") that control the selection of information to be displayed on the screen. For example, in the jump that occurs between Figures III-3 and III-4 below, a "branch-only" parameter is set and the level-clipping and line-truncation parameters are set to "ALL".

One may jump to a selected statement, to a statement that is structurally related to the selected statement, to a statement having a given name, or to a statement specified by a "link" (a textual construct that symbolically points to a particular statement within the file or within another file).

In addition, stacks of display-start statement identifiers and VIEWSPECS are maintained for the last sevaral locations viewed within the file and for the last several files accessed, and jumps may be made relative to these stacks so as to restore a previous view.

b. Notes on Illustrations

In describing the display views and the processes they reflect, we assume that the reader is familiar with the user features of NLS to the extent that they are explained in Appendix A.

Appendix A also contains a glossary of special NLS terminology.

In examining the photographs, note that the name of an NLS command appears at the top of the display; this is the command currently specified by the user.

Immediately to the left of this "command feedback line" is a small block of text called the "VIEWSPEC area," where VIEWSPEC parameters are displayed on two lines.

When the VIEWSPECS are displayed in small characters, they are currently in effect; when they are enlarged, it means that the user has just set them and they will go into effect when the display is next re-created.

The upper line shows the current level-clipping and line-truncation parameters (see Appendix A for explanation). The lower line in the VIEWSPEC area shows code letters indicating the status of various display features that are controlled by VIEWSPECS.

c. Scenario and Illustrations

Note: The illustrations for this scenario are grouped together, beginning on page 35.

We log into the timesharing system, increase our drum allocation, and enter the NNLS (New NLS) subsystem giving a set of initials and a user name (see Figure III-1). The NLS display comes up (see Figure III-2), and we load the NLS system-guide file, SYSGD, with level and line truncation set to one and with blank lines displayed between statements (see Figure III-3).

The major sections of the SYSGD file (see Figure III-3) are as follows:

(1) Program Overlay Structure

A diagram showing the overlay structure used in the current implementation of NLS (see Figure III-7 below).

(2) Global Documentation

Links to other files that describe the file structure, design philosophy, system architecture, and commonly used terminology of the NLS system (see Figures III-4 and III-5).

(3) Overlay Index

A list of all of the overlays in the system (each overlay represents a functional area of the system) with pertinent data about each overlay (see Figures III-8, III-9, III-24, III-25, and III-26 below).

(4) Categories for Procedures

Categorical list of procedures, by function, to be used with the keyword retrieval system.

(5) List of Known Bugs

A place where bugs may be recorded.

(6) Map of Symbols

A map of symbols to be used with the DDT debugging system.

(7) Thoughts for Improvements

A place where ideas about the further development of NLS, TODAS, and NUTILITY may be recorded (see Figure III-6).

The initials of the creator (or last modifier) and the date and time of creation (or modification) are stored internally as the "signature" of each statement. Since statement signatures are available for display upon request, one can easily determine who made or last commented upon a given statement (see Figure III-6).

The SYSGD file is basically a directory file with a sufficient amount of descriptive material so that one can use the various retrieval tools of NLS to locate any desired procedure or documentation.

To illustrate how the SYSGD file is used, we go through the steps necessary for a programmer to implement a new command in NLS. The new command will be called "transpose" and will interchange two textual entities. The command language for using this new command will be modeled after that of the "move" command.

Let us begin (Figure III-7) with the diagram of the NLS overlay structure. From the global documentation of the system we know that the main-control (mnctrl) overlay consists of the code that implements the command language for NLS commands. We therefore use the "Jump to Vectorlabel" command to go to the branch within the overlay index branch that contains information about the mnctrl overlay.

This information includes the following (see Figure III-8):

(1) A link to the source-code file for this overlay

(2) A list of all of the procedures within the overlay

(3) Brief documentation as to the functions of each procedure

(4) Links to each procedure's source code. See

Figure III-9; whereas Figure III-8 shows only the first line of each statement, Figure III-9 shows all lines -- otherwise the two displays are the same.

If we take the link to the "wc" procedure (Figure III-10), we see the top-level code for the NLS command language, written in one of the Special-Purpose Languages (see Section II-B-1) designed for this use. Jumping to the "move" command code (Figure III-11) and increasing the level-clipping parameter by one, we see the top-level command-language code for the "move" command. Since this is to serve as a model for our proposed "transpose" command, we will copy the branch and make the necessary changes to it.

The necessary changes are substituting the string "Transpose" for the string "Move", "qt" for "qm", and "tr" for "m" (see Figure III-12).

If we look at the code for the "move" command (Figure III-13), we see that it makes calls on routines in the parameter-specification (prmspc) overlay to allow the user to make selections on the screen, and to routines in the text-editing (txtedt) overlay to implement the algorithms for the commands.

Since the syntax for a procedure call permits an entire link to be used in place of a procedure name, we can use the "jump to link" command to see what these routines in txtedt actually do.

If we jump "up" from the procedure pointed to by the link (Figure III-14), we see that once again this code can act as a model since it only implements the delimiting of the selected entities and goes to another routine, movetx, to actually move the text. We therefore copy this branch also and make the necessary changes for the "transpose" command (see bottom of Figure III-14; Figure III-15 is the same as Figure III-14 except that all lines are shown).

The necessary modifications are changing the procedure names to correspond to our code in the mnctrl overlay and changing the name of the routine that is called to actually do the transposition.

The "jump to name" command takes us to the movetx routine (Figure III-16), which again serves as a model for our new routine, trantx. After outputting the file we do "Jump File Return" twice to get back to the SYSGD file.

Thus our new command has been implemented into the source code for NLS. We must now compile the main-control and text-editing overlays and reload the system. Rather than do the compilation from NLS, we will use the NUTILITY subsystem to both compile the files and load a new version of the system, which can then be run experimentally under DDT (see Figures III-17, III-18, and III-19). In Figures III-19 and III-20, the new command is successfully tested.

Let us now continue our examination of the SYSGD file (for a review, see Figure III-3). As shown above, the SYSGD file can be used to help one examine and modify the source code. There are also ways in which it can help one locate procedures of a given functional type. The use of the keyword system with the "index" branch anu the use of the content analyzer on the "ovlind" branch are two examples of this aid.

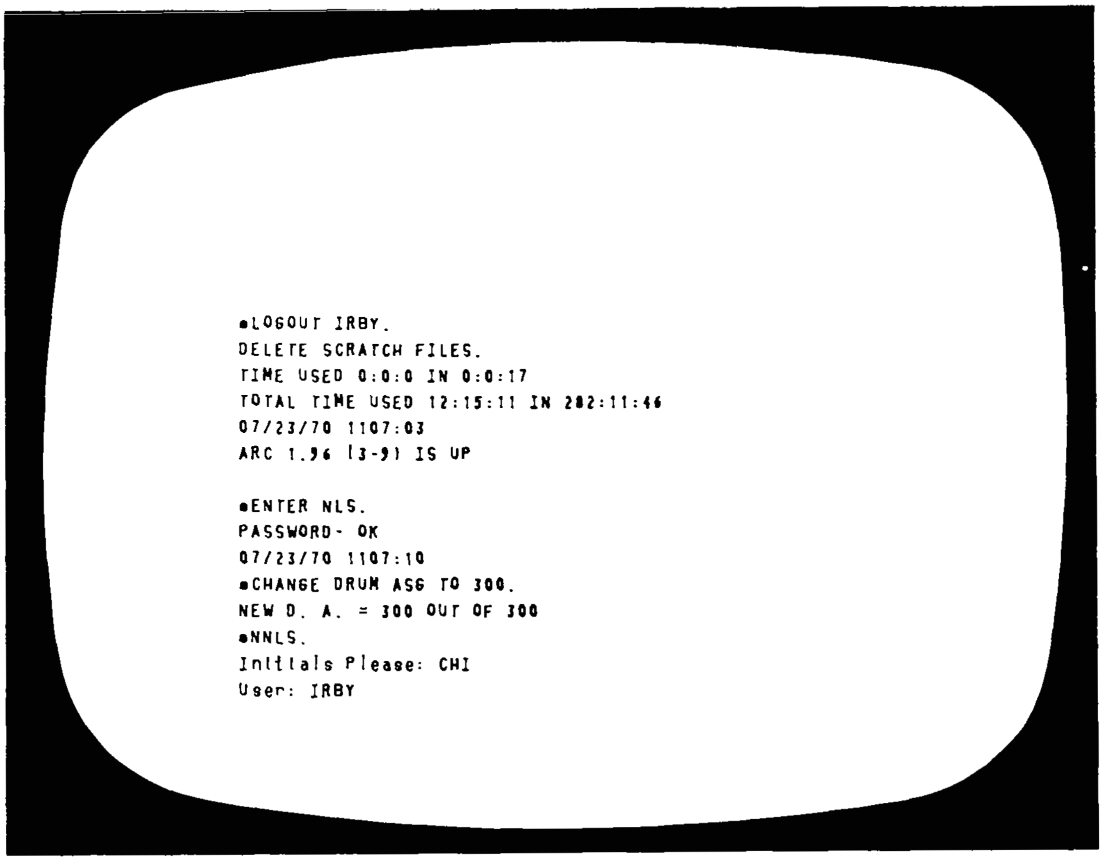
The "index" branch contains categorical listings of the procedures in the system by function. Using the keyword retrieval system, one can select and weight categories and have the named entries (links to procedures in this case) presented in order of decreasing relevance, according to the user's weighting and order of selection. One may then take the links and examine the documentation and code for each procedure (see Figures III-21, III-22, and III-23).

The "ovlind" branch of the SYSGD file contains information about each overlay in the system, including lists of all of the procedures, organized on an overlay basis, with one-line explanations of functions (see Figures III-24 and III-25).

Since a brief description of each procedure's function is associated with its name, we can use the content analyzer to help us find procedures of a given type. For example, we can insert a content-analysis pattern to search for statements containing the string "display" (see Figure III-26).

We compile the pattern (see Figure III-27) and re-create the display with the content-analysis filtering in effect. Now only statements containing the string "display" will be shown (see Figure III-28).

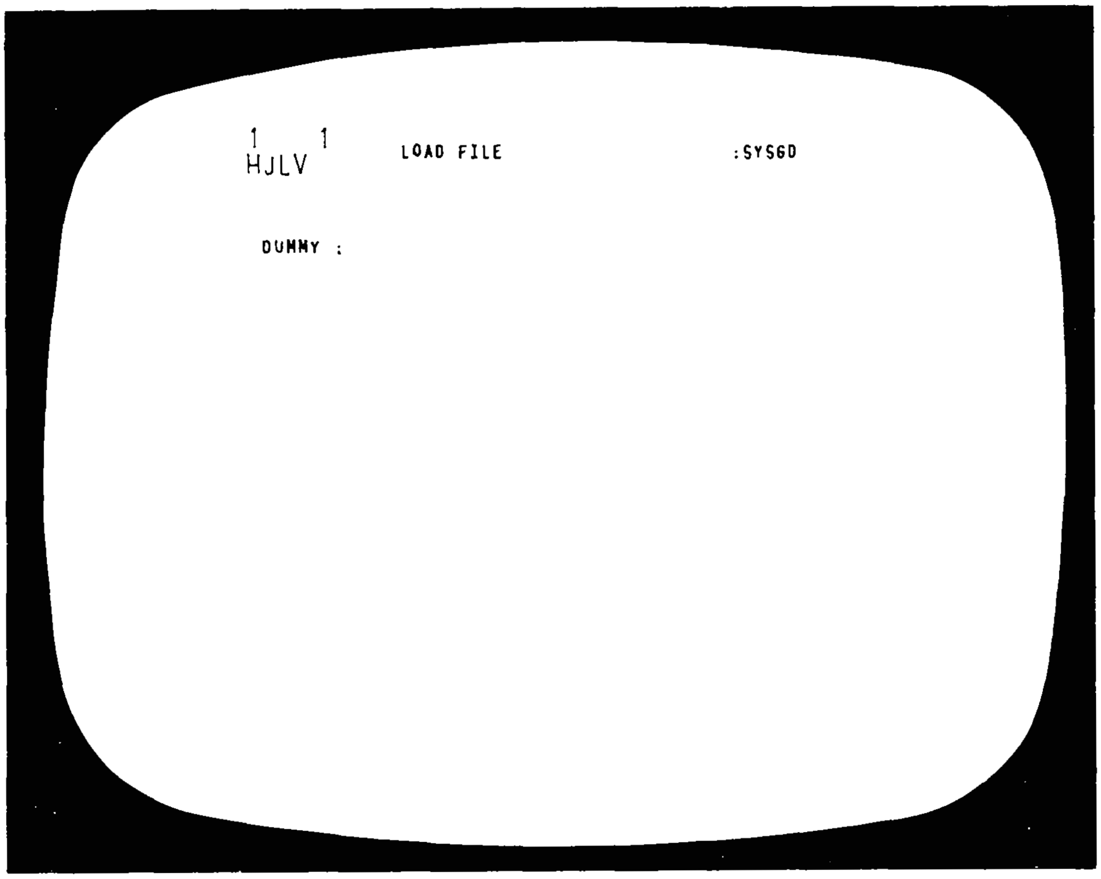
The result of this content filtering on this branch of the file is a list of procedures that in some way deal with the display. This is a useful way to retrieve up-to-date lists of procedures of a given functional type.



```
•LOGOUT IRBY.
DELETE SCRATCH FILES.
TIME USED 0:0:0 IN 0:0:17
TOTAL TIME USED 12:15:11 IN 282:11:46
07/23/70 1107:03
ARC 1.96 (3-9) IS UP

•ENTER NLS.
PASSWORD- OK
07/23/70 1107:10
•CHANGE DRUM ASG TO 300.
NEW D. A. = 300 OUT OF 300
•NNLS.
Initials Please: CHI
User: IRBY
```

TA-7079-11

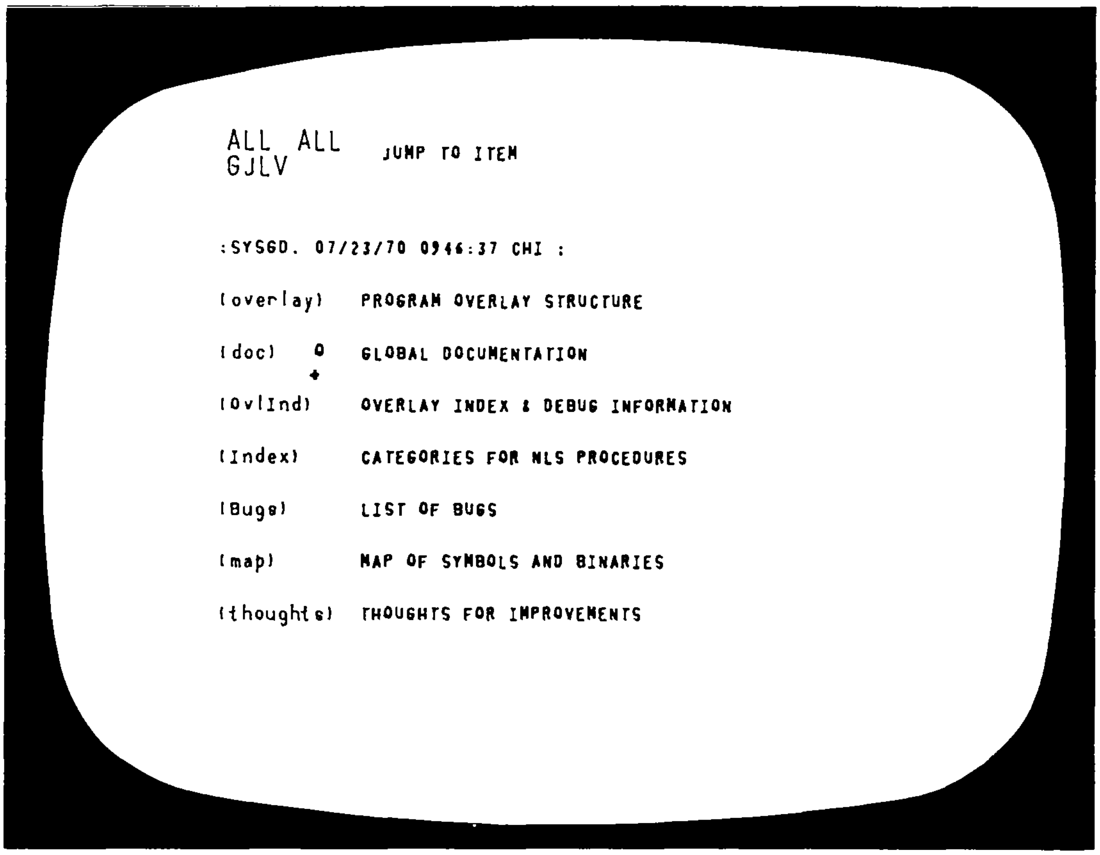
FIGURE III-1 LOGGING IN TO TIMESHARING SYSTEM AND ENTERING **NLS** SUBSYSTEM



```
1      1
HJLV        LOAD FILE                    :SYSGD

   DUMMY :
```

TA-7079-12

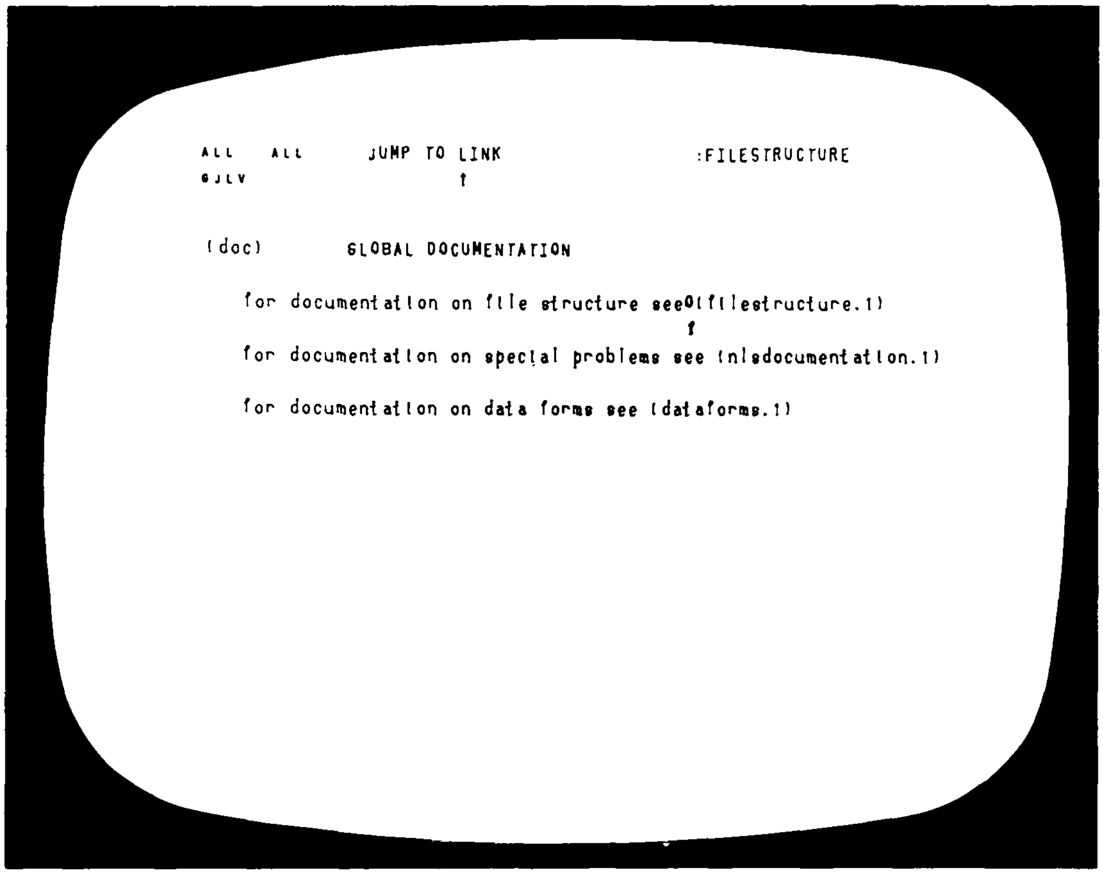
FIGURE III-2 **NLS** DISPLAY BEFORE A FILE IS LOADED OR CREATED. A "*Load File*" command has been specified but not executed.



```
ALL  ALL
GJLV        JUMP TO ITEM

:SYSGD. 07/23/70 0946:37 CHI :

(overlay)    PROGRAM OVERLAY STRUCTURE

(doc)    o   GLOBAL DOCUMENTATION
         +
(OvlInd)     OVERLAY INDEX & DEBUG INFORMATION

(Index)      CATEGORIES FOR NLS PROCEDURES

(Bugs)       LIST OF BUGS

(map)        MAP OF SYMBOLS AND BINARIES

(thoughts)   THOUGHTS FOR IMPROVEMENTS
```

TA-7079-13

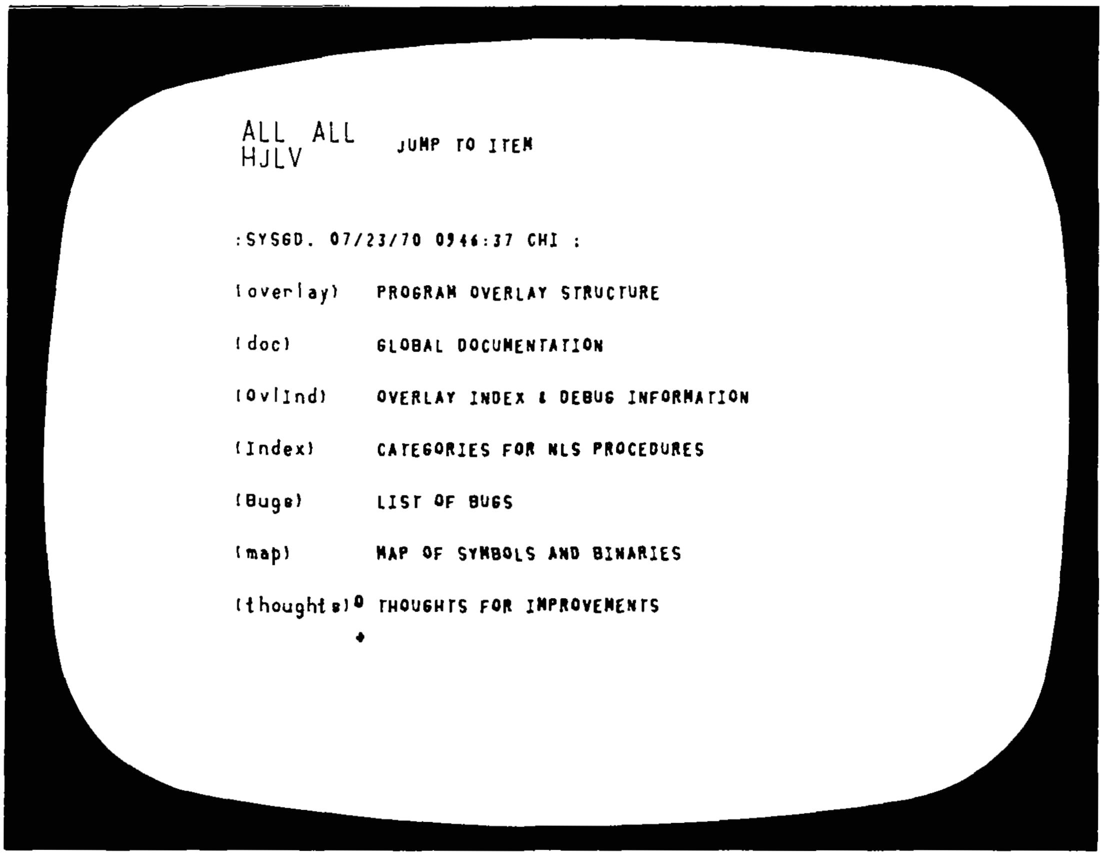
FIGURE III-3 MAJOR SECTIONS OF THE **NLS** SYSTEM-DIRECTORY FILE, **SYSGD.** The command specified in Figure III-2 has just been executed: a "Jump" command to display the "Global Documentation" branch has been specified but not executed.



```
ALL   ALL     JUMP TO LINK                :FILESTRUCTURE
@JLV                 ↑

(doc)       GLOBAL DOCUMENTATION

   for documentation on file structure see0(filestructure.1)
                                          ↑
   for documentation on special problems see (nlsdocumentation.1)

   for documentation on data forms see (dataforms.1)
```

TA-7079-14

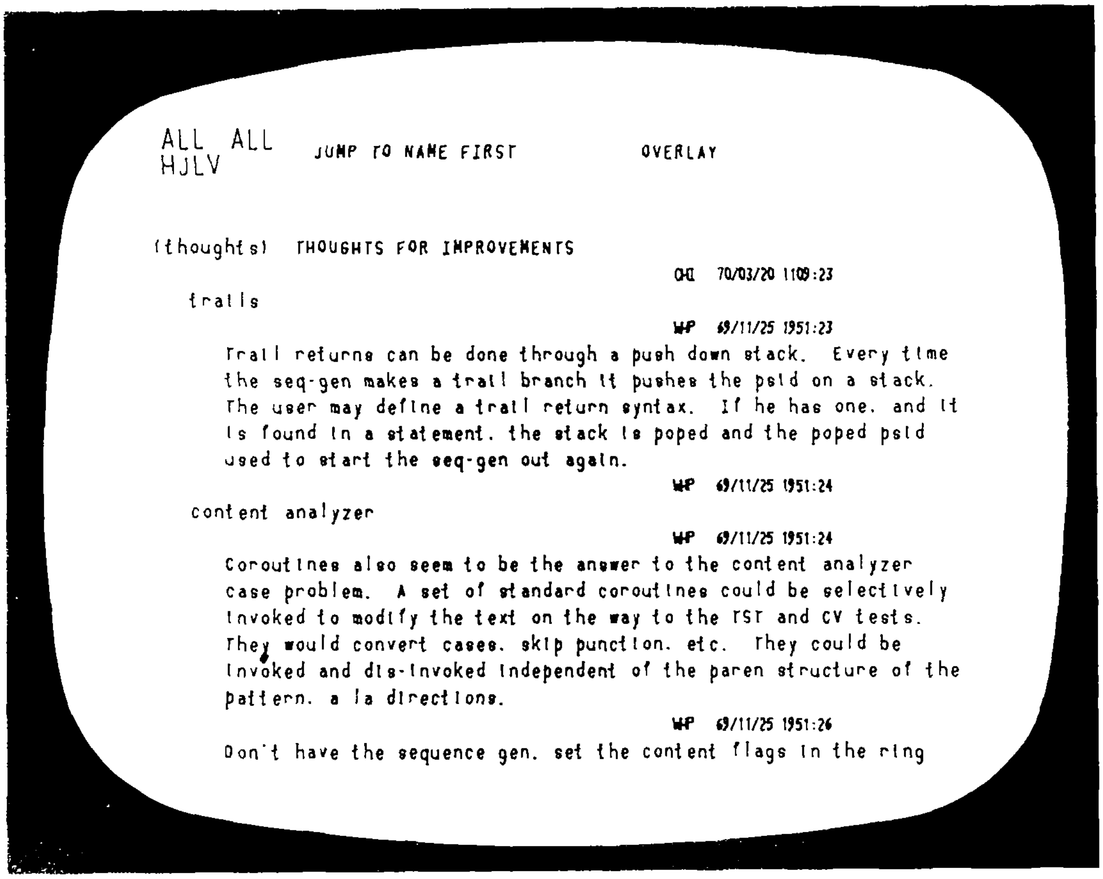
FIGURE III-4 "GLOBAL DOCUMENTATION" BRANCH OF **SYSGD** WITH "BRANCH-ONLY" FEATURE IN EFFECT. The user is about to follow a link which will cause the file-structure documentation to be displayed.



```
ALL  ALL
HJLV         JUMP TO ITEM

:SYSGD. 07/23/70 0946:37 CHI :

(overlay)    PROGRAM OVERLAY STRUCTURE

(doc)        GLOBAL DOCUMENTATION

(OvlInd)     OVERLAY INDEX & DEBUG INFORMATION

(Index)      CATEGORIES FOR NLS PROCEDURES

(Bugs)       LIST OF BUGS

(map)        MAP OF SYMBOLS AND BINARIES

(thoughts)°  THOUGHTS FOR IMPROVEMENTS
          ♦
```

TA-7079-15

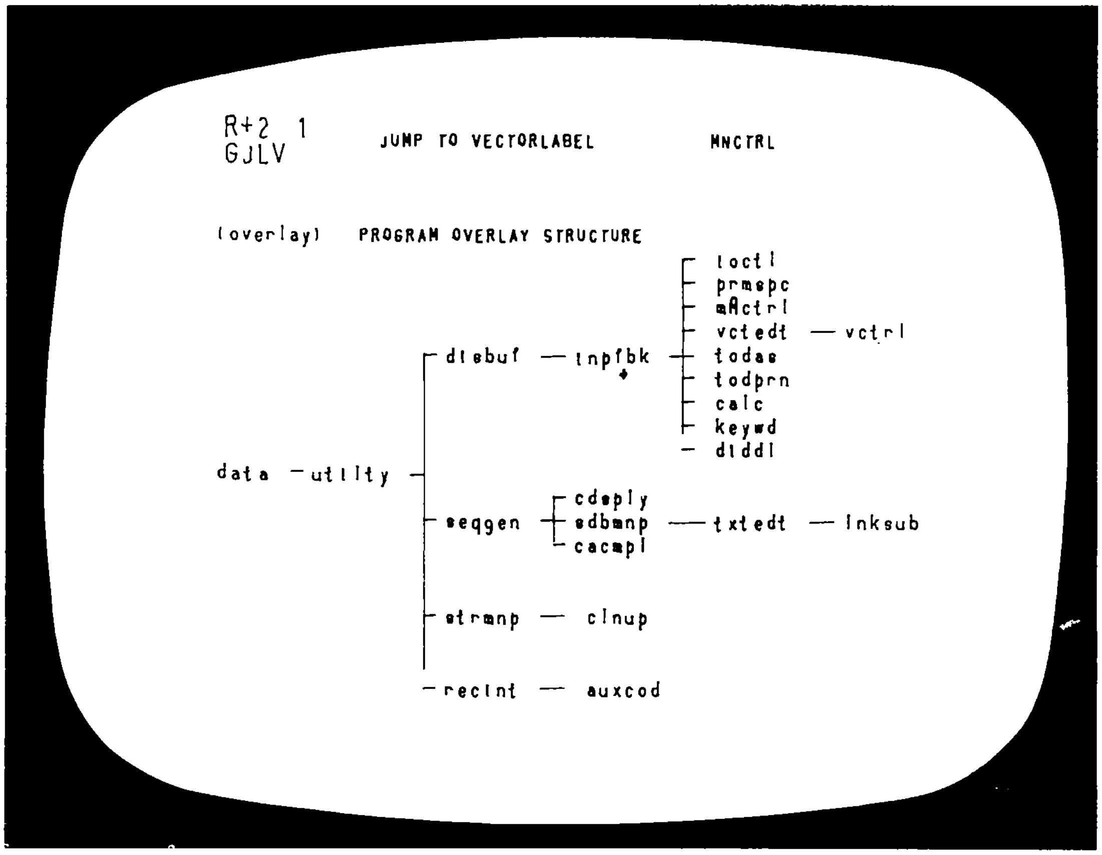
FIGURE III-5 The user has now returned to the same view shown in Figure III-3 (several steps have been omitted from the sequence). A "Jump" command to display the "Thoughts for Improvements" branch has been specified but not executed.



```
ALL  ALL
HJLV        JUMP TO NAME FIRST            OVERLAY


(thoughts)  THOUGHTS FOR IMPROVEMENTS
                                          CHI  70/03/20 1109:23
   trails
                                          WP  69/11/25 1951:23
      Trail returns can be done through a push down stack.  Every time
      the seq-gen makes a trail branch it pushes the psid on a stack.
      The user may define a trail return syntax.  If he has one, and it
      is found in a statement, the stack is poped and the poped psid
      used to start the seq-gen out again.
                                          WP  69/11/25 1951:24
   content analyzer
                                          WP  69/11/25 1951:24
      Coroutines also seem to be the answer to the content analyzer
      case problem.  A set of standard coroutines could be selectively
      invoked to modify the text on the way to the TST and CV tests.
      They would convert cases, skip punction, etc.  They could be
      invoked and dis-invoked independent of the paren structure of the
      pattern, a la directions.
                                          WP  69/11/25 1951:26
      Don't have the sequence gen. set the content flags in the ring
```

TA-7079-16

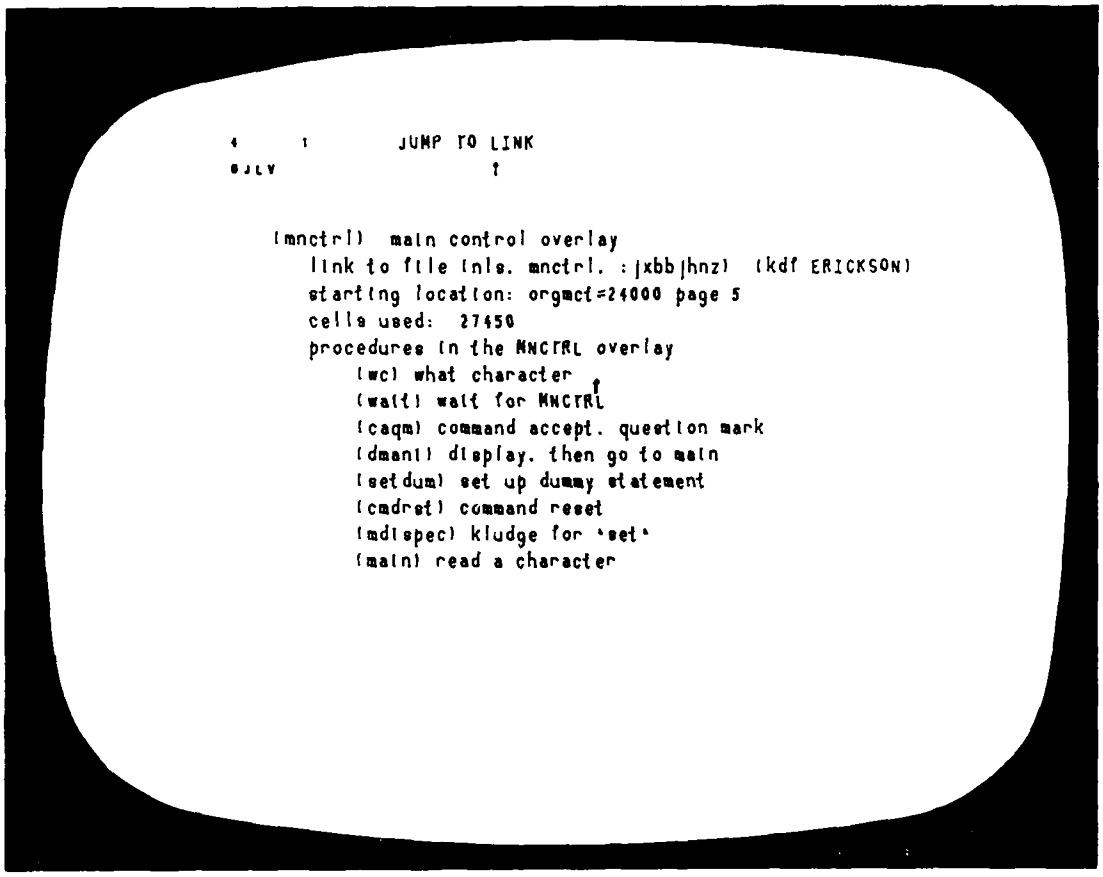
FIGURE III-6 The "Thoughts for Improvements" branch serves as a repository for ideas about what could be done to improve the system. A "Jump to Name" command has been specified to display a statement named "overlay"—the user has specified this name by typing it in.



```
R+2  1
GJLV        JUMP TO VECTORLABEL          MNCTRL

(overlay)   PROGRAM OVERLAY STRUCTURE
                                            ┌ ioctl
                                            ├ prmspc
                                            ├ mActrl
                                            ├ vctedt   — vctrl
                        ┌ dlsbuf  — inpfbk ─┼ todas
                        │           +       ├ todprn
                        │                   ├ calc
                        │                   ├ keywd
                        │                   - dlddl
data  —utility  ─┤
                        │         ┌ cdsply
                        ├ seqgen ─┼ sdbmnp  —— txtedt  — lnksub
                        │         └ cacmpl
                        │
                        ├ strmnp  —  clnup
                        │
                        - reclnt  —  auxcod
```

TA-7079-17

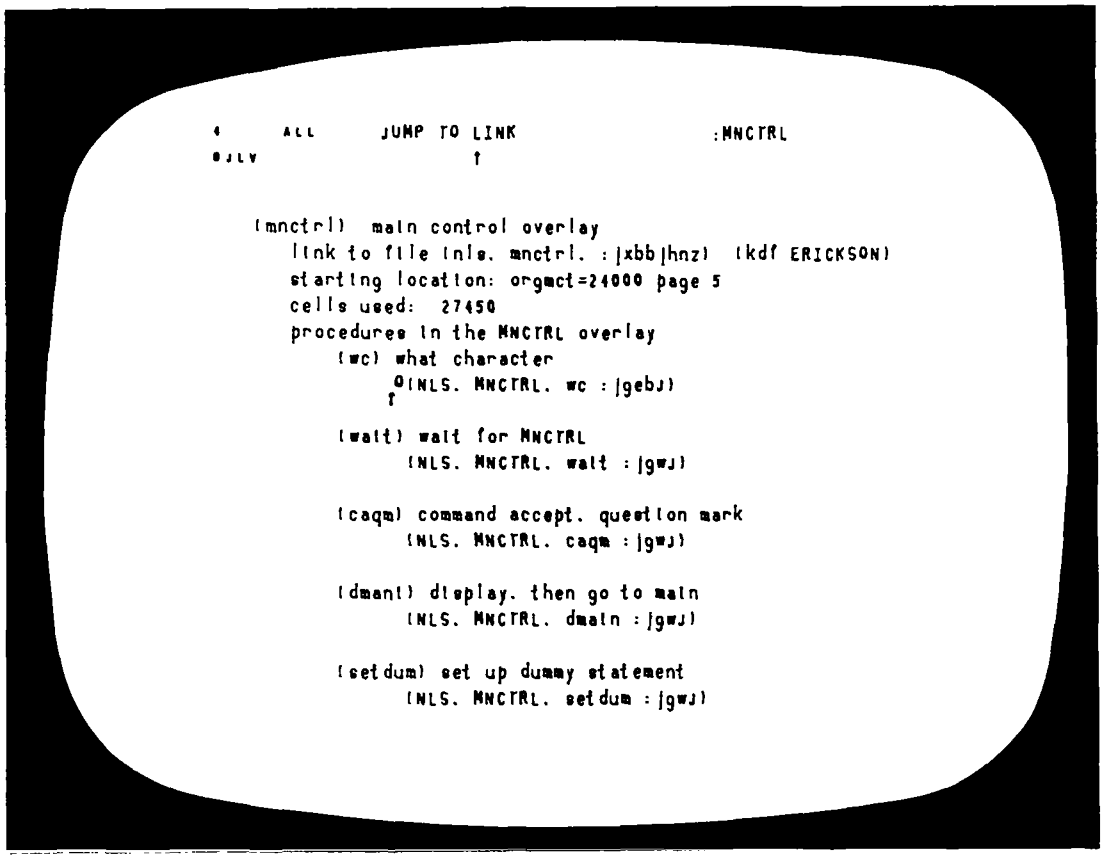
FIGURE III-7   FIRST BRANCH OF **SYSGD** FILE: DIAGRAM OF OVERLAY STRUCTURE USED BY **NLS** SYSTEM



```
4      1          JUMP TO LINK
●JLV                  ↑


  (mnctrl)  main control overlay
     link to file (nls. mnctrl. :jxbbjhnz)  (kdf ERICKSON)
     starting location: orgmct=24000 page 5
     cells used:  27450
     procedures in the MNCTRL overlay
         (wc) what character  ↑
         (wait) wait for MNCTRL
         (caqm) command accept. question mark
         (dmanl) display. then go to main
         (setdum) set up dummy statement
         (cmdrst) command reset
         (mdispec) kludge for 'set'
         (main) read a character
```

TA-7079-18

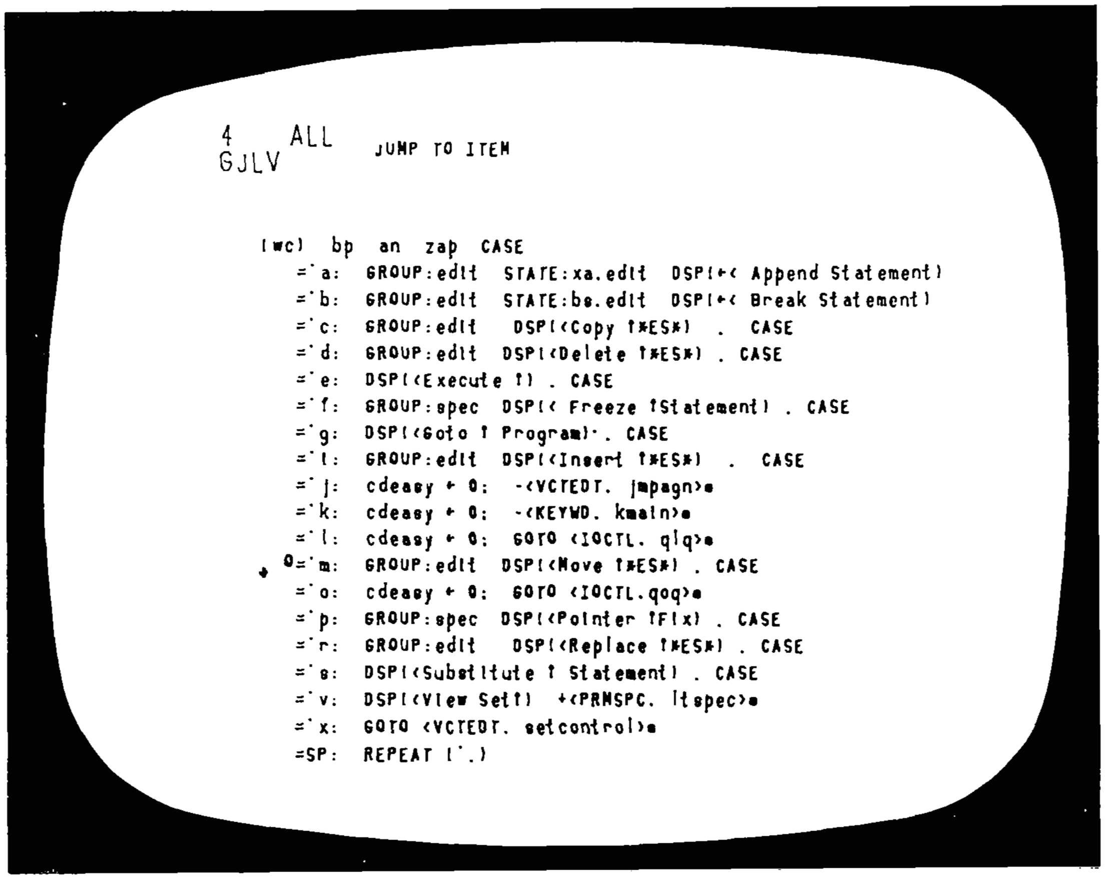
FIGURE III-8 A BRANCH GIVING SUPPORTIVE INFORMATION FOR MAIN-CONTROL (MNCTRL) OVERLAY. Display shows only the first line of each statement. A "Jump to Link" command has been partially specified—no selection has been made.



```
4       ALL       JUMP TO LINK                    :MNCTRL
0JLV                    ↑


    (mnctrl)  main control overlay
       link to file (nls, mnctrl, :jxbbjhnz)  (kdf ERICKSON)
       starting location: orgmct=24000 page 5
       cells used:  27450
       procedures in the MNCTRL overlay
           (wc) what character
                0(NLS, MNCTRL, wc : jgebJ)
                ↑

           (wait) wait for MNCTRL
                (NLS, MNCTRL, wait : jgwJ)

           (caqm) command accept, question mark
                (NLS, MNCTRL, caqm : jgwJ)

           (dmant) display, then go to main
                (NLS, MNCTRL, dmain : jgwJ)

           (setdum) set up dummy statement
                (NLS, MNCTRL, setdum : jgwJ)
```

TA-7079-19

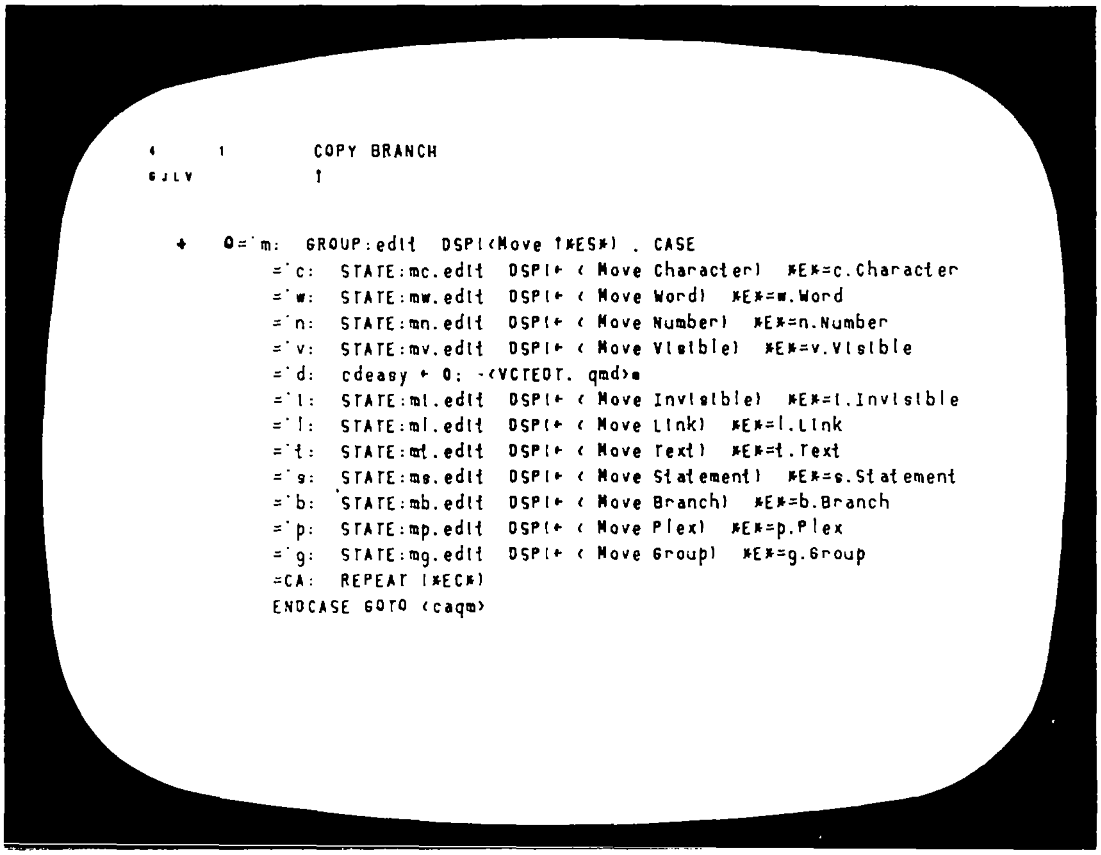
FIGURE III-9 SAME AS FIGURE III-8, EXCEPT THAT DISPLAY SHOWS ALL LINES OF EACH STATEMENT. The link in the statement named "wc" has been selected; this link specified another statement named "wc" in a file called "MNCTRL" under User "NLS" (the characters "jgebJ" in the link are VIEWSPEC codes).



```
4    ALL
GJLV      JUMP TO ITEM


(wc)  bp  an  zap  CASE
   =`a:  GROUP:edit  STATE:xa.edit  DSP(←< Append Statement)
   =`b:  GROUP:edit  STATE:bs.edit  DSP(←< Break Statement)
   =`c:  GROUP:edit   DSP(<Copy ↑*ES*)  .  CASE
   =`d:  GROUP:edit  DSP(<Delete ↑*ES*) . CASE
   =`e:  DSP(<Execute ↑) . CASE
   =`f:  GROUP:spec  DSP(< Freeze ↑Statement) . CASE
   =`g:  DSP(<Goto ↑ Program)· . CASE
   =`i:  GROUP:edit  DSP(<Insert ↑*ES*)  .  CASE
   =`j:  cdeasy ← 0:  -<VCTEDT. jmpagn>•
   =`k:  cdeasy ← 0:  -<KEYWD. kmain>•
   =`l:  cdeasy ← 0:  GOTO <IOCTL. qlq>•
 ♦ O=`m:  GROUP:edit  DSP(<Move ↑*ES*) . CASE
   =`o:  cdeasy ← 0:  GOTO <IOCTL.qoq>•
   =`p:  GROUP:spec  DSP(<Pointer ↑F(x) . CASE
   =`r:  GROUP:edit   DSP(<Replace ↑*ES*) . CASE
   =`s:  DSP(<Substitute ↑ Statement) . CASE
   =`v:  DSP(<View Set↑)  ←<PRMSPC. ltspec>•
   =`x:  GOTO <VCTEDT. setcontrol>•
   =SP:  REPEAT (`.)
```

TA-7079-20

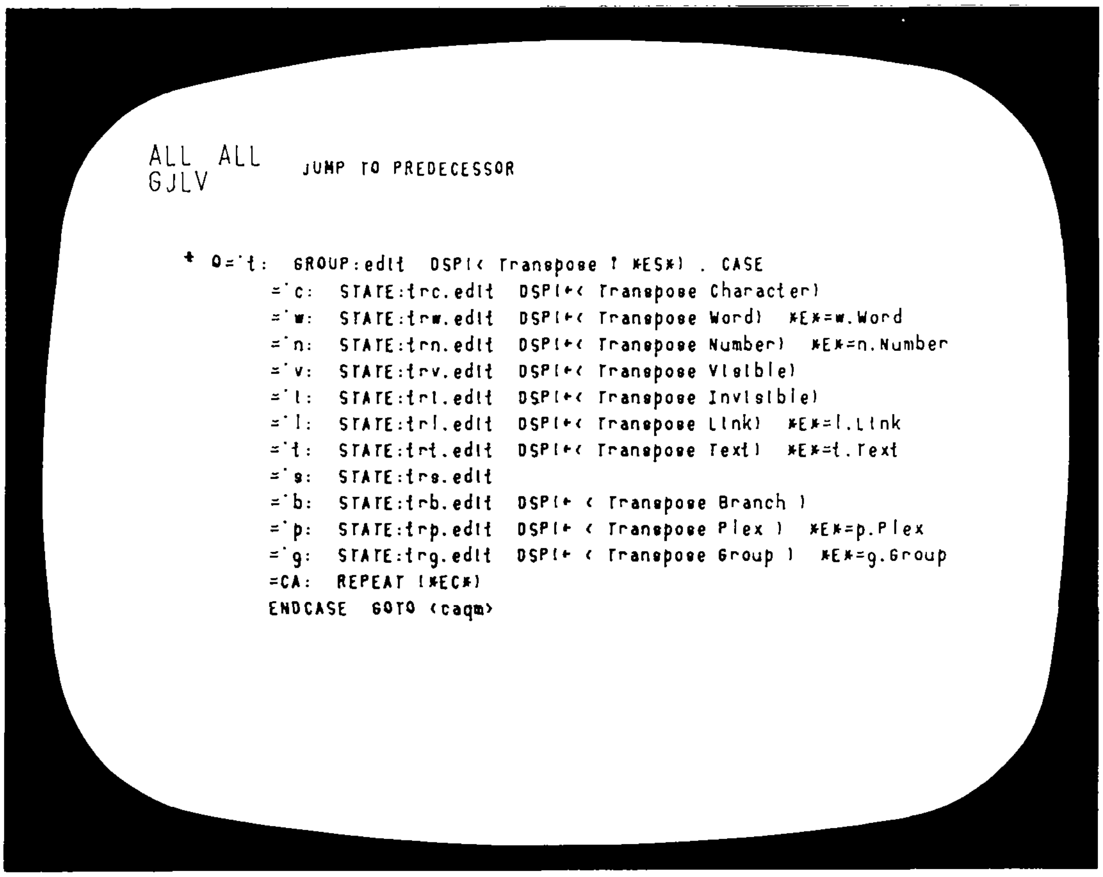
FIGURE III-10 RESULT OF "JUMPING" ON THE LINK SELECTED IN FIGURE III-9 (from Index in SYSGD File to Source Code for "wc" Procedure in Main-Control Overlay). A "Jump" command has been specified to display a particular branch of this code.



```
4       1          COPY BRANCH
6JLV               ↑


  +   O='m:  GROUP:edit  DSP(<Move ↑*ES*) . CASE
          ='c:  STATE:mc.edit  DSP(+ < Move Character)  *E*=c.Character
          ='w:  STATE:mw.edit  DSP(+ < Move Word)  *E*=w.Word
          ='n:  STATE:mn.edit  DSP(+ < Move Number)  *E*=n.Number
          ='v:  STATE:mv.edit  DSP(+ < Move Visible)  *E*=v.Visible
          ='d:  cdeasy + 0; -<VCTEDT. qmd>.
          ='i:  STATE:mi.edit  DSP(+ < Move Invisible)  *E*=i.Invisible
          ='l:  STATE:ml.edit  DSP(+ < Move Link)  *E*=l.Link
          ='t:  STATE:mt.edit  DSP(+ < Move Text)  *E*=t.Text
          ='s:  STATE:ms.edit  DSP(+ < Move Statement)  *E*=s.Statement
          ='b:  STATE:mb.edit  DSP(+ < Move Branch)  *E*=b.Branch
          ='p:  STATE:mp.edit  DSP(+ < Move Plex)  *E*=p.Plex
          ='g:  STATE:mg.edit  DSP(+ < Move Group)  *E*=g.Group
          =CA:  REPEAT (*EC*)
          ENDCASE GOTO <caqm>
```

TA-7079-21

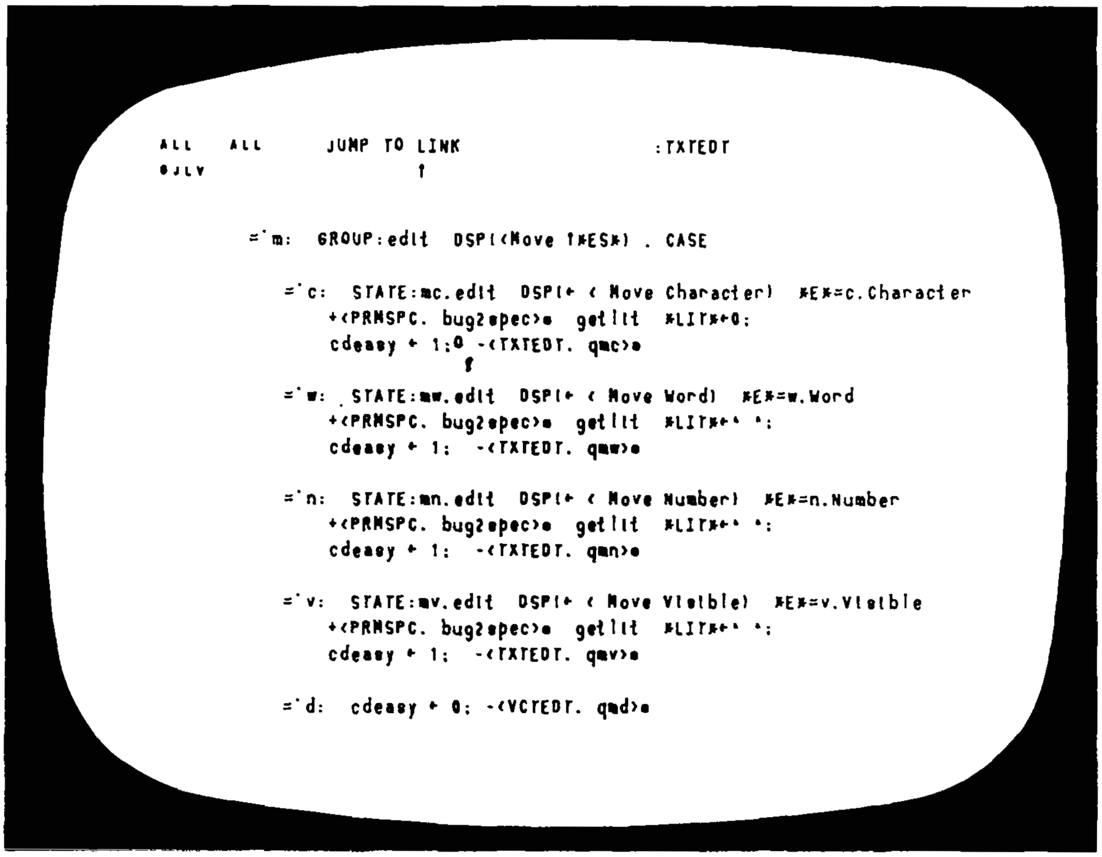
FIGURE III-11 CODE DESCRIBING COMMAND LANGUAGE FOR "MOVE" COMMANDS, IN A SPECIAL-PURPOSE LANGUAGE. A "Copy Branch" command has been specified to create a duplicate copy of this command-language code.



```
ALL  ALL      JUMP TO PREDECESSOR
GJLV

 + O='t:  GROUP:edit  DSP(< Transpose ! *ES*) . CASE
       ='c:  STATE:trc.edit  DSP(+< Transpose Character)
       ='w:  STATE:trw.edit  DSP(+< Transpose Word)  *E*=w.Word
       ='n:  STATE:trn.edit  DSP(+< Transpose Number)  *E*=n.Number
       ='v:  STATE:trv.edit  DSP(+< Transpose Visible)
       ='i:  STATE:tri.edit  DSP(+< Transpose Invisible)
       ='l:  STATE:trl.edit  DSP(+< Transpose Link)  *E*=l.Link
       ='t:  STATE:trt.edit  DSP(+< Transpose Text)  *E*=t.Text
       ='s:  STATE:trs.edit
       ='b:  STATE:trb.edit  DSP(+ < Transpose Branch )
       ='p:  STATE:trp.edit  DSP(+ < Transpose Plex )  *E*=p.Plex
       ='g:  STATE:trg.edit  DSP(+ < Transpose Group )  *E*=g.Group
       =CA:  REPEAT (*EC*)
       ENDCASE  GOTO <caqm>
```

TA-7079-22

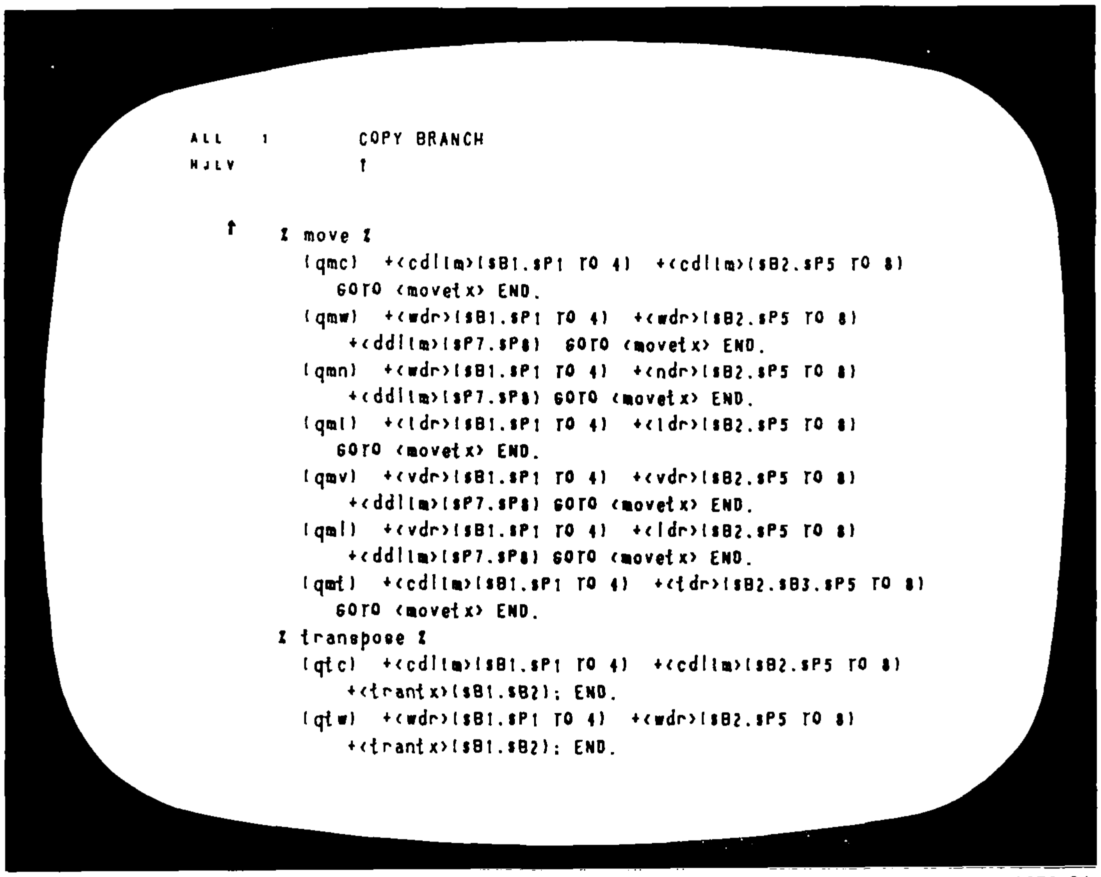
FIGURE III-12 By executing the "copy branch" command, then substituting "Transpose" for "Move," "qt" for "qm," and "tr" for "m," we produce the command-language code for the "transpose" command. A "Jump to Predecessor" command has been specified to redisplay the unaltered command-language code for "Move."



```
ALL    ALL       JUMP TO LINK                    :TXTEDT
•JLV                      ↑


          ='m:  GROUP:edit  DSP(<Move ↑*ES*) . CASE

             ='c:  STATE:mc.edit  DSP(+ < Move Character)  *E*=c.Character
                 +<PRMSPC. bug2spec>•  getlit  *LIT*+0;
                 cdeasy + 1;0 -<TXTEDT. qmc>•
                            ↑
             ='w:  STATE:mw.edit  DSP(+ < Move Word)  *E*=w.Word
                 +<PRMSPC. bug2spec>•  getlit  *LIT*+' ';
                 cdeasy + 1;  -<TXTEDT. qmw>•

             ='n:  STATE:mn.edit  DSP(+ < Move Number)  *E*=n.Number
                 +<PRMSPC. bug2spec>•  getlit  *LIT*+' ';
                 cdeasy + 1;  -<TXTEDT. qmn>•

             ='v:  STATE:mv.edit  DSP(+ < Move Visible)  *E*=v.Visible
                 +<PRMSPC. bug2spec>•  getlit  *LIT*+' ';
                 cdeasy + 1;  -<TXTEDT. qmv>•

             ='d:  cdeasy + 0; -<VCTEDT. qmd>•
```

TA-7079-23

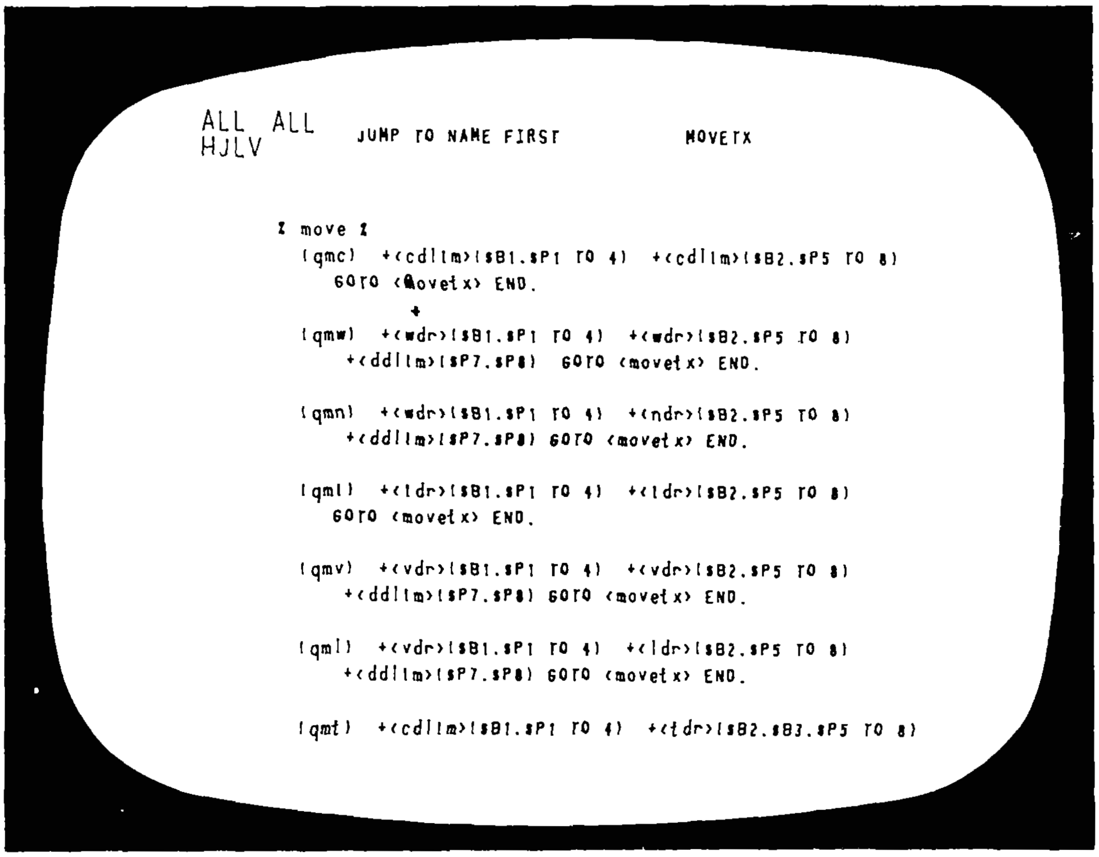
FIGURE III-13 COMMAND-LANGUAGE CODE FOR "MOVE" COMMANDS, WITH ALL LEVELS SHOWING. We now "Jump" on a link to a routine named "qmc" in the "TXTEDT" overlay file which implements the "Move Character" command (the link in this case is a subroutine call in the "Move" command-language code—photo shows display just before "Jump" command is executed).



```
ALL    1        COPY BRANCH
HJLV            ↑

   ↑   % move %
         (qmc)  +<cdllm>($B1.$P1 TO 4)  +<cdllm>($B2.$P5 TO 8)
            GOTO <movetx> END.
         (qmw)  +<wdr>($B1.$P1 TO 4)  +<wdr>($B2.$P5 TO 8)
             +<ddllm>($P7.$P8)  GOTO <movetx> END.
         (qmn)  +<wdr>($B1.$P1 TO 4)  +<ndr>($B2.$P5 TO 8)
             +<ddllm>($P7.$P8) GOTO <movetx> END.
         (qml)  +<ldr>($B1.$P1 TO 4)  +<ldr>($B2.$P5 TO 8)
            GOTO <movetx> END.
         (qmv)  +<vdr>($B1.$P1 TO 4)  +<vdr>($B2.$P5 TO 8)
             +<ddllm>($P7.$P8) GOTO <movetx> END.
         (qml)  +<vdr>($B1.$P1 TO 4)  +<ldr>($B2.$P5 TO 8)
             +<ddllm>($P7.$P8) GOTO <movetx> END.
         (qmt)  +<cdllm>($B1.$P1 TO 4)  +<tdr>($B2.$B3.$P5 TO 8)
            GOTO <movetx> END.
       % transpose %
         (qtc)  +<cdllm>($B1.$P1 TO 4)  +<cdllm>($B2.$P5 TO 8)
             +<trantx>($B1.$B2); END.
         (qtw)  +<wdr>($B1.$P1 TO 4)  +<wdr>($B2.$P5 TO 8)
             +<trantx>($B1.$B2); END.
```

TA-7079-24

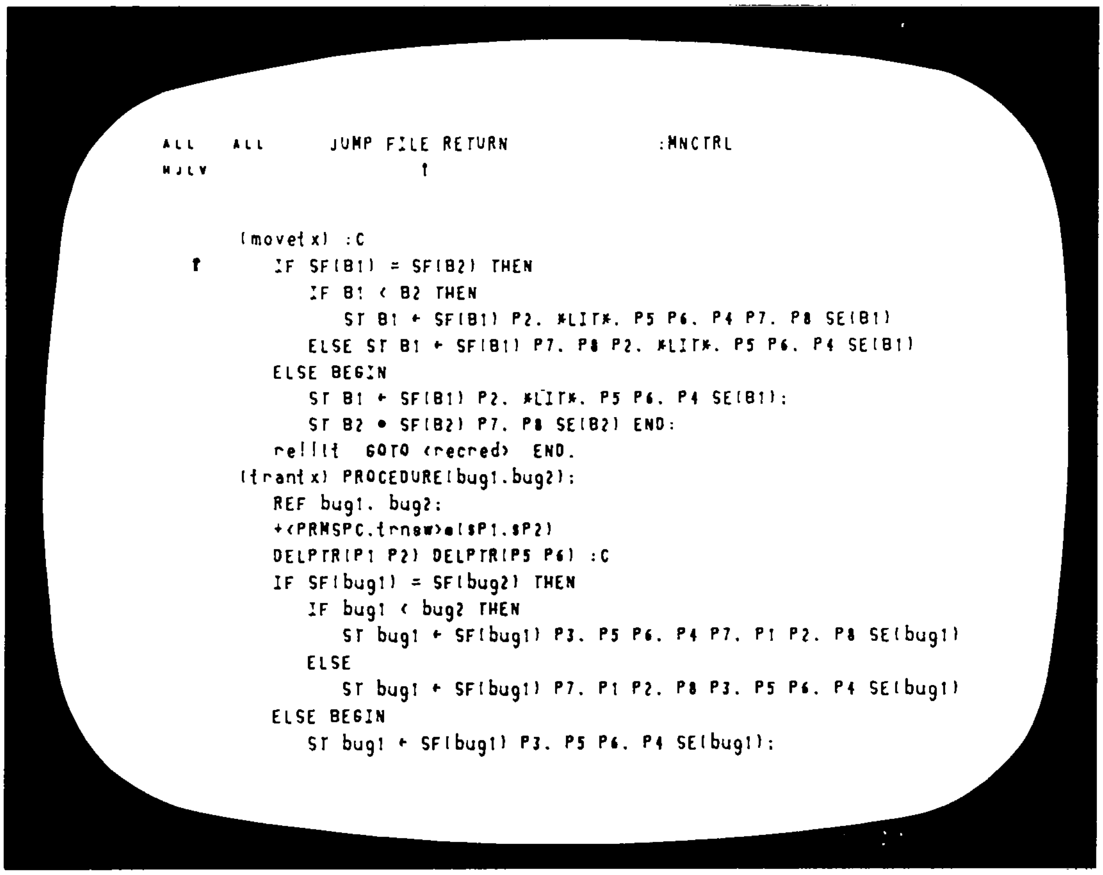
FIGURE III-14 We have executed the "Jump to Link" command from the previous view, and then executed a "Jump to Up" command (omitted from sequence of photos) to display the source statement of "qmc." We see that the code which implements the "move" commands can be used as a model for the "transpose" command. Again, "Copy Branch" is used to make a duplicate copy of this code, and the copy is altered to produce code for the "Transpose" command (bottom of photo).



```
ALL  ALL
HJLV       JUMP TO NAME FIRST             MOVETX


        % move %
          (qmc)  +<cdllm>($B1.$P1 TO 4)  +<cdllm>($B2.$P5 TO 8)
             GOTO <Movetx> END.
                   +
          (qmw)  +<wdr>($B1.$P1 TO 4)  +<wdr>($B2.$P5 TO 8)
              +<ddllm>($P7.$P8)  GOTO <movetx> END.

          (qmn)  +<wdr>($B1.$P1 TO 4)  +<ndr>($B2.$P5 TO 8)
              +<ddllm>($P7.$P8) GOTO <movetx> END.

          (qml)  +<ldr>($B1.$P1 TO 4)  +<ldr>($B2.$P5 TO 8)
             GOTO <movetx> END.

          (qmv)  +<vdr>($B1.$P1 TO 4)  +<vdr>($B2.$P5 TO 8)
              +<ddllm>($P7.$P8) GOTO <movetx> END.

          (qml)  +<vdr>($B1.$P1 TO 4)  +<ldr>($B2.$P5 TO 8)
              +<ddllm>($P7.$P8) GOTO <movetx> END.

          (qmt)  +<cdllm>($B1.$P1 TO 4)  +<tdr>($B2.$B3.$P5 TO 8)
```

TA-7079-25

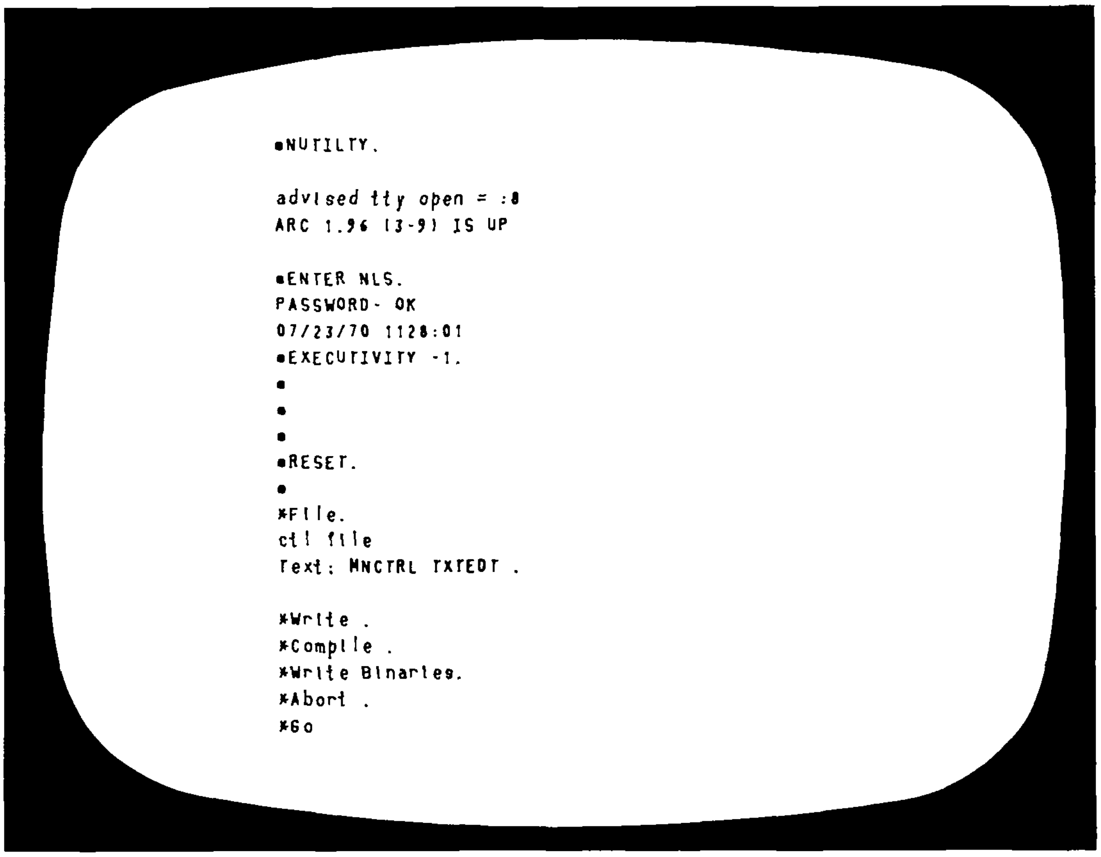
FIGURE III-15 SAME DISPLAY AS FIGURE III-14 BUT WITH ALL LINES SHOWING. We now follow a "GOTO" instruction by using the "Jump to Name" command (photo shows display just before "Jump" command is executed).



```
ALL    ALL       JUMP FILE RETURN               :MNCTRL
HJLV                     ↑


            (movetx) :C
     ↑         IF SF(B1) = SF(B2) THEN
                  IF B1 < B2 THEN
                     ST B1 ← SF(B1) P2. *LIT*. P5 P6. P4 P7. P8 SE(B1)
                  ELSE ST B1 ← SF(B1) P7. P8 P2. *LIT*. P5 P6. P4 SE(B1)
               ELSE BEGIN
                  ST B1 ← SF(B1) P2. *LIT*. P5 P6. P4 SE(B1);
                  ST B2 ← SF(B2) P7. P8 SE(B2) END;
               re!!!t  GOTO <recred>  END.
            (trantx) PROCEDURE(bug1.bug2);
               REF bug1. bug2;
               ←<PRMSPC.trnsw>@($P1.$P2)
               DELPTR(P1 P2) DELPTR(P5 P6) :C
               IF SF(bug1) = SF(bug2) THEN
                  IF bug1 < bug2 THEN
                     ST bug1 ← SF(bug1) P3. P5 P6. P4 P7. P1 P2. P8 SE(bug1)
                  ELSE
                     ST bug1 ← SF(bug1) P7. P1 P2. P8 P3. P5 P6. P4 SE(bug1)
               ELSE BEGIN
                  ST bug1 ← SF(bug1) P3. P5 P6. P4 SE(bug1);
```

TA-7079-26

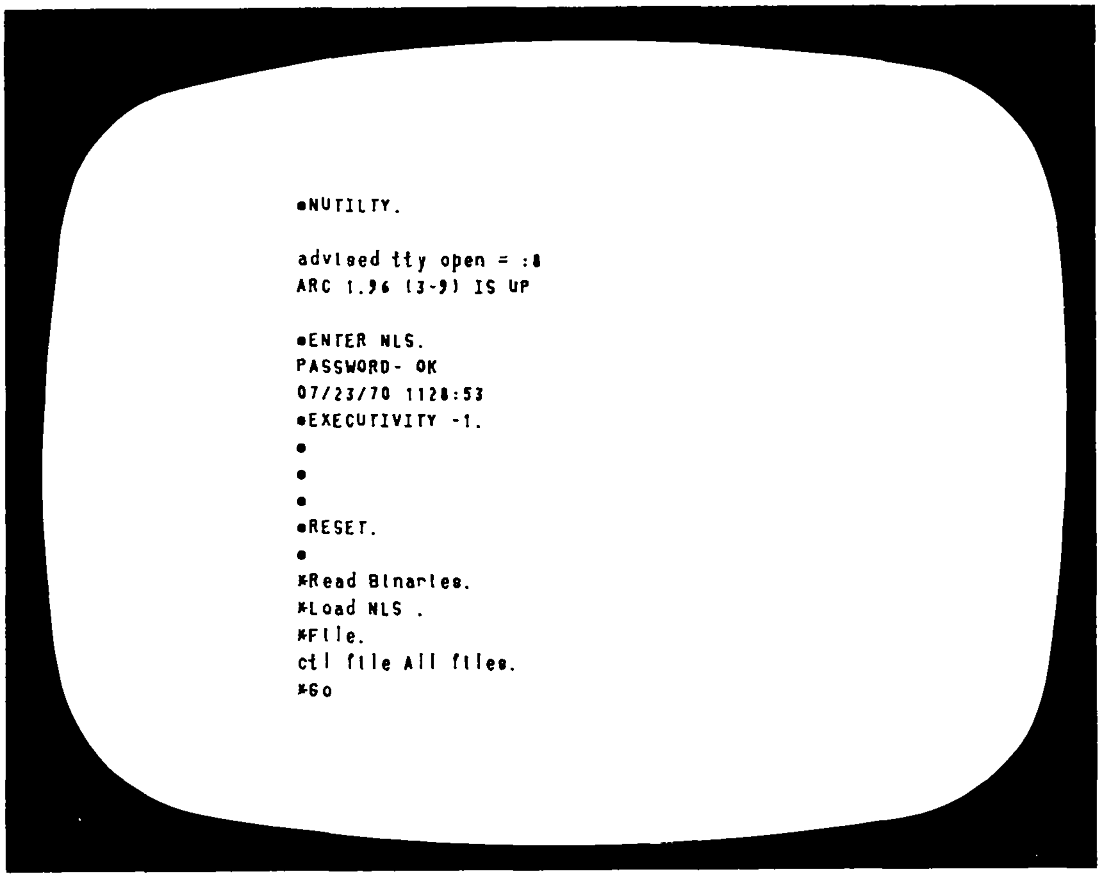
FIGURE III-16 The "movetx" code has been copied and altered to produce the "trantx" code for the new command. We now return to the previous file by using the "Jump File Return" command (photo shows display just before "Jump" command is executed).



```
•NUTILTY.

advised tty open = :8
ARC 1.96 (3-9) IS UP

•ENTER NLS.
PASSWORD- OK
07/23/70 1128:01
•EXECUTIVITY -1.
•
•
•
•RESET.
•
*File.
ctl file
Text: MNCTRL TXTEDT .

*Write .
*Compile .
*Write Binaries.
*Abort .
*Go
```

TA-7079-27

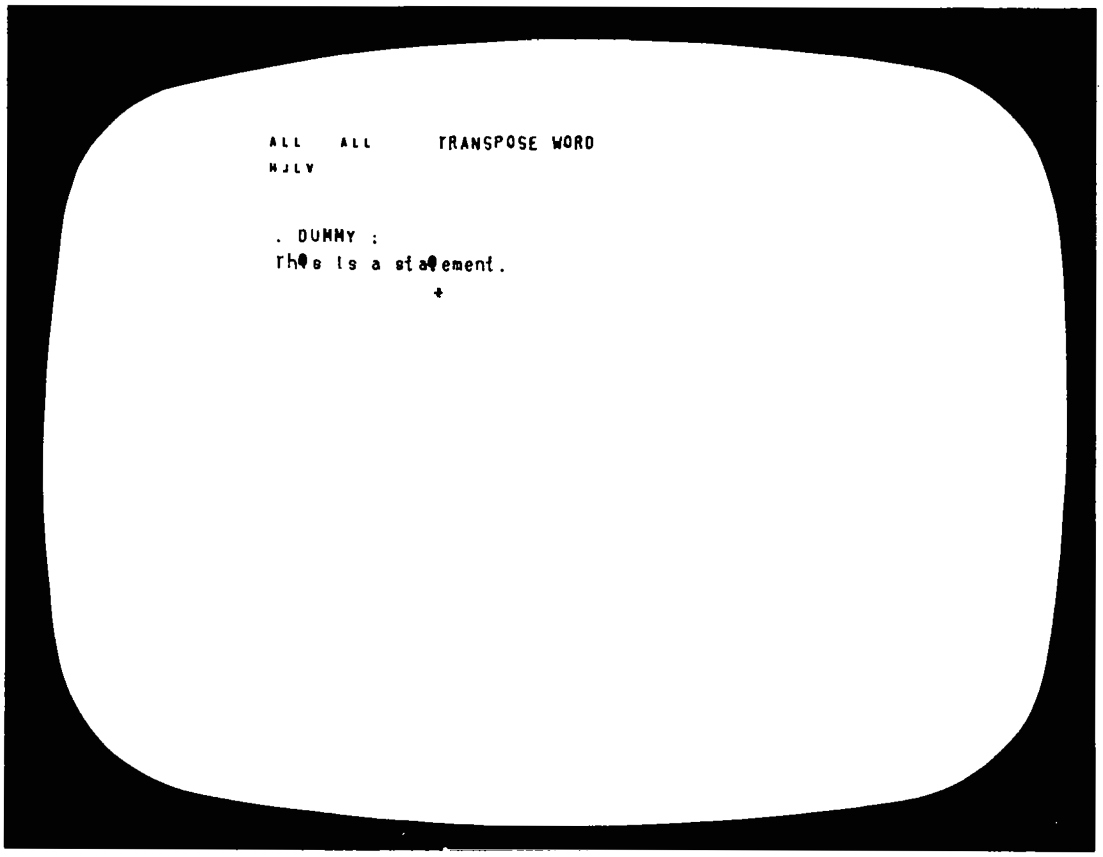
FIGURE III-17 The NUTILITY subsystem allows one to automatically archive, retrieve from archive, compile, print, and cross-reference files (display is now running in Teletype-simulation mode, as it was before NLS was started up in Figure III-1).



```
•NUTILTY.

advised tty open = :8
ARC 1.96 (3-9) IS UP

•ENTER NLS.
PASSWORD- OK
07/23/70 1128:53
•EXECUTIVITY -1.
•
•
•
•RESET.
•
*Read Binaries.
*Load NLS .
*File.
ctl file All files.
*Go
```

TA-7079-28

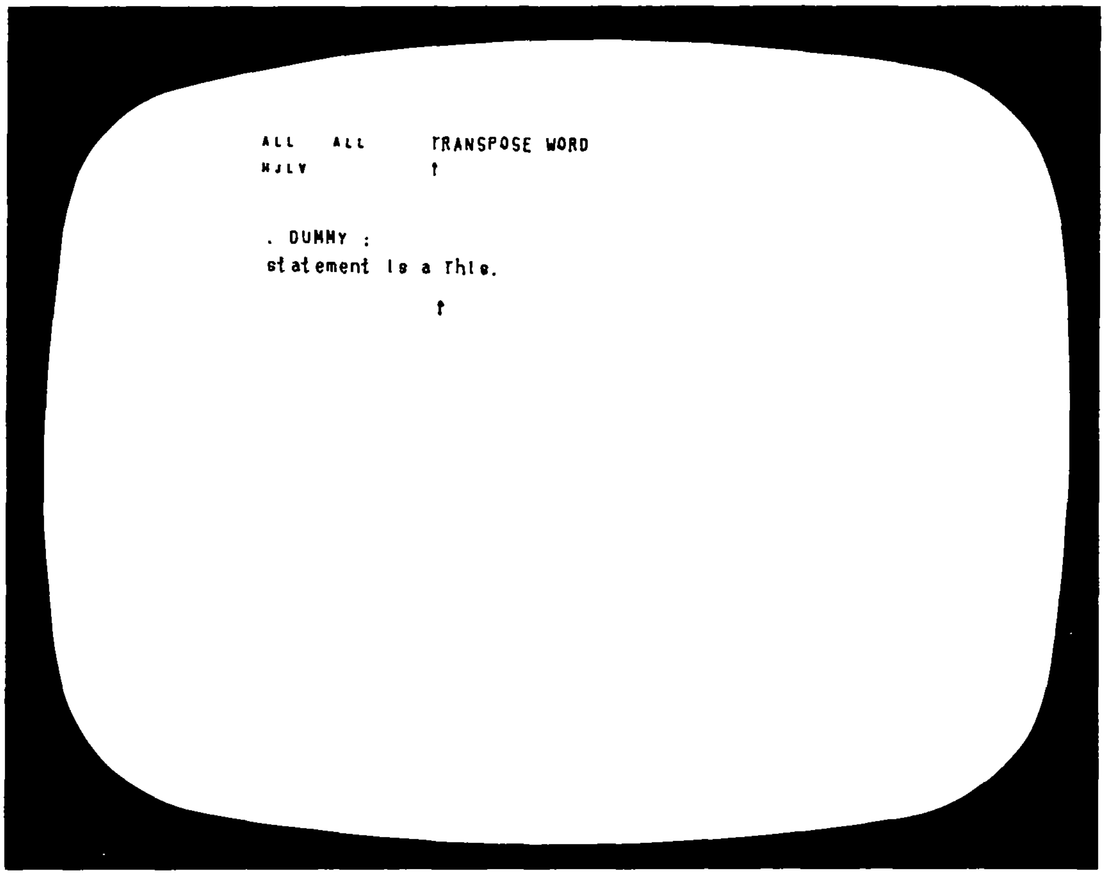
FIGURE III-18 Once the changed code has been compiled, NUTILITY can be used to automatically load a new version of the system.



```
ALL    ALL       TRANSPOSE WORD
MJLV


. DUMMY :
This is a statement.
                +
```

TA-7079-29

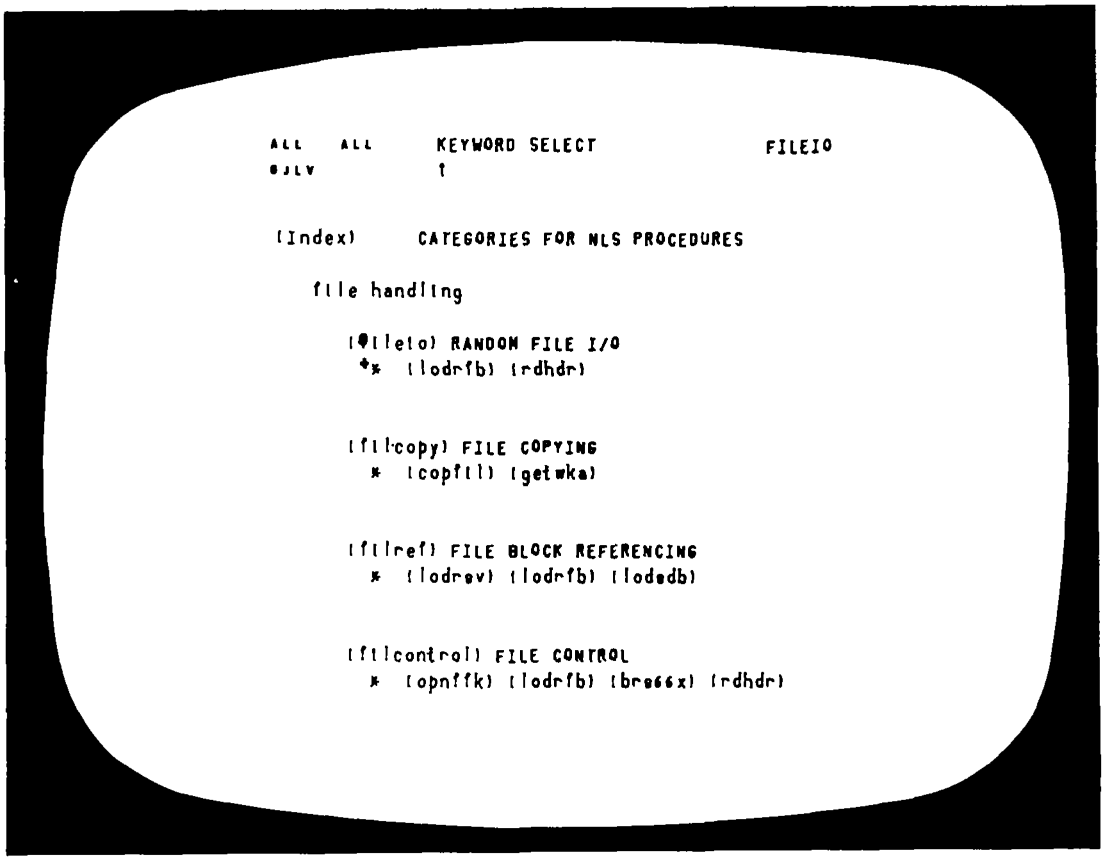
FIGURE III-19 Under DDT we run the new experimental NLS and build a dummy statement to test the new "transpose" command. In this photo the command has been specified and two words selected.



```
ALL   ALL      TRANSPOSE WORD
HJLV           ↑

. DUMMY :
statement is a this.
               ↑
```

TA-7079-30

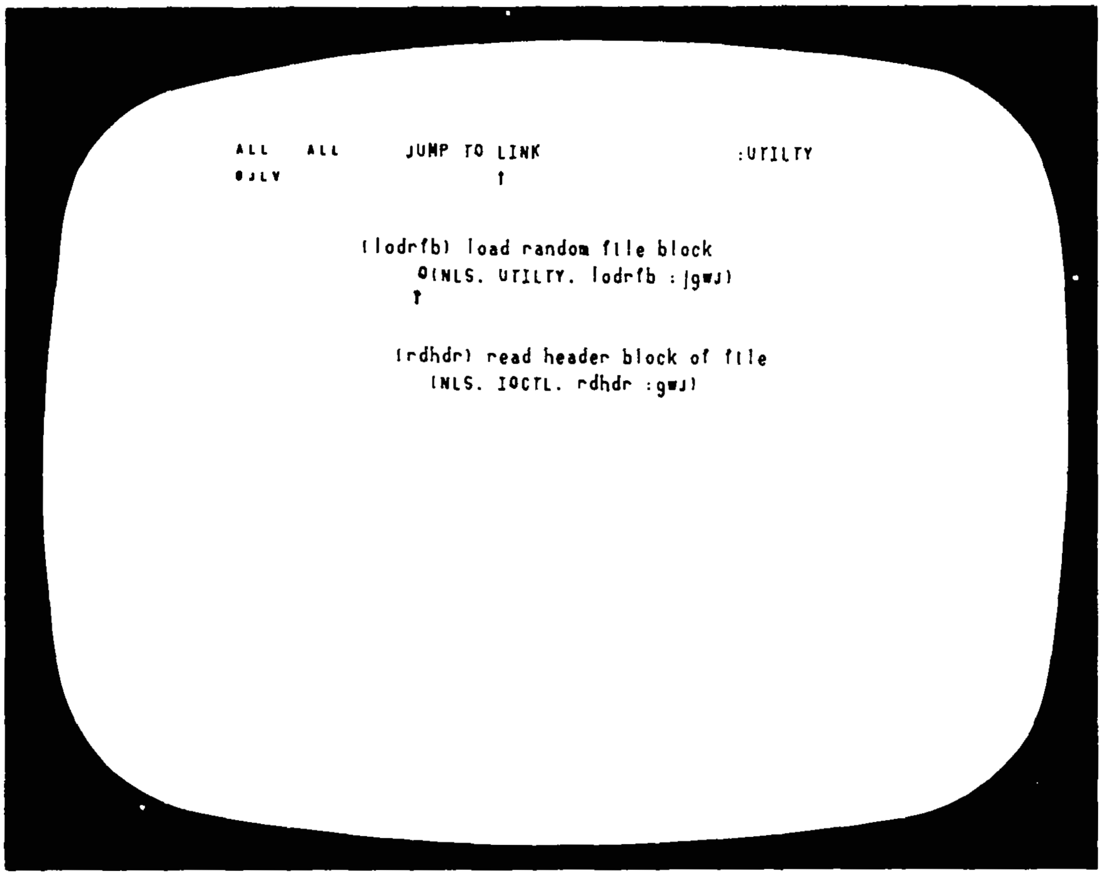
FIGURE III-20 Here the "transpose" command has been executed.



```
ALL   ALL      KEYWORD SELECT                  FILEIO
●JLV           t

(Index)     CATEGORIES FOR NLS PROCEDURES

   file handling

      (fileio) RANDOM FILE I/O
       +*  (lodrfb) (rdhdr)


      (fil-copy) FILE COPYING
        *  (copfil) (getwka)


      (filref) FILE BLOCK REFERENCING
        *  (lodrev) (lodrfb) (lodedb)


      (filcontrol) FILE CONTROL
        *  (opnffk) (lodrfb) (bre66x) (rdhdr)
```

TA-7079-31

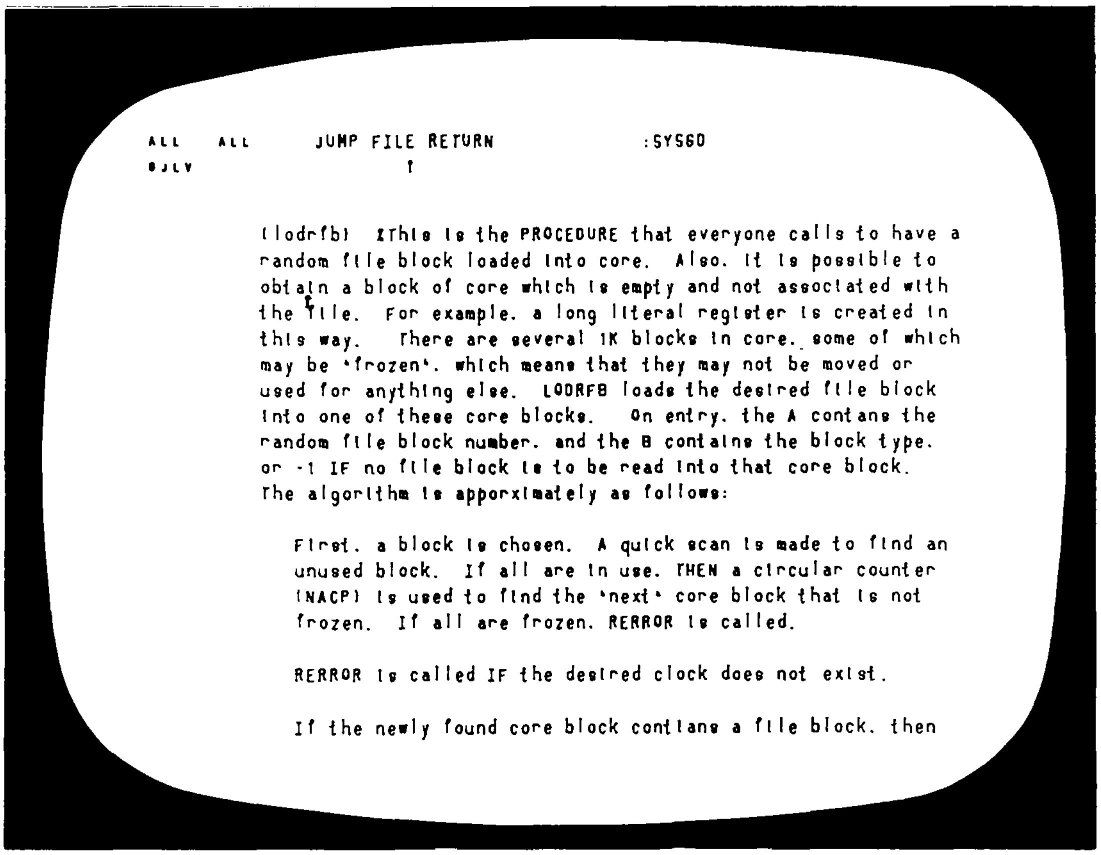
FIGURE III-21 Another branch of the SYSGD file allows one to select procedures according to function with the "Keyword" commands. In this photo the user has selected the word "fileio" as a keyword; we see that the statement named "fileio" contains two other names, "lodrfb" and "rdhdr," in a special format.



```
ALL    ALL       JUMP TO LINK                      :UTILTY
0JLV                     ↑

            (lodrfb) load random file block
                 0(NLS. UTILTY. lodrfb : jgwJ)
                ↑

               (rdhdr) read header block of file
                   (NLS. IOCTL. rdhdr :gwJ)
```

TA-7079-32

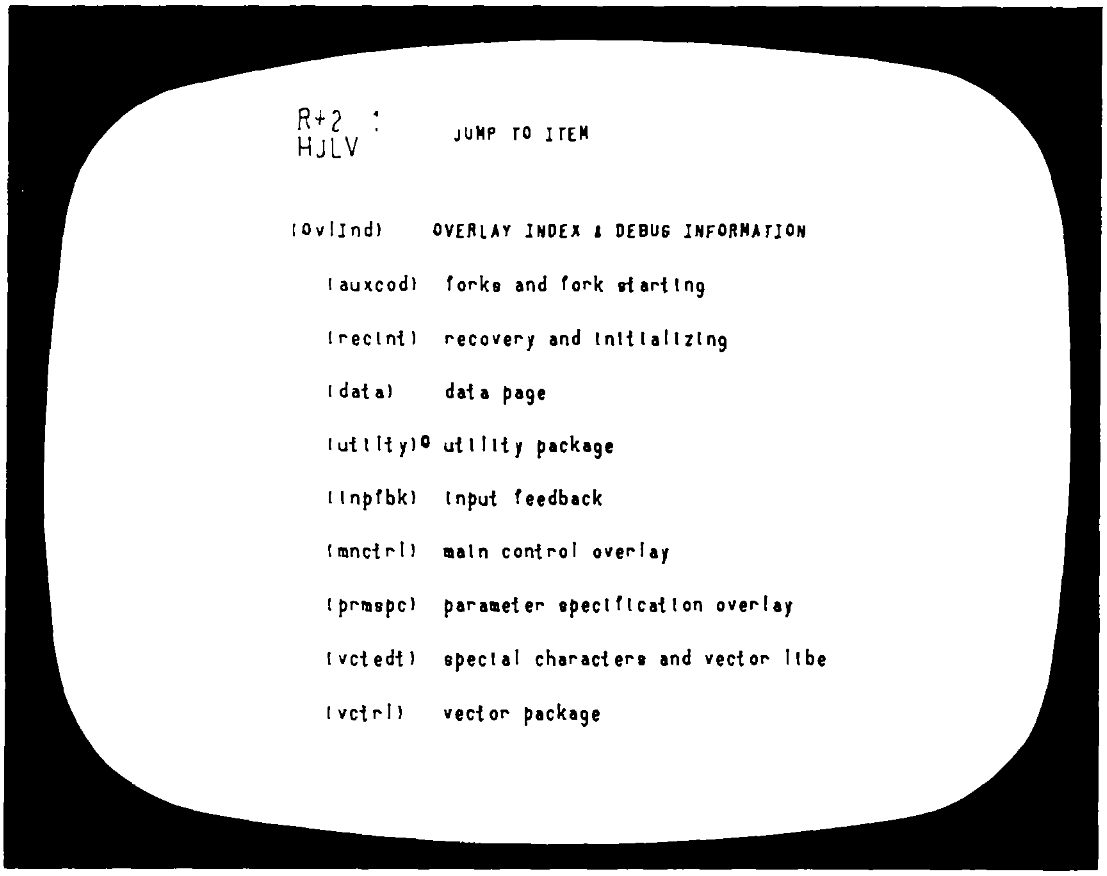
FIGURE III-22 DISPLAY PRODUCED BY KEYWORD SYSTEM. When the selected procedures are presented, one can "Jump," via the associated links, and examine the documentation and code for the procedure.



```
ALL   ALL       JUMP FILE RETURN              :SYSGD
•JLV                    †

          (lodrfb) :This is the PROCEDURE that everyone calls to have a
          random file block loaded into core. Also. it is possible to
          obtain a block of core which is empty and not associated with
          the file. For example. a long literal register is created in
          this way.  There are several 1K blocks in core. some of which
          may be 'frozen'. which means that they may not be moved or
          used for anything else. LODRFB loads the desired file block
          into one of these core blocks.  On entry. the A contans the
          random file block number. and the B contains the block type.
          or -1 IF no file block is to be read into that core block.
          The algorithm is apporximately as follows:

            First. a block is chosen. A quick scan is made to find an
            unused block. If all are in use. THEN a circular counter
            (NACP) is used to find the 'next' core block that is not
            frozen. If all are frozen. RERROR is called.

            RERROR is called IF the desired clock does not exist.

            If the newly found core block contians a file block. then
```

TA-7079-33

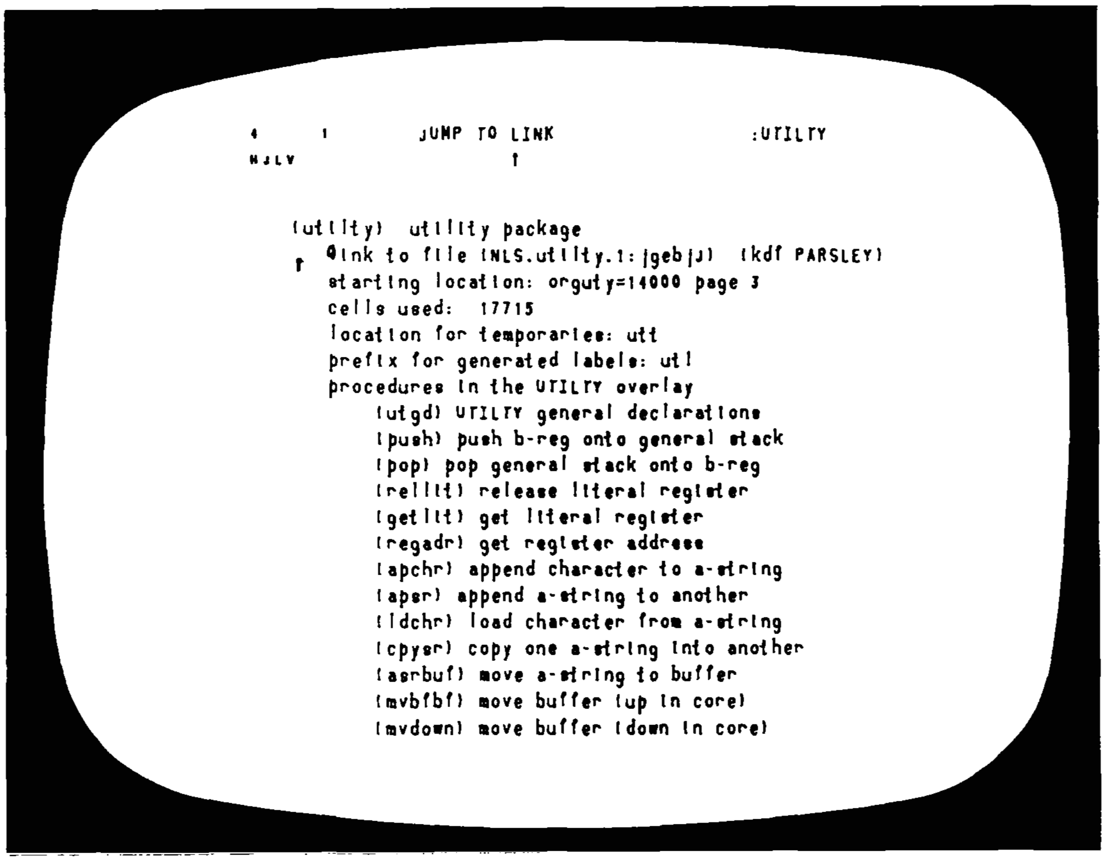
FIGURE III-23 DOCUMENTATION OF **LODRFB** PROCEDURE. The "Jump to Link" command specified in Figure III-22 has been executed, and the user has set up a "Jump File Return" to display the SYSGD file again.



```
R+2 :
HJLV        JUMP TO ITEM

(OvlInd)    OVERLAY INDEX & DEBUG INFORMATION

   (auxcod)  forks and fork starting

   (recint)  recovery and initializing

   (data)    data page

   (utility)0 utility package

   (inpfbk)  input feedback

   (mnctrl)  main control overlay

   (prmspc)  parameter specification overlay

   (vctedt)  special characters and vector libe

   (vctrl)   vector package
```

TA-7079-34

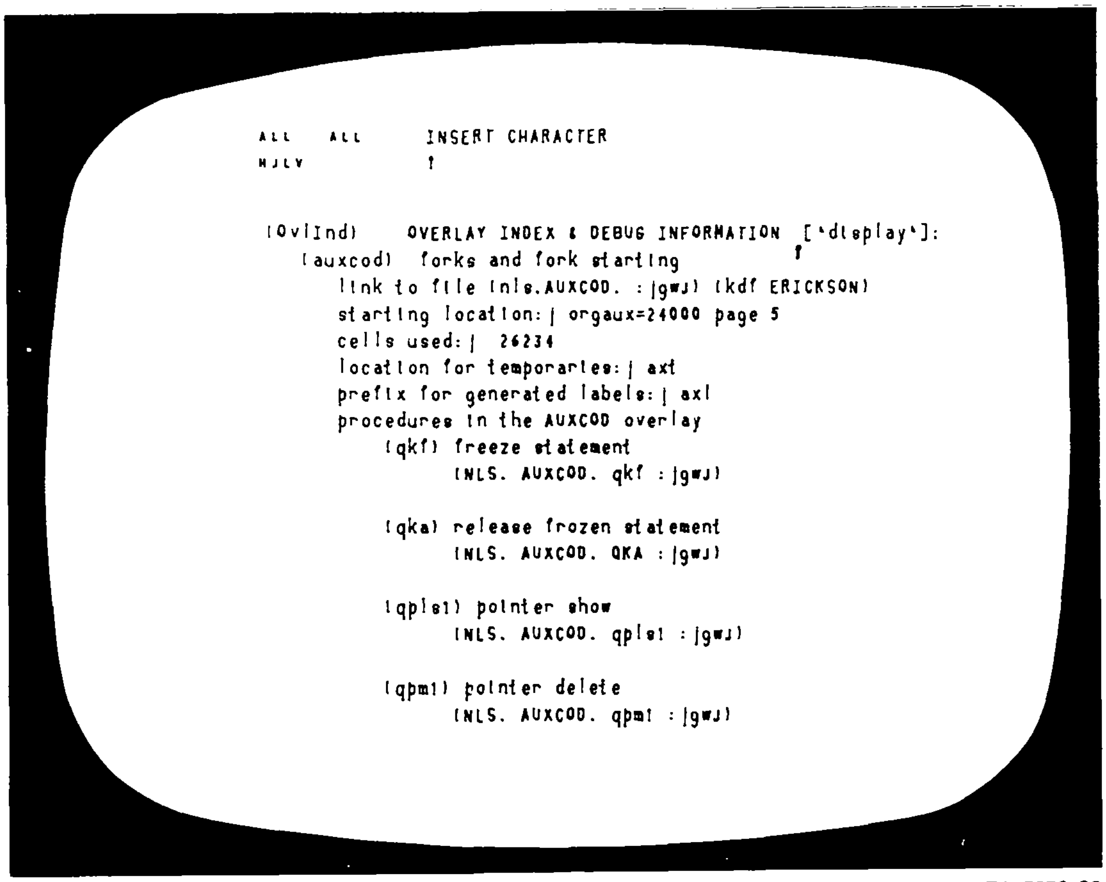
FIGURE III-24 The overlay index branch of the SYSGD file contains information about each overlay in the system. A "Jump" has been specified to display the branch on the utility package. The VIEWSPEC code "R+2" appearing at the upper left corner of the screen shows that in executing the "Jump," two additional structural levels of text are to be displayed.



```
4       1        JUMP TO LINK                  :UTILTY
HJLV                      ↑

    (utility)  utility package
   ↑  @link to file (NLS.utility.1:jgebjj)  (kdf PARSLEY)
      starting location: orguty=14000 page 3
      cells used:  17715
      location for temporaries: utt
      prefix for generated labels: utl
      procedures in the UTILTY overlay
          (utgd) UTILTY general declarations
          (push) push b-reg onto general stack
          (pop) pop general stack onto b-reg
          (rellit) release literal register
          (getlit) get literal register
          (regadr) get register address
          (apchr) append character to a-string
          (apsr) append a-string to another
          (ldchr) load character from a-string
          (cpysr) copy one a-string into another
          (asrbuf) move a-string to buffer
          (mvbfbf) move buffer (up in core)
          (mvdown) move buffer (down in core)
```

TA-7079-35

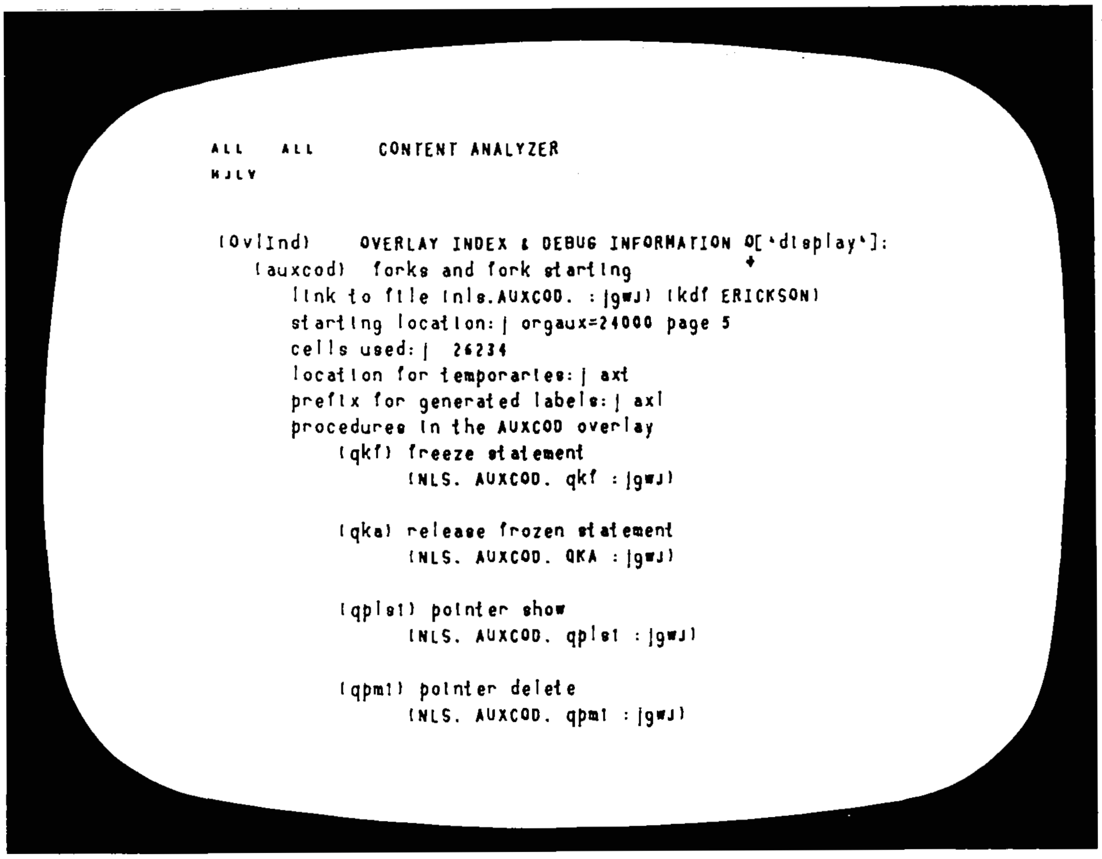
FIGURE III-25 INFORMATION ON THE "UTILITY" OVERLAY, Including Part of the List of Procedures in the Overlay. The first substatement in this branch contains a link which the user is about to follow.



```
ALL    ALL       INSERT CHARACTER
HJLV             ↑


(OvlInd)    OVERLAY INDEX & DEBUG INFORMATION  ['display']:
   (auxcod)  forks and fork starting              ↑
      link to file (nls.AUXCOD. :jgwJ) (kdf ERICKSON)
      starting location:| orgaux=24000 page 5
      cells used:|  26234
      location for temporaries:| axt
      prefix for generated labels:| axl
      procedures in the AUXCOD overlay
          (qkf) freeze statement
                (NLS. AUXCOD. qkf :jgwJ)

          (qka) release frozen statement
                (NLS. AUXCOD. QKA :jgwJ)

          (qplsl) pointer show
                (NLS. AUXCOD. qplsl :jgwJ)

          (qpml) pointer delete
                (NLS. AUXCOD. qpml :jgwJ)
```

TA-7079-36

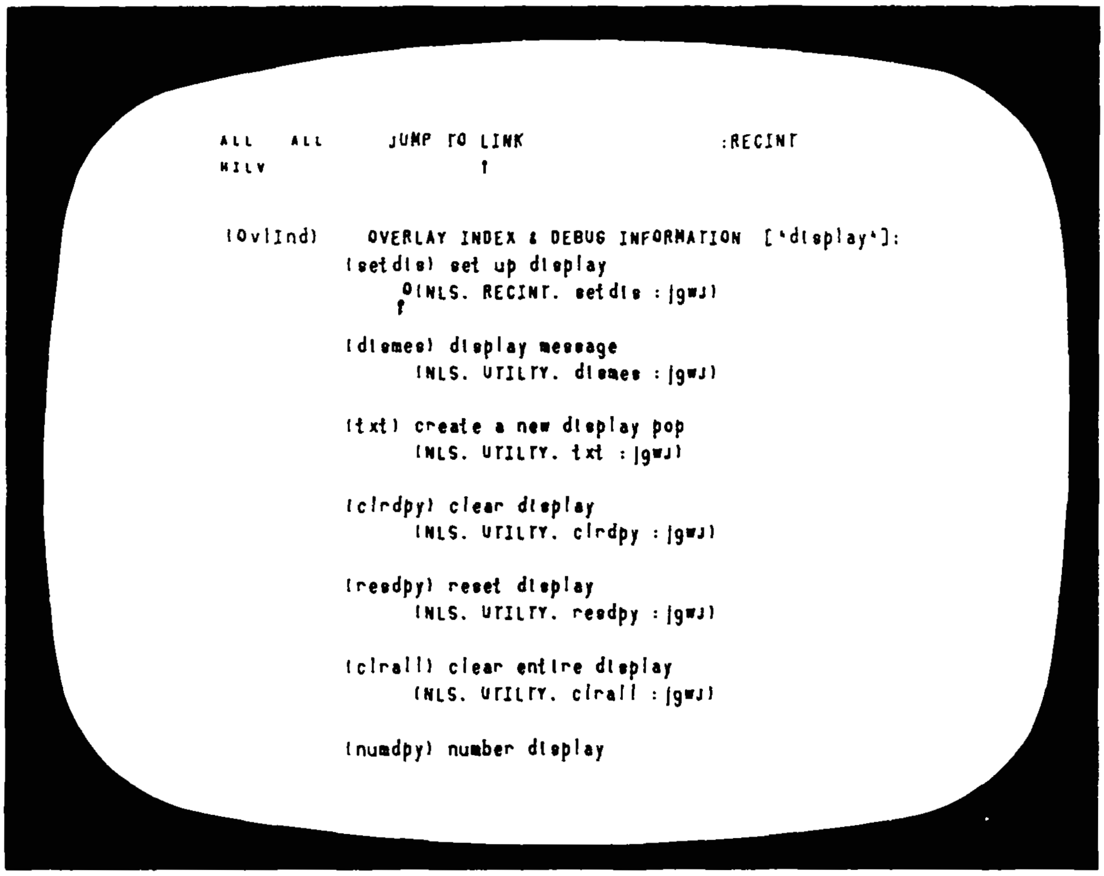
FIGURE III-26 Here the user is back to the SYSGD file (several steps have been omitted from the sequence). In the first line of text on the display he has inserted a content-analyzer pattern for finding procedures containing the string "display."



```
ALL   ALL      CONTENT ANALYZER
HJLV


(OvlInd)    OVERLAY INDEX & DEBUG INFORMATION O['display']:
   (auxcod)  forks and fork starting          +
      link to file (nls.AUXCOD. :jgwJ) (kdf ERICKSON)
      starting location:| orgaux=24000 page 5
      cells used:|  26234
      location for temporaries:| axt
      prefix for generated labels:| axl
      procedures in the AUXCOD overlay
          (qkf) freeze statement
                (NLS. AUXCOD. qkf :jgwJ)

          (qka) release frozen statement
                (NLS. AUXCOD. QKA :jgwJ)

          (qplst) pointer show
                (NLS. AUXCOD. qplst :jgwJ)

          (qpml) pointer delete
                (NLS. AUXCOD. qpml :jgwJ)
```

TA-7079-37

FIGURE III-27 The pattern is compiled with command "Content Analyzer." The display is not yet affected——the content-analysis code compiled from the pattern will be run when the user changes a VIEWSPEC.



```
ALL    ALL       JUMP TO LINK                       :RECINT
HILV                     ↑

(OvlInd)    OVERLAY INDEX & DEBUG INFORMATION  ['display']:
            (setdis) set up display
                °(NLS. RECINT. setdis : jgwJ)
                ↑

            (dismes) display message
                 (NLS. UTILTY. dismes : jgwJ)

            (txt) create a new display pop
                 (NLS. UTILTY. txt : jgwJ)

            (clrdpy) clear display
                 (NLS. UTILTY. clrdpy : jgwJ)

            (resdpy) reset display
                 (NLS. UTILTY. resdpy : jgwJ)

            (clrall) clear entire display
                 (NLS. UTILTY. clrall : jgwJ)

            (numdpy) number display
```

TA-7079-38

FIGURE III-28 RESULT OF CONTENT ANALYSIS: a list of procedures that in some way deal with the display (since they contain the word "display")

Section III
APPLICATIONS AND EXPERIENCE

C. The Augmented Manager

1. Introduction

a. General Considerations

Most people need to play the role of "manager" at various times, whether that role is their primary function or simply a necessary element for the accomplishment of their other work.

We will use the terms "manager" and "management" in this expanded sense, applying them to an individual's management of his own time, resources, and skills as well as to individual and collective management of the work of other individuals and groups.

Our experience with augmented management can be described best in terms of the augmentation tools we have developed and the methods we have evolved for exploiting these tools in the task of management.

Once a tool goes into use, methodology becomes as complex and critical a concern as the design and development of the tool itself, and our research in the area of augmenting management has been concerned principally with the development of special methodologies for using our basic augmentation tools with only secondary emphasis on the development of new special-purpose tools.

We have identified the following as some of the major needs relevant to fulfilling the role of manager in our augmented community.

(1) Fast and flexible means for accessing and studying management working files and other group working files

(2) Techniques for locating specific pieces of information in a complex file or collection of files

(3) Efficient methods for entering, updating, and storing management information

(4) Techniques for interacting with other group

members to verify and comment on information within the group's working records

(5) Facilities for rapid and flexible composition, modification, and publication of management documentation

(6) Special tools for the processing of management information (e.g., the NLS calculator facility).

Some of the specific tools and methods developed in response to the needs outlined above are described briefly below.

Our collection of on-line files now contains considerable management information of various kinds, and we have pursued our investigation of management-information operations by using NLS and TODAS to provide and develop aids for management of the ARC on-line community.

There are many areas of potential application for on-line aids, and we have chosen those which appear to be most useful operationally for the purposes of experimentation and development.

b. Cost-Accounting Files

While still under continuous development, special cost-accounting files (described in detail in Section III-C-2 below) are now in routine use. These files are extremely useful in terms of getting work done, and our experience with developing and using them has constituted some of our most fruitful research on the intensive exploitation of the overall on-line system.

The files have an intricate structure, designed for maximum mobility in accessing information and for maximum efficiency in interfacing with the NLS calculator facility. Intelligent use of these files includes the entry of new information and the occasional restructuring of the information to take advantage of new possibilities in the system and to meet newly discovered needs. While fundamentally simple, this kind of usage requires the coordinated operation of the most advanced features of NLS; thus the cost-accounting files are a leading edge of our research on highly interactive information structures.

c. Work-Planning Files

We have begun experiments in using NLS files for the planning and monitoring of tasks within our project activities. Our first experiment involved the management of the TODAS development activity, for which we created a file called UPLAN to use in formulating specifications for TODAS and in planning for the implementation of the specified features.

This file contains a branch for each identified task with the following information organized for easy study and retrieval:

(1) Description of the task, with links to other working files used in its development

(2) Comments on the relationship of the task to other ARC tasks

(3) Estimates of implementation costs (mainly manpower) and schedule of work and implementation stages

(4) Comments on the current status of planning and implementation.

We also created a separate file called UMEET containing agendas and notes for the TODAS development activity meetings.

We made use of a research assistant working on-line to take notes during these meetings. The research assistant worked from an agenda composed before the meeting so as to have a convenient framework for note-taking, and was also available for finding on-line information as needed during the meetings. Successive meeting agendas and minutes were kept in a single file so as to make it easy for us to search for records of all discussion related to any given topic.

On a less formal basis, we have used various other files for work planning throughout the project's history.

This has ranged from the construction of lists of tasks in such a way as to reflect their interdependencies, to the maintenance by individuals of files listing their personal tasks on hand along with plans for completing these tasks.

d. Personnel Files

Some of our personnel information is now kept in on-line files. Features in the timesharing system permit us to restrict access to some of these files, while letting others remain "public." Even though our personnel roster is relatively small, we find it advantageous to have the ability to specify automatic searches on the personnel files, especially for such tasks as cost estimation.

e. Documentation

All of our documentation, from large final reports to the brief circulars distributed at conferences, is composed, stored, and updated in on-line files. Our flexible composition and editing capabilities and fast publication technology (from on-line file to formatted hardcopy in a few seconds per page) give management controls over the documentation process that are not possible without augmentation.

f. On-Line Dialogue

We use the term "dialogue" to indicate the process of formal communication via interactions with a common information base. This kind of dialogue plays an important role in the formulation and monitoring of plans for project activities, and while we are still using informal experimental methods to achieve this, we are in the process of formulating the initial plans for a long-term investigation of advanced techniques for augmenting dialogue processes.

Subsequent sections offer detailed descriptions of several management applications that have been developed using NLS and TODAS, illustrated with photographs taken from NLS display screens to show sequences of information-manipulation operations. A familiarity with the basics of NLS (to the extent that they are explained in Appendix A) is assumed.

In following the descriptions, it should be kept in mind that the speed with which NLS serves its users is an important part of its utility. The photographs indicate transitions that take only one or two seconds under good conditions. This speed lends great power and flexibility to the relatively simple service functions performed by NLS.

2. Project Cost Records

In our routine operations, we need frequent and rapid access to summary data on project costs, with little reference to lower-level details except when the costs are first checked for reasonableness and accuracy. Therefore we decided to start by putting summary data on-line at ARC. As needed in the future, we can add more levels of detail.

We first constructed a cost-history file for 1968-1969 costs on SRI projects ESU 7101 (RADC Contract F30602-68-C-0286) and ESU 7079 (NASA Contract NAS 1-7897). This file is called HISCO.

The elements of HISCO included the following for each of the two projects, on the basis of 4-week accounting periods (as used by SRI's accounting system):

(1) Salary

(2) Burden

(3) Overhead

(4) Total cost

(5) Fee

(6) Total charges.

See Figures III-29, III-30, and III-31 (illustrations for this section are grouped together, beginning on page 75). Each of these three figures shows a display of one branch of the file, containing the information for a specific project and year.

We also needed a section showing combined salary costs and combined total charges for all of our projects (see Figures III-32 and III-33). We put these costs in separate branches of the file. The last branch shows total costs for both projects combined. We retroactively studied existing records for all 1968 data and kept up the 1969 costs every 4 weeks, entering the new data by hand.

The usual way of accessing HISCO was via pre-established links from other working files whenever the user had a question about recent costs. The VIEWSPECS in the link usually caused HISCO to be brought in with only high-level statements on display, showing only the headings for project name, combined salary, total charges, and total ARC costs (see Figure III-34).

The user could then select the project he was interested in (by the command Jump to Item), open up an additional level for viewing, and see column headings and numerical data (Figures III-29, III-30, and III-31).

Then he could jump down through the accounting periods to the one he was looking for.

If he was making a calculation (perhaps already started in the file he was working in before he loaded HISCO), he could then call the calculator and add, subtract, multiply or divide by any of the numbers in HISCO. His previous calculations in the previous file would remain intact.

If finished with HISCO, he could then return to the previous file (by the command Jump File Return) and continue with the calculation, having found in HISCO the input number or numbers he was looking for.

As an arena for experimentation, HISCO proved valuable. Operationally, it was useful from time to time but revealed a need for more frequent updating of the summary data. Our experience with HISCO led to the development of a redesigned cost-history file called COSTS which is updated weekly, with 4-week and cumulative summaries.

The cost elements in this file are:

(1) Salary costs

(2) Total personnel costs

(3) Non-labor costs

(4) Total costs

(5) Total charges with fee

(6) Balance remaining.

See Figures III-35, III-36, and III-37. Figures III-35 and III-36 show the same branch of the file with different VIEWSPECS; Figure III-36 displays one more level than Figure III-35, and this level shows the weekly data. Figure III-37 shows the weekly data for another project.

We also included funding information showing current totals, unfunded totals, and total contract amounts in the "cost," "fee," and "total" categories.

We use separate branches for each project and for total ARC project costs (Figure III-38). The skeleton format for the file was set up in advance for the entire year of 1970.

Before entering any actual data, we copied the first top-level branch (containing some 70 statements) within the file at the same level four or five times so as to create blank format branches. Then we simply inserted the project name headings for each project into a corresponding blank branch. We keep one blank format branch in the file in case any new projects should arrive.

Like HISCO, COSTS is usually reached through a link from some other working file, perhaps while a study of near-future costs is in progress, or from a proposal cost estimate under development. Again the file is usually entered with only the top-level statements or project headings showing (see Figure III-39).

If a particular project is of interest, the corresponding branch is selected and another level opened for view. The second level shows period-by-period subtotals in each cost category. If weekly data are desired, another level is opened by changing the VIEWSPECS and a particular week is selected by the command Jump to Item.

The statement for each week has the week-ending date as its name. The reason for using this form of name construction is not only so that the statement for a particular week can be accessed by the Jump to Name command using the ending date, but also so that the date may optionally be suppressed from the display. (NLS has the capability of suppressing all statement names from the display.)

The normal way of viewing these particular files is with names suppressed; thus the dates do not clutter the display. However, a user who needs to know the ending date for a particular week can see it by executing a single command.

To access the information for another project within COSTS, one executes Jump to Return twice to see the top-level statements again (Figure III-39).

One can move very quickly and accurately through a file that is set up in this fashion, even without any familiarity with the information it contains.

The primary function of COSTS is to show a consistent week-by-week progression, by category, of costs for each project. The file can also be used for study purposes, through the use of content-analyzer patterns, some of which are stored in the origin statement (see Figure III-40, which is the same as Figure III-39 but with different VIEWSPECS). Any other patterns can be created as needed.

This allows a user to extract special categories of information from the file very quickly. For example, a user may easily create a display showing all project costs for the eighth week of 1970, for each ARC project. It is also possible to output such a "filtered" display via a line printer, thus obtaining hard copy of a special-purpose extract from the total file.

The content analyzer is helpful when using the calculator on all the data for one week, project by project, to find total ARC charges by category.

When only one week's data are displayed (Figure III-41), one can add items down each column and insert the answer in the "ARC total" space. One can then clear the accumulator, and add down the next column. This is done very rapidly through bug selection of input numbers and keyset entry of commands -- Add, Add, Add, Add, Insert, Clear, Add, Add, Add, Add, Insert, Clear, and so forth.

The COSTS file is now operationally useful to us, and we expect it to be useful for future experimentation with automatic processing techniques.

3. Estimates for Proposals

a. Personnel Costs

An estimator working on a proposal can select from an on-line file, by labor category, representative people who may be involved with the proposed work; as he selects them, he can transfer their names and relevant information about them to a file where he is building up his estimate.

The estimator loads a special file, maintained by himself, which is a directory to all of his other files and perhaps to a few files belonging to other people. Figures III-42 and III-43 are two displays of a user's file directory. In Figure III-42, only first-level statements are shown; these are used for establishing categories. In Figure III-43, another level is shown, containing the actual directory listings in each category.

This "file directory" contains links to each of the files that it lists. In the present case the files probably would be cost histories, personnel listings, previous special studies of costs, and other administrative information.

He loads a previous cost estimate, makes a working copy of it, changes the heading to reflect the name of the new proposal estimate, and eliminates the amounts from the old estimate.

This produces a blank cost-estimate format. If any items from the old estimate are inappropriate, they are easily deleted; new items are easily added as separate statements. When the format is ready, it is output as a new file.

He can then load a file that lists names of people in the group and some projection of expected additions. Figures III-44, III-45, and III-46 show portions of such a file.

Using this personnel-listing file, he obtains information about labor categories. A branch containing content-analyzer patterns is kept in the file. These can be easily reached by jumping to a link that causes all the patterns to be displayed (Figure III-47).

Each pattern will select some particular category of statements from the file. For example, the estimator will need to know which people have the status of Senior Professional.

He selects the appropriate pattern with the command Execute Content Analyzer, and then jumps on a link that turns on the content analyzer, starting the search at the beginning of the branch containing personnel listings and restricting the search to that branch.

This produces a display showing only the listing of senior professionals in the group. This set of statements can then be transferred to the new proposal cost-estimate file.

Other patterns can be used to extract sets of statements according to other criteria -- for example, all the hardware or software people in the group (Figures III-48 and III-49).

Thus the estimator can select, by labor category, representative people who may be involved with the proposal; as he selects them, he can transfer their names and the information that goes with them to the file where he is building up his estimate.

The payroll burden and overhead rates are checked for currency and inserted into the estimate, using the calculator to apply them to the direct labor. At this point the labor portion of the estimate is completed.

b. Non-Labor Costs

A typical estimate will involve some travel costs, some consultant costs, and some report costs. Data supporting the cost of consultants may be checked by reviewing current consultants' costs by project and by consultant. These are kept in a separate file and reached through a link for review. The data may be copied into the estimate if desired.

Report-production costs are estimated using current Institute schedules, which are based primarily on the number of pages expected in the end product. These computations can be made using the calculator and the existing cost factors from the last proposal (checked for current applicability).

In addition, the proposal may contain plans to add equipment. In this case, the estimator will use an equipment study written in another file by the people involved in hardware design.

The equipment costs contained in the special study are summarized in total and reached by a link. The special study can be viewed and updated as appropriate and can be copied to go with the proposal as an appendix or used later for backup.

In this fashion, various information is gathered from various files and transferred into the developing cost estimate. Figures III-50, III-51, and III-52 show portions of a completed on-line cost estimate actually used for a recent ARC proposal.

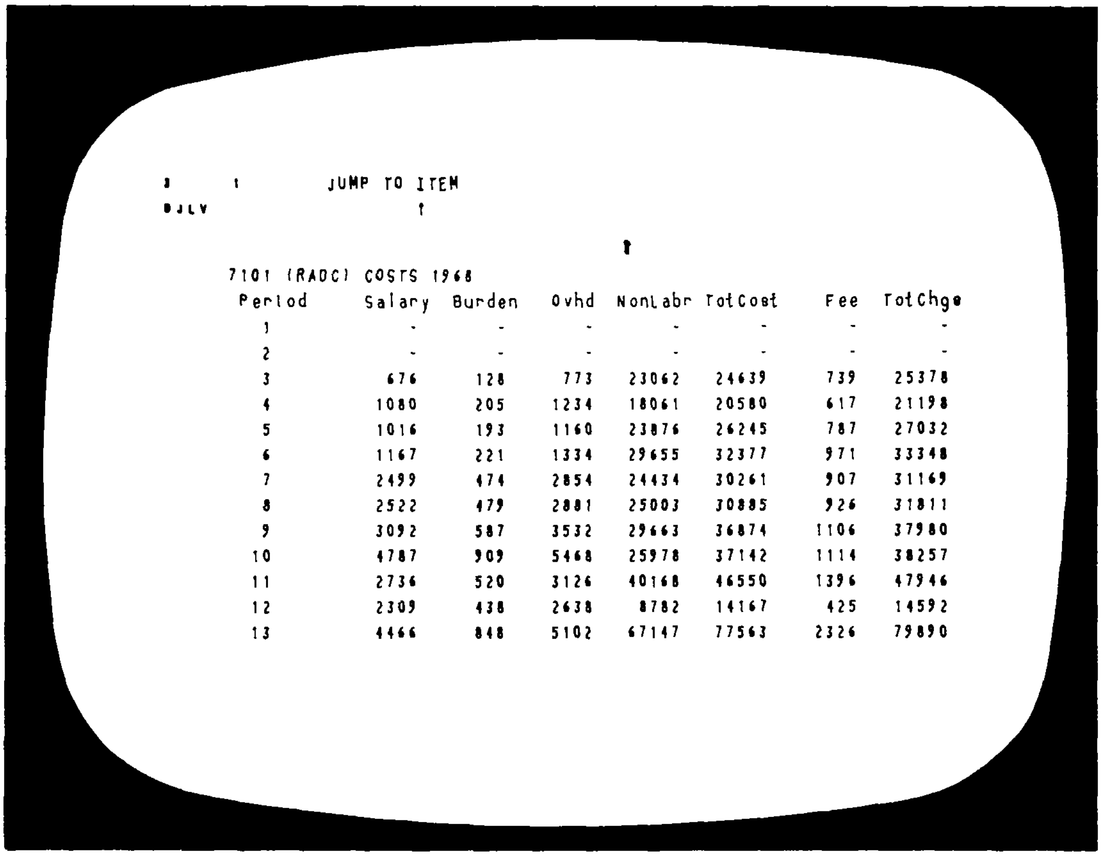
4. Purchase-Order Processing

In making estimates of costs for new equipment being constructed at ARC, reference to previous cost information is very useful. We constructed a purchase-order/requisition processing file that contains a separate statement for each item purchased for the past few years. Figure III-53 shows a portion of this file.

All outstanding orders are contained at a second level within a single branch (see Figure III-54); therefore the distinction between outstanding and completed orders is easy to see by reference to level. To reduce clerical error, we consider an order completed when the comp pattern is inserted and the statement is moved to its alphabetical position on the top level.

This file can be searched using the content analyzer in some interesting ways. If we wonder what we purchased on PR A08927, the question can be answered simply by executing a content-analyzer pattern specifying the number. We can quickly see all outstanding orders charged to a particular project. Figure III-55 shows a content-analyzer pattern that has been temporarily written into the file, for finding any entries pertaining to orders for relays under Project 7101. Figure III-56 shows a view generated by using this pattern.

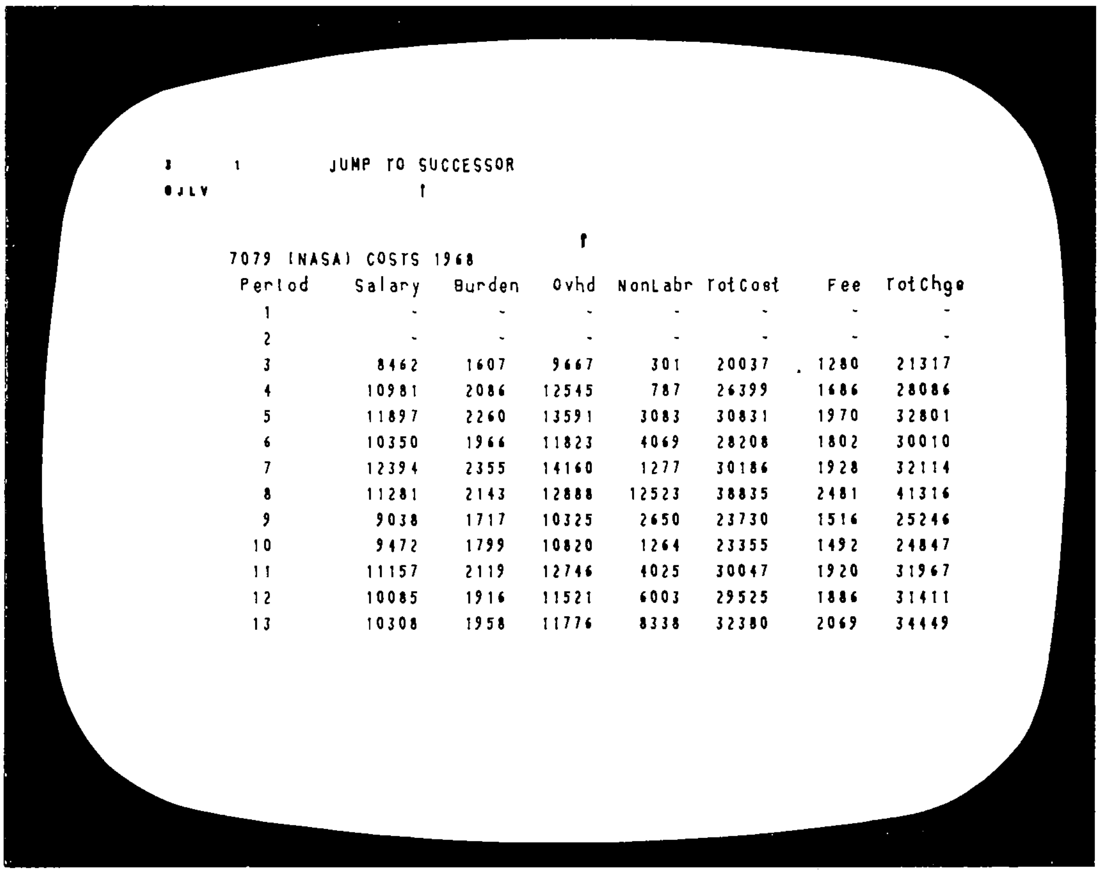
This file is kept up-to-date by the secretary of the hardware group, who is most involved with requisitioning. She does this updating entirely with TODAS.



```
3       1          JUMP TO ITEM
BJLV                     t

                                             t
    7101 (RADC) COSTS 1968
     Period      Salary  Burden    Ovhd  NonLabr TotCost     Fee  TotChge
          1           -       -       -        -       -       -        -
          2           -       -       -        -       -       -        -
          3         676     128     773    23062   24639     739    25378
          4        1080     205    1234    18061   20580     617    21198
          5        1016     193    1160    23876   26245     787    27032
          6        1167     221    1334    29655   32377     971    33348
          7        2499     474    2854    24434   30261     907    31169
          8        2522     479    2881    25003   30885     926    31811
          9        3092     587    3532    29663   36874    1106    37980
         10        4787     909    5468    25978   37142    1114    38257
         11        2736     520    3126    40168   46550    1396    47946
         12        2309     438    2638     8782   14167     425    14592
         13        4466     848    5102    67147   77563    2326    79890
```

TA-7079-39

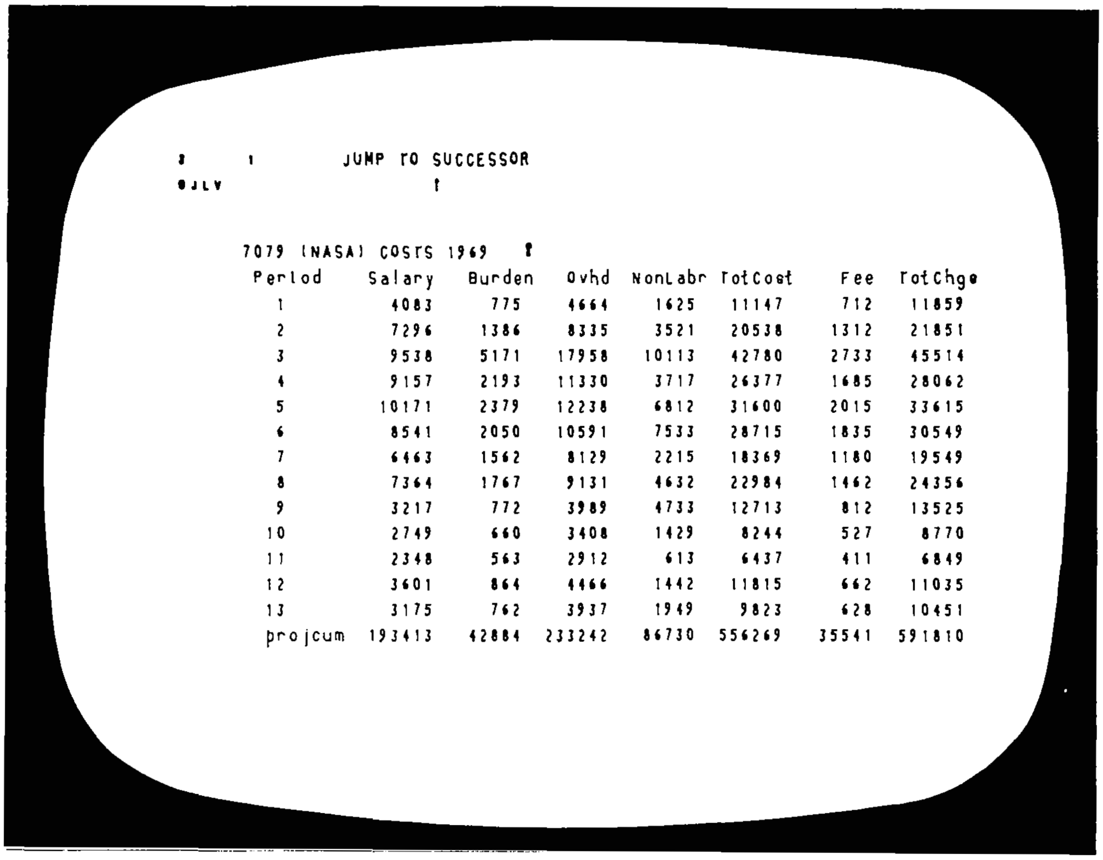
FIGURE III-29 A BRANCH OF FILE HISCO



```
3      1          JUMP TO SUCCESSOR
@JLV                      ↑

                                              ↑
   7079 (NASA) COSTS 1968
    Period     Salary   Burden    Ovhd  NonLabr TotCost     Fee  TotChge
         1          -        -       -        -       -       -        -
         2          -        -       -        -       -       -        -
         3       8462     1607    9667      301   20037  . 1280    21317
         4      10981     2086   12545      787   26399    1686    28086
         5      11897     2260   13591     3083   30831    1970    32801
         6      10350     1966   11823     4069   28208    1802    30010
         7      12394     2355   14160     1277   30186    1928    32114
         8      11281     2143   12888    12523   38835    2481    41316
         9       9038     1717   10325     2650   23730    1516    25246
        10       9472     1799   10820     1264   23355    1492    24847
        11      11157     2119   12746     4025   30047    1920    31967
        12      10085     1916   11521     6003   29525    1886    31411
        13      10308     1958   11776     8338   32380    2069    34449
```

TA-7079-40

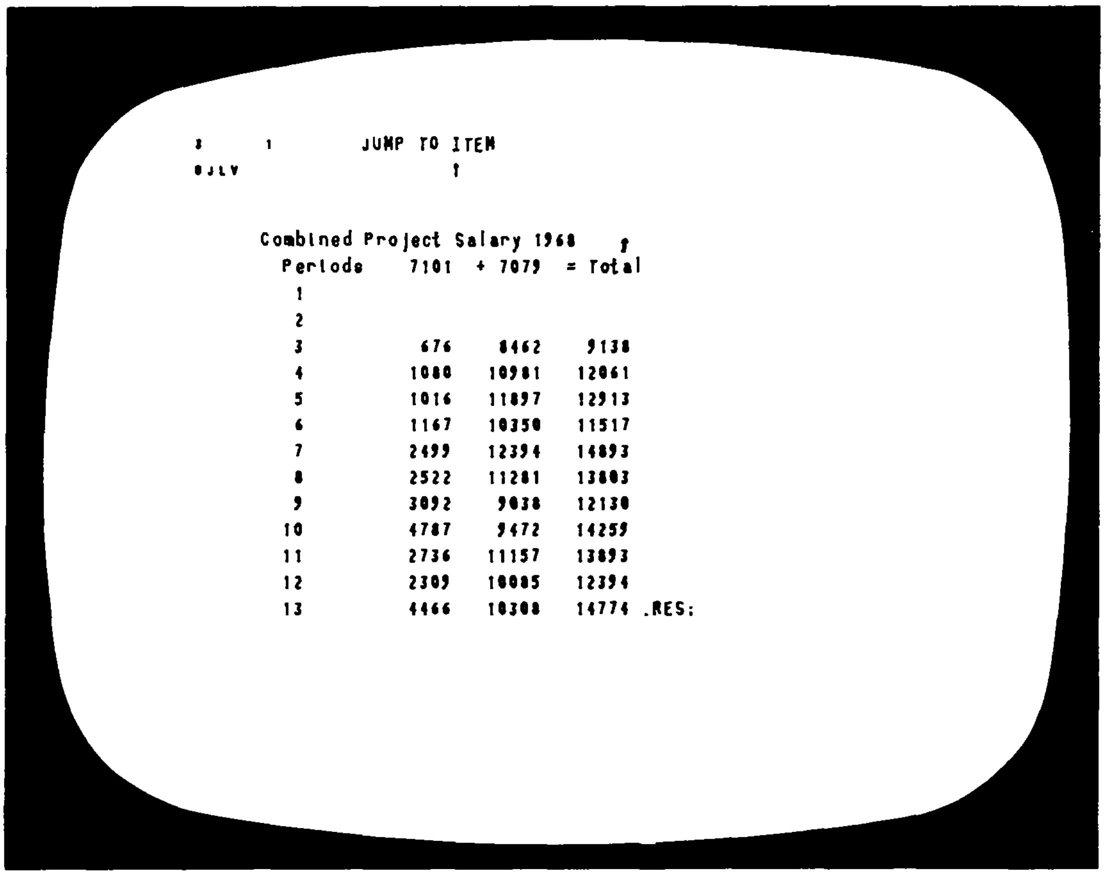
FIGURE III-30   A BRANCH OF FILE HISCO



```
•      ı        JUMP TO SUCCESSOR
•JLV                     !


    7079 (NASA) COSTS 1969    !
     Period      Salary    Burden     Ovhd  NonLabr TotCost     Fee  TotChge
       1           4083       775     4664     1625   11147     712    11859
       2           7296      1386     8335     3521   20538    1312    21851
       3           9538      5171    17958    10113   42780    2733    45514
       4           9157      2193    11330     3717   26377    1685    28062
       5          10171      2379    12238     6812   31600    2015    33615
       6           8541      2050    10591     7533   28715    1835    30549
       7           6463      1562     8129     2215   18369    1180    19549
       8           7364      1767     9131     4632   22984    1462    24356
       9           3217       772     3989     4733   12713     812    13525
      10           2749       660     3408     1429    8244     527     8770
      11           2348       563     2912      613    6437     411     6849
      12           3601       864     4466     1442   11815     662    11035
      13           3175       762     3937     1949    9823     628    10451
      projcum    193413     42884   233242    86730  556269   35541   591810
```

TA-7079-41

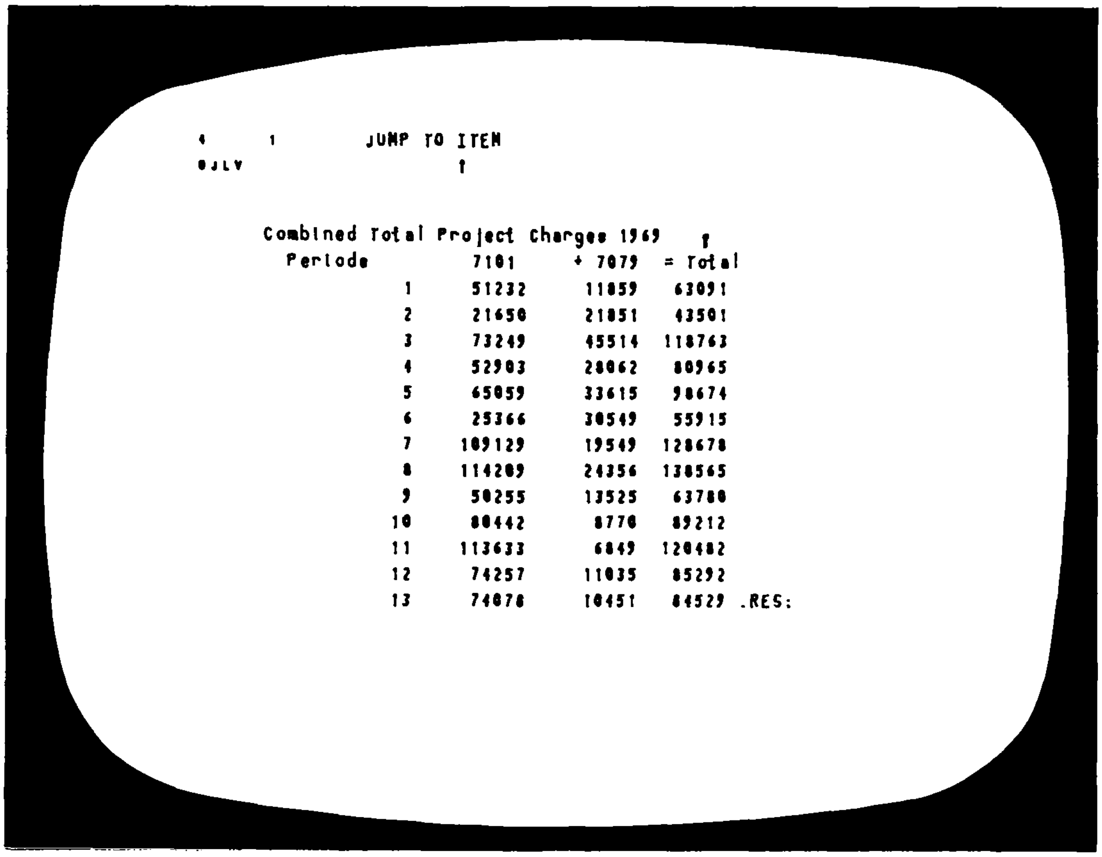
FIGURE III-31   A BRANCH OF FILE **HISCO**



```
3       1          JUMP TO ITEM
8JLV                        ↑


        Combined Project Salary 1968     ↑
          Periods     7101   + 7079   = Total
            1
            2
            3           676     8462     9138
            4          1080    10981    12061
            5          1016    11897    12913
            6          1167    10350    11517
            7          2499    12394    14893
            8          2522    11281    13803
            9          3092     9038    12130
           10          4787     9472    14259
           11          2736    11157    13893
           12          2309    10085    12394
           13          4466    10308    14774 .RES:
```

TA-7079-42

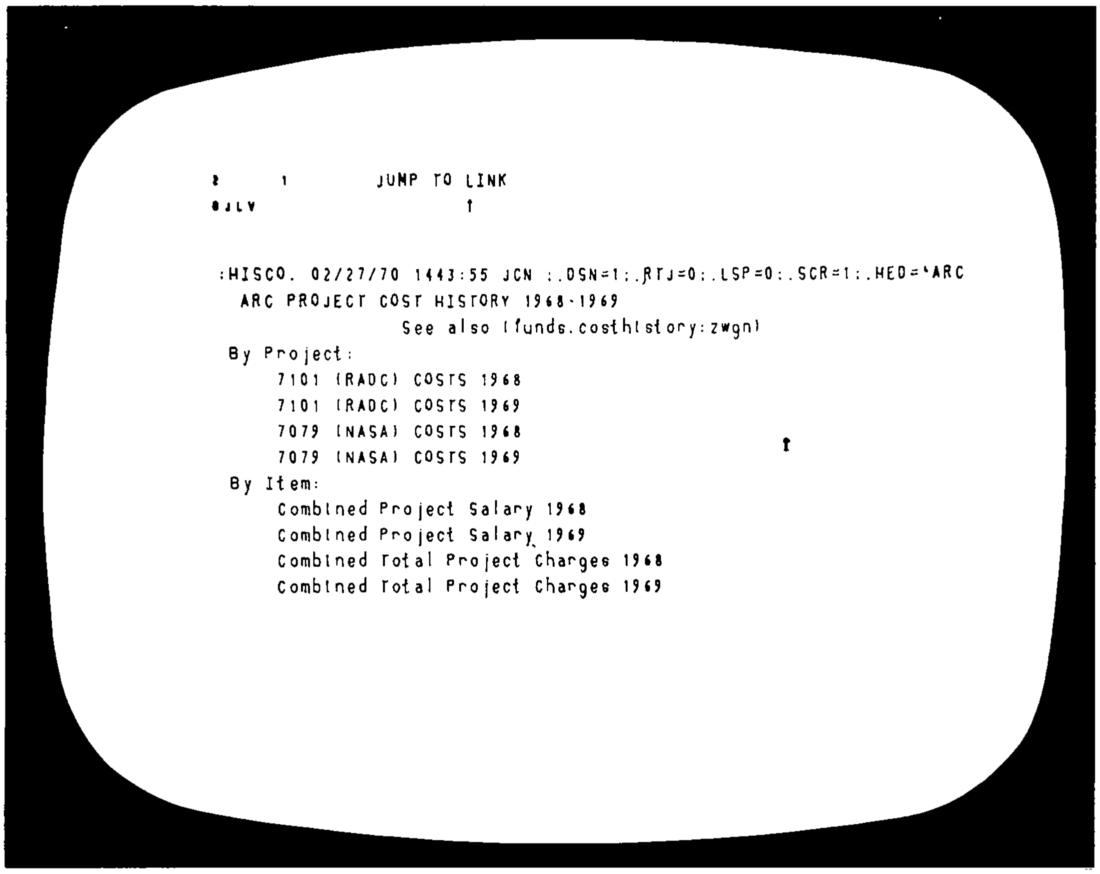
FIGURE III-32   A BRANCH OF FILE **HISCO**



```
4       1          JUMP TO ITEM
0JLV                    ↑


     Combined Total Project Charges 1969   ↑
       Periods           7101      + 7079   = Total
                    1     51232     11859    63091
                    2     21650     21851    43501
                    3     73249     45514   118763
                    4     52903     28062    80965
                    5     65059     33615    98674
                    6     25366     30549    55915
                    7    109129     19549   128678
                    8    114209     24356   138565
                    9     50255     13525    63780
                   10     80442      8770    89212
                   11    113633      6849   120482
                   12     74257     11035    85292
                   13     74078     10451    84529 .RES:
```

TA-7079-43

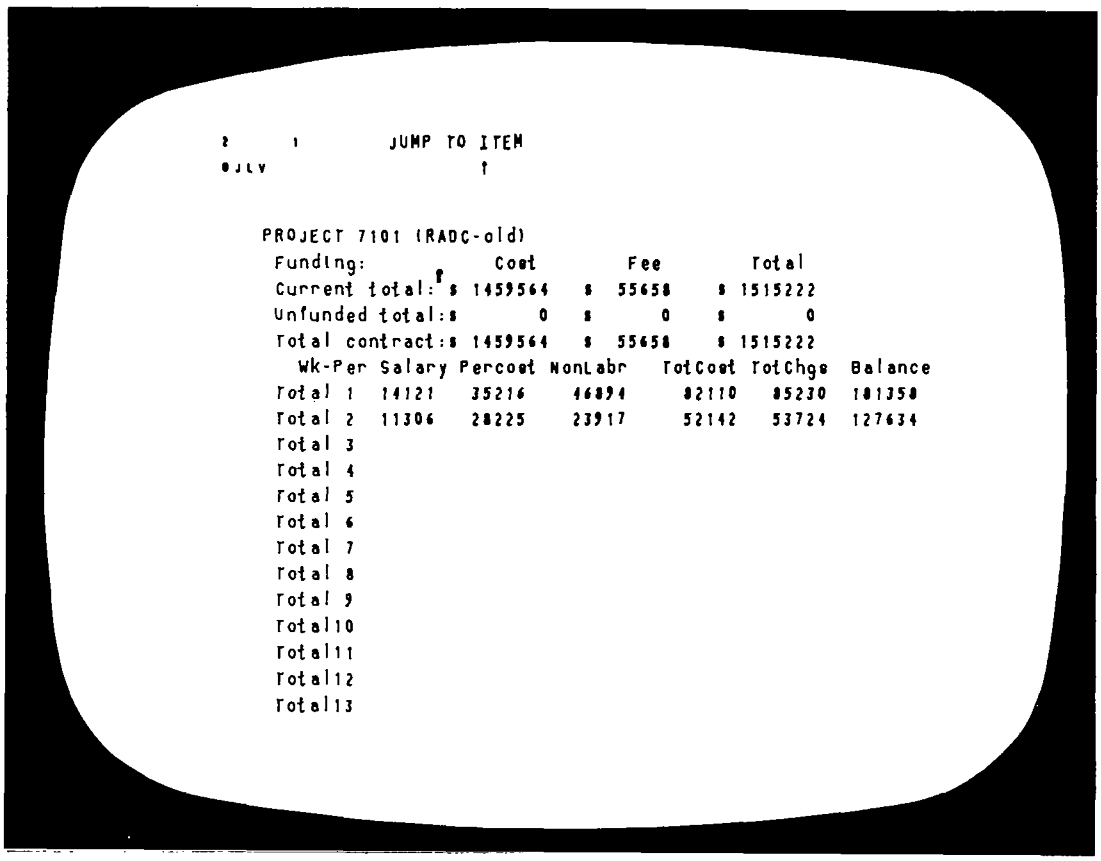
FIGURE III-33   A BRANCH OF FILE **HISCO**



```
1      1          JUMP TO LINK
JLV                       ↑


:HISCO. 02/27/70 1443:55 JCN :.DSN=1:.RTJ=0:.LSP=0:.SCR=1:.HED='ARC
  ARC PROJECT COST HISTORY 1968-1969
                  See also (funds.costhistory:zwgn)
 By Project:
     7101 (RADC) COSTS 1968
     7101 (RADC) COSTS 1969
     7079 (NASA) COSTS 1968                                  ↑
     7079 (NASA) COSTS 1969
 By Item:
     Combined Project Salary 1968
     Combined Project Salary 1969
     Combined Total Project Charges 1968
     Combined Total Project Charges 1969
```

TA-7079-44

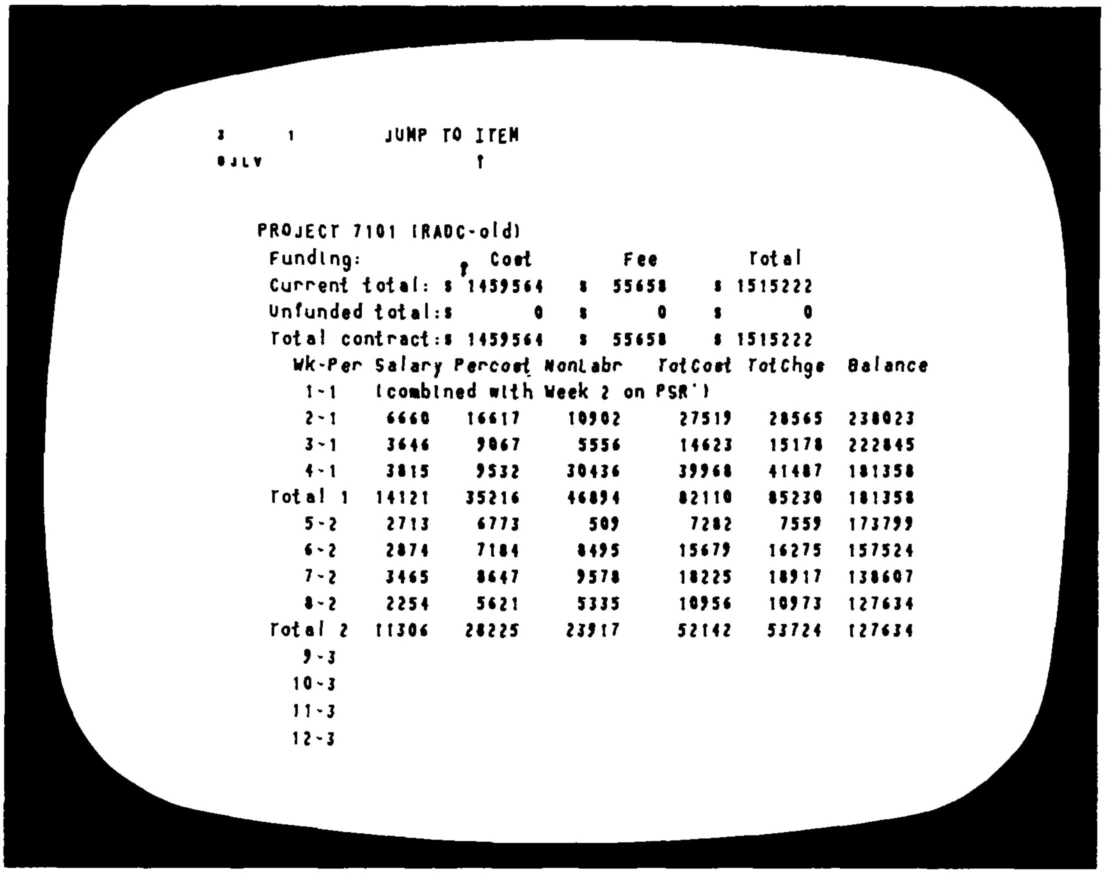
FIGURE III-34   INITIAL VIEW OF FILE **HISCO** UPON ENTRY VIA LINK



```
2        1           JUMP TO ITEM
●JLV                        †

    PROJECT 7101 (RADC-old)
     Funding:                Cost         Fee         Total
     Current total: $ 1459564    $  55658     $ 1515222
     Unfunded total:$       0    $      0     $       0
     Total contract:$ 1459564    $  55658     $ 1515222
       Wk-Per Salary Percost NonLabr   TotCost TotChgs  Balance
     Total 1  14121   35216   46894     82110   85230  181358
     Total 2  11306   28225   23917     52142   53724  127634
     Total 3
     Total 4
     Total 5
     Total 6
     Total 7
     Total 8
     Total 9
     Total10
     Total11
     Total12
     Total13
```

TA-7079-45

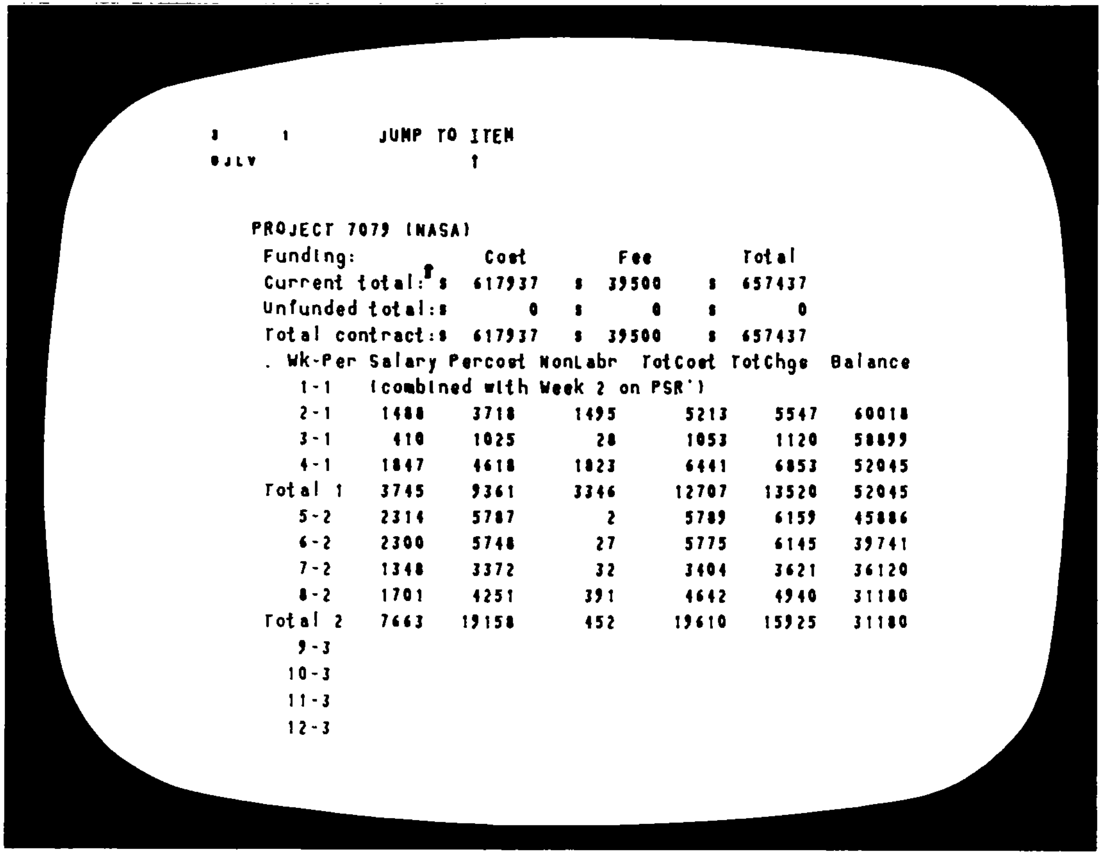
FIGURE III-35 A BRANCH OF FILE COSTS SHOWING ENTRIES FOR 4-WEEK ACCOUNTING PERIODS



```
3      1        JUMP TO ITEM
BJLV                    ↑


    PROJECT 7101 (RADC-old)
     Funding:          ↑ Cost        Fee        Total
     Current total: $ 1459564   $  55658    $ 1515222
     Unfunded total:$       0   $      0    $       0
     Total contract:$ 1459564   $  55658    $ 1515222
       Wk-Per Salary Percost NonLabr   TotCost TotChge  Balance
         1-1   (combined with Week 2 on PSR')
         2-1    6660   16617   10902     27519   28565   238023
         3-1    3646    9067    5556     14623   15178   222845
         4-1    3815    9532   30436     39968   41487   181358
     Total 1   14121   35216   46894     82110   85230   181358
         5-2    2713    6773     509      7282    7559   173799
         6-2    2874    7184    8495     15679   16275   157524
         7-2    3465    8647    9578     18225   18917   138607
         8-2    2254    5621    5335     10956   10973   127634
     Total 2   11306   28225   23917     52142   53724   127634
         9-3
        10-3
        11-3
        12-3
```

TA-7079-46

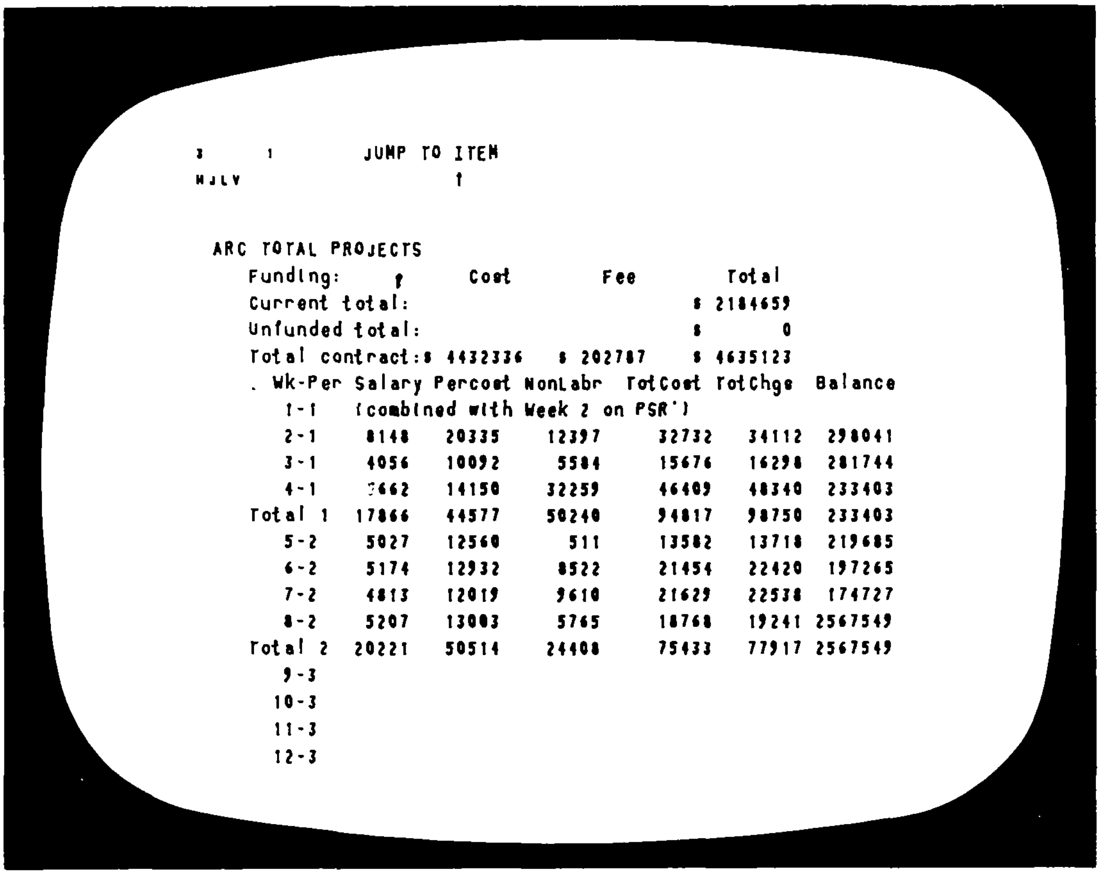
FIGURE III-36   SAME AS FIGURE III-35 BUT EXPANDED TO SHOW WEEKLY ENTRIES



```
3      1        JUMP TO ITEM
0JLV                  ↑


   PROJECT 7079 (NASA)
    Funding:               Cost          Fee          Total
    Current total:↑$  617937   $  39500    $  657437
    Unfunded total:$        0   $      0    $       0
    Total contract:$  617937   $  39500    $  657437
    . Wk-Per Salary Percost NonLabr  TotCost TotChge  Balance
       1-1   (combined with Week 2 on PSR')
       2-1    1488    3718     1495      5213    5547    60018
       3-1     410    1025       28      1053    1120    58899
       4-1    1847    4618     1823      6441    6853    52045
    Total 1   3745    9361     3346     12707   13520    52045
       5-2    2314    5787        2      5789    6159    45886
       6-2    2300    5748       27      5775    6145    39741
       7-2    1348    3372       32      3404    3621    36120
       8-2    1701    4251      391      4642    4940    31180
    Total 2   7663   19158      452     19610   15925    31180
       9-3
      10-3
      11-3
      12-3
```

TA-7079-47

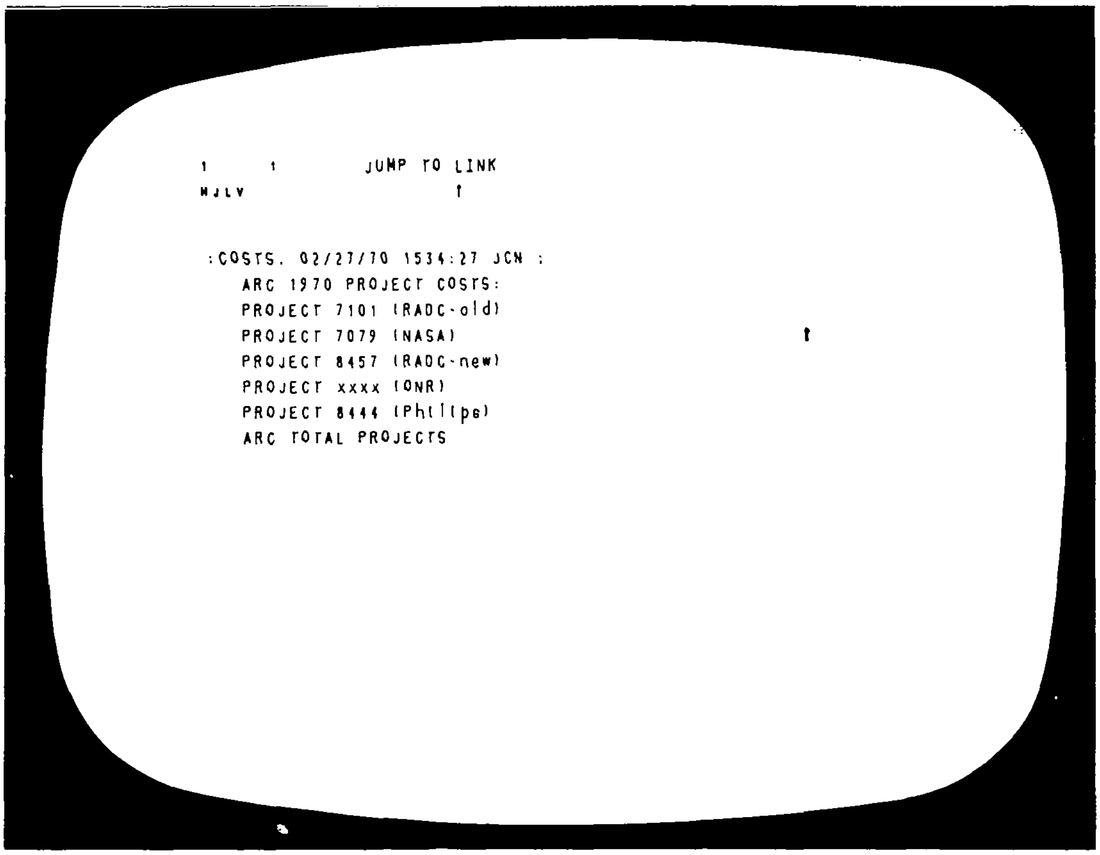
FIGURE III-37 SAME AS FIGURE III-36 BUT FOR A DIFFERENT BRANCH OF FILE COSTS SHOWING DATA FOR A DIFFERENT PROJECT



```
3      1        JUMP TO ITEM
HJLV                  ↑


 ARC TOTAL PROJECTS
    Funding:     $       Cost         Fee         Total
    Current total:                              $ 2184659
    Unfunded total:                             $       0
    Total contract:$ 4432336    $ 202787        $ 4635123
   . Wk-Per Salary Percost NonLabr  TotCost TotChge  Balance
       1-1   (combined with Week 2 on PSR')
       2-1    8148   20335    12397     32732   34112  298041
       3-1    4056   10092     5584     15676   16298  281744
       4-1    5662   14150    32259     46409   48340  233403
    Total 1  17866   44577    50240     94817   98750  233403
       5-2    5027   12560      511     13582   13718  219685
       6-2    5174   12932     8522     21454   22420  197265
       7-2    4813   12019     9610     21629   22538  174727
       8-2    5207   13003     5765     18768   19241 2567549
    Total 2  20221   50514    24408     75433   77917 2567549
       9-3
      10-3
      11-3
      12-3
```

TA-7079-48

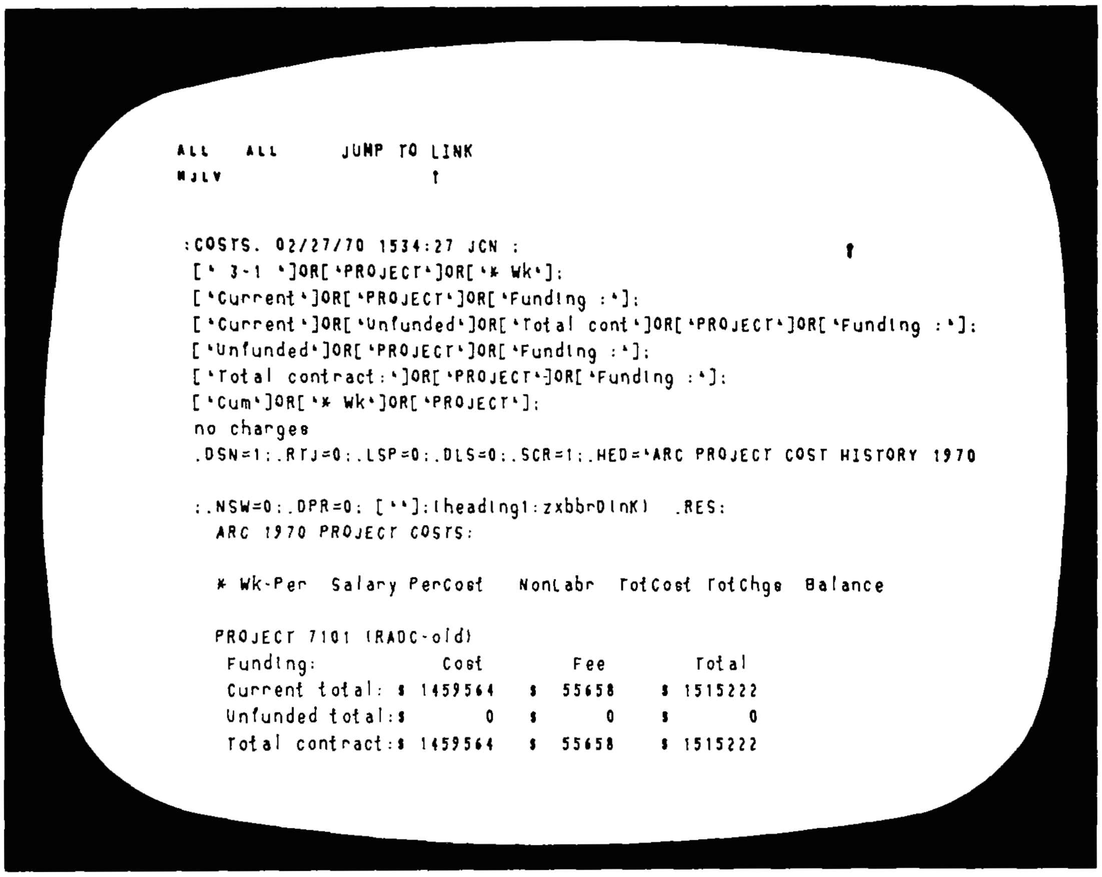
FIGURE III-38 A BRANCH OF FILE **COSTS** SHOWING COMBINED DATA FOR ALL **ARC** PROJECTS



```
1      1          JUMP TO LINK
NJLV                     ↑

 :COSTS. 02/27/70 1534:27 JCN :
   ARC 1970 PROJECT COSTS:
   PROJECT 7101 (RADC-old)
   PROJECT 7079 (NASA)                         ↑
   PROJECT 8457 (RADC-new)
   PROJECT xxxx (ONR)
   PROJECT 8444 (Phillps)
   ARC TOTAL PROJECTS
```

TA-7079-49

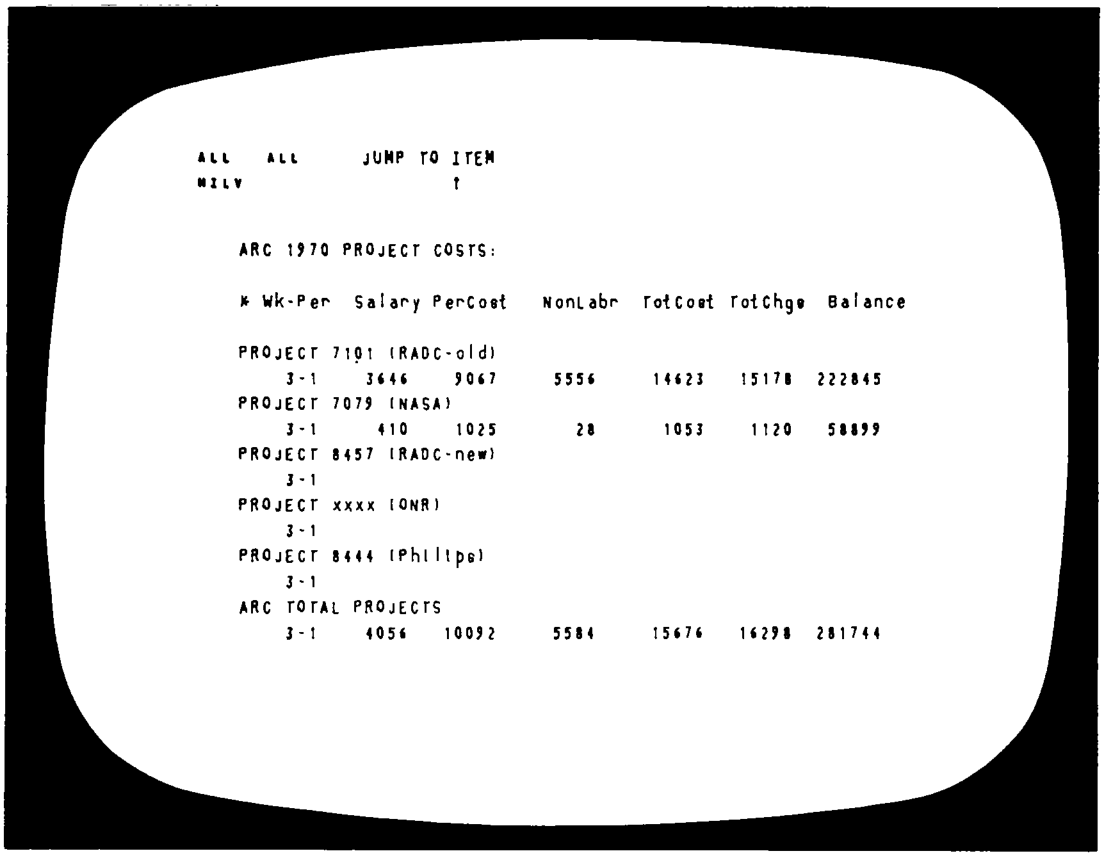
FIGURE III-39 INITIAL VIEW OF FILE **COSTS** UPON ENTRY VIA LINK



```
ALL   ALL      JUMP TO LINK
MJLV                  ↑


:COSTS. 02/27/70 1534:27 JCN :                            ↑
 [' 3-1 ']OR['PROJECT']OR['* Wk']:
 ['Current']OR['PROJECT']OR['Funding :']:
 ['Current']OR['Unfunded']OR['Total cont']OR['PROJECT']OR['Funding :']:
 ['Unfunded']OR['PROJECT']OR['Funding :']:
 ['Total contract:']OR['PROJECT']OR['Funding :']:
 ['Cum']OR['* Wk']OR['PROJECT']:
 no charges
 .DSN=1:.RTJ=0:.LSP=0:.DLS=0:.SCR=1:.HED='ARC PROJECT COST HISTORY 1970

 :.NSW=0:.DPR=0: ['']:(heading1:zxbbrD(nK)  .RES:
   ARC 1970 PROJECT COSTS:

   * Wk-Per  Salary PerCost   NonLabr  TotCost TotChge  Balance

   PROJECT 7101 (RADC-old)
    Funding:              Cost        Fee       Total
    Current total: $ 1459564   $  55658   $ 1515222
    Unfunded total:$       0   $      0   $       0
    Total contract:$ 1459564   $  55658   $ 1515222
```

TA-7079-50

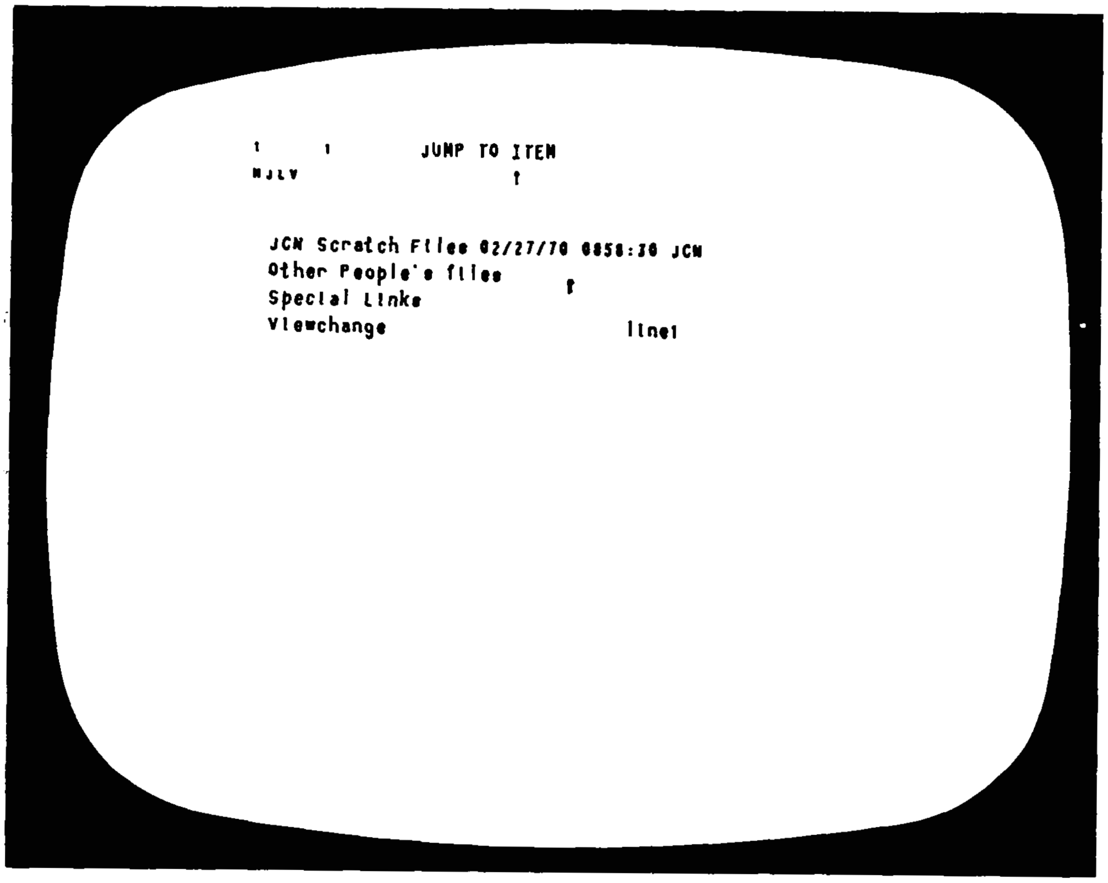
FIGURE III-40 SAME AS FIGURE III-39 BUT WITH DIFFERENT **VIEWSPECS** TO SHOW CONTENT-ANALYZER PATTERNS STORED IN FIRST STATEMENT OF FILE



```
ALL   ALL      JUMP TO ITEM
NILV                  ↑


    ARC 1970 PROJECT COSTS:

    * Wk-Per  Salary PerCost   NonLabr  TotCost TotChge  Balance

    PROJECT 7101 (RADC-old)
         3-1    3646    9067     5556     14623   15178  222845
    PROJECT 7079 (NASA)
         3-1     410    1025       28      1053    1120   58899
    PROJECT 8457 (RADC-new)
         3-1
    PROJECT xxxx (ONR)
         3-1
    PROJECT 8444 (Phillps)
         3-1
    ARC TOTAL PROJECTS
         3-1    4056   10092     5584     15676   16298  281744
```

TA-7079-51

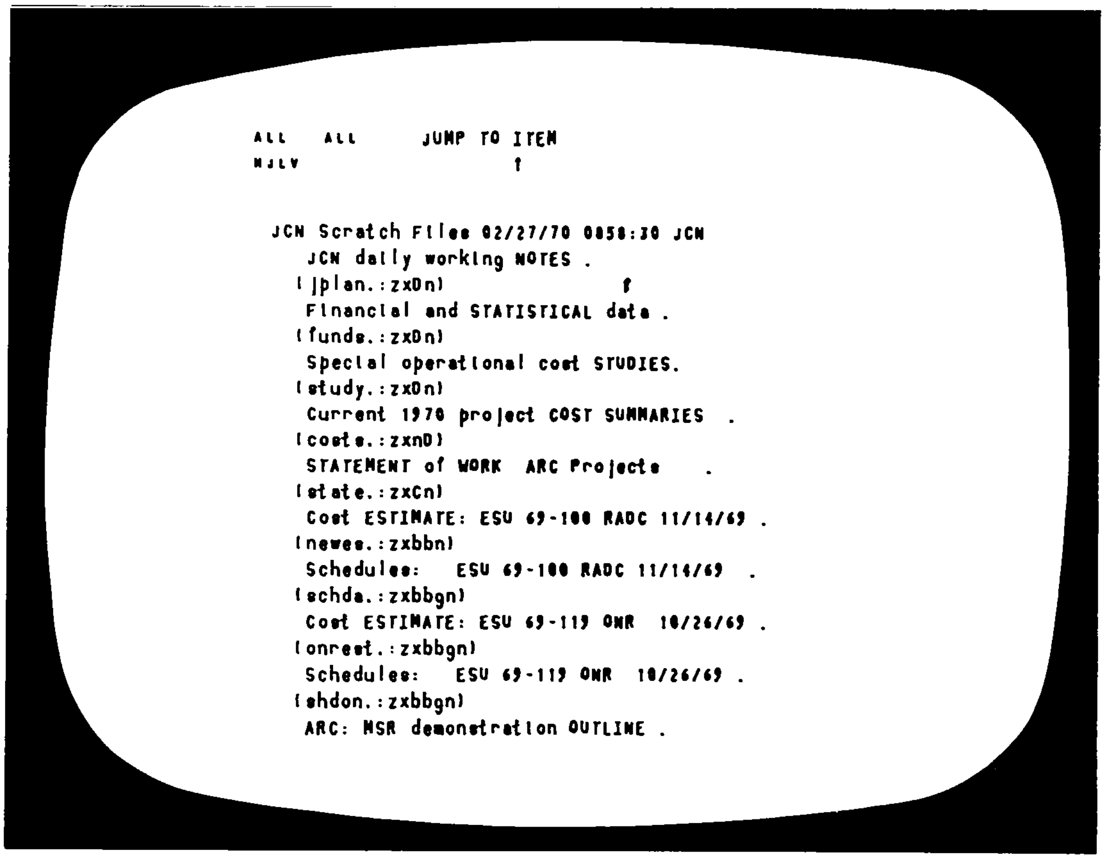
FIGURE III-41 VIEW OF FILE **COSTS** WITH CONTENT ANALYZER IN OPERATION, SHOWING DATA FOR A SINGLE WEEK ONLY. This is done by using the first pattern appearing in square brackets in Figure III-40.



```
1      1         JUMP TO ITEM
WJLV                    ↑


  JCN Scratch Files 02/27/70 0858:30 JCN
  Other People's files        ↑
  Special Links
  Viewchange                       line1
```

TA-7079-52

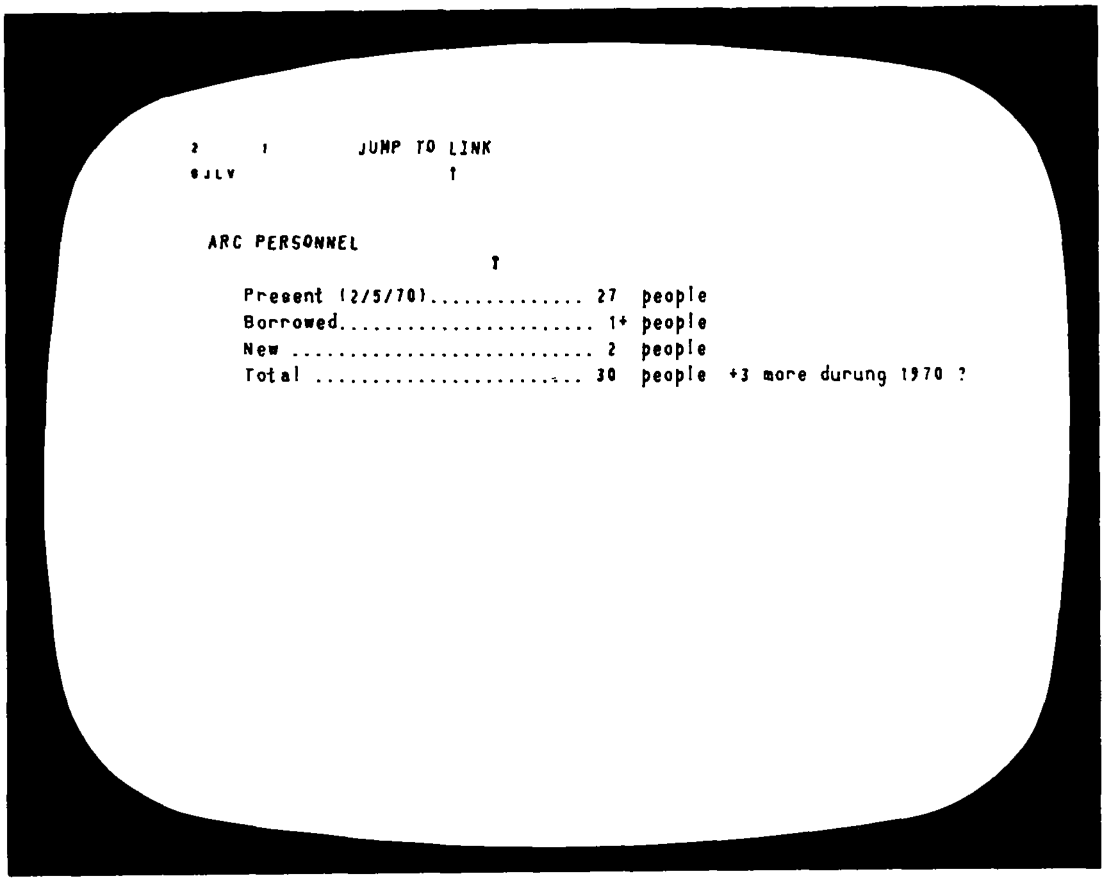
FIGURE III-42 VIEW OF A USER'S FILE DIRECTORY, SHOWING FIRST-LEVEL STATEMENTS ONLY



```
ALL   ALL      JUMP TO ITEM
NJLV                  ↑

  JCN Scratch Files 02/27/70 0858:30 JCN
     JCN daily working NOTES .
    (jplan.:zxOn)                ↑
     Financial and STATISTICAL data .
    (funds.:zxOn)
     Special operational cost STUDIES.
    (study.:zxOn)
     Current 1970 project COST SUMMARIES  .
    (costs.:zxnO)
     STATEMENT of WORK  ARC Projects    .
    (state.:zxCn)
     Cost ESTIMATE: ESU 69-100 RADC 11/14/69 .
    (newes.:zxbbn)
     Schedules:   ESU 69-100 RADC 11/14/69  .
    (schds.:zxbbgn)
     Cost ESTIMATE: ESU 69-119 ONR  10/26/69 .
    (onrest.:zxbbgn)
     Schedules:   ESU 69-119 ONR  10/26/69 .
    (shdon.:zxbbgn)
     ARC: MSR demonstration OUTLINE .
```

TA-7079-53

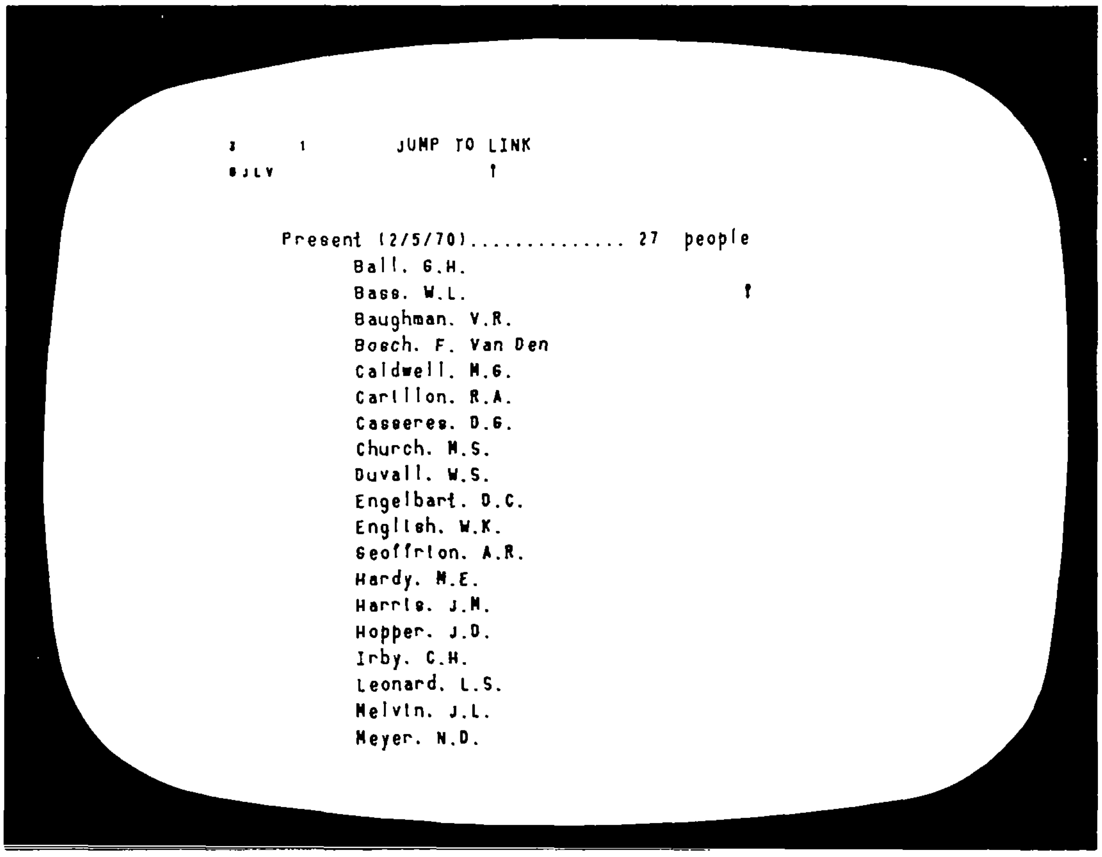
FIGURE III-43   SAME AS FIGURE III-42 BUT WITH ALL LEVELS DISPLAYED



```
2       1         JUMP TO LINK
●JLV                    ↑


 ARC PERSONNEL
                          ↑
    Present (2/5/70).............. 27  people
    Borrowed...................... 1+  people
    New .......................... 2   people
    Total ........................ 30  people  +3 more durung 1970 ?
```

TA-7079-54

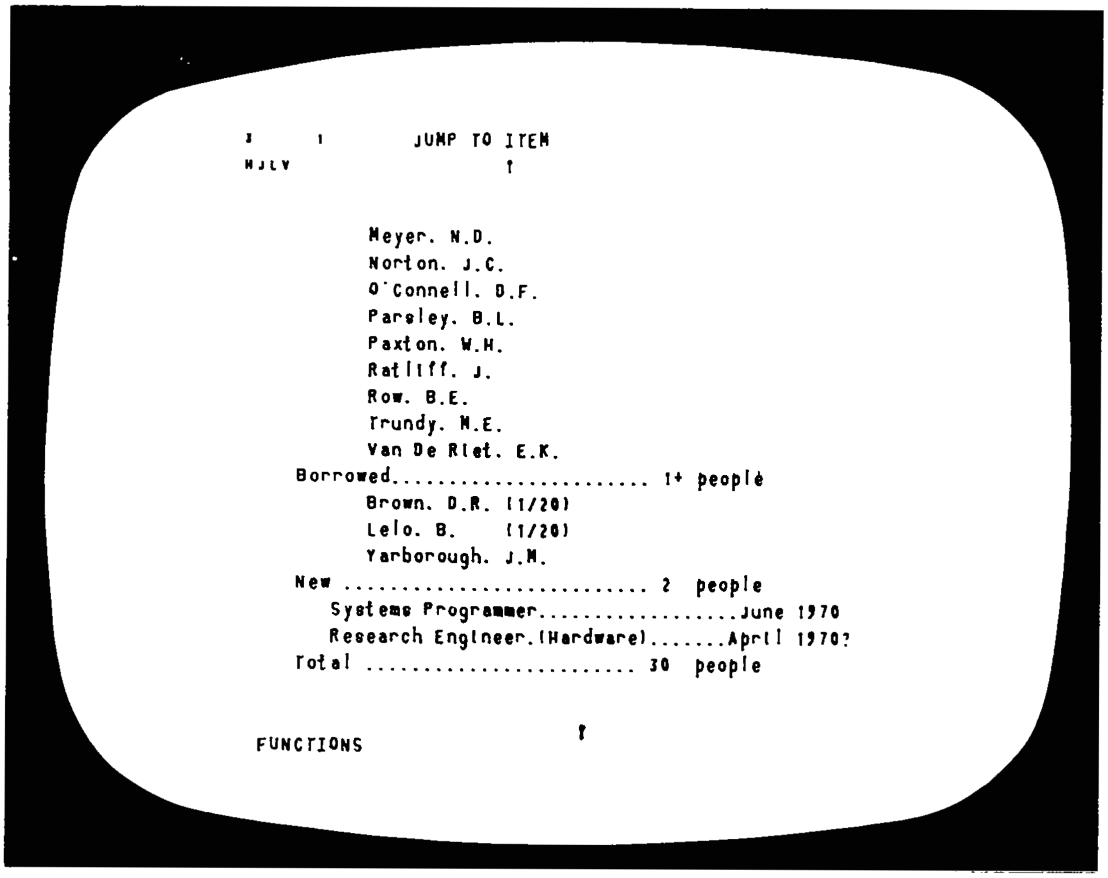
FIGURE III-44 PART OF A FILE CONTAINING INFORMATION ON ARC PERSONNEL. (Not all levels are shown.)



```
3     1        JUMP TO LINK
SJLV                 ↑

     Present (2/5/70)............. 27  people
            Ball. G.H.
            Bass. W.L.                        ↑
            Baughman. V.R.
            Bosch. F. Van Den
            Caldwell. M.G.
            Carlilon. R.A.
            Casseres. D.G.
            Church. M.S.
            Duvall. W.S.
            Engelbart. D.C.
            English. W.K.
            Geoffrion. A.R.
            Hardy. M.E.
            Harris. J.M.
            Hopper. J.D.
            Irby. C.H.
            Leonard. L.S.
            Melvin. J.L.
            Meyer. N.D.
```

TA-7079-55

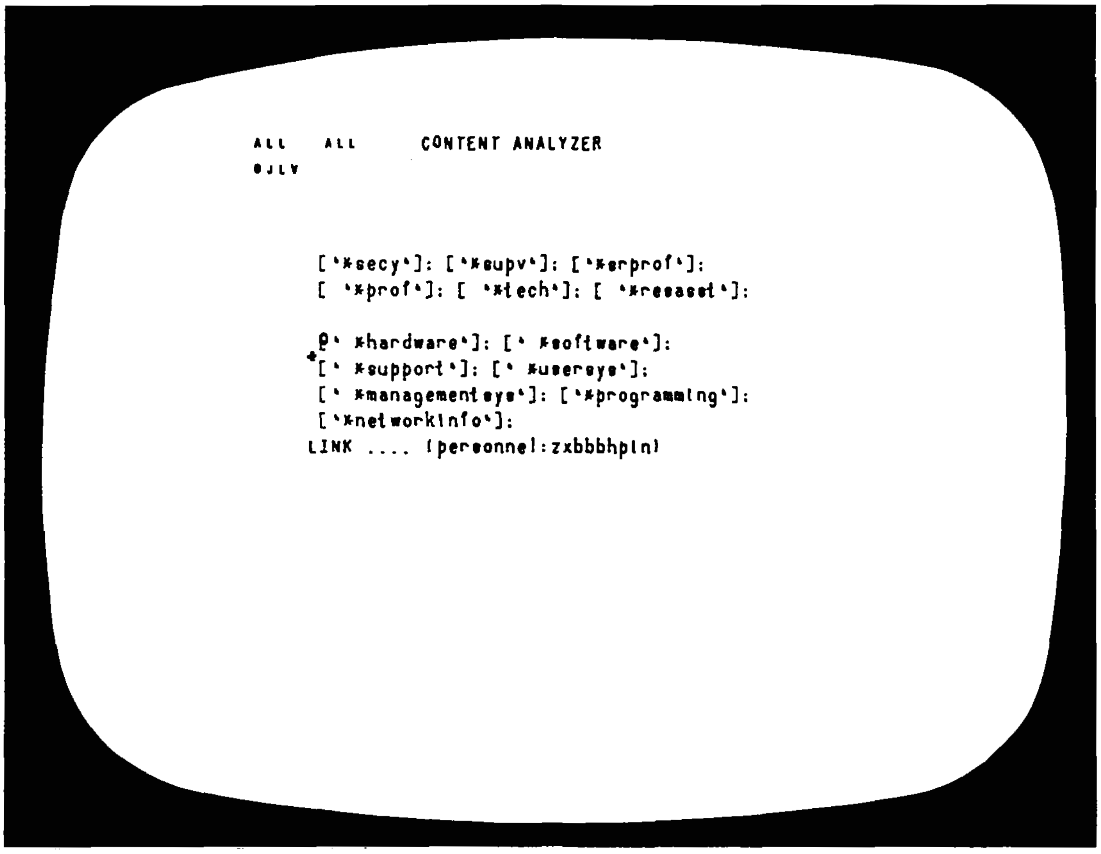
FIGURE III-45 A VIEW OBTAINED BY JUMPING TO ONE OF THE STATEMENTS SHOWN IN FIGURE III-44 AND OPENING AN ADDITIONAL LEVEL



```
3      1         JUMP TO ITEM
MJLV                  T


             Meyer. N.D.
             Norton. J.C.
             O'Connell. D.F.
             Parsley. B.L.
             Paxton. W.H.
             Ratliff. J.
             Row. B.E.
             Trundy. M.E.
             Van De Riet. E.K.
      Borrowed......................... 1+ people
             Brown. D.R. (1/20)
             Lelo. B.    (1/20)
             Yarborough. J.M.
      New ............................ 2  people
         Systems Programmer..................June 1970
         Research Engineer.(Hardware).......April 1970?
      Total ......................... 30  people


  FUNCTIONS                     T
```

TA-7079-56

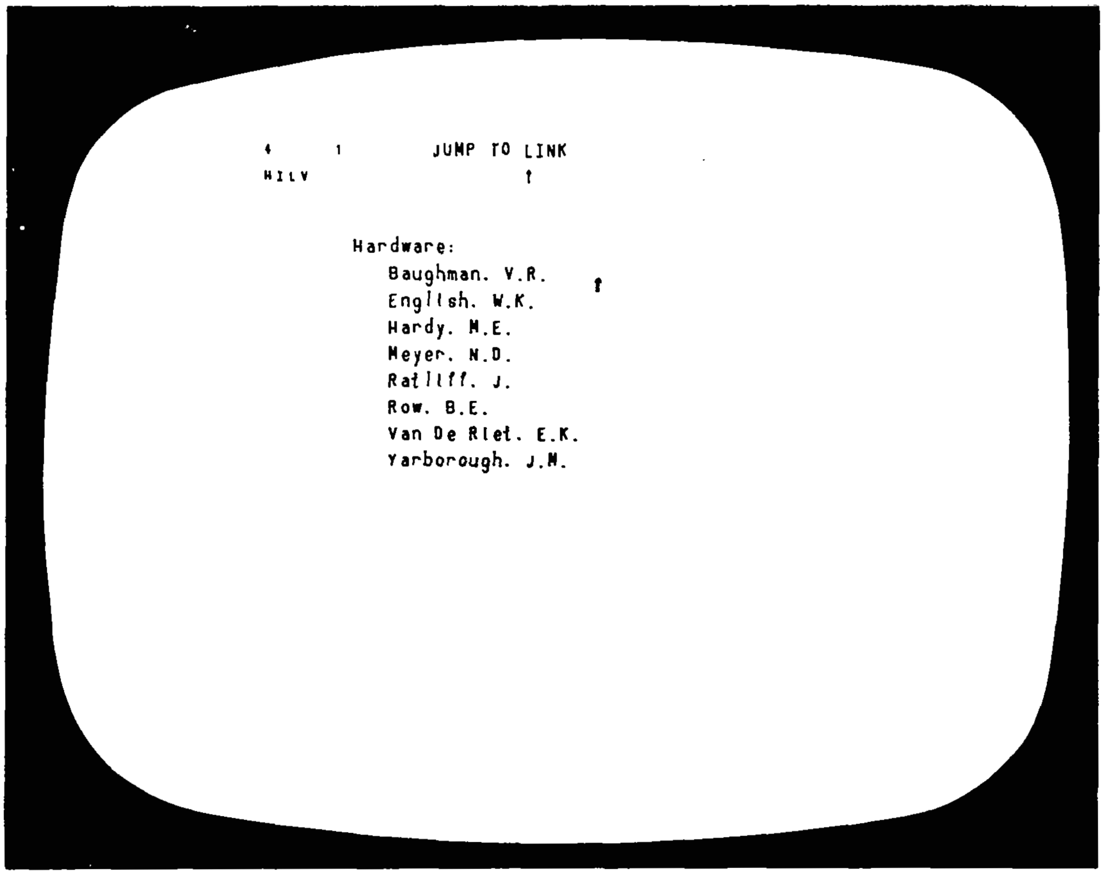
FIGURE III-46 A VIEW OBTAINED BY JUMPING TO THE LAST STATEMENT SHOWN IN FIGURE III-45



```
ALL    ALL        CONTENT ANALYZER
OJLV



     ['*secy']: ['*supv']: ['*srprof']:
     [ '*prof']: [ '*tech']: [ '*resasst']:

     [' *hardware']: [' *software']:
    +[' *support']: [' *usersys']:
     [' *managementsys']: ['*programming']:
     ['*networkinfo']:
    LINK .... (personnel:zxbbbhpln)
```

TA-7079-57

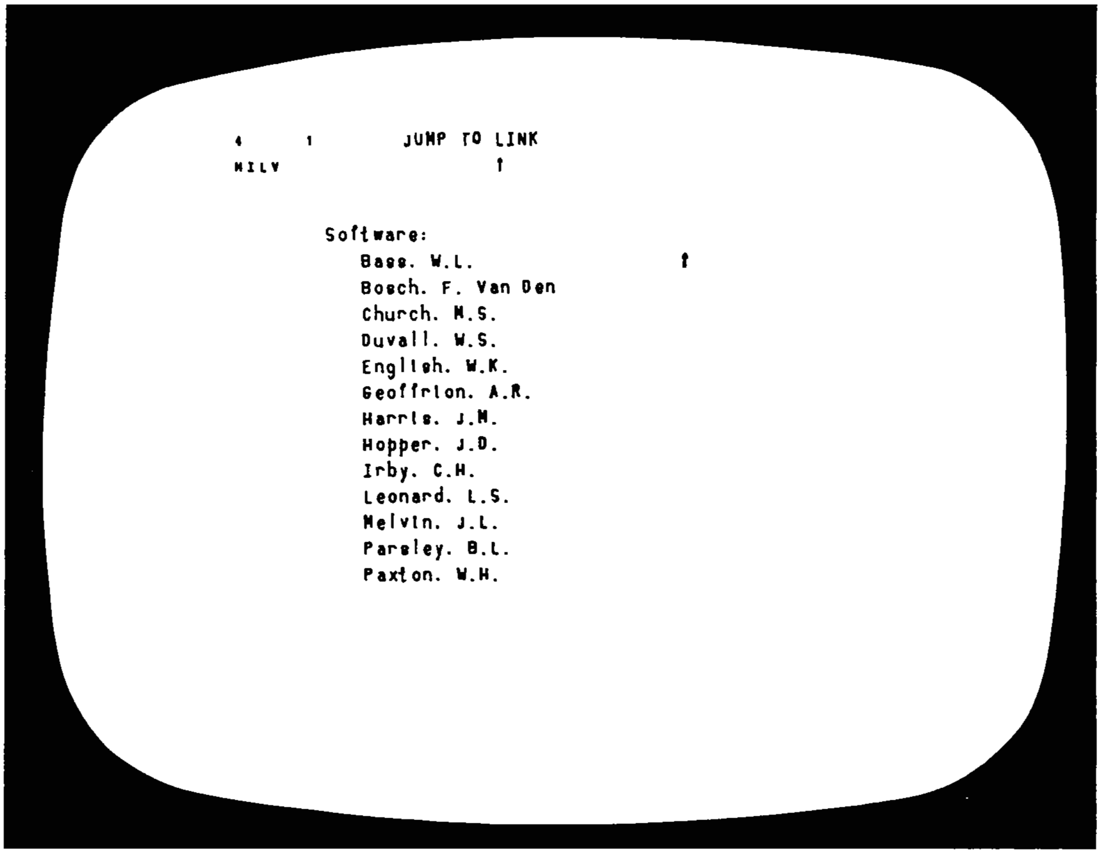
FIGURE III-47 CONTENT-ANALYZER PATTERNS STORED IN THE PERSONNEL-INFORMATION FILE. Each set of square brackets contains one pattern, used to search for hidden "tags" in statements in the file.



```
4      1          JUMP TO LINK
HILV                    ↑


           Hardware:
              Baughman. V.R.    ↑
              English. W.K.
              Hardy. M.E.
              Meyer. N.D.
              Ratliff. J.
              Row. B.E.
              Van De Riet. E.K.
              Yarborough. J.M.
```

TA-7079-58

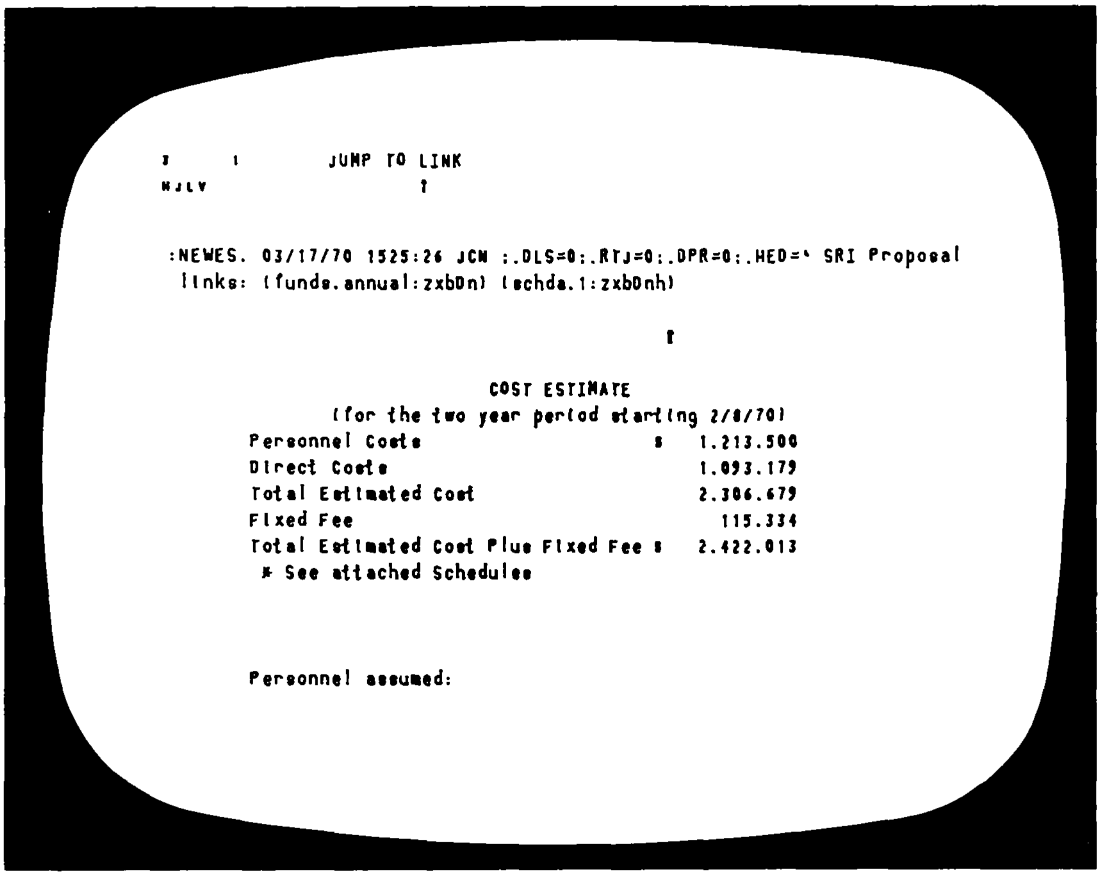
FIGURE III-48 VIEW OBTAINED BY USING CONTENT ANALYZER TO SELECT ENTRIES IN PERSONNEL-INFORMATION FILE THAT ARE TAGGED FOR "HARDWARE"



```
4      1        JUMP TO LINK
MILV                  ↑

          Software:
             Bass. W.L.                    ↑
             Bosch. F. Van Den
             Church. M.S.
             Duvall. W.S.
             English. W.K.
             Geoffrion. A.R.
             Harris. J.M.
             Hopper. J.D.
             Irby. C.H.
             Leonard. L.S.
             Melvin. J.L.
             Parsley. B.L.
             Paxton. W.H.
```

TA-7079-59

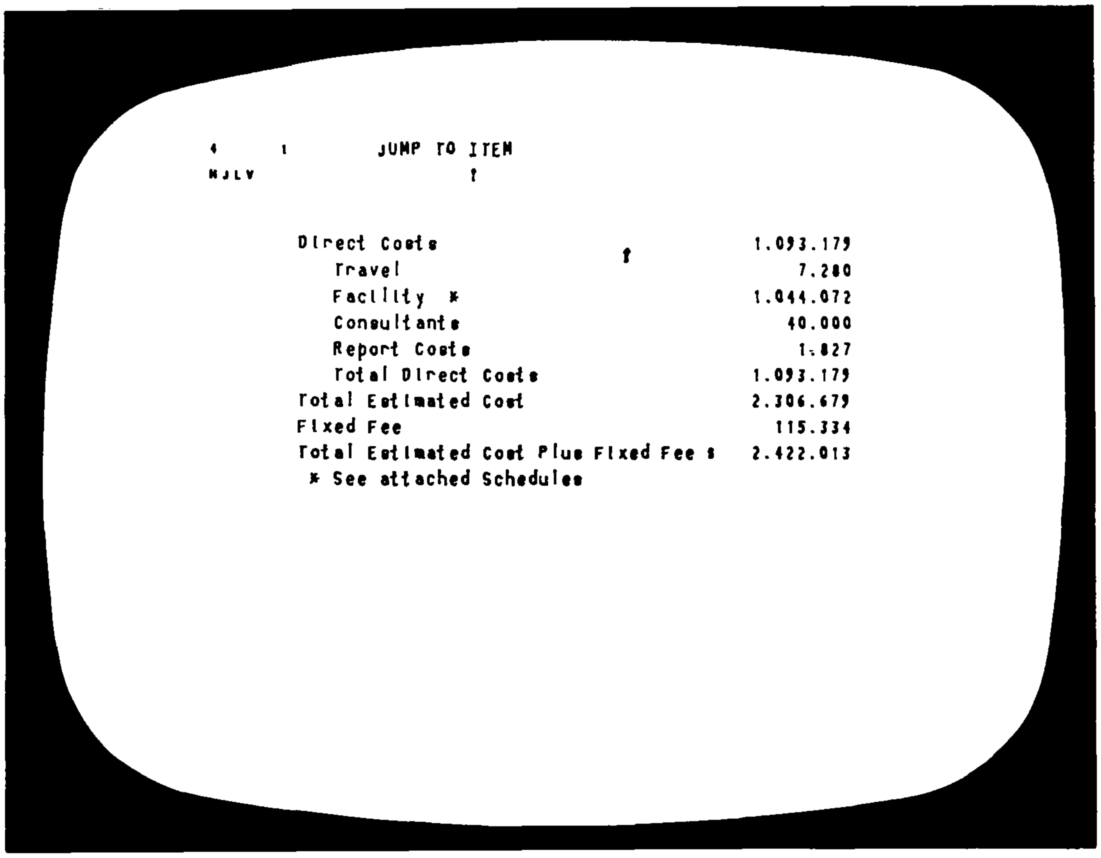
FIGURE III-49 VIEW OBTAINED BY USING CONTENT ANALYZER TO SELECT ENTRIES IN PERSONNEL-INFORMATION FILE THAT ARE TAGGED FOR "SOFTWARE"



```
3      1          JUMP TO LINK
HJLV                   1


:NEWES. 03/17/70 1525:26 JCN :.DLS=0:.RTJ=0:.DPR=0:.HED=' SRI Proposal
 links: (funds.annual:zxbOn) (schds.1:zxbOnh)

                                                      1

                                   COST ESTIMATE
                  (for the two year period starting 2/8/70)
         Personnel Costs                          $   1.213.500
         Direct Costs                                 1.093.179
         Total Estimated Cost                         2.306.679
         Fixed Fee                                      115.334
         Total Estimated Cost Plus Fixed Fee $        2.422.013
          * See attached Schedules



         Personnel assumed:
```

TA-7079-60

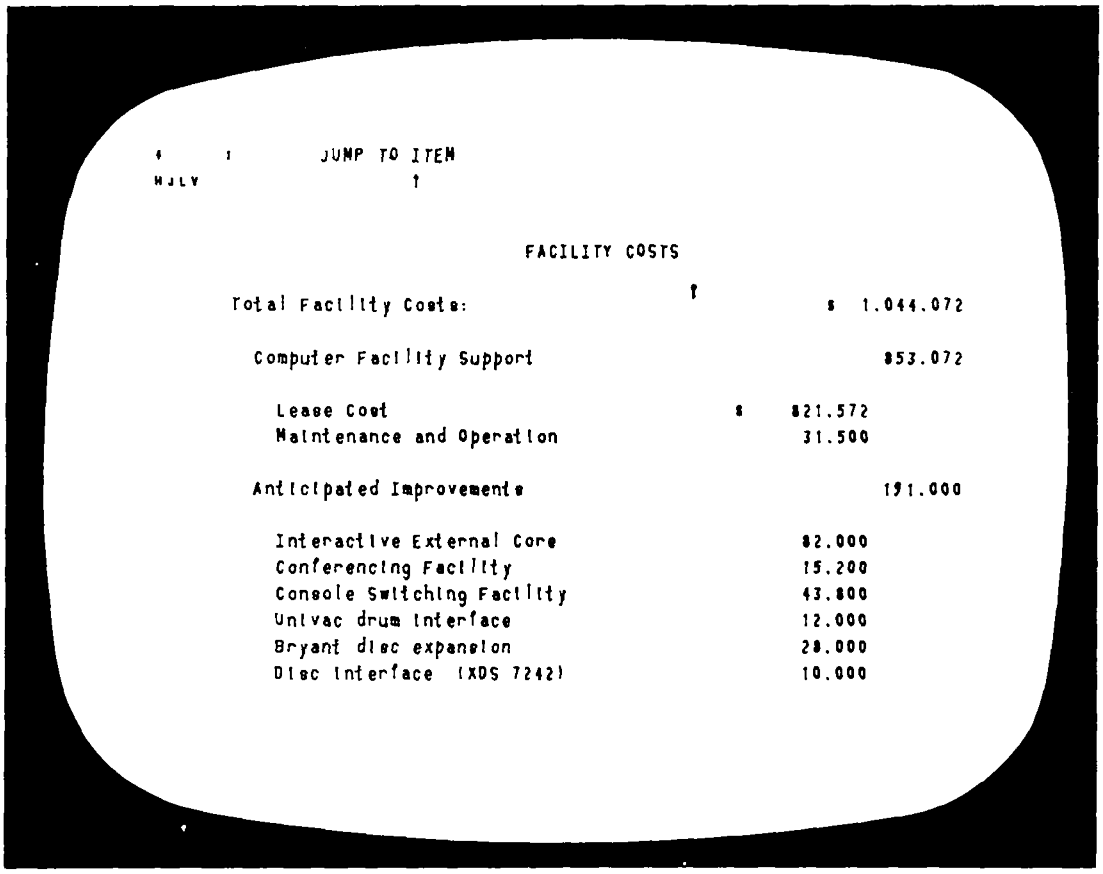
FIGURE III-50   PART OF AN ON-LINE COST ESTIMATE FOR USE IN A PROPOSAL



```
4      1         JUMP TO ITEM
NJLV

          Direct Costs                           1.093.179
             Travel                                  7.280
             Facility  *                         1.044.072
             Consultants                            40.000
             Report Costs                            1.827
             Total Direct Costs                  1.093.179
          Total Estimated Cost                   2.306.679
          Fixed Fee                                115.334
          Total Estimated Cost Plus Fixed Fee $  2.422.013
           * See attached Schedules
```

TA-7079-61

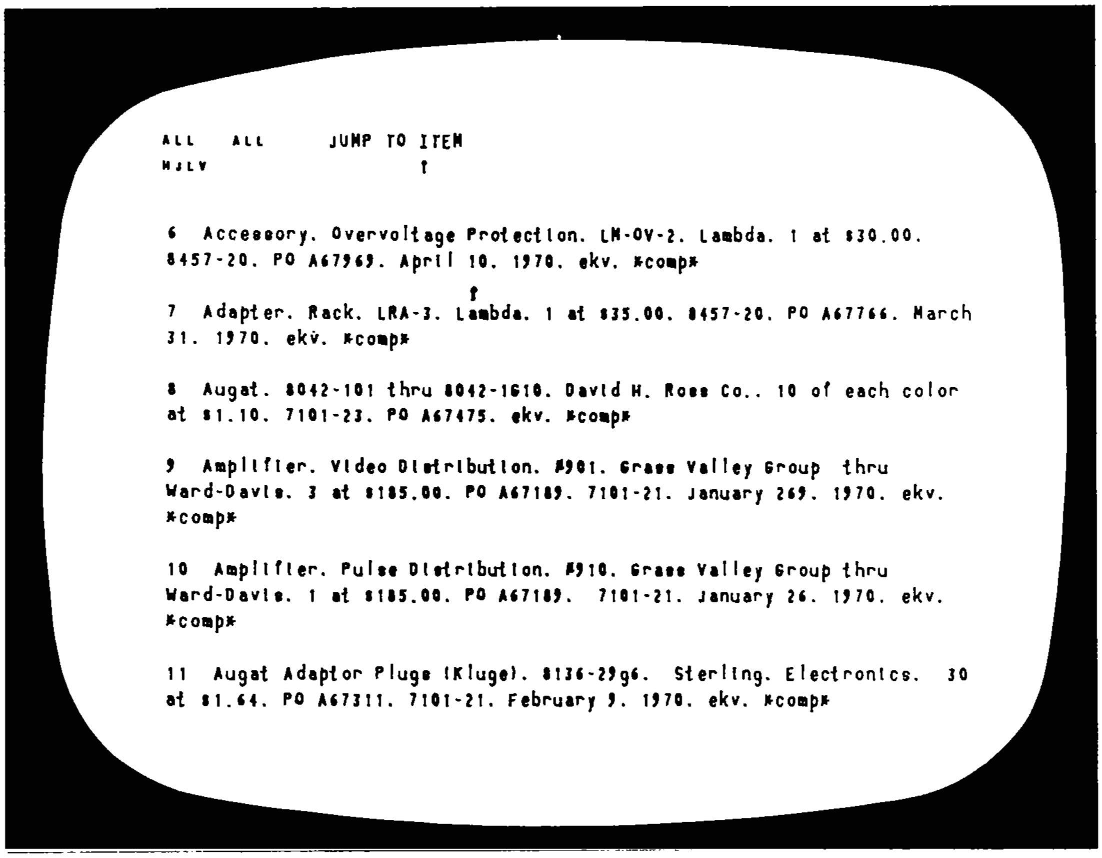
FIGURE III-51   PART OF AN ON-LINE COST ESTIMATE FOR USE IN A PROPOSAL



```
+      I         JUMP TO ITEM
HJLV                      †


                                        FACILITY COSTS
                                                 †
        Total Facility Costs:                                  $  1.044.072

          Computer Facility Support                                 853.072

            Lease Cost                                  $   821.572
            Maintenance and Operation                        31.500

          Anticipated Improvements                                  191.000

            Interactive External Core                        82.000
            Conferencing Facility                            15.200
            Console Switching Facility                       43.800
            Univac drum interface                            12.000
            Bryant disc expansion                            28.000
            Disc interface  (XDS 7242)                       10.000
```

TA-7079-62

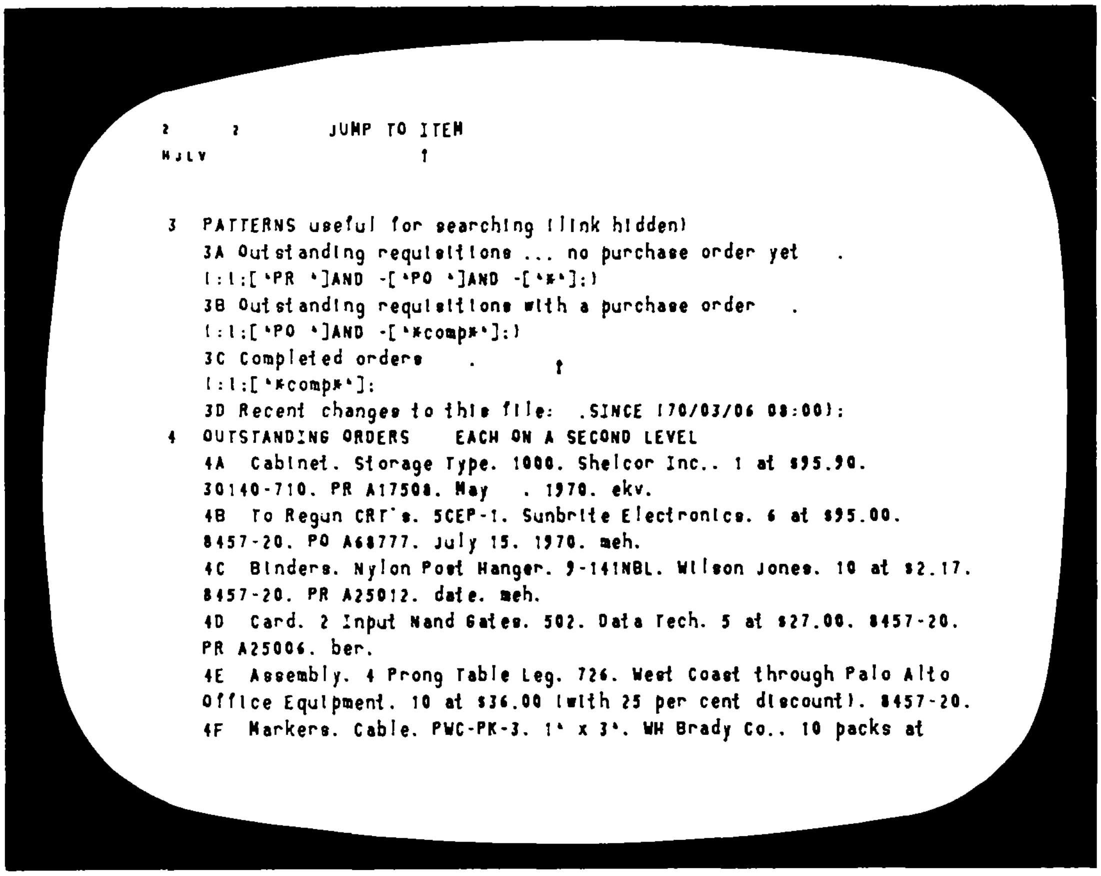
FIGURE III-52   PART OF AN ON-LINE COST ESTIMATE FOR USE IN A PROPOSAL



```
ALL   ALL       JUMP TO ITEM
MJLV                    ↑


6  Accessory. Overvoltage Protection. LM-OV-2. Lambda. 1 at $30.00.
8457-20. PO A67969. April 10. 1970. ekv. *comp*
                              ↑
7  Adapter. Rack. LRA-3. Lambda. 1 at $35.00. 8457-20. PO A67766. March
31. 1970. ekv. *comp*

8  Augat. 8042-101 thru 8042-1610. David H. Ross Co.. 10 of each color
at $1.10. 7101-23. PO A67475. ekv. *comp*

9  Amplifier. Video Distribution. #901. Grass Valley Group  thru
Ward-Davis. 3 at $185.00. PO A67189. 7101-21. January 269. 1970. ekv.
*comp*

10  Amplifier. Pulse Distribution. #910. Grass Valley Group thru
Ward-Davis. 1 at $185.00. PO A67189.  7101-21. January 26. 1970. ekv.
*comp*

11  Augat Adaptor Plugs (Kluge). 8136-29g6.  Sterling. Electronics.  30
at $1.64. PO A67311. 7101-21. February 9. 1970. ekv. *comp*
```

TA-7079-63

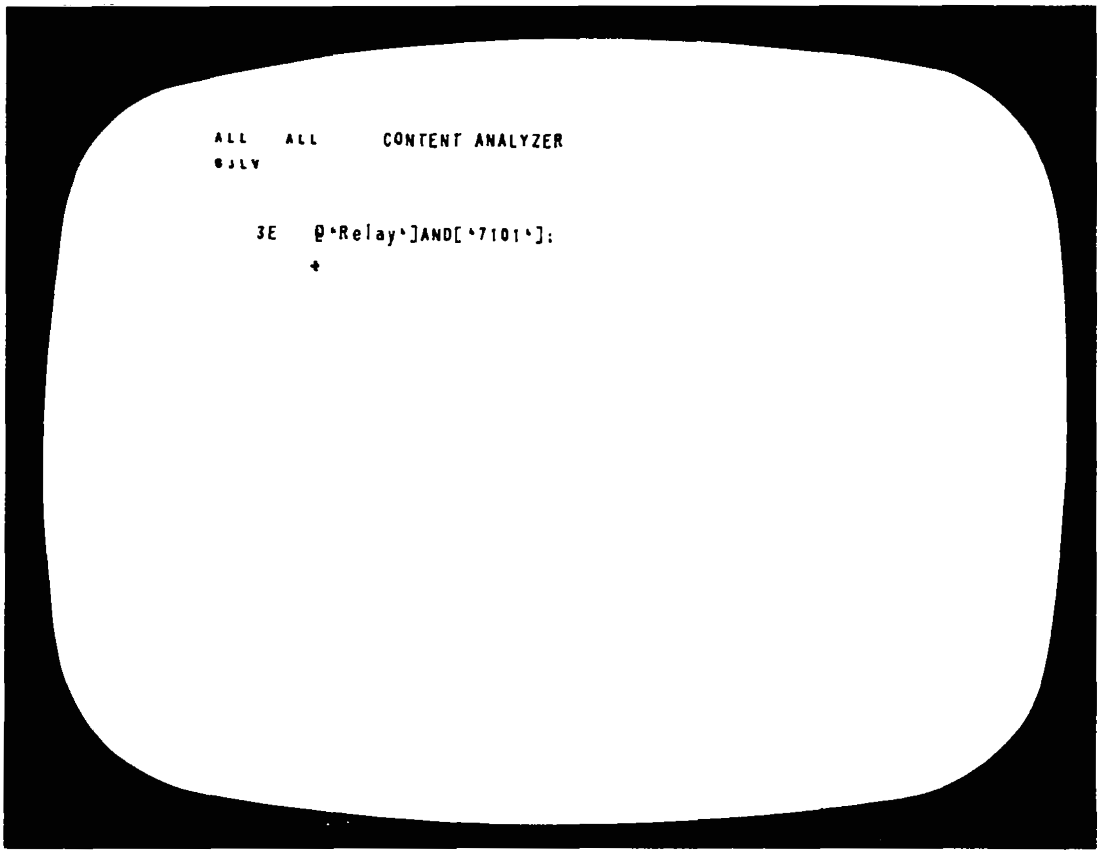
FIGURE III-53   VIEW OF A PORTION OF THE PURCHASE-ORDER PROCESSING FILE, SHOWING CONTENTS OF INDIVIDUAL STATEMENTS



```
2      2         JUMP TO ITEM
HJLV                  †


 3  PATTERNS useful for searching (link hidden)
    3A Outstanding requisitions ... no purchase order yet   .
    (:l:['PR ']AND -['PO ']AND -['*']:)
    3B Outstanding requisitions with a purchase order   .
    (:l:['PO ']AND -['*comp*']:)
    3C Completed orders     .        †
    (:l:['*comp*']:
    3D Recent changes to this file:  .SINCE (70/03/06 08:00):
 4  OUTSTANDING ORDERS    EACH ON A SECOND LEVEL
    4A  Cabinet. Storage Type. 1000. Shelcor Inc.. 1 at $95.90.
    30140-710. PR A17508. May   . 1970. ekv.
    4B  To Regun CRT's. 5CEP-1. Sunbrite Electronics. 6 at $95.00.
    8457-20. PO A68777. July 15. 1970. meh.
    4C  Binders. Nylon Post Hanger. 9-141NBL. Wilson Jones. 10 at $2.17.
    8457-20. PR A25012. date. meh.
    4D  Card. 2 Input Nand Gates. 502. Data Tech. 5 at $27.00. 8457-20.
    PR A25006. ber.
    4E  Assembly. 4 Prong Table Leg. 726. West Coast through Palo Alto
    Office Equipment. 10 at $36.00 (with 25 per cent discount). 8457-20.
    4F  Markers. Cable. PWC-PK-3. 1" x 3". WH Brady Co.. 10 packs at
```

TA-7079-64

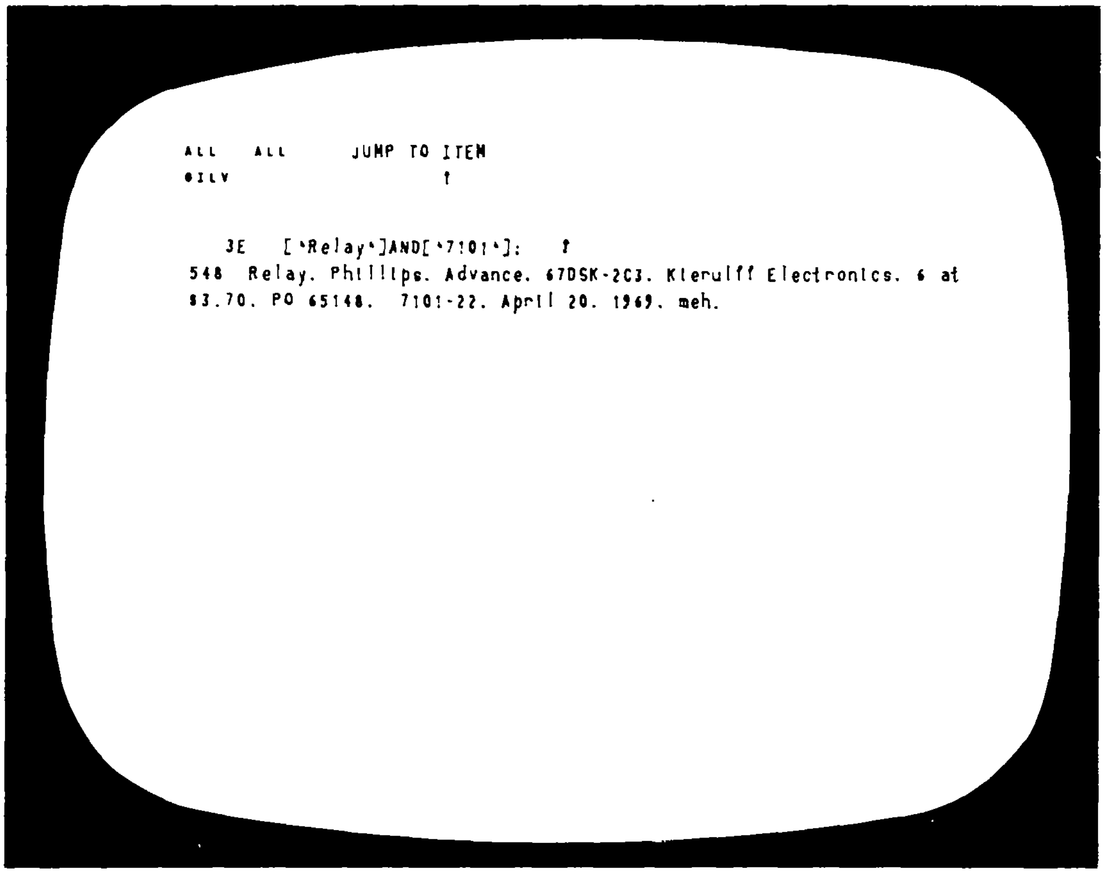
FIGURE III-54 VIEW OF A PORTION OF THE PURCHASE-ORDER PROCESSING FILE, SHOWING OUTSTANDING ORDERS LOCATED IN A SEPARATE BRANCH. Upper part of screen shows a branch containing Content-Analyzer patterns.



```
ALL    ALL        CONTENT ANALYZER
6JLV

    3E    @'Relay']AND['7101'];
          +
```

TA-7079-65

FIGURE III-55 A CONTENT-ANALYZER PATTERN FOR SEARCHING IN THE PURCHASE-ORDER FILE



```
ALL   ALL      JUMP TO ITEM
●ILV                  ↑


    3E   [‘Relay‘]AND[‘7101‘]:    ↑
548  Relay. Phillips. Advance. 67DSK-2C3. Kierulff Electronics. 6 at
$3.70. PO 65148.  7101-22. April 20. 1969. meh.
```

TA-7079-66

FIGURE III-56 VIEW GENERATED BY A SEARCH ON THE PATTERN SHOWN IN FIGURE III-55

D. The Augmented Report-Writing Team

1. Introduction

This section discusses the generation of large, contractually required reports, since these test our capabilities most thoroughly. Smaller, less formal reports involve a great deal less effort; the advantages discussed below are equally present, but the problems are insignificant in comparison to the ones that arise with, for example, a final report.

It should be noted that much of this discussion is applicable to proposals and other types of documents, as well as reports.

2. Team Approach

As in many other research groups, none of our major reports are written by a single individual.

Because of the broad scope of activities reported -- ranging from management systems research to the detailed design of software implementations -- we use a team approach, involving one or two individuals who coordinate the effort and numerous others who contribute sections of material covering their particular specialties.

The present report contains material written by at least six people -- the exact number is hard to determine because some material is adapted from previous documents, using archived files saved for this purpose.

Also involved in the process are individuals who must formally approve sections as they are written, the principal investigator, who must approve the overall report at various stages of completion, and a technical writer/editor.

Typically, these persons as well as the coordinators are also major contributors to the text of the report.

This list excludes other SRI personnel outside of ARC -- editors, illustrators, officers who must approve the report, etc. Part of the report-writing team's job is to conduct liaison with these people.

Augmentation bears upon all aspects of this team effort. Because of the flexibility of the tools (principally NLS), the various individuals involved use the system in various individualistic ways.

To some, NLS is simply a super-typewriter; to others, it is a high-powered study aid for searching existing material and extracting information needed in the current report.

Some people prefer to do certain types of work with hard copy rather than at a console. These people use NLS for rapid production of hard copy of latest versions of parts of the report; after working on the hard copy, they use NLS for rapid incorporation of their changes and additions into the master files on line.

To the coordinators, augmentation means the ability to create an outline of topics to be covered in the report, with notations indicating persons responsible for writing particular sections, comments as to desired amounts of material and depth of coverage, deadlines for successive drafts, etc.

Such an outline can be continually updated by changing the basic plan of the report, adding or deleting sections, altering deadlines and assignments, etc.; thus the outline becomes a continuing status report of considerable complexity, yet with all information readily accessible because of NLS features such as the content analyzer.

As sections of the report begin to take shape as NLS files, links to these files can be inserted at the appropriate locations in the outline. Now the coordinators, examining the status of the report effort, can jump instantaneously from a planning description of part of the report to a view of the actual work in progress; conversely, authors working on report sections can examine and reexamine the outline as they work.

It is difficult to make a meaningful comparison between report-writing at ARC and elsewhere, because the process always depends heavily on the individuals involved and the organization's general philosophy about reports, as well as the facilities available. However, we can safely say that this kind of close teamwork would be quite impossible for us without the augmentation aids that support it. To collaborate as tightly as we do without augmentation, we would need an immense amount of time and a staff of full-time coordinators, clerks, and typists.

3. Advantages

a. NLS

As a tool for coping with the mechanical problems of report writing -- input of material, assembly of material into a report structure, organization within the structure, text editing, and final output -- NLS has proven to be superb. For some operations, such as automatic searching of thousands of words of text for particular words or phrases, NLS is several orders of magnitude more efficient than paper-and-pencil technology.

To illustrate the advantages of NLS, let us briefly consider a few specific examples.

For entering text, NLS becomes a super-typewriter. This is probably the simplest possible use of NLS. The advantages of NLS over a typewriter come from several factors:

(1) Backspace-character and backspace-word keys, which cause immediate deletion of the last input character or word.

(2) The user's knowledge that further corrections can easily be made later.

(3) The availability of NLS's study capabilities. This is the most critical advantage, since it allows the writer to go back rapidly over what he has already written and reorient himself as he moves from one topic to another.

As a tool for assembling, organizing, and reorganizing material from diverse sources (recent input, existing files, etc.) NLS replaces such technology as the loose-leaf binder, the blackboard full of notes, the extra information written on small slips of paper and clipped to pages in the looseleaf binder, etc.

In our present state of evolution, this replacement is necessarily incomplete -- the older methods are used in coordination with NLS. However, NLS dominates the overall technology for organizing material and to this extent it greatly increases efficiency.

The master copy of a report in progress is a set of NLS files. The insertion of new material as it becomes available is accomplished quickly and smoothly, and the working material is completely legible at all stages.

The difference between working with a formatted NLS display and working with a binder of cut, stapled, pasted, and pencil-marked typewriter copy must be experienced to be appreciated. Moreover, a complete, fresh, fully formatted printout of the existing draft can generally be obtained in a few minutes, at any stage of the job.

The term "editing" is used here to mean the task of going through a draft or a section of a draft and correcting errors of spelling, grammar, style, and (within limits) content.

Our reports generally receive two editing passes: one by ARC's technical writer, using NLS, and a second by one of the SRI editing staff using pencil-and-paper methods.

NLS permits the augmented editor to work a great deal faster than he could with paper and pencil. Of course, he works with the knowledge that another editor will go over the draft and habitually leaves many decisions up to him.

The advantages of NLS for editing stem from two basic factors -- sheer speed, and the power of automatic searching.

The speed is just brute force in action: it is quicker, for example, to specify the command

"Replace Word," point to a location, type the new word, and hit the "command accept" button than it is to make the equivalent marks on hard copy. Using "Jump" commands to locate cross-referenced locations in the text is faster, by orders of magnitude, than flipping pages in a binder.

Automatic searching for specified strings of text permits operations that are virtually impossible for the unaugmented editor.

For example, many writers consistently misspell certain words. The augmented editor can execute a "Substitute" command that will correct all ocurrences of several different consistent misspellings, throughout a file, in a single operation.

For another example, suppose that halfway through a lengthy draft, the editor suddenly finds something that makes him uneasy about the way a particular term is being used -- a term used many times in the early portions of the draft.

The unaugmented editor would be faced with the prospect of reexamining many pages of text, looking closely enough to find all occurrences of the suspect term. This situation is one that editors encounter quite frequently, and the work involved is both lengthy and fatiguing.

With NLS, the problem disappears; the editor uses the content analyzer to find all occurrences of the term automatically. If he then decides that a different word should be used, he can use a "Substitute" command to make the change throughout the file or in specified portions of the file.

b. A Note on Structured Text

Our use of "structured text" -- i.e., a format that reflects hierarchical relationships among "statements" or paragraphs -- is somewhat controversial with some of our readers.

Many of NLS's most valuable features depend upon structured text. Moreover, structured text has advantages of its own. The special formatting carries information that is not in the text itself, thus increasing the overall "bandwidth" and permitting greater economy in writing.

Structured text is a direct consequence of the use of the computer as a writing medium, just as standardized punctuation resulted from the introduction of movable type as the medium for publishing. Hopefully, structured text will turn out to be only the first step in the development of new ways to show, by formatting and other nonverbal means, various relationships among the units of information that make up a document.

c. Assembly of Reports

As noted above, NLS is used to put together material from various sources to produce a complete draft. This raises the possibility of keeping on hand a large collection of material that describes our work in its various aspects, frequently updated so that at any time we can extract what is needed for a report. This material, ideally, would make up a very sizable part of each report and would require only simple rewriting, thus greatly reducing the labor of report-writing.

This idea has been with us for years, and we have made some progress in implementing it. To the extent that it actually happens, it is very worthwhile; however, only a very small part of a typical report is actually done this way.

In the present report, for example, only the preface, the bibliography, and the appendix are taken directly from existing material. Larger portions are derived from existing material, but with complex and extensive rewriting to suit the needs of this report.

4. Problems

The problems of augmented report-writing fall into several categories, described in the following sections.

To some extent, the problems are not merely report-writing problems but augmentation problems, i.e., problems created by side-effects of augmentation upon the total ARC system.

a. Technical Problems

The simplest problems are technical in origin. For the most part, they seem not to relate to gaps in technical capability -- i.e., "missing features" -- but rather to inadequate performance of the technical systems on which we have come to depend.

As noted elsewhere in this report, system response is degraded when several users are working simultaneously; because of our team approach to report-writing, this degradation tends to be worst at the worst possible time -- just when several people are trying to complete their contributions to the report.

A more critical problem arises from technical limitations on file storage. Two storage media, disc and tape, are available for report purposes.

Tape, in the current system configuration, is inconvenient and can only be used for long-term backup copies.

In the last few months regular procedures have been established for tape archiving; so far, however, relatively few files have been saved on tape in such a way as to be reasonably accessible.

Disc storage space is severely limited. Reports take up a great deal of space compared to other types of files in use by ARC, while at the same time they are less useful in day-to-day work by ARC personnel.

Work on a report generally involves considerable shuffling of files in order to obtain space for the new report files; files not related to the report are moved into out-of-the-way locations, and we sometimes lose track of them in this process. Duplicate copies are destroyed to make more room, and here there is a chance of inadvertently destroying the last copy of a file. Furthermore, if only one copy of a file is kept we face the possibility that a system error will destroy it.

The actual loss of a file has become a relatively rare occurrence, as users have become aware of the problems and developed habits and procedures to safeguard their files. However, we expend a great deal of time and energy on these safeguarding measures, and this seriously slows our report-writing efforts as well as all of our other on-line work.

When work on a report is finished (or temporarily halted) a reverse process takes place: the report files are hidden away, with a risk of losing them.

b. Problems of Explaining the "Augmentation Culture"

NLS and our other augmentation systems are part of an exceedingly complex total system that includes all of our set procedures for doing things, our management methods, our goals, and our strategies and priorities for doing research. This total system is the tangible part of the "augmentation culture"; the intangible part consists of personalities, emotional interactions, intellectual orientations, etc.

It is a tiny and incomplete culture with a brief history -- a laboratory model. Nevertheless, like most cultures it is incompletely understood by the people inside it. The long-term side-effects of any innovation are unpredictable, and occasionally such side-effects give rise to problems.

The problems of the augmentation culture affect all of our activities to varying degrees; with reports, however, these problems are multiplied, because the central task is (in principle) to explain this same culture to a reader who has no direct experience of it.

In practice, we rarely or never attempt a complete explanation; instead we describe various important aspects of our work, one at a time. Unfortunately, this leads to a fragmented picture in which the true context of our work -- a whole-system approach -- tends to become attenuated.

c. Problems of Organization and Technique

Practically nobody likes to write reports. What is worse, very few people are capable of writing good reports on difficult topics without very great expenditures of time and energy. At ARC, we have responded to these problems in several ways:

(1) We have developed an elaborate system of computer aids for writing (as well as other purposes).

(2) We have negotiated our contracts in such a way as to require a minimum number of reports.

(3) We have used live demonstrations and films to supplement the communication function of reports.

(4) We have experimented continually with different ways of organizing a report-writing team to function within our culture.

Some commentary on these responses may serve to illuminate the problems.

With regard to the first item, the computer aids are magnificent (as described above).

Reducing the total number of reports required is debatable; it can be argued that it would be easier and more effective to produce small reports at more frequent intervals, thus making final reports much less critical and easier to write, since they could lean upon the information contained in recent short reports.

It is probably true that films and live presentations are more communicative of what we are doing than any written report could be. However, they do not lessen the demand for good reports and they cannot be stored on line for computer access.

Experimentation with ways of organizing the reporting effort will probably be the most fruitful way of solving the problems.

We are still only beginning to develop a good organizational approach to the report problem. The present report is being produced under a rather elaborate plan of highly distributed authorship with a single individual as planner, coordinator, and "pusher."

At this writing, our principal difficulties are in getting all the various authors to finish their contributions on time and in sticking to any one scheme for the organization of the many sections. Although early analysis may well be misleading, the situation suggests that we have concentrated too hard upon the distribution of responsibility for writing sections, thus neglecting other equally important requirements for a workable delegation of authority, better forecasting of the effort involved in various phases of the work, better timing, etc.

E. The Augmented Presentation

1. General

When a group of people meets for purposes of discussion, briefing, planning, etc., it is often necessary to communicate a prepared body of information to them for review, orientation, or tutorial purposes.

The usual mode in such a session (which might be a meeting, lecture, seminar, or any other sort of gathering dealing with any sort of subject from poverty problems to sales policy to biochemistry) is for one person at a time to speak, with various degrees of preparation, and various levels of support from audio-visual aids. Effective use of these aids usually requires special preparation of the materials (tapes, slides, charts, etc.), or special effort at the time to write on a blackboard or image-projector suface -- although occasionally there is an opaque projector or TV circuit that enables the speaker to integrate some of his normal working material into his presentation without special preparation.

To any of us who are seasoned users of NLS, participating in conventional presentation processes has become even more painful than it was before we became accustomed to augmentation. We have grown so used to the ability to move around easily within any of our informaton bases (designs, documentation, reference material) and to switch quickly among the many useful ways of viewing the information, that we wish that sort of capability were available to the speaker.

If a closed-circuit TV system is available to distribute the computer-display images to a group, then a speaker with on-line access to a computerized information system can indeed make an "augmented presentation."

2. Early Experiments

In October 1967 we set up a special conference room, with TV monitors arranged so that some twenty people, sitting around a rectangular table, could watch an augmented presentation. We inaugurated the facility with a two-day meeting between our staff and the technical monitors from the four agencies then sponsoring our work.

At all times during the meeting, the speaker sat at the NLS control console (a position at the table that had a keyboard, keyset, and mouse in addition to a monitor) and could instantly access and display information in any of our total collection of files. Some of the material was specially prepared reference material for agenda purposes, but most of it was our actual working material -- planning, design, documentation, and scratch-note files.

A trial agenda was prepared ahead of time in an NLS file, but from the outset it was subject to ad hoc review and modification.

To give the participants a better means of talking about the displayed material (i.e., voicing questions, challenges, or suggestions), each had access to a mouse which could control a second tracking spot on the screen. This spot was used as one would use a wooden pointer to draw attention to an item on a blackboard; it was of different shape from the control spot used by the speaker, so it was easy to keep track of who was pointing.

This mode of running a project-review meeting proved very rewarding. We subsequently used the setup frequently for special presentations (mostly for visitors), also with great success, and perhaps a dozen times for working meetings of our group. In mid-1968 we dismantled the setup, since we needed the space for expanded shop area and the TV monitors for more NLS consoles.

To get at least the capability for a larger audience to view a display, we have had a flexible arrangement for attaching two or more TV monitors to one NLS console, We hold many demonstrations and presentations this way, for groups up to about 20 people.

3. Development of Improved Capabilities

By mid-1968 we had acquired some special-effects video equipment that gave us the following capabilities:

Video monitoring and switching: An operator monitors images generated from a number of sources and can select from these the signals to be channeled to various destinations.

Video mixing: Two signals can be blended, with adjustable strengths, to superimpose two images.

Signal fading: A selected signal source can be slowly diminished or strengthened to achieve smooth, gradual transistions between image-composition states.

Image splitting: The screen image is divided geometrically into two parts, each being the corresponding frame part from a different image source. A horizontal or vertical dividing line can be set at any postion, or a rectangular inset can be made in one corner of the frame and its size adjusted.

We also located a NASA-owned Eidophor Video Projector at the nearby Ames Research Center and were fortunate enough to be able to borrow it on several occasions. It projects our video images onto a large single screen with very high quality and is suitable for presentations to large audiences.

As we were experimenting with augmented presentations, we learned of a special technique developed by W. A. Palmer Films Inc. of San Francisco for directly filming the video image (and sound from a microphone circuit) to produce a motion picture. On a number of occasions, we have arranged for a camera and operator to film our presentations.

We also developed special coupling equipment enabling us to lease voice- and video-grade transmission channels to a remote site. From the remote site, we can then use our computer system to support two regular NLS consoles and any associated video-control and -projection equipment for conducting full presentations.

4. Large-Scale Augmented Presentations

We successfully used the full range of these techniques in two large-scale presentations -- in December 1968 at the FJCC in San Francisco, and in October 1969 at the annual ASIS meeting in San Francisco.

These presentations were set up with the main speaker seated at the front of the auditorium, at one of our regular NLS consoles, to one side of the large screen but in full view of the audience. He had a microphone, and could directly address them as though delivering an ordinary speech.

A TV camera mounted near him caught a full-size face view, and a "production manager" for the presentation, seated at the rear of the auditorium with switching, mixing, and frame-splitting control, could put the face view on the screen for a much more effective feeling of direct contact between speaker and audience.

When the speaker chose to use NLS, and turned his head slightly to see the NLS display screen, the projected image would be switched to show his computer-display view to the audience.

Frequently, images captured on mobile TV cameras in our laboratory at SRI were switched onto the projection screen, so that the presentation could include both a tour of our laboratory and participation from people located there.

Four different researchers at the laboratory worked during the presentation at consoles in our work room (various shots during the presentation verified that they were there in the background, actually using the consoles while the rest of the show went on).

At various times, the full face of one of the four was brought on screen, dialogue went on between him and the main speaker to introduce him and his topic, and then he took over and ran a portion of the presentation in much the same manner as the main speaker had been doing -- the screen being able to show alternate (or split and/or mixed) images of him and his display-screen image.

For both conferences, the complete presentations (each about one hour and forty minutes in length) were captured on 16-mm film (black and white with optical sound). The ASIS film is the better of the two, and we have made eight copies that have been in active loan circulation since the first of the year. It is better than any earlier self-contained form for conveying an overall picture of our augmentation system. The copies have been loaned to some 70 organizations, including several in Canada and one in France, and we hear of many repeat showings put on by a borrowing organization as a result of the response of the initial audience.

One must see one of the movies to appreciate the difference between the augmented presentation system and what could almost have been substituted for it -- e.g., a previously made set of slides of all the display-screen views, and a very fast-acting and large-capacity slide-selection system under control of the speakers.

For classroom lectures, for project briefings in connection with complex system-developmment projects, for presentation of complex issues to a reviewing body (such as a congress), etc., these extensions of our augmentation system toward an augmented presentation system seem extremely promising.

We intend to push further development of these techniques for application by smaller groups within a system-development team -- group meetings for brainstorming, plan and design review, etc.

5. Conclusions

Assuming that techniques similar to ours are bound to evolve for widespread use in intense intellectual work both by individuals and in small groups, it is quite obvious that extensions to the techniques, such as described above, for augmenting presentations to larger groups will become quite important. Although pursuit of presentation techniques is not our central goal, we would like to encourage a general appreciation of its potential.

IV THE NETWORK INFORMATION CENTER

A. Overview of the ARPA Network

1. Basic Description

The Advanced Research Projects Agency (ARPA) of the Department of Defense has inaugurated a novel experiment, the purpose of which is to explore the possibility for fostering effective, real-time cooperation and interaction between approximately fourteen of its contractors, all of which are doing research on the development of advanced information-processing techniques.

To carry out this experiment, the Agency is in the process of establishing the ARPA Network, which is a data-communication network linking the computer facilities of these fourteen research centers.

Each of these centers has a coordinated collection of resources -- both technological resources such as computers, storage facilities, input/output devices, and software systems, and human resources such as knowledge, skills, and methodologies.

It has been necessary, in general, for these centers to engage in wasteful duplication of basic resources and to live with adverse restrictions on the number and quality of special-purpose resources available to the researchers at any one center. But wide-band connections between centers will bring many -- perhaps eventually all -- of any center's computer resources within the reach of every user in the Network.

Two basic domains of work are involved in this experiment:

(1) Development of technology for providing intercommunication between programs in the different centers

(2) Integration of the distributed resources of these centers into new patterns of support for users of the Network.

Detailed information-network technology is described in a set of companion papers presented at the 1970 Spring Joint Computer Conference (see Refs. 19 through 23). The following simplified description is intended as background for the discussion of the Network Information Center contained in subsequent sections.

2. Host-to-Host Communication

Each research center, or Network "node," will provide access to one or more local computers dedicated to research on information-processing techniques. These local computers are called "hosts."

As part of the Network, ARPA supplies to each participating center a small, specially programmed computer called an Interface Message Processor (IMP) which is connected to the host (or hosts) for that center.

The IMP provides each host computer with a uniform functional interface into a network consisting of 50-kilobaud transmission lines leased from telephone companies.

Each IMP controls traffic to and from its host or hosts, as well as traffic along the Network channels. As far as each host is concerned, the Network may be treated as a multi-port black box with IMPs serving as the ports.

Very much like voice-path connections between users of a telephone system, one-way message paths called "links" may be set up between specified hosts by the IMPs.

For example, "Link AD3" might designate a path from Host A to Host D differentiated from other possible links from A to D by the label "3."

Special Network-monitor programs resident in each host computer activate such links and negotiate among themselves to allocate them to specific communications-path purposes.

The "0" links are reserved for control messages directly between the monitors.

3. Program-to-Program Communication

Program A in Host A may ask for a communication link to be established from it to Program B in Host B. As currently planned, the following chain of operations would occur:

Program A submits the request to Monitor A.

Monitor A uses Control Link AB0 to negotiate this with

Monitor B. After Monitor B ascertains that Program B is available for such communication, the two monitors together assign one of the ABn links to this communication purpose (for example, Link AB4).

Monitor A verifies to Program A that its request has been filled, and that Link AB4 is now connected to Program B.

Program A is now linked to Program B by what appears to the two programs as a private communication channel between them; any data sent out by Program A via Link AB4 is delivered directly to Program B.

Any sort of information that could be sent between two programs within the same computer may be sent along such a channel.

For example, Program A might send control information to Program B asking it to set up a reverse link, and to transmit along that link the results of processing a File B that resides within Host B. Or conceivably, Program B could be asked to link to Host C to get and process File C.

Program A could be an interactive preprocessor serving a User A who is connected directly and locally to Host A and for whom most of the service is being supplied by a Program B. Program B could be a sophisticated user subsystem such as Mathlab at MIT, the Culler-Fried system at UCSB, or our NLS.

In any of the more sophisticated, display-oriented subsystems, User A's hardware could be different from the hardware used at Program B's home site.

Program A might provide an adaptation for the particular terminal hardware being used by User A.

It probably would also provide some of the local interactive feedback such as character echoing at a full-duplex terminal.

B. Some of Our Prospective Uses for the Network

1. Basic Processes

A variety of Program A that is likely to be established early at each host is a process that allows a typewriter terminal at that host to link to the timesharing executive in Host B.

Such a process would enable User A to log into the Host B's timesharing system and work as though he were a local user at Site B, using any of the typewriter-oriented subsystems that may exist in the B system.

2. Bootstrapped System-Transfer

For our first experiment in Network usage we have established an interface to the University of Utah's Network host, a PDP-10 computer, so that a programmer at one of our consoles can use the editor and loader-debugger at Utah to develop and check out programs to be run for our special use.

> This is the first step in a sequence of experimental uses of the Network that promises to help us significantly in preparing to transfer our software systems onto a new PDP-10 computer system next Fall.
>
> In carrying out this experiment, we will be coordinating a very large collection of technological and methodological resources in a novel way.
>
> > We are producing modified versions of all our language compilers. These new versions will still be run on our XDS940 system but will produce object code for the PDP-10.
> >
> > Our programmers will compose, modify, study, and compile the software being written for the PDP-10 using only the resources of our own Center, principally NLS and the special compilers.
> >
> > Then, when there is need to test a piece of the new software, they will activate Network links to put them in communication with Utah's PDP-10 executive system. The compiled code will be shipped over the

Network to Utah, where it will be loaded and executed.

Still working at our on-line consoles, our programmers will be able to debug the object code using the PDP-10's debugging facilities and, then, rapidly switch back to NLS to correct the source code as well.

In this way we plan to bootstrap the development of a PDP-10 version of NLS so that we can be up and running very quickly when our own PDP-10 is installed. We hope NLS will have been entirely written and mostly debugged by then. The entire power of NLS will have been used during cyclic stages of converting/rewriting, debugging, source-code updating, and documentation.

It is interesting to note that another PDP-10, which could be used for program checkout via magnetic tapes carried back and forth, is located just down the hall from our present computer. However, it promises to be significantly more convenient for us to use Utah's PDP-10 connected directly to our computer via the Network.

We can communicate our programs to it much more conveniently; and, while sitting at our own consoles to debug the programs, we can quickly switch to local NLS to inspect or modify the source-code files or to enter notes and documentation.

3. Centralized File Archiving

Another straightforward way we can benefit from the basic Network is by using the file-storage system at the University of California at Santa Barbara as our file-archive resource.

Besides providing service to the local ARPA project, UCSB's host IBM 360/75 is the central resource of the University's Computation Center.

IBM 2314 disc-file transports, programs to store and retrieve files, on-duty operators to run back-up dumps onto magnetic tape, and accounting and billing procedures to handle service to miscellaneous users are all part of this resource.

We consider using their file-storage system as our file-archive resource.

We would interface this archive system to our On-Line System so that the response to a call for some service in storing or retrieving an archive file could function exactly as though we were storing and managing the file locally.

Other Network users could also use this file-archive system either by interfacing to it directly or by interfacing to it through us.

This begins to suggest some of the power inherent in the network concept, since the number of subsystems available to each network participant can "pyramid" on the basis of a relatively small number of interfacing tasks.

We could gain considerable economic advantage from being able to share with many other users the capitalization and operating-accounting expenses involved in providing a basic disc-file archive system.

Installing and operating such a service ourselves would be a distraction from how we want to use our manpower -- but we have to have this capability.

Being able to use the Network to share this resource could be an important benefit to us.

4. Data-Base Management Service

We also hope to utilize one of the big-computer hosts with large discs and an operating staff, to help install a commercial data-base management system for use by us and the rest of the Network.

Again, the interface process would reside in our host and give our users a data-base management service that they can interact with as if it were a local service. None of the protocol required to fire up the remote system, properly format service requests, and specify delivery of its products need involve our users.

To present our users with uniform conventions over all of our different service functions, we would want to be able to compose service requests in an NLS file using our own particular syntax.

In addition, to take advantage of NLS power in study and modification, we would want to have query results converted into NLS-file form so that they can be integrated directly into our regular working files.

C. Need for a Network Information Center

To pursue such relatively straightforward uses of the Network, a user will need to know answers to such questions as:

What resources are available within the Network?

What conditions are associated with the use of a particular resource at a given node?

What interfacing is required to couple to this resource?

Who should be contacted to arrange for this?

More generally, each site will also need answers to questions about other nodes and about the Network such as the following:

What are the hardware and software resources?

What are the research activities?

What are the operating instructions for a given resource?

When will a prospective resource become available?

Who is interested in new display systems?

Who went to a particular conference?

Who else was interested in NIC Memo 7721?

The whole Network experiment will consist of negotiating, implementing, and operating many such arrangements. To carry out these negotiations effectively and to document the results of Network experiments, Network participants need access to information of the kinds mentioned above as well as a medium for recording their experiences; it is the role of the Network Information Center (NIC) to service these needs.

D. NIC Development Activity

1. Background

Our involvement in the development of the Network Information Center began shortly after the official announcement of plans to proceed with the development of the Network itself.

Realizing the importance of this information center to the success of the Network experiment, we volunteered to undertake the task of designing, implementing, and operating the NIC.

Since then, we have been gradually evolving and implementing plans for providing a coordinated and useful collection of services to Network participants. This has proved to be a far more difficult task than we originally envisioned for several reasons.

The Network will provide a completely unfamiliar working environment, and it is far from obvious just what kind of information services will be most conducive to effective action within that environment. Our contracts have contained no specific guidelines as to what form NIC services should take, and other Network participants have been equally vague and often contradictory in voicing their expectations with regard to the NIC.

We also have had no previous experience in providing information services to users outside our own Center, and have learned the hard way that this kind of operation requires a kind of organizational structure and management different from that which had been appropriate in the early years of our research activities.

In designing the NIC, we were conscious of the fact that more was needed than just good library facilities.

The technical sophistication of the tools we had available and of the users we would be serving demanded that we seek to make it possible for this community to derive significant advantages from the Network in terms of increased communication of ideas, designs, criticisms, and comments on needs and possibilities.

We recognized the challenge involved in fostering an environment that would encourage increased collaboration among individuals and groups at different nodes, and felt that the NIC would bear much of the responsibility for setting the direction of efforts to meet this challenge.

And in designing the NIC we were aware that it would not be sufficient merely to provide good technical features, but that the design must reflect the kind of coordinated user-system / service-system interface that we have felt is so important in the bootstrapping approach to the development of our own augmentation aids.

2. Orientation

In order to gain perspective into what was expected or desired in the way of service from the NIC, we talked with a number of Network site managers as well as with library science specialists.

Most of the managers were uncertain about the nature and extent of their probable participation in the Network and did not know what they wanted from the NIC.

What expectations we were able to uncover frequently showed extreme differences from person to person.

Some thought that there was little need for a NIC, while others thought that the NIC should supply initiative and leadership in the development of overall Network conventions and methodologies.

Some thought that the NIC was going to serve as a intermediary for handling working communication between hosts, providing teletype buffering and spooling operations, or that it would be responsible for specifying protocols for communication between different types of terminal devices.

The types of service to be provided, in terms of media for storage of documents and methods of enquiry, were perceived at points all along the continuum from completely off-line to completely on-line.

And the types of retrieval capabilities envisioned ranged from fully automatic to fully manual.

However, one reasonably consistent picture did emerge: almost all the managers contacted expressed a concern for the poor state of documentation within their projects and a hope that the NIC would be able to help them not only with Network-oriented documentation but with their in-house needs as well.

The need for improved documentation was seen not only in the area of "formal" documentation, such as user manuals and reference guides, but also in the realm of the "folklore" that evolves around every system and that is an absolute necessity for making effective use of a system.

Some managers expressed the opinion that a remotely located documentation-aid system might receive heavier use than one at their own site because there would not be the usual conflict between using the local facility's resources for documentation as opposed to using them for programming or other work, the latter always receiving higher priority in most researchers' minds.

Part of the "folklore," or informal documentation, would play the role of clearing house for communication on defects, bugs, needs, possibilities, etc., and would help to provide feedback on the status and accecptability of existing or needed programs and formal documentation.

The outcome of these discussions, and of analysys of the several tentative NIC designs that resulted, has been to take the position that we should initially provide Network users with a few relatively basic services and let the development of the "optimum" NIC evolve in a natural way as usage experience builds up.

Thus, we are planning to help Network users gradually to apply our concepts, conventions, tools, and techniques to relevant aspects of their own working environments, while at the same time providing them with a reasonable initial corpus of reference material.

We will make every attempt to encourage and facilitate feedback from these users so that we can expand NIC services to meet their needs for information access and exchange.

E. Current NIC Plans

1. Introduction

We aim to be prepared to provide certain basic types of services through the NIC, each of which will be expanded or elaborated as needed. This approach is our way of accommodating the uncertainty about the user-population's information activity described above. This section outlines the measures we have taken to accommodate these basic service needs.

2. Basic Library Services

The foundation of NIC operation must be a flexible library service, the development of which requires us to:

Accumulate a physical collection of information items,in various sizes and media, and then store them in a secure and orderly manner.

Provide a catalog in which the descriptions of these items are maintained, and from which can be obtained the keys (clues) necessary to locate the physical items.

Provide indices and associated query procedures to aid users in finding catalog items of interest.

Provide direct means for a user to obtain access to documents for which he possesses keys, and enable "browsing" where possible.

Provide direct reference help via two-way dialogue with an expert.

We will initially provide this basic library service in ways not too dissimilar to those used by other special information centers and clearing houses.

That is, we expect to be interacting with users through mail and voice-telephone channels, to be distributing hard-copy indices and catalogs, and to be mailing special query-response listings and references.

3. On-Line Services

We are also seeking to harness computer, display, microform and communication technologies to elevate what a "library" represents to its community toward new levels of "augmenting the intellectual processes of communication and collaboration."

To this end we are taking special measures toward "the way we do our library work here" -- the nature of the collection, the form of catalog and indices, the procedures for developing and maintaining them, etc.

And we are investigating ways of improving special aspects of the user's interface to the NIC -- his query and browsing techniques, his means for doing bibliographic work, his means for publishing communiques to the community, his means for staying in touch with current activities, etc.

As the technology and methodology of Network utilization develop, there will be a rapidly increasing number of widely distributed terminals through which a person can gain on-line access to the NIC, and we are taking very strong measures to be able to serve them.

Besides developing the services described below, we have invested a great deal of effort toward increasing our service capacity with special system studies and the installation of a bank of external core and three high-speed swapping drums; and this fall we will be shifting to a new computer (see Section II).

We are prepared to supply some on-line query service as soon as the Network can support host-to-host full-duplex typewriter transactions -- the simplest kind of general Network usage. Continual development of the variety and sophistication of the service will follow lines discussed in the rest of this section.

During the next year most remote on-line users will be served by TODAS, a typewriter-oriented version of our On-Line System (NLS).

Initially, both access to our subsystems and the number of Network users that we can accommodate will be quite restricted, but we plan for steady improvement in both interactive quality and amount of the service available to Network users.

We feel ready to support three Network TODAS users now; when installation of the new drums and external core is completed late this summer, we might be able to increase this to five; and, depending upon the response possible using our new computer, we may be able to support as many as fifteen by the end of the year.

Those sites having display terminals that can emulate typewriters with high transmission rates will be able to get fast-response from our typewriter-oriented system -- service that is really quite respectable even without direct screen-selection means such as a mouse or tablet.

We are currently modifying an Imlac display console for remote use on our system, and we hope that it will become fully operational by early fall.

It will eventually have full NLS capabilities including our standard mouse and keyset input devices, and any other Network user possessing such a terminal could quickly avail himself of similar service.

As soon as we have transferred our present systems onto the new PDP-10 computer and have achieved reasonable stability in system reliability and response, we intend to concentrate upon developing techniques to permit general use of our NLS capabilities on a wide variety of remote display terminals, and we anticipate having a growing Network load of this type by next summer.

V CONCLUSIONS, RECOMMENDATIONS, AND PLANS

A. Introduction

Our experience in developing and using augmentation systems, some of which was described in Section III, gives us confidence in the validity of our long-term goals and approach to augmentation research. We are beginning to have strong feelings that our efforts, still proceeding under the guidance of our bootstrapping strategy, are on the threshold of yielding significant payoffs.

The concept of augmentation systems is no longer in question -- such systems are bound to come. The only uncertainties involve the rate and direction of development and what can be done to improve both, so as to foster more efficient evolution and ensure the earliest possible application of these systems to solving real problems facing society.

This section summarizes the conclusions we have drawn from our research program and outlines some of our plans for future activities.

B. Conclusions Relevant to Other Augmentation-System Developers

1. General Comments

The experience we have had in setting up the Augmentation Research Center facility may not be directly applicable to other system developers, but there still may be something to be learned from our accomplishments and disappointments in implementing some of the components of our system.

In this section, therefore, we will take a look at our facility, evaluating some of its unique aspects and discussing some of the major problems that have been encountered in its evolution.

This facility is described briefly in Section II-B, and a more complete description is contained in Ref. 17.

2. Accomplishments

a. Display System

(1) General

Our display system uses centrally located display-generating equipment and a closed-circuit television system to distribute images to individual consoles.

This approach to display-system design was chosen on the basis of cost and flexibility, and a detailed description of the system and of considerations that went into its design is given in an earlier report (Ref. 11).

After considerable experience operating this system, we are still pleased with the basic approach.

(2) Usage Advantages

The closed-circuit television system offers several distinct advantages over other means of producing displays at a user console.

Since only a television monitor and a video line are required to present the display at each NLS console, the design and location of consoles is flexible enough to facilitate experimentation with different types and to permit moving them about with a minimum of cabling problems.

The video signal can be inverted to provide a dark-on-light display, which is usable in higher ambient light conditions than the more common light-on-dark presentation and which makes flicker in the display image less noticeable to the user.

A significant storage time can be obtained on the vidicon surface by proper adjustment of the television camera. This reduces the flicker effect that is present in the original CRT display, and with this system we find it possible to produce satisfactory images with a regeneration rate of about twenty per second.

(3) Maintenance Features

Since the display equipment at each NLS console is simply a television monitor, it can be replaced by a spare for maintenance without taking the console out of service.

The location of the display-generating equipment in the computer room, where complex maintenance and repairs are more convenient, makes possible an uncluttered office environment in the user's working area.

Having two identical display systems, from display controller through actual monitors, is a major factor in maintaining up-time in spite of the high level of maintenance that has been required on these systems.

Since there is not a fixed one-to-one relationship between display-generating equipment and NLS consoles, users may select consoles freely on the basis of current needs even when part of the display system is out of service.

(4) Potentials for Extending Capabilities

Since a great variety of commercially available equipment is compatible with our video system, there are many potentials for innovative use of our system that would be much harder to realize with conventional display-distribution techniques.

For example, we have used high-quality projection television as part of our presentations at the 1968 Fall Joint Computer Conference and at the ASIS Conference in 1969 (see Section III-E).

It is possible to use multiple TV monitors or intermediate-size projection equipment for smaller groups, and experimentaion with these techniques will be a major element of the team-augmentation work to be carried out under our next contract.

The video capability offers additional flexibility in the images that may be presented on the screen.

For example, in the conferences mentioned above, live television pictures of the people and equipment involved were freely used, both alone and after being mixed with computer-generated images.

This, also, will be a significant factor in team collaboration at a distance where pictures of the people involved can be used, either mixed with or inserted alongside of the computer-generated images.

Another potential use of the video is the viewing of microform documents.

Many systems are now available that use closed-circuit television for the storage, retrieval, and viewing of microform documents, and we expect this kind of system to become more prevalent in the future.

We already have plans for experimenting with various switching techniques in our display-distribution system, and the inclusion of provisions for microform viewing would require very little additional effort.

b. NLS Consoles

(1) Console Design

As mentioned above, the use of video for displays allows considerable flexibility in console design. We have experimented with many arrangements over the past few years and are now using the three basic designs shown in Figures V-1, V-2, and V-3.

(2) Keyboards

The keyboards in use have gone through several stages of design with special attention to touch and layout. They now have a key force of 80 grams and a good "feel." They are well accepted, and we find that new users rapidly accommodate to the locations of special keys such as command accept and backspace.

(3) Mouse

We find that the mouse is still the most convenient locating device for our purposes. It is described in Ref. 7, along with some experiments in the use of various selection devices. Available commercially, it is now used on other systems as well as our own.

(4) Keysets

The keysets in use were designed with special attention to "feel." Tolerances on the moving parts are very close to give smooth, positive action. Key

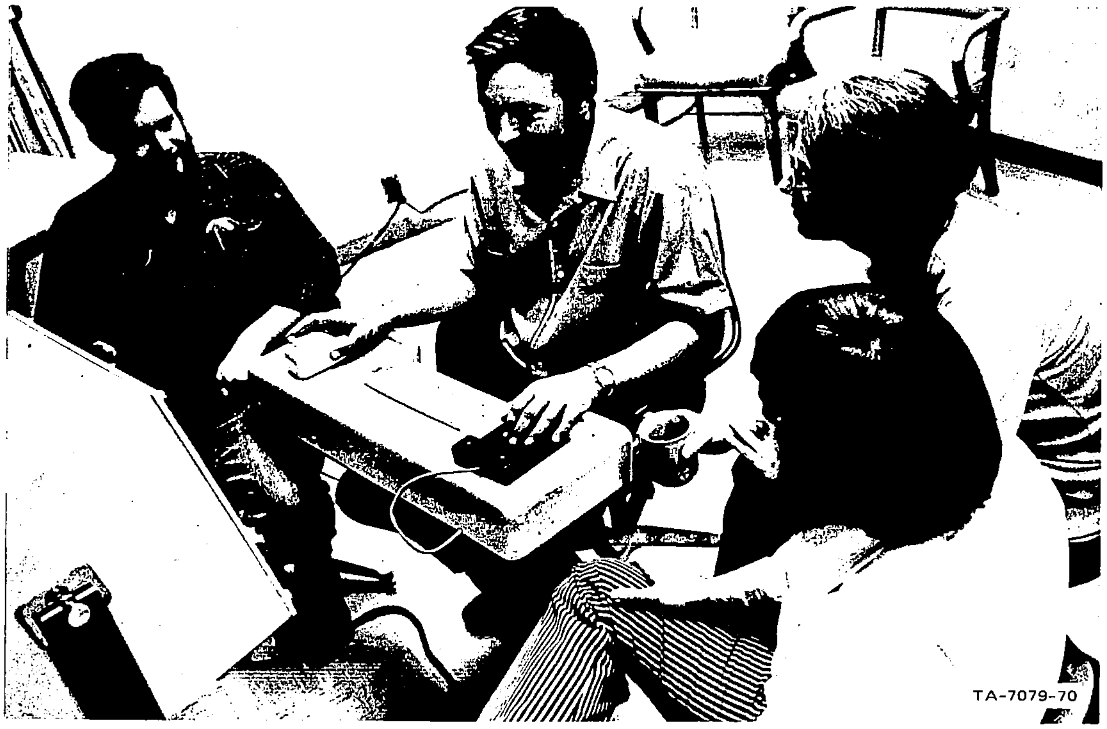


FIGURE V-1 "HERMAN MILLER" CONSOLE. This console, designed in cooperation with Herman Miller Research, was first used for the Fall Joint Computer Conference in 1968. The keyboard-keyset-mouse unit is attached to a swivel chair, with the TV monitor on a separate stand. It is a little confining to many people and the limited space for operating the mouse is an annoyance. However, it is reasonably comfortable and is useful for meetings and demonstrations because the user can turn and be part of a group while working.

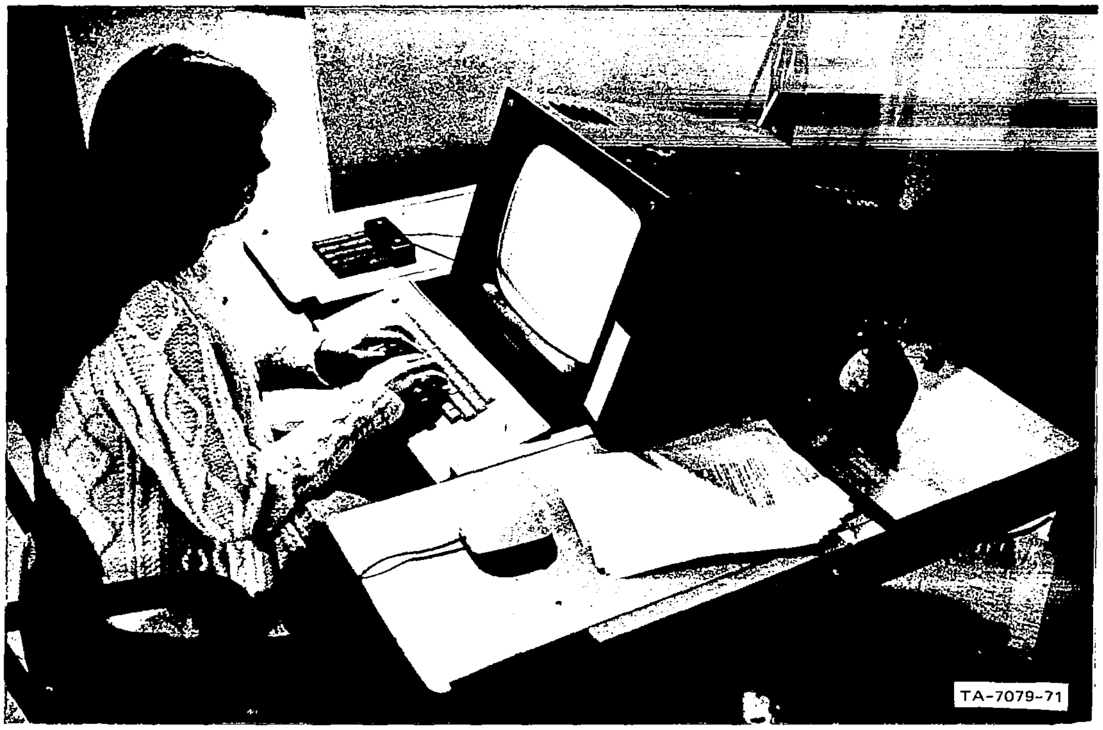


FIGURE V-2 ONE-PIECE CONSOLE. This console, designed about two years ago, basically a table on wheels with the TV monitor mounted at a suitable angle for viewing. Both sides of the table have pull-out shelves for mouse, keyset, and working papers. This console now seems too rigid to most users. The TV monitor is too close for some, and there is not enough extra working space for notebooks and papers.

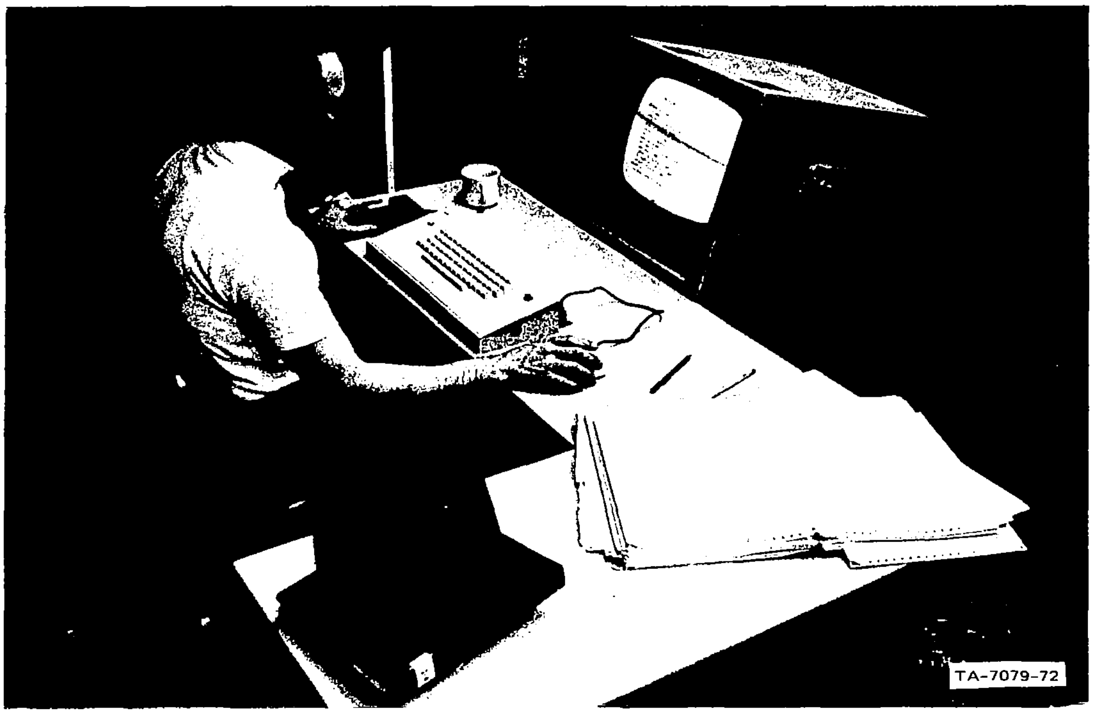


FIGURE V-3 CURRENT CONSOLE DESIGN. This console consists of a small table with integral keyboard. Connectors for the mouse and keyset are immediately behind the keyboard. The TV monitor is on a separate stand. The table stands low, placing the keys at a convenient level for typing, and other tables can be placed on either side for additional working space. This arrangement is very flexible and allows plenty of working space as needed. It is particularly good for group collaboration, since the monitors may be turned for better group viewing, and entire consoles are quickly moved into different working arrangements as needed.

spacing and size approximate those of a piano.

c. Printer

We value a high-quality line printer that produces upper- and lower-case alphabetics and a full complement of ASCII symbols. The one now in use is a Data Products model M600-11A, which has been very reliable and maintains high-quality output.

We normally use specially perforated paper along with our output-formatting programs to produce hardcopy, complete with pre-punched holes, which is ready to be inserted into a standard 8.5x11 three-ring binder. This feature has been enthusiastically accepted and is of particular value in the production of reports and other documentation.

d. Software Architecture

The overall architecture of our software systems is described briefly in Section II.

The use of a compiler-compiler to augment the development of a versatile set of special-purpose languages has proved quite valuable in facilitating debugging, rapid modification of existing features and addition of new features, and orientation of new programmers.

Our efforts to use higher-level languages wherever possible should result in additional payoffs when we transfer our software to a new computer this fall.

e. The On-Line System

NLS has been evolving for many years and has proved its value both as a tool for bootstrapping its own evolution and as a flexible laboratory for developing experimental working methodologies (see, for example, Section III).

3. Problems

The major problems we have encountered in developing our system stem from four basic difficulties:

(1) Trying to fit a very large operational system into hardware too small to support it

(2) Using an available timesharing system that has proved inadequate for our purposes

(3) Developing too much special hardware with the associated problems of unreliability and missed schedules

(4) Unreliability of some components of the facility, particularly file storage devices.

a. Hardware Limitations

Limitations imposed by both the hardware configuration and the timesharing service system have resulted both in worse response and in the ability to handle a smaller number of on-line users than we had expected.

Moreover, this has resulted in the necessity for expending more resources on programming the service system than we had originally anticipated, and our planned work on evolving the user system has suffered accordingly.

We originally had hoped to be able to service ten on-line NLS consoles at a time, but we now find that only five can be operated with reasonable response to the users. The primary factors affecting this response time are swapping (core-drum transfers) and file I/O (core-disc transfers).

To analyze factors affecting the response of the timesharing system, we developed a highly parameterized, discrete simulation model of the system (Ref. 17).

This was used to evaluate the impact of changes in the hardware configuration, such as faster drums or larger core memory, as well as the effect of various mixes of user demands.

Changes in the scheduling and swapping algorithms were also tested in the simulation, with the results being subjected subsequently to experimental verification.

The major hardware limitations of the facility are the maximum core available (the 940 can accommodate only 64,000 words), and the limited address space available (a program can address only 16,000 words).

Maximum core limitation results in the swapping bottleneck mentioned above.

Limited address space requires an overlay structure inordinately complex for a program the size of NLS.

This means that programmers must expend a great deal of energy designing overlay structures and are constantly running into problems of full overlays as the system is expanded.

This factor alone probably increased the cost of programming the system by ten to twenty percent.

In retrospect, we recognize the error in the following design decisions:

The decision to refresh displays from main core memory was a bad one. Memory bandwidth is no problem, but tying up the memory with display images (which require "frozen" pages of core) reduces the space available for programs and makes the swapping bottleneck even worse.

With the present system design, providing feedback to a user requires that a program unique to each user be swapped in for every character of input. A different approach might have consolidated some of the interactive feedback, thus reducing the swapping requirements.

b. Timesharing System Defects

The timesharing system obtained for use on our computer has been another significant facility limitation. Although this timesharing system was one of the most sophisticated available at the time we acquired it, it has proved itself inadequate to provide the service which we must have.

Partly this is because the designers of the system did not anticipate accommodating programs as large as NLS. We are constantly running into size limitations built into the assembler, loader, and debugger.

Another problem is that we are the only people using this timesharing system for an application like NLS and, therefore, have had to carry on alone its maintenance and evolution.

There still are many bugs in the system that would be totally unacceptable if we were not a research-oriented organization with few naive users dependent on the system for service.

Some of these bugs have been known about for two years or more, but higher-priority demands on our programmers' time have always prevented us from locating them. With transfer to a new computer now imminent, we plan to live with them a little longer rather than wasting valuable energy trying to fix them at this time.

c. Problems with Custom-Built Hardware

Much of the hardware in our system has been custom built to our specifications, either by our own staff or under contract.

We have badly underestimated the resources needed for constructing some of this hardware, the results being missed schedules, failure to meet specifications (for the display system in particular), and high maintenance costs.

In putting together an experimental facility for augmentation research, we now see the desirability of restricting special hardware development to those areas most directly associated with user features -- e.g., user-console design.

Wherever possible, use should be made of commercially available computers, interface devices, display-generation equipment, etc.

d. File-Storage Device Unreliability

File unreliability has also resulted in considerable loss of time to both programmers and other users.

Although the disc-file system we are using is quite good when compared with others in the computer field, adequate file backup features were not incorporated into the system design.

This means that users must either go through elaborate procedures to keep several copies of important files or occasionally suffer significant lost time when a file goes bad.

Until file storage devices are much more reliable than those now available, sophisticated automatic backup facilities should be designed into any system.

4. Maintenance Experience

a. General

General reliability of the facility has been good. Computer up-time has been high, although the reliability of the disc-file system has been only fair.

We had a period of several months of above-normal error rate, with five days down while clock tracks were rewritten.

The chain printer we originally bought had marginal print quality, was unreliable, and we had difficulty getting satisfactory maintenance service.

Consequently, we have replaced the unit with a Data Products drum printer that has 96 printing characters per line with upper- and lower-case alphabet. The print quality is excellent (witness this report), and so far it has been fairly reliable.

b. Display System

We have spent more effort on maintenance of the display system than on any other part of the facility. Since our display system is somewhat unusual, we will discuss some of the problems encountered in so far as they reflect considerations relevant to the design of other similar systems.

(1) One basic system limitation has been the inability of the display-generator CRTs to produce adequate light for the vidicon pickups. This means that many elements of the display system are operating at marginal levels.

The display-generator CRTs must be run at such high intensity that their life is relatively short.

This high intensity also causes difficulties in. maintaining good focus over the entire image.

To operate with these low light levels, the vidicons must be quite sensitive; since sensitivity drops off with age, they, also, have a relatively short useful life.

(2) Because the writing speed of the display generators is lower than we had specified, we have a flicker problem when all six screens on the system in use are reasonably full of text.

We can compensate to some extent for this flicker by careful adjustment of the vidicon beam current and target, but this adjustment needs frequent attention.

We have considered using longer-persistance phosphors on the TV monitors and will experiment with this in the near future.

(3) In addition to these difficulties there are some basic weaknesses in both the display generators and the television system which could be corrected in future designs by more careful attention to component quality control and the inclusion of circuit-design and layout features which would facilitate maintenance.

C. Our General Orientation Towards Future Research

Over the years we have frequently been faced with the problem of attempting to explain to people outside of ARC just what our On-Line System (NLS) is.

It is usually fairly easy to get across the idea of augmentation in the abstract, but it is much more difficult to convey to people who have not made extensive use of NLS just how powerful it is as an augmentation tool -- it is very easy to get trapped into looking at NLS as just a nice text-editing system without seeing all the power that resides behind that particular aspect of its surface appearence.

Recently we have developed a way of looking at NLS which helps to convey some of the power that we know is present, and that is to view NLS as a worker's on-line "office" -- that is, his normal, daily, local working environment. The analogy follows from the observation that to a naive visitor an office can look like just another "room," but to the person who uses that office it serves as an interface to many of the capabilities of an entire organization.

The office serves as a place where a person can work at organizing his ideas, studying correspondence, reports, etc., and formatting his own written materials. The office has associated with it communication links such as mail and telephone as well as access to secretarial and clerical services that are used on a daily basis.

This same sort of picture can be applied to NLS, for it is used on a daily basis to help with organizing, studying, formatting, etc., and provides now, or soon will provide, many of the other communication and clerical services normally associated with "office work."

NLS also has many similarities to an office in the way that both act as interfaces to extensive external capabilities.

Just as one would not expect to find complete publication, laboratory, library, and filing facilities in the average office, one should at present not expect any single augmentation facility such as NLS to provide all the computational facilities that a person might wish to call upon. But, in the same sense that a person should expect to be able to access extensive organizational resources from his office, he should be able to access extensive computational resources from an NLS-like facility.

For example, NLS provides all the capabilities needed for constructing, studying, and editing programs in any of several languages; however, the facilities for printing, compiling, archiving, etc. are not considered to be integral parts of NLS and must be activated as "external processors" by the NLS user.

Similarly, we have plans for implementing various message transmission and information-retrieval systems as external processors, so that all the formatting of input to these systems as well as the studying of the output from them can be done using the powerful NLS mechanisms, while the actual processing can be carried out by independent programs running as background jobs on our timesharing system -- or even by programs operating on computers at remote sites.

This office picture is very important to understanding how a research center such as ARC can make the most effective use of the resources of a network of interconnected computers such as the ARPA Network described in Section IV.

We anticipate that whenever we plan to make extensive use of the resources of a particular node of the Network, we will add facilities to NLS so that it can be used as an interactive interface with those resources.

For example, if we were planning to use an information-retrieval system at some other site, we would make provisions for the following:

(1) Permitting the retrieval requests to be formulated using regular NLS techniques

(2) Translating the request from our syntax to that required by the remote system and transmitting the reformatted request out over the Network

(3) Receiving the response from the remote site and, finally, translating this output into a properly formatted NLS file so that it can be viewed and manipulated using tools already familiar to the ARC staff.

This approach appeals to us because it promises to permit a member of our research community to use subsystems running at widely distributed computer facilities without learning a new set of conventions and user techniques for each different system. We believe that this way of viewing the role to be played by an organization's local computational facility will become more widely accepted as computer utilities and networks proliferate.

When organizations become able to access a powerful and varied collection of resources merely by interfacing their local facilities to a network, it will become more generally recognized how wasteful it is for each local facility to design and implement all the computational capabilities it needs.

It is likely that various computer "utilities" will evolve to serve the needs of large communities of users for specialized on-line services that cannot be provided economically at the local level.

By taking advantage of the load-leveling that results from serving a large number of customers, utilities could develop service facilities to fill the needs for large high-speed calculations, archival file storage, searches over large data bases, etc., with very good average response times and at relatively low cost.

Bringing the dispensing of specialized computation services into the "market place" provided by resource distribution networks could foster competition that would both accelerate development of this kind of service and make possible significant reductions in the cost of computational services in general.

In the context of this kind of development, individual research centers such as ours will be relieved of much of the chore of implementing and maintaining the basic service capabilities necessary for daily operations.

This chore, now duplicated from center to center, consumes an excessive portion of our collective resources. Effective computer networks should permit computer science researchers to reduce this duplication of effort so as to increase the rate of progress possible with available professional manpower and computational resources.

These research groups will then be able to focus more energy on the problems that they are particularly well-equipped to handle.

In our case this might involve basic research to find the most satisfying ways to interface a specific community of users to a network providing general capabilities, while other groups could apply their talents to areas of interest such as mathematical manipulation, computer graphics, and so on.

This consideration becomes particularly important when one realizes, as we are coming to realize to an ever greater extent, that the tools and techniques needed to constitute a "complete" augmentation system are far beyond the development capabilities of any one research center such as ours.

In coming years, we believe that significant developments in the computer sciences will come about more and more as a result of the cooperative efforts of many research centers, each working on particular aspects of augmentation.

The preceding comments should convey some feeling for the power that we feel is inherent in a network of research-oriented computer centers and provide a background for understanding our enthusiasm for participating in the ARPA Network. We believe that being a part of the ARPA Network will be valuable to us in achieving the goals of augmentation research for several reasons.

(1) It will give us experience in working with a community of researchers more varied and widely distributed than our own team of staff members.

This will permit us to gain additional insight into the validity of our approaches to augmentation research by observing how and to what extent our facilities can be of service to computer users who are not captive members of our local research community.

(2) Network participation will provide impetus for our efforts in the direction of team augmentation by creating a community of researchers who are working on different aspects of a common problem at widely distributed geographical locations and who need new tools and

techniques for communication and cooperation to succesfully achieve their common goals.

By working with such an extended community, we will be able to set goals and priorities in our research program more knowledgeably than we would be able to do if we were limited to augmenting only teams working in our own Center.

(3) As mentioned above, we at ARC have come to realize that attaining the goals of augmentation research on a reasonable time scale is a task far beyond the capabilities of any one small research center such as ours, and we are very optimistic about the possibility of achieving a significant acceleration in the progress of augmentation research through network participation with other research centers having similar goals.

D. Overview of Current Augmentation Research Center Plans

1. General

a. Introduction

The previous section should give the reader some feeling for the forces that have been shaping our plans for future research. Internal forces have combined with those generated by our Network participation to produce a shift in our research emphasis in the direction of two general activities:

(1) Research on team augmentation

(2) Development of a system design discipline.

In addition, increased awareness of our need to communicate and interact with the outside world is leading us into the development of a new area of specific concern, discussed below under "Transfer of Results."

b. Team Augmentation

Whereas in the past we gave most of our attention to augmenting the individual worker, we are now focusing on the augmentation of a team of collaborating workers, each of whom is individually augmented.

Section V
CONCLUSIONS, RECOMMENDATIONS, AND PLANS

The high mobility and manipulative capability of a skilled "augmented individual" has a unique potential which can be most fully realized when a number of augmented individuals join to form a collaborative team.

Not only can each individual move very rapidly through joint working files to study them, enter new information, and update old material, but the group processes of intercommunication and coordination can be facilitated by special computer aids, conventions, and techniques.

The contemplated efforts in "team augmentation" involve the development of several facets:

(1) Conventions and procedures for organizing the working records of our plans, designs, objectives, design principles, and schedules to give members of a team effective mutual "task orientation" by making all information related to the team's objective optimally accessible.

(2) A "Dialogue Support System" to facilitate rapid evolution of these working records through dialogue among members of the design team.

(3) Techniques to facilitate simultaneous collaboration among people at physically remote on-line terminals by giving them direct communication with one another, independent of their current individual work interactions with the computer. This includes provision, where feasible, for the following:

(a) Video and/or voice intercommunication

(b) Easy and flexible control of means for duplicating, at any terminal, all or part of the type-out or display from another terminal

(c) Ready transfer of control of one terminal's computer interaction to another terminal's input devices.

(4) Special management and system-building techniques for use by an augmented team, ~~and~~

provision of applicable technical intelligence and user training capabilities.

These techniques are expected to evolve within ARC under conditions of use in our own coordinated system-development work, and to be applied over a wide range of collaborative actions, from simple question-answering facilities to complex design work involving intense mutual participation by team members.

As applicable techniques become effective within ARC, we will explore their value for the following:

(1) Support of Network Information Center (NIC) services such as teaching, question-answering and some types of query servicing

(2) Working collaboration between ARC staff and personnel at other Network sites

(3) Working collaboration between people at remote Network sites, independent of ARC staff.

c. Development of User- and Service-System Design Disciplines

The functional features of the "user system" -- the collection of computer aids available to an ARC worker -- have evolved with some ingenuity, a great deal of cut-and-try experimentation in actual-usage conditions, and a certain special orientation offered by our overall research framework. Until now, however, there has been a significant lack of objective, methodical engineering design in the development of the overall user system.

A user-system design discipline is definitely needed, and we intend to devote an increasing amount of effort toward developing such a discipline.

Like the user system, the "service system" -- the hardware and software underlying the features for augmenting users -- has evolved in an ad hoc fashion.

Here there is also a significant need for a system-design discipline.

Such system-design disciplines would have communicable, teachable, and generally applicable frameworks, supporting coordinated sets of concepts, terminologies, principles, methodologies, and special tools.

d. Transfer of Results

Behind these basic aspects of our work in the ARC (team augmentation and design disciplines) lies an essential feature of our long-term strategy, namely, the goal of producing results that will be of direct value to other groups of system developers -- in particular, to those who will be developing augmentation systems.

This is in contrast with being of direct value to customers who want systems for their own direct use -- e.g., to augment a manager, a designer, an editor, or a scientist.

Display terminals, communication channels, and computer service are destined to become both cheap and plentiful, and it is certain that a very large number of organizations will want to use them. They must rely upon system developers who should be capable of the following:

(1) Analysis of system-usage environments

(2) Design and implementation of a smooth, powerful, and coordinated system of user aids, conventions, and methods

(3) Training and "education" of users unfamiliar with the potential of this new technology

(4) Subsequent monitoring of user performance so as to implement the changes necessary to track the evolution of users' attitudes, concepts, skills, usage habits, and wants.

Although it is important for us to stimulate the eventual customers for augmentation systems by making them aware of the potential for these systems in their work, we feel that our results should be directed primarily towards helping system developers. We plan to do this by pursuing the following long-term goals:

(1) Making visible an advanced, integrated system,

operating in a heavy-usage environment, that can orient system developers to the available cost/benefit tradeoffs

(2) Developing an effective system-design discipline to aid in developing augmentation systems, whether or not these systems resemble ours

(3) Maintaining thorough, highly current, comprehensive documentation, designed for quick location of relevant material

(4) Establishing wide-band communication channels over which a dynamic interchange of information can take place, so that the maximum amount of our knowledge can be quickly available in useful form

(5) Offering a complete prototype design of an augmentation system especially designed for augmenting system development.

This system would be compatible with the system-design disciplines described above, and would include techniques for planning, analyzing, designing, programming, debugging, documenting, and teaching.

Our current approach to implementing the transfer of results discussed above is to plan for what we call the System Developer Interface Activity (SYDIA). We expect to approach representative candidates during 1970 with proposals for multiple sponsorship. The initial purpose of the SYDIA will be to develop the following:

(1) A facility for an effective interchange of information and skills between ARC and the existing and potential community of augmentation-system developers

(2) The ability to assist other groups in transferring our system, or parts of it, directly into other hardware and user environments.

Later, with specific individual funding arrangements, we expect to begin developing close interchange relationships with various system-development groups who could adapt our augmented techniques to their own system-development work.

2. Team Augmentation Research Plans

a. Introduction

We have already discussed the evolution that has been taking place in the goals of our research program.

Having made significant progress towards realizing the objective of augmenting the individual intellectual worker, we find that the greatest augmentation need evidenced within our own Center at the present is for developing tools that will facilitate collaboration among members of a project-oriented or problem-solving team of augmented individuals.

It is not that we have already accomplished our original objectives and feel that we can now turn our attention elsewhere, but that team augmentation -- seen in the light of our bootstrapping strategy -- offers the greatest promise of hastening the eventual realization of these goals.

We view the "augmented team" as a group of workers sharing a common base of working files and using the mechanical elements of their augmentation system as both a medium for goal-related communications and a laboratory for carrying out relevant experiments.

At present we are most interested in exploring the possibilities for augmenting the activities of teams whose purpose is the development of advanced computer systems such as our own.

We feel that this is a profitable way of investing the resources with which we are entrusted, not only from the standpoint of our bootstrapping orientation, but also because augmenting this type of team now is most likely to have the greatest payoffs in the long run for society as a whole.

Over the past year we have identified a set of basic capabilities that seem to meet the major needs of the augmented system-development team.

The following description of these basic capabilities can be viewed as representing a framework for the system-development activity that must take place in the process of designing and implementing systems to realize these capabilities.

It also indicates, indirectly, the nature of other related activities that will be concerned with integrating these capabilities into our working methodology, applying them directly to the operation of our Center, and analyzing their effectiveness so as to provide direction to subsequent developments.

b. Fast Editing and Publication

Our already fast computer editing techniques will naturally continue to evolve, and we are in the early stages of developing a powerful "Output Processor" capability.

The Output Processor is envisioned as a coordinated set of techniques for producing hard copy through a variety of media, such as microform and direct publication on paper, using conventions that are compatible with those by which the associated file material can be studied and manipulated on line.

We plan to concentrate early upon automatic production, from our on-line files, of hard copy in which one can very flexibly specify the composition of text, diagrams, tables, equations, footnotes, indices, etc.

During the production process, such operations as converting intrafile links to page references and interfile links to footnotes would be performed automatically, with associated conversion tables being saved for future use.

One of our goals for the next few years is to develop a system coupling a typewriter-like computer terminal with a microform reader that can be positioned to any page within its "library" upon direction from the central computer.

This system would use conversion tables generated by the Output Processor during publication of the microform library to drive the reader in response to directions from the user. This would give the user much of the power for studying large bodies of interreferenced documents that currently can be obtained only through the use of a display-oriented system like NLS.

c. Plexdocs

A team tackling a complex system-development project must provide itself with the highest possible visibility over its working environment -- i.e., over the following:

Planning: plans, contingency alternatives, resource commitments, status, criticisms

Designing: designs, design principles, constraints, estimates, analyses, supportive data, relevant needs and possibilities

Operating: roles, task definitions, assignments, policies, operational procedures and conventions.

We currently have quite powerful techniques for aiding an individual or small report-writing team in producing documents of the usual research-report size and complexity. But in our approach to team augmentation, we consider it essential to expand upon these techniques so as to facilitate the development and production of very large, very complex documents containing many details that are highly cross-dependent.

We use the term "plexdoc" to denote the concept of such a complex document, which would, in fact, be composed of a possibly quite large collection of NLS files, cross-linked in complicated ways.

The stem "plex" comes from the Latin "plexus," which means woven or intertwined, and supports the image of a body of information that has been woven into a coherent fabric by a group of people through the use of special indices, footnotes, reader-contributed comments, specific cross-references, etc.

We intend to develop and keep up to date a large, detailed, highly cross-referenced and well-indexed "plexdoc" that contains a description of our own project-team activity fulfilling the needs listed above. Our techniques to facilitate its modification and republication will be under constant evolutionary pressure.

d. Dialogue Between On-Line Collaborators

(1) The Dialogue-Support System

On-line access by collaborators to each other's files, as provided by a number of today's timesharing systems, leaves much to be desired in supporting effective dialogue.

In this context, we use "dialogue" to refer to the incremental building up of a group expression of ideas on any given subject through the addition of "comments" to some set of relevant files.

We are attempting to meet the need for more flexible and powerful means of facilitating such dialogue through the development of a "dialogue-support system," which we consider an essential element of any team augmentation system.

The dialogue-support system must function smoothly in conjunction with the plexdoc conventions to provide capabilities such as are described below.

(2) Commenting

Any team member working at a display console must be able swiftly to access for study any portion of a plexdoc's structured files, and he must be able conveniently to add his contributions to the on-going dialogue that is contained in these files.

Whenever he wishes, he should be able to introduce comments that are freely sprinkled with explicit references to any specific item (character, word, statement, line, curve, box, expression, etc.) within any prior entry -- as though he were pencil-marking a paper draft with marginal comments, underlines, encircled passages, arrows, and the like.

When creating a comment entry, he needs flexible aids and methods for the following:

Arranging display of the various passages he is referencing relative to the content of the comment he is creating

Designating the explicit entities he wishes to reference

Having the current comment-creation state preserved temporarily while he checks on some related material.

This must be managed by the computer so that it does not matter if other people are concurrently scanning the same material or affixing comment references to the same items.

(3) Study

His study techniques should enable him flexibly to select which comments will be displayed for him and which ones will remain invisible (or whose presence will be made known to him without appearing in their entirety). For example, he may wish to be aware of only those comments that reference a given citation in a text, term in an equation, or label in a diagram.

He quite likely does not want to see reference indicators for all such prior comments, so he needs flexible mechanisms for specifying which are to be visible -- e.g., by author, creation time (relative or absolute), specific content, prior-assigned comment-set membership, author-affixed category designations, etc.

Also, whenever he sees indication that an interesting type of comment is associated with some item in the studied passage, he needs considerable flexibility for designating how he will be shown such selected comments relative to the referenced material -- for example, in one of the following ways:

With a split screen (original reference text on the left and comments on the right)

Comments enclosed within boxes and the boxes embedded within the original text

Ability to "flip" between views of the reference material and views of the comments, etc.

(4) Notification

Provisions need to be developed to enable setting up "annunciator calls" to various people, or sets of people, to request their special attention (at some level of priority) to a given comment. This might call for actions such as the following:

(1) An approval signoff to record the fact that the comment has been noted by the party or parties to which it was addressed

(2) Some kind of special vote, automatically tallied and recorded on the annunciator specification in that comment

(3) A need to observe a "point of order" in the special methodology the team has adopted -- e.g.:

"I protest this decision and call for a review, citing Policy X, relative to Budget Item Y and Design Principle Z."

(5) Retrieval

All dialogue entries immediately become part of the plexdoc containing the complete working records (and much of the history) of the augmented team. Since comments and other working record entities can refer to each other in indefinite extension, it will be possible to build up a very complex network of relationships among these comments and the substantive records about which the dialogue is swirling.

Although these relationships need never be ambiguous, it will be difficult for even a knowledgeable team member to keep track of them in such a way that he can effectively "navigate" through the plexdoc to follow all the relevant developments that may be taking place concurrently.

This is about the toughest central challenge in effectively augmenting a team -- that of developing computer aids, working methods, etc. to allow a skilled person to be highly effective in digesting the content and implications of such a record, and to develop a substantive next-stage design or plan that integrates the dialogue contributions.

Essentially similar techniques are required to augment any individual's central intellectual capability for synthesizing the next stage of development in plan or design -- and to the extent that we are successful with this, we should be able to offer strong guidance for capability augmentation over wide ranges of individual and team activities.

Our initial activities in response to this problem will be in the direction of providing powerful retrieval tools that will enable a user flexibly to specify, by content, which elements of the plexdoc are of greatest interest to him at any moment.

We also have plans for developing techniques that will permit a user to construct specific indices and catalogues over a given plexdoc and to create and manipulate arbitrary "sets" of entities that are of immediate interest.

Many of the tools developed to fulfill these needs will also receive extensive use in other ARC activities such as the Network Information Center.

e. Distributed Dialogue

We consider it important for people other than the central, highly trained, display-equipped team to participate as well as possible in dialogue of the type discussed above.

We seek to provide capabilities that function effectively over the widest possible range in sophistication of computer, communication, and terminal facilities and in level of experience and training of the individual users -- and with as much independence as possible of geographical location.

As a first step we intend to work on the problem of developing organization and formatting conventions for presenting plexdocs for study on devices other than our centrally located selective-view displays.

We assume that for significant participation via any type of coupling with a low information-transfer rate (such as a Teletype), the user will need to have on hand comprehensively indexed hard-copy reference material that is republished relatively often (e.g., weekly or even daily).

We are already working on mechanisms for producing high-quality hard copy in both paper and microform (as described above under "Fast Editing and Publication").

Our goal is to be able automatically to publish reference material (probably in microfiche) in such a way as to make feasible a frequent-republication operation servicing a moderate number of remote participants.

We have also begun to investigate many different kinds of remote printing devices and are particularly excited about the high-speed, high-quality, scan-driven hard-copy devices now appearing on the market.

We also are investigating techniques for allowing a remote affiliate, such as a participant at one of the other ARPA Network sites, to use a manual microform reader (or even a volume of paper printouts) in conjunction with a typewriter-like computer terminal through which we could provide computer aids for locating items of interest and following the various kinds of cross-reference links.

A next step (nearing operational status) is to provide facilities for direct, on-line, dialogue participation using TODAS (our Typewriter-Oriented Documentation-Aid System). We are giving this special emphasis so as to provide for early access to Network Information Center files.

We are actively pursuing an extremely promising possibility associated with an emerging line of microfiche readers that can be operated under direct computer control and which permit jumping to any frame of a fiche in a fraction of a second.

Most of these readers also allow jumping to any fiche within a cartridge, or even to any cartridge within a larger store, in times comparable to those we currently experience in studying files on line using our display system.

Such a reader, loaded with updated cartridges from us, where the reader and a typewriter terminal both connect through the Network to our computer, can provide a person with very powerful help in his plexdoc studying.

He would be able to follow links, jump to an index and from there to selected points, jump successively to the candidate selections produced by a retrieval query, indicate where he wants to direct a comment reference (via typewriter entry of his comment) -- all via quick directions on the typewriter, abbreviated by cues (which the computer knows about) that he sees on the screen of the microfiche reader.

In some applications a frame-jumping microfiche reader/typewriter terminal system could be quite competitive with our high-response, on-line display console system, and we are very interested in developing experimental versions of such a system for exploratory use in the Network Information Center.

f. Conference Dialogue

The team augmentation techniques we have discussed so far are all directed at aiding team members working at individual computer terminals.

There are times, however, when such a team will wish to convene as a whole to review some new proposal, debate a pressing issue, or collaborate actively in some particular phase of their work.

The "complete" team augmentation system must provide mechanisms for facilitating this kind of conference dialogue activity.

We already have experimented with using NLS as a sophisticated "blackboard" where one person can make a presentation to a group seated within viewing range of one of our regular NLS consoles. This gives new power for presenting material and answering questions as well as providing a very flexible medium within which the record of the discussion can evolve.

In cases where there are more viewers than can be comfortably accommodated around the "chairman's" console, our display transmission mechanisms make it trivial for us to hook up additional "slave" monitors at convenient locations.

At two of our major conference presentations (see Section III-E) we have used video projection equipment to allow us to give live demonstrations of our on-line system to large audiences, and we are planning to purchase similar equipment for use in conference augmentation experiments at our own Center.

The next step along this avenue of research is the development of techniques for allowing several users, each working at his own terminal, to collaborate in real-time work on a common set of working files.

We have experimented a little in allowing multi-console simultaneous access to a single file in which one user (the "chairman") has full control while the other users are restricted to merely positioning a personal bug-mark on the display, each using his own mouse.

With the evolution of multiple viewing windows, a flexible voice intercom system, and other techniques will come opportunities for more sophisticated forms of interaction; and we are optimistic about the possibility of achieving significant increases in team effectiveness through advances in real-time dialogue augmentation.

Further down the road, we see a real need for developing techniques that, without requiring an extensive training period, will extend the ability to participate effectively in augmented conferences to individuals with little experience in using the complex set of coordinated skills needed to competently operate our present on-line system. One possibility we envision is to give them a "chauffeur" to operate their on-line vehicle.

g. Voice Dialogue

We hope to begin experiments in the near future using techniques for the digitization of normal speech strings, developed by Glenn Culler while he was at the University of California, Santa Barbara. Our plans are to modify NLS so that a "statement" can contain not only the present text and/or graphic material but also a digital representation of a speech string.

Then, with only minor changes to NLS, we would be able to provide techniques for breaking long speech strings into shorter ones, hierarchically organizing them, and providing cross-reference links between voice strings and normal text.

These capabilities will permit us to integrate actual spoken dialogue into the dialogue mechanisms previously discussed, providing an extremely powerful addition to our repertory of team-augmentation techniques.

They would be of great help to remote dialogue participants -- or even to our own staff when away from the office -- since phoned-in comments could be integrated into an on-going dialogue record, and they open new doors in the realm of conference augmentation as well.

Moreover, the anticipated gradual evolution of speech-processing techniques will provide increasingly powerful benefits from this voice dialogue approach.

h. Technical Intelligence

To satify our own needs as a research team, we have been accumulating for many years a significant corpus of "intelligence" (bibliographic) data about activities and products of organizations outside our own that are involved in related work, and we have committed ourselves to shaping up this collection in order to provide at least parts of it (properly catalogued and indexed) to other groups, particularly NIC users.

In addition to maintaining an up-to-date collection of standard bibliographic items, we plan to expand into new areas that are specifically related to the needs of system-development teams. We intend to begin seeking out and collecting data such as the following:

(1) Characteristics of, and user experience with, various commercially available and custom-made system elements

(2) Reference material and user commentaries on externally developed systems and techniques

(3) Intelligence on the status and results of related work by other groups.

We have begun development of a flexible set of information-retrieval tools that will be used extensively in the maintenance and interrogation of our intelligence collection as well as in the management of our complex working records (as discussed above).

i. User Training

With any user-oriented system as complex and versatile as ours, there is a continuous problem in helping new users to attain competence in operating the system so as to obtain the maximum benefits from it. This problem is magnified many times when users can work from many widely separated sites, making it impossible for them to receive personal help from experienced staff on a minute-to-minute basis.

With a system evolving as rapidly as ours, it is difficult to keep even the central staff informed of all new system features and special methodologies.

We are making some progress in the preparation of training and reference materials for our user system, but there remains much to be done, not only in actually providing such materials, but in discovering what forms of indoctrination materials (films, video tape, introductory manuals, reference manuals, brief reference guides, etc.) are most useful.

j. Special Management Techniques

The management of a technically sophisticated problem-solving team requires the use of some methodological techniques that are common to the management of any organized group, as well as many others of a more specialized nature.

There needs to be an accepted methodology for managing the files containing the team's working records so as to ensure the most effective use of these files.

Effective procedures need to be worked out for developing plans for the future activities of the team, for negotiating and reviewing task designations and individual roles, and for allocating and accounting for the resources possessed by the team.

Finally, there must be well-understood and accepted ways of defining, representing, and monitoring operational procedures and of resolving conflicts between elements of the team regarding the allocation of resources among the activities of the team that must compete for them.

k. Special System-Building Techniques

In addition to the capabilities described above, which are relevant to the needs of all problem-oriented teams, there are some considerations that are uniquely applicable to a team whose domain is system development.

The system-development team needs to have a consistent, if not complete, set of principles for designing the overall software-architecture of interactive systems.

It must develop an understanding of the principles underlying the design and analysis of interactive systems from the standpoint of user services.

It must have effective techniques for training new users of the systems it has implemented and for generating useful reference materials for these systems.

It must have flexible methods for obtaining and analyzing performance data on a system.

It should have powerful ways for simulating significant parts of an existing system and for simulating newly conceived or modified systems in order to diagnose existing bugs, predict future performance under various conditions, and compare the performance of proposed system configuratons with that of the existing one.

Finally, it must have highly augmented techniques for creating, compiling, and debugging the huge collection of programs used in the implementation of a large software system.

3. Development of System Design Disciplines

a. Analysis and Design Principles for On-Line User Systems

Designing a whole augmentation system involves a balanced consideration of many factors, all of which are subject to reevaluation and change in response to increased understanding gained through experience. The following are examples of such factors:

(1) Ways in which users conceptualize their working tasks

(2) Methods for representing and recording these concepts

(3) Procedures for developing the concepts and products associated with a given task

(4) Forms of computer aids and utilization methodologies needed to obtain maximum help in carrying out such working procedures

(5) User techniques for negotiating service transactions with the system in various contexts.

At present there is no design discipline encompassing this range of interdependent system factors, but one really must evolve if any large-scale benefit is ever to be derived from the computer services that we clearly see coming over the horizon.

By a "design discipline" we mean a coherent, communicable, generally applicable framework comprising a coordinated set of concepts, terminologies, principles, methodologies, and special tools.

This design discipline must provide a common ground for analyzing the needs of a community of users, formulating easily communicable designs for systems to meet these needs, and providing adequate controls over the implementation process.

These capabilities should make it possible to offer clients accurate cost/benefit pictures prior to any extensive commitment of resources and should make the design and implementation processes more efficient.

Consider the predictable tremendous increases in speed, capacity, and economic availability of computational resources, and then realize how the quality of service represented by these resources can be enhanced through the application of ever more powerful artificial intelligence techniques.

We consider it desirable to avert the possibility of all this power being harnessed in ways that leave these mechanical helpers remote and uninvolved from our minute-by-minute human activity.

We need to learn how to design such systems and how to adapt our thinking and working methodologies so that quick little bits of service can be harnessed on our own terms.

A significant challenge is involved in finding ways for matching these computer services to human perceptual, cognitive, and motor mechanisms so as to optimize working capabilities and provide a natural and satisfying extension of human capacities.

Design of the repertoire of user services to be provided by a computer/communication/terminal facility is a process requiring the kind of coherent design discipline associated with any complex system design process, and we consider the development of such a discipline for augmentation-system design to be a vital component of our next phase of research.

b. Software Architecture Principles for Interactive System Design

(1) Overview

Just as important to us as the development of a discipline dealing with the analysis and design of user services is the development of a companion discipline dealing with the design of software architecture for interactive systems offering these services.

These two disciplines are actually just opposite faces of the same coin and must be interfaced so as to form a single overall discipline that provides a unified approach for going from user needs straight through to integrated operational systems satisfying those needs.

To organize the conceptualization of such a system design into appropriate levels and areas, and to develop effective representations of the design specifications and principles at each such level, seems necessary if we are ever to transform interactive system design from an art into a profession.

Our present software architecture approach stresses these things and promises to make useful headway in evolving conventions, procedures, and aids that could help other people approach and operate on the architecture of complex computer systems.

We describe below some of the approaches that have evolved in our work and which we feel offer the foundations for a design discipline of the type we are seeking.

(2) Compiler-Compiler Techniques

Almost all our programming is done in some member of a whole family of languages based on our Tree Meta syntax-directed compiler (see Section II-B-2-b and Ref. 17).

This technique provides us with great flexibility in fitting each programming task into a language particularly appropriate to the requirements of that task, yielding more efficient use of vital programmer time as well as code that is sufficiently self-documenting to aid considerably both in the debugging of our system and in the orientation of new programmers.

(3) Format Conventions for Source-Code Language Files

A very important feature of these special languages is that their format and structuring conventions have been designed so as to fit particularly well into our augmentation-system usage environment. This provides a very rewarding and powerful degree of facility for composing, studying, and modifying source code.

The plexdoc approach described previously has evolved from a belief that all levels of design and analysis thinking and data should be integrated into one compatible system of files over which the designers and supervisors, and later maintenance personnel, colleagues, etc., can roam freely and adroitly.

This approach has been applied to some extent in our present collection of system-program files. This collection forms a plexdoc by virtue of the fact that interfile links can be used as the arguments of procedure calls; the internal system documentation is also linked into this plexdoc.

Dialogue-support techniques have an important contribution to make at every level of this process in providing an integrated approach to recording, documentation, and monitoring.

(4) System Architecture Principles Fostering Evolutionary Development by an Augmented Team

We have begun to apply various modularization concepts to our working and thinking methodologies so as to foster the ability of individual members of our research/development team to work effectively on specific aspects of our project with the least possible chance of destructively interfering with other activities.

These concepts are also important to the way in which we view participation in the ARPA Network, in that we are seeking flexible ways of interfacing various modules of our own operating systems to computation components available at other sites through the Network.

We also seek to exploit the properties of modularization to promote design clarity and facilitate transfer of programs and techniques to other systems and groups.

The extensive attention to conceptual partitioning through use of a variety of special-purpose programming languages is one example of this.

Appendix A
USER FEATURES OF NLS AND TODAS

I The On-Line System (NLS)

A. Introduction

NLS, as currently implemented, is essentially a highly sophisticated text-manipulation system oriented primarily toward on-line use; i.e., it is not primarily oriented toward production of hard copy, although fairly sophisticated hard-copy formatting and output are included in the system.

NLS is intended to be used on a regular, more or less full-time basis in a time-sharing environment, by users who are not necessarily computer professionals. The users are, however, assumed to be "trained" as opposed to "naive." Thus the system is not designed for extreme simplicity, nor for self-explanatory features, nor for compatibility with "normal" working procedures.

Rather, it is assumed that the user has spent considerable time in learning the operation of the system, that he uses it for a major portion of his work, and that he consequently is willing to adapt his working procedures to exploit the possibilities of full-time, interactive computer assistance.

Thus the practices and techniques developed by users for exploiting NLS are as much a subject of research interest as the development of NLS itself.

NLS is supplemented by a typewriter-oriented counterpart called TODAS, which is discussed in Section II of this appendix. Section III describes the NLS/TODAS capabilities for producing flexibly formatted hard copy from on-line files.

Section IV of this appendix is a glossary of special NLS/TODAS terminology.

B. NLS Console

The user sits at a console whose main elements are a display screen, a typewriter keyboard, a cursor device called the "mouse," and a set of five keys operated by the left hand, called the "keyset."

The screen is used for displaying text, in various formats. The top portion of the screen (approximately

1/5 of the total area) is reserved for feedback information of various kinds: the name of the user command mode currently in effect, a "register" area used for various kinds of textual/numerical feedback, an "echo register" which displays the last six characters typed by the user, and other items that are explained below.

The keyboard closely resembles a conventional typewriter keyboard, with a few extra keys for special characters and control functions. It is used for typing text as content for a file and for specifying commands, which are given as two- or three-character mnemonics.

The mouse is a roughly box-shaped object, about four inches on its longest side, which is moved by the right hand. It is mounted on wheels and rolls on any flat surface. The wheels drive potentiometers that are read by an A/D converter, and the system causes a tracking spot ("bug") to move on the screen in correspondence to the motion of the mouse.

The user specifies locations in the displayed text by pointing with the mouse/bug combination. This eliminates the need for specifying a location by entering a code of some kind. Use of the mouse is very easily learned and soon becomes unconscious.

On top of the mouse are three special control buttons, whose uses are described below.

The keyset has one key for each finger of the left hand. The keys are struck in combinations called "chords," and each chord corresponds to a character or combination of characters from the keyboard. There are 31 possible chords; beyond this, two of the buttons on the mouse may be used to control the "case" of the keyset, giving alternative meanings to each chord. There are four possible cases, for a total of 124 possible combinations.

A simple binary code is used and has proved remarkably easy to learn. Two or three hours' practice is usually sufficient to learn the most commonly used chords and develop reasonable speed.

The keyset was developed to increase the user's speed and smoothness in operating NLS. It was found that users normally keep the right hand on the mouse, because the great majority of command operations involve a pointing action; efficient use of the keyboard, however, requires the use of both hands, and shifting the right hand (and the user's attention) to the keyboard is distracting and annoying if it must be done for each two- or three-letter command mnemonic.

Use of the keyset permits the user to keep his right hand on the mouse and his left on the keyset, reverting to the keyboard only for entry of long strings of text (typically five or more characters).

Originally, the keyset exactly duplicated the keyboard in function; in the development of NLS, however, certain control functions have been made two-stroke operations from the keyset where they would be three- or four-stroke operations from the keyboard. Nevertheless, it is still possible to operate all of the features of NLS without using the keyset; thus the beginner may defer learning the keyset code until he has gained some degree of mastery over the rest of the system.

C. Structured Text

"Text" is used here as a very general term. A "file" of text (corresponding very roughly to a "document" in hard copy) may consist of English or some other natural language, numerical data, computer-program statements, or anything else that can be expressed as a structure of character strings. Simple line drawings can also be included in a file.

All text handled by NLS is in "structured-statement" form. This special format is simply a hierarchical arrangement of "statements," resembling a conventional "outline" form.

Each statement in a file may be considered to possess a "statement number," which shows its position and level in the structure. Thus the first statement in a file is Statement 1; its first substatement is 1a, and its next substatement is 1b; the next statement at the same level as the first is Statement 2; and so forth. Statement numbers have been suppressed in printing out most of this document but are printed out for the remainder of this section as an example. .DSN=0;

Every statement also bears a "signature" that may be displayed on command. The signature is a line of text giving the initials of the user who created the statement (or modified it most recently) and the time and date when this was done.

A *statement* is simply a string of text, of any length; this serves as the basic unit in the construction of the hierarchy. In English text, statements are normally equivalent to paragraphs, section and subsection headings, or items in a list. In other types of text, statements may be data items, program statements, etc.

Each paragraph and heading in this document is an NLS statement. Each statement is indented according to its "level" in the hierarchy; this paragraph is a substatement of the one above, which is in turn a substatement of another statement. A statement may have any number of substatements, and the overall structure may have any number of levels.

Note that when a user creates a file, he may let all of his statements be first-level ones, i.e., 1, 2, 3, etc. In this case he will not have to consider a hierarchical structure but simply a linear list, as is found in conventional text.

However, many of the features of NLS are oriented to make use of hierarchy, and the benefits of these *features are lost if hierarchy is not exploited.*

This is an example of an NLS feature to which the user must accommodate his methods; however, the experience of users has been that hierarchical structure very rapidly becomes a completely "natural" way of organizing text. Many automatic features of NLS make the structure easy to use: for example, statement numbers are created automatically at all times and the user need not even be aware of them. It is sufficient, when the user creates a statement, to specify its level relative to the preceding statement. .DSN=1;

D. Use of the System

Text manipulation is considered to involve three basic types of activity by the user: composition, study, and modification. In practice, the three activities are so intermingled as to be indistinguishable.

1. Composition

Composition is simply the creation of new text material as content for a file.

In the simplest case, the user gives the command "Insert Statement" by typing "is". He then points (with the mouse) to an existing statement; the system displays a new statement number which is the logical successor, at the same level, as the statement pointed to. The user may change the level of this number upward by typing a "u" or downward by typing a "d".

NOTE: If no previous statement has been created, the system displays a "dummy" statement at the top of the text-display area, and the user points to this dummy in order to insert his first statement.

The user then types the text of the new statement from the keyboard. On the screen, the top part of the text-display area is cleared and characters are displayed here as they are typed. When the statement is finished, the user hits a CA (command accept) button on the keyboard or mouse, and the system re-creates the display with the new statement following the one that was pointed to.

New material may also be added to existing statements by means of commands such as Insert Word, Insert Text, and others. Properly speaking, these operations are modification rather than composition, and are discussed below.

Simple line drawings may be composed and added to the file by means of the "vector package." This is discussed in another section of this report.

2. Study

The study capabilities of NLS constitute its most powerful and unusual features. The following is only a brief, condensed description of the operations that are possible.

a. Jumping

NLS files may, of course, contain a great deal more text than can be displayed on the screen, just as a document may contain more than one page of text. An NLS file is thought of as a long "scroll." The process of moving from one point in the scroll to another, which corresponds to turning pages in hard copy, is called "jumping." There is a very large family of Jump commands.

The basic Jump command is Jump to Item. The user specifies it by entering "ji" and then points to some statement with the mouse. The selected statement is moved to the top of the screen, as if the scroll had been rolled forward.

Most of the Jump commands reference the hierarchical structure of the text. Thus Jump to Successor brings to the top of the display the next statement at the same level as the selected statement; Jump to Predecessor does the reverse; Jump to Up starts the display with the statement of which the selected statement is a substatement, and so forth.

The Jump to Name command uses a different way of addressing statements. If the first word of any statement is enclosed in parentheses, the system will recognize it as the "name" of the statement. Then, if this word appears somewhere else in the text, the user may jump to the named statement by pointing to the occurrence of the name, or by typing the name.

This provides a cross-referencing capability that is very smooth and flexible; the command Jump to Return will always restore the previous display, so that the user may follow name references without losing his place.

It is also possible to jump to a statement by typing its statement number.

b. View Control

If a file is long, it may be impossible for the user to orient himself to its content and structure or to find specific sections by jumping through it. The principal solution to this problem is provided by level control and line truncation.

Level control permits the user to specify some number of levels; the system will then display only statements of the specified level or higher. Thus if three levels are specified, only first-, second-, and third-level statements are displayed.

Line truncation permits specification of how many lines of each statement are to be displayed. Thus if one line is specified, only the first line of each statement will be displayed.

Common usage is to use the first two or three levels in a file as headings describing the material contained under each heading in the form of substatements. Thus the user may start by looking at a display showing only the first-level statements in the file, one line of each. This amounts to a table of contents.

He may then select one of these statements and jump to it, specifying one more level. He will then see more details of the content of that part of the file. This process of "expanding the view" may be repeated until the user has found what he is looking for, at which point he may specify a full display of the text.

Users soon develop a habit of structuring files in such a way that this process will work well. As it happens, such a structure is usually a good, logical arrangement of the material, reflecting the relationships inherent in the content.

The level and truncation controls are designed so that the necessary specifications may be made with only one or two strokes of the keyboard or keyset. These controls are only the most important of a large set of view-control parameters called "VIEWSPECS."

Other VIEWSPECS control a number of special NLS features affecting the display format.

An example of the use of VIEWSPECS is given below in Section I-D-2-e of this appendix.

c. Content Analysis

The NLS content analyzer permits automatic searching of a file for statements satisfying some content pattern specified by the user. The pattern is written in a special language as part of the file text.

Content patterns may be simple, specifying the occurrence of some word, for example. They may also be highly complex, specifying the order of occurrence of two or more strings, the absence of some text construct, conditional specifications, etc. Simple patterns are extremely easy to write; complex ones are correspondingly more difficult.

d. "Keyword" System

A "keyword statement" is a named statement that references other statements in the file by name, in a special format. The name of the keyword statement is then understood to be a "keyword" applying to the statements referenced by the keyword statement.

Suppose that a file contains a list of keyword statements. The user may study this list and select several keywords with the Keyword Select command (pointing to the keywords with the mouse).

He may specify a weight from 1 to 10 for each keyword; if no weight is specified, a weight of 1 is assumed.

When the user gives the Keyword Execute command, a searching/scoring process is executed. Each of the selected keyword statements is scanned for the names of statements that it references. Each referenced statement receives a "score" equal to the weight of the keyword. If a statement is referenced in more than one keyword statement, the scores add.

When this process is completed, NLS constructs a display picture showing only the statements that have received nonzero scores, in order of decreasing scores.

In other words, each keyword is the name of a statement that defines some arbitrary category of statements in the file. When a user selects and weights keywords, he is expressing his interest in certain of these categories. NLS then displays all of the statements in these categories, beginning with the "most interesting."

Because the relationships used in this system are set up explicitly when a user writes keyword statements, the system is very flexible although not highly automated. It may be regarded as a generalized method of reordering some of the statements in a file on the basis of user-selected criteria chosen from a supplied list (the keyword statements).

Note that this reordering is on the display, not in the file proper. The file proper is not affected in any way, except that the list of selected keywords and weights is saved in the file.

This list may be displayed on command. Individual keywords may be deleted from the list or their weights changed, or the whole list can be deleted on command.

e. Link Jumping

A "link" is a string of text, occurring in an ordinary file statement, that indicates a cross-reference of some kind. It may refer to another statement in the file, or to a statement in some other file, possibly belonging to another NLS user. The text of the link is both human-readable and machine-readable, and the command Jump to Link permits the user to point to the link with the mouse and immediately see the material referred to.

An example of a link is (Smith, Plans, Longrange:ebgtn).

The first item in the link indicates that the referenced file belongs to a user named Smith; the second is the name of the file; the third is the name of a statement in the file (a statement number may also be used); and the string of characters following the colon controls the VIEWSPECS to set up a particular view of the material.

The Jump to Link command, executed by pointing to this link, would cause Smith's file "Plans" to be automatically loaded and the display start set to the statement whose name is "Longrange." VIEWSPECS would be automatically set as follows:

The "e" causes the number of levels displayed to be set equal to the level of the display-start statement.

The "b" increments this by one.

The "g" specifies that the display is to be limited to the "branch" defined by the display-start statement -- i.e., only that statement, its substatements, their substatements, etc.

The "t" specifies display of only the first line of each displayed statement.

The "n" specifies that statement numbers are to be suppressed from the display.

Thus the net result is to display the first lines of statement "Longrange" and its highest-level substatements, without statement numbers.

The use of interfile links permits the construction of large linked structures made up of many files, and study of these files as if they were all sections of a single document. The Jump to File Return causes the file previously viewed to be reloaded and displayed, with the same display start and VIEWSPECS that were in effect just before the link jump; thus the user may execute link jumps freely without losing track of his original location.

3. Modification

A large repertoire of editing commands is provided for modification of files. The basic functions are Insert, Delete, Move, and Copy.

These functions operate upon various kinds of text entities. Within statements, they may operate upon single characters, words, and arbitrary strings of text defined by pointing to the first and last characters.

This set of commands is not restricted to operation within one statement at a time; for example, a word may be moved or copied from one statement to another.

The editing functions also operate at the structural level, taking statements or sets of statements as operands. A number of special entities have been defined for this purpose: for example, a "branch" consists of some specified statement, plus all of its substatements, plus all of their substatements, etc. A branch can be deleted, moved to a new position in the structure, etc.

As noted above, the modification activity tends to merge, in practice, with study and composition.

E. Summary

It must be noted that NLS is not a system designed foı general usage, but a specialized tool designed for a group of people working on the development of computer aids to human intellectual processes. It is for this reason, for example, that NLS is not really a text-editing system oriented toward hard-copy production, but rather something simultaneously more general and more specialized.

It is in the process of manipulating a file -- studying it, making modifications, adding new material as an integrated process lasting for minutes or hours at a time and having a continuity extending for days, weeks, or even years -- that the real benefit of NLS appears.

An NLS file tends to become an evolving entity, subject to constant modification, updating, and reevaluation. Its development may have no clearly defined endpoint. It may cease to exist as a file by being incorporated in another file, or it may eventually be abandoned or superseded; however, in most cases it will never be "finished" in the usual sense of the word.

Continuous use of NLS to store ideas, study them, relate them structurally, and cross-reference them results in a superior organization of ideas and a greater ability to manipulate them further for special purposes, as the need arises -- whether the "ideas" are expressed as natural language, as data, as programming, or as graphic information.

II The Typewriter-Oriented Documentation-Aid System (TODAS)

TODAS is a text-handling system designed as a "typewriter" counterpart to NLS. In principle, TODAS can be operated from a Teletype or any other sort of hard-copy terminal, including terminals linked to the 940 through acoustic couplers and ordinary telephone lines (as opposed to NLS, which requires special transmission arrangements).

The present implementation allows for the use of Teletype Models 33, 35, and 37, Terminet and Execuport terminals (the latter being portable, with a built-in acoustic coupler), and NLS display consoles.

Each of these terminals has its own character set, no two sets being exactly the same except Teletype Models 33 and 35. As a result, special-character assignments are device-dependent. A TODAS feature allows the user to redefine characters at will to suit his immediate purposes.

An important use of TODAS is for access, within the ARPA Computer Network, to the Network Information Center (NIC) operated by ARC. TODAS will give Network users access to files of information created either with TODAS or with NLS, since files created with the two systems are identical in structure and format.

TODAS has many of the same capabilities as NLS for the manipulation of text; it differs from NLS as required by the use of a "typewriter" device instead of a display. The important differences arise from the fact that TODAS has no analog cursor device to correspond to the NLS mouse and from the generally slower operation of TODAS.

For this reason, editing of text within a statement cannot be done by means resembling those of NLS, since all of the NLS editing operands are indicated by the user with the mouse. TODAS uses two alternative methods.

One is the TODAS "alter" command, which operates very much like the "modify" command of the QED line-editing system developed by Project GENIE at UC. "Alter" creates a new statement to replace the original one, by going through the original from beginning to end; under user control, characters are (1) copied from the old statement to the new, (2) skipped over, or (3) inserted into the new statement from the keyboard.

The other is the TODAS "substitute" command, which allows the user to specify that a certain string of characters in the statement is to be found by TODAS and replaced with another specified string.

At the structural level (where the user wishes to manipulate statements and sets of statements as units), NLS permits the user to identify statements by pointing with the mouse; TODAS requires that statements be identified from the keyboard. Considerable flexibility is provided in this operation.

The user may identify a statement directly by typing its statement number or its name; he may also identify it indirectly by specifying its structural relationship to some other statement whose number or name he knows off-hand.

Indirect specification corresponds to the use of NLS commands such as "jump to head," "jump to successor," etc., but with the added feature that relationships may be concatenated -- thus the user may, in a single operation, specify a complex relationship such as the successor of the first substatement of the predecessor of a given statement.

A special TODAS capability (not yet implemented in NLS) is "executable text."

A TODAS statement may consist of the string of characters that a user would type from the keyboard to perform some sequence of operations. This statement may then be executed with a special command, and the result will be exactly as if the user had actually typed these characters, causing the sequence to be carried out.

The sequence may, in principle, be arbitrarily complex; an executable statement might, for example, contain the following sequence:

(1) Load a file whose name is specified elsewhere in the current file

(2) Search this file with the content analyzer, finding statements with a specified pattern of content

(3) Write these statements out in a temporary "buffer" file

(4) Reload the original file

(5) Copy the statements in the "buffer" file into a specified location in the working file.

A special "switch" character may be used in the executable text. When the switch character is encountered, execution of the text is interrupted and control reverts to the keyboard. The user then enters part of the control sequence manually; when he types the switch character from the keyboard, execution of the executable statement resumes at the point where it left off. This feature affords great flexibility, since it allows part of the sequence to be specified ahead of time and part at "execution time."

Besides its primary purpose as a Network user's interface to the NIC, TODAS is used within ARC as a supplemental tool to NLS.

TODAS can be used conveniently for many tasks that do not require the rapid display response of NLS, and has the advantage of creating significantly less load on the overall timesharing system. We currently have one clerical worker, who is not an NLS user, operating TODAS routinely for entry of information and for some limited retrieval work.

Additionally, we find TODAS useful for remote accessing of our system. We have made TODAS available to selected consultants, who use home terminals with acoustic couplers, and regular ARC personnel occasionally do work from their homes by the same means.

The prototype version of TODAS went into service in September 1969; a second version, with greatly expanded capabilities, became operational early in 1970.

III Output Facility

NLS and TODAS both use the same facilities for producing formatted hard-copy output from NLS/TODAS files.

The devices in ordinary use at ARC for hard-copy output are a line printer that produces upper/lower-case print of adequate quality for local use, and a paper-tape-driven automatic typewriter used for final output of reproducible copy for reports, proposals, etc.

The Output Processor (previously known as "PASS4") can be controlled by the user to a considerable extent. This is done by means of "directives" embedded in the file text. The directives can be used to reset page parameters, control page numbering, and turn various format features "on" or "off."

For example, directives can be used to suppress indentation of statements or change the amount of indentation, to create "running headers" that are automatically printed at the top of each page, suppress statement numbers, etc. One of the directives causes all directives to be suppressed from the output.

In addition to the line printer and the automatic typewriter, the Output Processor can output a file to magnetic tape, appropriately formatted to drive CRT-to-film conversion equipment for production of microfilm.

In all cases, the user may elect to output an entire file or only part of the file. In the latter case, he may cause output to begin at some specified point in the file instead of at the beginning, and he may cause the printout to be limited by the same kinds of criteria that may be used on the display -- i.e., content analysis, limited number of structural levels, etc.

IV Glossary of Special NLS/TODAS Terminology

BRANCH: A specified statement, plus all of its substructure -- i.e. all of its substatements, plus all of their substatements, etc.

BRANCH ONLY: A VIEWSPEC parameter that restricts the text displayed (NLS) or typed (TODAS) to a single branch (see VIEWSPECS).

BUG: The cursor mark on the screen that is moved by the mouse and that is used for selecting (pointing to) entities on the display.

When the bug is "active," i.e., when a selection can be made, it appears as an up-arrow; when it is inactive it appears as a plus sign.

CHARACTER: Any letter, digit, punctuation mark, space, tab, or carriage return; an indivisible entity.

CHORD: A combination of keys on the keyset (see KEYSET).

CLIPPING: See LEVEL CLIPPING.

END: The last statement in any branch; specified by specifying the branch.

FILE: A complete tree structure of statements with a single root (the origin statement).

FILENAME: The name of a file. It appears as the first word in the origin statement of an existing file, and must be supplied by the user in creating a new file.

GAP CHARACTER: Any space, tab, or carriage return.

GCHAR: Abbreviation for GAP CHARACTER.

GROUP: A subset of a plex, consisting of all branches from one specified branch to another, inclusive.

HEAD: The first statement in a sublist.

The head is specified by pointing to any statement in the sublist.

INVISIBLE: Any consecutive string of gap characters, bounded by (but not including) printing characters or the end of a statement: see PRINTING CHARACTER, GAP CHARACTER, STATEMENT.

Specified by pointing to any character in the string. If a single printing character lying between two invisibles is pointed to, both invisibles (and the printing character) are selected.

KEYSET: The device at the left-hand side of the NLS console. When a combination of keys (a chord) is depressed on the keyset, the effect is the same as striking a key on the keyboard.

KEYWORD: The name of a "keyword statement."

KEYWORD STATEMENT: A statement that lists, in a special format, the names of all statements in the same file that fall into some arbitrary category.

The "keyword system" of NLS/TODAS commands, operating upon keyword statements, performs information-retrieval operations based on the sets of statements defined in keyword statements.

LABEL: A string of text placed in a picture by means of a command in the vector package.

LEVADJ: The specification of level when a statement, branch, plex, or group is newly created or moved.

LEVEL: The "rank" of a statement (see STATEMENT) in the hierarchy of the file (see FILE).

The level is equal to the number of fields of letters or digits in the statement number; thus Statement 3 is a first-level statement, Statement 4a10g3 is a fifth-level statement, etc. Level is of great importance in understanding the hierarchical structure of an NLS file.

LEVEL CLIPPING: Restricting the text displayed (NLS) or printed (TODAS) to statements no deeper than a specified level, which is set by a VIEWSPEC parameter (see VIEWSPECS).

LINE TRUNCATION: Restricting the amount of text displayed (NLS) or printed (TODAS) to the first N lines of each statement, where N is specified by setting a VIEWSPEC parameter (see VIEWSPECS).

Each statement is formatted into lines upon output to the display or printing device; thus the amount of text in a line depends upon the device and upon other parameters.

LINK: A specially formatted string of text in a statement, analogous to a reference or cross-reference citation in a conventional document.

A link specifies a user, a file belonging to that user, a location in the file, and VIEWSPECS. A user may "follow" a link by means of the Jump to Link command (NLS) or an indirect address (TODAS). In either case, the system detects the link, interprets it, loads the specified file, and displays or prints the specified portion of it with the specified VIEWSPEC parameters.

MOUSE: The device at the right-hand side of the NLS keyboard. When it is rolled around on the tabletop, it causes the bug (cursor mark) to move correspondingly on the screen.

NAME: If the first word of a statement is enclosed in parentheses, it is the NAME of the statement.

The command Jump to Name can then be used to place the statement at the top of the display. This is done by entering the name from the keyboard or keyset, or by finding an occurrence of the name as text on the display and pointing to it with the bug.

NLS: Acronym for "On-Line System."

ORIGIN: The first statement in a file; it contains information about the file, plus any other text the user inserts. It has a level of 0, and hence no statement number.

OUTPUT PROCESSOR: Program used by NLS and TODAS for producing formatted output.

PASS4: Alternate name used for the Output Processor.

PATTERN: A string of special-language text in a statement that may be compiled via the command Execute Content Analyzer. When compiled, it produces a program that is used by the content-analyzer feature.

PCHAR: Abbreviation for PRINTING CHARACTER.

PLEX: Another name for a SUBSTRUCTURE, used in command specifications.

A plex is specified by pointing to any one of its highest-level statements.

POINTER: A string of up to three characters that is attached to some character in the text with the Pointer Fix command.

PREDECESSOR: The statement preceding a specified statement in a SUBLIST.

PRINTING CHARACTER: Any letter, digit, or punctuation mark.

SOURCE: The statement of which a specified statement is a substatement.

SIGNATURE: Information stored with a statement (and displayed on command) giving the initials of the user who created the statement (or most recently modified it) and the time and date when this occurred.

STATEMENT: The basic structural unit of a file of text in NLS. Formally, it is a string of text and/or pictures that is bounded at the beginning by the end of the previous statement or the beginning of the file, and bounded at the end by the beginning of another statement or the end of the file.

Statements are arranged in a tree structure or hierarchy and are assigned "statement numbers" indicating their positions in the structure. Each statement has a number, made up of alternating fields of digits and letters; the number of fields indicates the "level" of the statement (see LEVEL).

A statement is specified by pointing to any character in the string.

SUBLIST: The set of all substatements of a specified statement (not including the substatements of the substatements).

SUBSTATEMENT: A statement "X" is called a substatement of another statement "Y" if it is deeper in the structure than "Y," if it follows "Y," and if there is no intervening higher-order statement. "Y" is called the source of "X." The statement number of "X" will be the same as that of "Y" except that it will have one more field at the end. The value of this field gives its ordinal position in a "sublist" of the substatements of "Y."

A substatement is specified by pointing to the source statement.

SUBSTRUCTURE: The set of all substatements of a specified statement, plus all their substatements, etc. until no more are found. The set of all branches defined by statements in the sublist of a given statement.

SUCCESSOR: The statement following a specified statement in a sublist.

TAIL: The last statement in a sublist.

The tail is specified by pointing to any statement in the sublist.

TEXT: Any string of characters within a statement, bounded by (and including) two specified characters: see CHARACTER, STATEMENT.

TODAS: Acronym for Typewriter-Oriented Documentation-Aid System.

TRUNCATION: See LINE TRUNCATION.

VECTOR: A line in a picture.

VIEWSPECS: View-control parameters controlling a number of special NLS features affecting the display format. See, for example, BRANCH ONLY, LEVEL CLIPPING, and LINE TRUNCATION.

VISIBLE: Any consecutive string of printing characters, bounded by (but not including) gap characters or the end of a statement: see PRINTING CHARACTER, GAP CHARACTER, STATEMENT.

Specified by pointing to any character in the string. If a single gap character between two visibles is pointed to, then both visibles (and the gap character) are specified.

WORD: Any consecutive string of letters and/or digits, bounded by (but not including) any other types of characters or the end of a statement: see STATEMENT.

Specified by pointing to any character in the string. If a single character is pointed to that is not a letter or digit and lies between two words, then both words (and the single character) are specified.

BIBLIOGRAPHY

The following is a chronological list of documents published by the Augmentation Research Center.

1. D. C. Engelbart, "Special Considerations of the Individual As a User, Generator, and Retriever of Information," Paper presented at Annual Meeting of American Documentation Institute, Berkeley, California (23-27 October 1960).

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

3. D. C. Engelbart, "A Conceptual Framework for the Augmentation of Man's Intellect," in Vistas in Information Handling, Volume 1, D. W. Howerton and D. C. Weeks, eds., Spartan Books, Washington, D.C. (1963).

4. D. C. Engelbart, "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), AD 640 989.

5. D. C. Engelbart and B. Huddart, "Research on Computer-Augmented Information Management," Technical Report ESD-TDR-65-168, Contract AF 19(628)-4088, Stanford Research Institute, Menlo Park, California (March 1965), AD 622 520.

6. W. K. English, D. C. Engelbart, and B. Huddart, "Computer-Aided Display Control," Final Report, Contract NAS1-3988, SRI Project 5061, Stanford Research Institute, Menlo Park, California (July 1965), CFSTI Order No. N66-30204.*

7. W. K. English, D. C. Engelbart, and M. L. Berman, "Display-Selection Techniques for Text Manipulation," IEEE Trans. on Human Factors in Electronics, Vol. HFE-8, No. 1, pp. 5-15 (March 1967).

8. D. C. Engelbart, W. K. English, and J. F. Rulifson, "Study For The Development of Human Intellect Augmentation Techniques," Interim Progress Report, Contract NAS1-5904, SRI Project 5890, Stanford Research Institute, Menlo Park, California (March 1967).

9. J. D. Hopper and L. P. Deutsch, "COPE: An Assembler and On-Line-CRT Debugging System for the CDC 3100," Technical Report 1, Contract NAS 1-5904, SRI Project 5890, Stanford Research Institute, Menlo Park, California (March 1968).

10. R. E. Hay and J. F. Rulifson, "MOL940: A Machine-Oriented ALGOL-Like Language for the SDS 940," Technical Report 2, Contract NAS 1-5904, SRI Project 5890, Stanford Research Institute, Menlo Park, California (April 1968).

11. D. C. Engelbart, W. K. English, and J. F. Rulifson, "Development of a Multidisplay, Time-Shared Computer Facility and Computer-Augmented Management-System Research," Final Report, Contract AF 30(602)4103, SRI Project 5919, Stanford Research Institute, Menlo Park, California (April 1968), AD 843 577.

12. D. C. Engelbart, "Human Intellect Augmentation Techniques," Final Report, Contract NAS 1-5904, SRI Project 5890, Stanford Research Institute, Menlo Park, California (July 1968), CFSTI Order No. N69-16140.*

13. D. C. Engelbart, W. K. English, and D. A. Evans, "Study for the Development of Computer-Augmented Management Techniques," Quarterly Progress Report 1, Contract F30602-68-C-0286, SRI Project 7101, Stanford Research Institute, Menlo Park, California (October 1968).

14. D. C. Engelbart and W. K. English, "A Research Center for Augmenting Human Intellect," in AFIPS Proceedings, Vol. 33, Part One, 1968 Fall Joint Computer Conference, pp. 395-410 (Thompson Book Co., Washington, D.C., 1968).

15. D. C. Engelbart and Staff of the Augmented Human Intellect Research Center, "Study for the Development of Human Intellect Augmentation Techniques," Semiannual Technical Letter Report 1, Contract NAS 1-7897, SRI Project 7079, Stanford Research Institute, Menlo Park, California (February 1969).

16. D. C. Engelbart and Staff of the Augmented Human Intellect Research Center, "Study for the Development of Human Intellect Augmentation Techniques," Semiannual Technical Letter Report 2, Contract NAS 1-7897, SRI Project 7079, Stanford Research Institute, Menlo Park, California (August 1969).

17. D. C. Engelbart and Staff of the Augmented Human Intellect Research Center, "Computer-Augmented Management-System Research and Development of Augmentation Facility," Final Report RADC-TR-70-82, Contract F30602-68-C-0286, SRI Project 7101, Stanford Research Institute, Menlo Park, California (April 1970).

BIBLIOGRAPHY

*Note: Reports with AD numbers are available from Defense Documentation Center, Building 5, Cameron Station, Alexandria, Virginia 22314. Items marked with an asterisk may be obtained from CFSTI, Sills Building, 5825 Port Royal Road, Springfield, Virginia 22151; cost $3.00 per copy or 65 cents for microfilm.

The following is a list of other documents cited in this report. The items are listed in the order in which they are cited.

18. "Specifications for the Interconnection of a Host and an IMP," Report No. 1822, Contract No. DAHC15-69-C-0179, ARPA Order No. 1260, Bolt Beranek and Newman Inc., Cambridge, Massachusetts (May 1969).

19. L. Roberts, "Computer Network Development to Achieve Resource Sharing," paper presented at 1970 Spring Joint Computer Conference, Atlantic City, New Jersey (May 1970); AFIPS Conference Proceedings, Vol. 36, p. 543 (AFIPS Press, Montvale, New Jersey, 1970).

20. F. Heart et al., "The Interface Message Processor for the ARPA Computer Network," paper presented at 1970 Spring Joint Computer Conference, Atlantic City, New Jersey (May 1970); AFIPS Conference Proceedings, Vol. 36, p. 551 (AFIPS Press, Montvale, New Jersey, 1970).

21. L. Kleinrock, "Analytic and Simulation Methods in Computer Network Design," paper presented at 1970 Spring Joint Computer Conference, Atlantic City, New Jersey (May 1970); AFIPS Conference Proceedings, Vol. 36, p. 569 (AFIPS Press, Montvale, New Jersey, 1970).

22. H. Frank, I. Frisch, and W. Chou, "Topological Considerations in the Design of the ARPA Computer Network," paper presented at 1970 Spring Joint Computer Conference, Atlantic City, New Jersey (May 1970); AFIPS Conference Proceedings, Vol. 36, p. 581 (AFIPS Press, Montvale, New Jersey, 1970).

23. S. Carr, S. Crocker, and V. Cerf, "HOST-HOST Communication Protocol in the ARPA Network," paper presented at 1970 Spring Joint Computer Conference, Atlantic City, New Jersey (May 1970); AFIPS Conference Proceedings, Vol. 36, p. 589 (AFIPS Press, Montvale, New Jersey, 1970).