STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U.S.A.

18 August 1969

Semiannual Technical Letter Report 2
Covering the Period 9 February 1969 through 8 August 1969
Stanford Research Institute Project 7079

# STUDY FOR THE DEVELOPMENT OF HUMAN INTELLECT AUGMENTATION TECHNIQUES

by

D. C. Engelbart

and Staff of the Augmented Human
Intellect Research Center

Contract NAS1-7897

Prepared for

National Aeronautics and Space Administration
Langley Research Center, Langley Station
Langley, Virginia 23365, Mail stop 126

Copy No. 53

## ABSTRACT

This report covers progress made during six months of a continuing research program in the Augmented Human Intellect Research Center (AHIRC) of Stanford Research Institute.

The program is directed toward the discovery of principles and techniques for the augmentation of human effectiveness in intellectual tasks by means of direct, on-line computer aids used on a full-time basis.

A considerable portion of the report is devoted to developments in the "On-Line System" (NLS), which is a unified system incorporating many of the computer aids that have been developed in the project.

Also described are a new text-manipulation system for use at typewriter terminals, a CRT-to-film document output system capable of handling mixed text and graphics, and the proposed organization of an information center for the ARPA computer network.

CONTENTS

# FOREWORD

The Augmented Human Intellect Research Center (AHIRC) operates under multiple sponsorship by NASA, ARPA, and the U. S. Air Force. Although this report applies specifically to NASA Contract NAS1-7879, not all of the work reported was funded exclusively by NASA. The Center's multiple funding is used in a highly integrated, flexible manner. For this reason, explicit separation of funding areas for a report such as this one would be difficult and would result in an unnatural division of material from a technical-information point of view.

AHIRC is a research center operating within Stanford Research Institute. It is devoted to research on techniques and principles for the augmentation of human intellectual processes by means of direct computer aid to intellectual (information-handling) tasks.

This report was composed, organized, and output for printing entirely by means of some of the computer aids developed by AHIRC.

# I GENERAL

## A. New Version of On-Line System ( NLS)

A new version of NLS has been implemented and is now in the final stages of debugging. This version of NLS, which contains a number of new features and is designed with further major improvements in mind, is described in Section II of this report.

## B. Typewriter-Oriented Documentation-Aid System (TODAS)

In order to provide access to AHIRC's computer-held files from terminals without CRT displays, an initial version of the Typewriter-Oriented Documentation-Aid System (TODAS) has been implemented. This system is described in Section III of this report.

## C. Graphics-Oriented Documentation-Output System (GODOS)

A system is now operational for output from NLS files via a CRT to either microfilm (roll or fiche format) or full-size paper copies. This system, which can handle mixed text and graphics, is described in Section IV of this report.

## D. New Physical Environment

Near the beginning of this report period, the AHIRC moved to another location in the same building. The new location occupies one end of a wing, and the offices are grouped around a large clear central area where the NLS consoles are located.

Besides affording a considerable expansion in office space to accommodate the staff expansion of AHIRC, the new location permits flexible rearrangement of the central area for experimentation with the effect of physical environment on group interaction in the use of NLS. This is an important component of our Management Systems Research activity, funded by RADC.

# II NEW VERSION OF NLS

## A. Introduction

A new version of NLS has been implemented and is just going into service at the time of this report. A number of bugs in the old version have been removed, and a number of new features have been added.

The text display now uses actual upper- and lower-case letters. The old system had only upper-case characters; these were used to represent lower case, and upper case was indicated by overbars.

## B. New User Features

### 1. File Cleanup

The old NLS incorporated a "File Cleanup" feature, used to detect and correct file errors caused by hardware malfunctions and software bugs.

In place of the old "File Cleanup" feature, the user now has two primary options and two secondary options.

The first primary option, called "File Cleanup," is essentially the same as the old "File Cleanup." Having specified this, the user may then specify "to Error" or "to Completion."

In a "File Cleanup to Error," the file is checked thoroughly for all sorts of structural errors, bad characters, etc. As soon as any one error is discovered, the process terminates and a message is displayed. The user will then normally reload the file and do a "File Cleanup to Completion."

A "File Cleanup to Completion" begins in the same fashion, but when errors are found, an attempt is made to repair them. This process may take several minutes to run through a large file, and imposes a considerable load on the system while it runs. This makes it inappropriate for routine use, especially since errors are actually quite rare.

The second primary option, called "File Checksum," is a feature that runs much more quickly and simply does checksums on the file. No attempt is made to repair errors. Again, the two secondary options are "to Error" and "to Completion."

"File Checksum to Error" terminates as soon as any one

error is found; "File Checksum to Completion" continues through the entire file listing errors.

This process detects almost all of the things that can go wrong with a file; if an error is found, the user may then use "Cleanup to Completion" to attempt to repair the error.

2. Name Delimiters

In the old NLS, a "name" was required to be delimited by parentheses. In the new version, the user may change the delimiters, using any characters he chooses.

3. NLS Calculator

The NLS Calculator Package has been implmented in the new NLS. This package, which has capabilities comparable to a good solid-state desk calculator but may also use information from a file as a "program," is described fully in Appendix A.

## C. New Implementation Features

Apart from the changes that appear as user features, a number of other changes have been made in the implementation of NLS.

The new system runs faster than the old, and it runs more smoothly because of extensive debugging.

There has been massive internal reorganization of NLS's code structure, both for greater efficiency and to allow for future implementation of features that are currently being designed or implemented but that are not yet ready for integration into NLS.

Finally, some of the new changes are related to NLS-TODAS compatibility (see Section III of this report).

# III TYPEWRITER-ORIENTED DOCUMENTATION AID SYSTEM (TODAS)

## A. Introduction

The initial version of the Typewriter-Oriented Documentation Aid System (TODAS) is now operational. In this implementation, TODAS is a very close counterpart of NLS, and can be used from any Teletype (or similar) terminal connected to the 940. Thus TODAS will be suitable for ARPA Network access to the AHIRC system.

## B. Purpose

Until the introduction of TODAS, NLS was the only means of accessing the AHIRC's collection of computer-held files. This posed a severe limitation upon users, because the total number of NLS consoles is limited and because system response becomes quite slow when a number of users are running NLS.

System improvements described elsewhere in this report will increase the number of consoles and improve system responsiveness; however, the limits will continue to be severe. In particular, the limits on NLS usage become prohibitive in the context of access to the Network Information Center by many remote ARPA Network users.

In terms of users within the AHIRC, it is recognized that much of the usage of NLS does not really require the great responsiveness and flexibility of the display system.

Thus there is a definite need for a text-manipulation system that does not depend on the use of a display console and does not load the computer as much as NLS, but that still permits all of the essential text composition/study/modification operations of NLS and is fully compatible with NLS (i.e., uses exactly the same file structure).

TODAS is a response to this need. It is designed for NLS compatibility at all levels, including user conceptualization. It runs interactively from any on-line hard-copy terminal (currently implemented for Model 33 and 37 Teletypes), permits all of the basic text operations, and imposes very little load on the system in comparison to NLS.

## C. Operation

The operation of the current TODAS is as much like NLS as possible.

The file I/O commands are identical to their NLS counterparts, as are the other file commands (queries, cleanups, etc.).

All of the structure-manipulation commands are essentially the

same as their NLS counterparts, except that typed-in statement addresses must be used instead of bug selections. Addresses may be statement numbers, statement names, and/or codes indicating a structural relationship to some predefined address.

A few NLS commands have no TODAS counterparts; these are commands related to such things as underlining, display flicker, etc., that are meaningless on a hard-copy terminal.

Some NLS features have been omitted from the initial TODAS. These include the content analyzer, the keyword system, the trail system, and link jumping.

The NLS "Jump" commands are replaced by the TODAS "Print" commands. These commands cause the terminal to print out a specified statement, branch, plex, or group. NLS-type VIEWSPECs may be used to control number of levels, number of lines per statement, suppression of statement numbers, etc.

As noted above, NLS bug selections are replaced by statement addresses in TODAS. However, this leaves no way to reference text entities within statements. The solution to this is to replace the NLS intrastatement editing functions with the TODAS "Alter" command.

Operation of the "Alter" command is almost the same as QED, the line-oriented text-editing system developed by Project GENIE for the SDS 940, except that "Alter" deals with an entire statement as QED deals with one line of text.

The statement is treated as a long string of undifferentiated characters, which is edited from beginning to end. User input of "control characters" causes various copy, deletion, insertion, and replacement operations to take place.

Appendix B is an initial version of the TODAS User's Guide, written for users who are familiar with NLS.

# IV GRAPHICS-ORIENTED DOCUMENTATION OUTPUT SYSTEM (GODOS)

## A. Introduction

GODOS operates via an out-of-house facility that uses Stromberg-Carlson 4060 CRT-to-film system. The final output may be either 35-mm or 16-mm microfilm.

At SRI, the microfilm may then be used in a Xerox Copyflo machine to produce high-quality hard copy or printing masters. The microfilm may also be converted to microfiche.

## B. Purpose

In the past, our only means of producing high-quality hard copy has been to punch paper tape for driving a Dura Mach 10 automatic typewriter. This system is fairly fast and produces hard copy of good quality, but offers no way of producing any graphic (i.e. nontextual) material. With the advent of first-stage graphics capability in NLS, it is now very desirable to make use of graphic representations of information, and a hard-copy output capability for mixed graphics and text is essential.

The current GODOS is a provisional attempt to develop and use such a capability. It also provides the ability to produce microfilm (roll or fiche format) without the need for intermediate full-size copy. This will be a great advantage in the operation of the Network Information Center.

## C. Operation

The user outputs a file or a portion of a file with the NLS command "Output Film File." This produces a disc file with the proper format for input to the out-of-house Stromberg-Carlson system. An operator then transfers this file to magnetic tape which is sent to the out-of-house facility. At present, turnaround time for the total operation is a few days.

## V NETWORK INFORMATION CENTER (NIC)

In addition to the planning activity described in Section VIII, AHIRC has begun the clerical and administrative tasks necessary to operate as an active information center.

### A. Solicitation of Documents

We have been soliciting documents from Network members for the past several months. Most of the documents in the current collection come from the initial four nodes of the Network (UCLA, UCSB, Utah, and SRI); from Bolt, Beranek, and Newman; from ARPA; and from the Network Working Group. The collection now contains about thirty documents.

During this initial period of solicitation, we have been establishing procedures to deal with the incoming flow of documents and have been training personnel to use these procedures. In the next few months we hope to find ways of encouraging contributions from other Network members and to double or triple the size of the collection.

### B. Transcription

AHIRC has been transcribing all of the documents received (which have been in hard-copy form) into machine-readable form. Nearly all of the documents received thus far have been transcribed. We have been experimenting with various procedures and standards for transcription, and training personnel in their use. We have established standards for transforming information such as equations and figures, which cannot be represented on standard typefaces, into a machine-readable representation.

In the next few months we expect the transcription service to be extremely busy; we may need to hire additional personnel for the transcription service to function effectively.

### C. Cataloging

We have been experimenting with various methods of cataloging the documents we have received. We have explored such schemes as keywords, keyword in context (KWIC) indices, and abstracting.

We hope, in the next few months, to establish a catalog oriented around keyword retrieval, which will also contain abstracts of all of the documents. We will also be training personnel in the use of this cataloging scheme, including the procedures for remote entry of new items into the catalog.

### D. Distribution

The Network will probably not begin functioning reliably until

late 1969 or early 1970, and the NIC will not be generally available until some time after that. Since Network members will require information before that time, AHIRC will distribute a hard-copy version of the collection to the initial Network participants. We are proceeding with the preparation of this version of the collection and expect that it will be available in the fall of 1969.

After procedures for this hard-copy distribution have been established, we intend to develop methods for handling special requests for hard-copy information from Network members.

# VI USER-SYSTEM RESEARCH

## A. Introduction

The "User System" comprises the user and his task environment in the context of the total AHIRC system -- as opposed to the "Service System," which refers to those activities that deal with implementation of the functional services needed in the user system and with the system programming and hardware that underlies those functional services.

This User-System/Service-System dichotomy is basic to the conceptual framework underlying current AHI planning.

Until recently, work on the User-System aspect has been on an intuitive, trial-and-error basis, with little in the way of an explicit conceptual framework and almost no formal analysis of User-System operation.

To begin work in this area, an explicit User System Research Activity has recently been established. The following is a brief provisional description of operations of this activity.

## B. Goals fo r User-System Research

### 1. Study of TODAS

Research on TODAS will be the primary focus of user system research over the next six months and should provide us with considerable insight into the methods of conducting user systems research. TODAS is an appropriate subject for user-system research, being a specific, relatively simple system that requires evaluation in order to guide its development. As time permits, we will also observe users on NLS.

### 2. Development of a Conceptual Framewo rk

As we proceed to analyze the user system, we will need a conceptual framework. This framework will aid communication with others, and it will aid us in organizing user-system technology. This conceptual framework must allow us to describe the entities relevant to our analysis and design questions, as well as the relationships and interactions between these entities.

The primary (high-level) entities that we will consider within this framework are the following:

#### a. Uses of the System

The system can be used in many ways. We need to understand

how the use relates to the user's needs for functional system services.

b. System Commands

These are the commands and command structure that are available to the user for the control of the system and the entry of information.

c. User Characteristics

Users will differ in their skills (manual, conceptual, and procedural), in their style of work, and in their psychological outlook.

3. Research on Techniques of User-System Research

In line with the general AHI "bootstrapping" approach to research, the User-System Research Activity will be working to develop new ways of conducting its own research, as well as making maximum use of existing techniques.

C. Techniques of Study

Having defined our goals we need to spell out the process for accomplishing them. This process is divided into several parts.

1. Gathering of Information

Information will be gathered by two methods:

Observation, which is carried out in the normal working environment, with a minimum amount of interference with normal working modes.

Experimental studies, which involve the use of controlled experiments -- in which, for example, subjects might be asked to perform specially designed tasks and then observed while performing them.

2. Analysis of Information Gathered

Analysis involves three major components:

One is purely descriptive -- the presentation of data in easily comprehended form.

The second is inferential -- the formation, from descriptive

data, of working hypotheses that can be used for further testing.

The third is theoretical -- the development of a theoretical structure that interrelates findings and makes them collectively coherent.

### 3. The Development of Classification Systems

In order to interrelate entities we need to define specific entities (concepts, subclasses) within the general classes mentioned above (uses of the system, system commands, and user characteristics). We expect to develop detailed classification systems as follows:

#### a. Uses of the System

Uses of the system can be described in terms of "operations" and "file characteristics."

"Operations" are the specific things done to a file by a user, such as restructuring, entering new material, studying, detail editing, etc. An operation may consist of many individual commands.

Operations might be classified in terms of the subset of system commands most heavily involved in each operation. Automatic measurement of the pattern of use of particular commands could then provide an operational definition for determining the incidence of the corresponding operations.

Uses of the system might also be classified according to the characteristics of files, such as the following:

Redundancy in the file at the character level -- English text has very high redundancy, while assembly code has very little.

Explicitness of the structure of the information -- program documentation usually is explicit in its structure while a novel usually is not.

#### b. System Command Structure

Command frequency has considerable bearing on the programming of the system and on the desired rapidity of response. A command which is rarely used may not need to

execute rapidly, whereas a command which is in constant use should presumably be made as fast as possible.

The concept of "command sequences" is used in discussion of the details of an "operation."

> The question is not only one of identifying the individual components of an operation, but of thoroughly exploring the pattern of usage of the components and determining why a particular technique sequence is used in one particular instance in place of another sequence that would achieve the same effect. This has been called the study of "user strategy."

c. User Characteristics

Users will differ in their skills (manual, conceptual, and procedural), and these differences provide the basis for a classification.

Users might be classified in terms of the intellectual and cognitive orientations that they bring to their work. This classification involves preferred modes of perception (e.g., graphic vs. verbal) and of thought (e.g., synthetic vs. analytic).

Users might also be classified in terms of their world view and style as it relates to their psychological and emotional orientation.

# VII SERVICE-SYSTEM DEVELOPMENTS

## A. Time-Sharing System

We activated Project GENIE's last version of the Time-Sharing System (TSS) in December. Liaison has continued with some of the individuals concerned with Project GENIE.

The following subsections describe briefly the principal efforts that have been taking place in the area of TSS.

### 1. System Maintenance and Debugging

Because of the heavy demands made on TSS by NLS, many of the frequent changes in NLS require changes in TSS. This work requires considerable ongoing effort.

### 2. File Handling

#### a. Making Colon Files Permanent

A "colon file" -- so called because of the use of a colon as a flag at the beginning of the file name -- is a disc file that can be accessed by the user directly from NLS (via TSS).

The original intention of Project GENIE was to treat these files exactly as drum files are treated; as it turned out, however, considerable work had to be done at AHIRC to make it possible for the user to set colon files to a "permanent" status, so that they would not be automatically deleted when the user logged out of the system.

#### b. Planning for an Archival File System

Another problem that arose in connection with colon files was that the new file-handling systems, once modified to handle colon files, became incompatible with the existing tape-file systems. As a result, it became impossible to copy random files from the disc or drum to tape.

The function of tape as an archival storage medium has been taken over by the KDF system, which stores files in a protected area of the disc. However, it is desirable to get rid of KDF because its use of disc space is extremely inefficient and files stored under KDF are not directly accessible -- they must first be copied to colon files.

The solution that is currently being worked on is to eliminate KDF, making virtually the entire disc available for permanent (or temporary) colon files.

It should be noted that it will not be immediately possible to actually use all of this space; there is a limitation imposed by the space allotted in each user's file directory, so that each user may have only about 100 files. However, the present limitation of colon files to only about one-eighth of the total disc is much more severe; it is usually impossible for a user to have more than about a dozen files.

Because colon files are not reliable enough for archival use, there will be an automatic tape backup system, so that if a colon file becomes unusable through system errors the user may retrieve a slightly older version from tape.

This requires revision of the TSS's internal working files to make it possible for colon files to be copied to tape. It is estimated that the tape backup system will be operating and KDF eliminated within two or three months; the result will be a great increase in the space available for storage of files that can be immediately accessed from NLS, with no loss in reliability of files.

### 3. Background Queueing and Processing System

Planning and design work is in progress on a "background processing system."

The purpose of this system is to let the user put time-consuming file I/O jobs (and some computing jobs) on a queue, and then return to his work instead of waiting, with a tied-up console, until the job is completed.

Part of this system -- the part that aquires a job from the user and queues it -- has now been coded; the programming for unloading the queue and causing jobs to be executed is still being planned.

## B. NLS and Related Software

The principal software efforts (apart from work on the time-sharing system) have been in the following areas:

Extensive internal reorganization of the system software structure to allow for the other work described here

The development of new compilers for the special programming languages used within AHIRC

The implementation of the new version of NLS and new features to be added to later versions of NLS.

## 1. New Compilers

The old Tree Meta metacompiler and the old MOL compiler produced symbolic (assembly-language) output, which then had to be assembled by the ARPAS assembler (provided as system software by SDS).

A minor consequence of this was that whenever changes were made in NLS, the program had to be first recompiled and then reassembled, a time-consuming process.

A much more important problem was that the assembly-language programs became so big as to overflow the symbol tables provided by ARPAS. It was decided that changing ARPAS would be less desirable than rewriting the compilers.

### a. New Tree Meta

The new Tree Meta compiler-compiler has an extended syntax, as compared to the old one, and is consequently more flexible and powerful.

The direct production of compiler programs as ready-to-run binary files is significantly faster than the old two-pass process and avoids the problem of symbol-table overflow.

### b. New MOL

The original MOL compiler was written in its own language and bootstrapped to produce symbolics which were then assembled. The resulting final version had a number of bugs in it. The new version was written in Tree Meta and compiled directly to binary by the metacompiler; as a result it is much cleaner than the old version.

The new MOL also incorporates a number of new features that permit the writing of more powerful constructs.

## 2. New NLS Implementations

The new (recently implemented) version of NLS is described in a separate section of this report. In addition to this implementation, work has been done on the following features for future versions:

a. "Substitute" Command

This command will cause a specified character string to be searched for throughout a file or specified portion of a file; wherever the string is found it will be replaced by a second specified string. This feature is expected to be very powerful in both programming and natural-language applications.

b. File Compactor

Because of certain design factors in NLS, files often require an amount of storage space out of proportion to the amount of actual text they contain. The file compactor will be a program, possibly integrated into NLS, which can process a file in such a way as to reduce its "apparent size" down to the minimum required for storage of the text and structural information in the file.

c. " Jump to Context" Command

This NLS command will take a character string as argument, and position the display start to the first (or possibly the "next") statement in the file that contains the specified string.

d. Portal Development

The "portal" is a means of interfacing NLS with a "file processor."

A "file processor" will be any one of a series of special programs. A file processor will take a file as input and will perform complex operations upon it. Some processors will produce a file as output; others will return formatted information for insertion into an existing file.

e. Improved Graphics Capability

A number of powerful new features are being implemented for the NLS Vector Package. These will include the provision of "boxes" of various shapes and sizes as primitive graphic entities, closer integration of graphic material with text, and more sophisticated means of manipulating graphic material.

f. New Display-Creation Routine

Creation of a display image is one of the most fundamental functions of NLS. The new routine for doing this will run a great deal faster than the current routine, and will also give the user much more control over the display.

## C. Hardware

### 1. Display Systems

Both of the Tasker display systems have been accepted and are in service. Lower-case alphabetic characters have been added to both.

We are still operating with only six displays on-line, primarily because of space limitations in the 940 core, but this will be alleviated by the external core system described below.

### 2. External Core System

An external core system for display refreshing is under construction and should be in operation by 15 October. The system consists of the following components:

A 32K core bank (24-bit words), with access switching to allow for connection of up to eight devices.

Provisions are included for expansion to 16 devices and two core banks of 64K each.

A controller (built by AHIRC) for transfers between the external core system and the SDS 940 main core.

The interface to the external core system will be the same as the interface to the 940 main core (through the multiplexer built by AHI for 940 core). Devices may be simply plugged into either core system.

Initially we will be operating both display systems and the ARPA Network terminal from the external core system.

### 3. Network I nterface

The design for the Network interface is complete and construction has begun on the IMP interface to communicate with the Network. This will be a full-duplex interface operating from external core (or, optionally, from 940 core).

# VIII FUTURE PLANS

## A. Presentation for ASIS

The next annual meeting of the ASIS (American Society for Information Science, formerly American Documentation Institute) will be held on 1-5 October in San Francisco at the Hilton Hotel. The AHIRC has been invited to participate.

We plan to take the same general approach as was used for our participation in the Fall Joint Computer Conference last December: we will give a formal presentation using on-line techniques with projection TV on 1 October, and then follow up with informal open-house sessions on the next two days.

The composition of the audience will be different from the FJCC, and this presentation will be slanted strongly toward the implications of our developments for information-system problems.

Non-labor costs for this participation will be borne by the ASIS, and personnel time will be charged as SRI overhead.

## B. Expansion of Terminal Facilities

In the near future, we expect to add a number of new terminals to our computer facility.

With the completion of the external core system, we will be able to add six new CRT terminals, for a total of 12, thus providing a user terminal for each of the 12 primary display CRT's.

We also plan to have 10 interactive-typewriter terminals with upper- and lower-case capability, suitable for operation of TODAS and other systems that do not depend on CRT display of information.

We are considering either Model 37 Teletypes or IBM Model 1052 terminals.

This will give us a total of 22 on-line terminals within AHIRC (in addition to a number of conventional Teletype terminals). Our goal is to ensure that anyone in AHIRC can find an unoccupied terminal of some sort at any time. With TODAS in use, this will mean that it will always be possible for a staff member to get immediate access to his on-line files.

## C. User-System Research

The User-System Research Activity will be developing from its current embryonic state toward becoming a major component of AHI

work. The following provisional listing of objectives is given as a brief indication of the direction and scope of this activity.

Long-Range Objectives

Increase our ability to analyze the user aspects of AHIRC's developing augmentation tools.

Develop specific "languages" and procedures appropriate to the analysis of user systems.

Improve our ability to train users in the use of AHI systems.

Develop the ability to compare conventional systems and other augmentation-oriented systems for professional work with AHI systems, in terms of the varieties and degrees of augmentation they offer.

Short-Range Objectives

Analyze the use of the Typewriter-Oriented Documentation-Aid System (TODAS) with respect to the creation of documentation in the area of computer-science research.

Develop a language and analytical tools suitable for characterizing the interaction of the user, the command structure, the problem, and the techniques in a way that will aid us in designing TODAS.

Guide the evolution of TODAS toward maximum power for composing, modifying, and querying documentation files.

Develop training techniques for TODAS.

## D. GODOS

Current plans for GODOS are mostly concerned with development of programs for file handling and provision of operator service, and with continuing development of the text-handling capabilities of GODOS.

### 1. File Handling and Operator Service

At present, there is no well-organized system for transferring an output file from the disc to magnetic tape or for sending the tape to the out-of-house production facility with the necessary information as to number and type of copies, etc. There is also no periodic basis for

collection and processing of material; runs are made on an ad hoc basis.

Ultimately, the user will be able to produce a disc output file, enter the necessary information about copies in a communication file (or perhaps at the front of the output file itself), and then return to other work without need for further intervention on his part.

An operator will check for new output files on a routine basis and will then do all of the necessary file-transfer operations and send the tape out for production.

2. Text-Handling Improvements

Features under consideration for future incorporation in GODOS (some of them depending upon developments in NLS and the new output processor) include the following:

Superimposition of text and graphic material

Control of character sizes

Special symbols

Italics and boldface

Half-tone graphics

Formatting with footnotes

Special formatting for equations, etc.

E. TODAS

The initial TODAS described in this report will be subject to intense study under the new User Systems Research activity, and will come into extensive use both as the primary remote interface with the NIC and as an alternative to NLS within the AHIRC.

Continuous improvement of TODAS may be expected as a consequence of needs that will become evident as it is studied and applied.

F. New Output Processor

The current output-processing system, PASS4, is used for all hard-copy production, film production, and file processing via the QED subsystem (currently used as a way of transferring information

from one file to another). We plan to replace PASS4 with a completely new output processor.

The new processor will perform essentially the same functions as PASS4, but will be much better integrated with NLS/TODAS and will allow the user far more flexible control over output formatting.

It will also have available more automatic features for format control, permitting the production of correctly formatted hard copy with fewer passes and less proofreading and correction by the user.

Formatting will be done on a page-by-page basis, instead of the line-by-line basis of PASS4. This will permit automatic elimination of widow lines, sophisticated algorithms for placement of graphic material on the page, etc.

Specific features currently under consideration for inclusion in the new output processor include the following:

Two-column page format (optional).

Conventional formatting of footnotes -- i.e., at bottom of pages.

This will involve devising some special convention for identifying text in an on-line file that is to be handled as a footnote in hard copy.

Indexing -- automatic generation of tables of contents, tables of statement names, keyword tables, word lists, etc., with page-number references.

A generalized way of specifying format for page headings, capable of reproducing almost any conventional heading format used in documentation.

Conversion of text links to page references.

An improved user language for control of output by means of "directives" embedded in the text.

## G. Network Information Center (NIC)

Considerable planning work has been done for the information center that the AHIRC will be operating in the ARPA Computer Network. The following is a brief resume of the current thinking on possibilities for NIC service.

## 1. Service Needs

Since the NIC and the Network are both experimental, we expect that the demands placed on the NIC will change as more researchers use the Network.

The amount of information that needs to be included in the NIC will increase.

Growth of the total bit mass of the collection requires that the data manipulation facilities accommodate expansion of the data base to many times its initial size.

Increase in the number of documents held in the collection will require cataloging and retrieval systems that can provide more and more aids to the user in finding his way to the information he needs.

The NIC users will have increasingly varied skills and needs as their numbers grow.

The experimental nature of the Network makes the characteristics of the eventual users difficult to predict; however, we expect the following:

(1) The initial user population will probably be composed of people proficient in the use of computer software.

(2) Researchers who have limited experience in the use of computers will begin to use the NIC to communicate with their counterparts at other installations, to search for information about their specific areas of research, and to keep documentation of their work in progress.

(3) Hopefully, administrative personnel will begin to find the NIC useful for such management tasks as report composition and project evaluation.

(4) Clerical and secretarial personnel may begin to enter reports into the collection and to conduct routine literature searches.

## 2. Proposed Services

The variety of skills of prospective NIC users suggests that

the system, to be useful to different users, will have to appear differently to each one -- i.e., it must be flexible enough to be usable in a variety of modes, which may be quite markedly different from one another.

a. Query, Documentation, and Retrieval Media

Research done at the AHIRC and described in previous reports indicates that for a trained user, who needs nearly instantaneous response, a high-performance CRT display device such as that used in NLS is the most useful medium for communicating with the machine. However, such devices are expensive and are not generally available.

Since typewriter terminals are readily available and inexpensive, NIC on-line services will initially be provided through typewriters (and Teletypes). The TODAS system, described elsewhere in this report, will be the primary tool available to NIC users. We hope to support low-performance CRT's in the future, and eventually high-performance display devices.

In many cases, the NIC user will not require immediate response. For example, a secretary may simply be transcribing a document, or a researcher may be searching for a list of documents that is not needed for a few hours, or even a few days. We plan to satisfy these delayed-action requirements in at least two ways.

First, we intend to provide a batched mode of NIC operation. In this mode, the user could enter a number of NIC commands (and text) through his typewriter, without having any of them executed. These operations would then be executed later, upon command from the user. This delayed-action response can be refined further to allow a user to specify when his batched operations should be performed -- immediately, overnight, or over the weekend, for example.

Secondly, we plan an off-line mode of machine interaction. In this mode, the user can create a stream of NIC operations with paper tape. These tapes can then be read into the machine and the commands on them executed. As with the batched mode of on-line operation, both editing and search operations can be performed in this fashion, and the user can specify when the operations are to be performed.

Some NIC operations will not require machine interaction. For example, extended examination of a large number of documents can often be done more quickly with hard copy than with a slow-speed typewriter. Many diagrams and photographs cannot be represented adequately with a typewriter; such figures require hard copy. In addition, since on-line resources are precious, some method needs to be provided to perform simple retrieval operations without using the machine. We plan to provide at least two methods for using the NIC collection without machine interaction.

First, we intend to distribute the NIC collection, as well as a catalog useful for off-line retrieval, in microform -- either as microfilm or as microfiche. The GODOS system, described elsewhere in this report, will be used for this purpose.

Secondly, we will make several provisions for paper copies of the collection and catalog. It will be possible to generate paper copy from the microform collection, either locally or at AHIRC. In addition, a user will be able to use a printer, either at his own installation or at AHIRC, to print documents or parts of documents.

In a typical on-line NIC session, the user may need to use the hard-copy collection. He may need to look at figures, or he may wish to retrieve hard copy of documents that interest him.

The user will be able to direct printing of documents on an on-line printer while continuing his work at a typewriter.

We hope to support in the future a work station consisting of an interactive device, such as a typewriter, and a microform viewer, controlled by the computer. The user could then perform search and retrieval operations using the typewriter. When he wished to examine a lengthy portion of text, or a figure, the machine could then position his microform viewer at the appropriate position in the microform collection, and he could proceed to look at the document.

b. NIC Commands

The variety of input/output media requires a variety of command sets. For example, the editing commands for TODAS,

where the text is being updated continuously, are not convenient in off-line paper tape operations, where the updating is done later.

The command sets used for editing and for search and retrieval will be different in the interactive, batched, and off-line modes of NIC operations.

Since a user may switch frequently from one mode of operation to another, we will maintain (as nearly as possible) logical and semantic consistency among the different command sets.

Varying degrees of sophistication in the use of NIC require that the commands be simple enough to be understood by a novice, yet powerful enough to satisfy the demands of an experienced user. We plan to accommodate these different user needs in several ways.

We will have several common search patterns stored on-line. The user will need only to insert specific information about his request to perform the search. For example, someone searching for documents by a specific author, written after a certain date, would only have to put the author's name and the date into a pattern and then execute the pattern.

More sophisticated users will be able to construct their own search patterns and execute them immediately, or store them on-line for subsequent use.

Experience at AHIRC indicates that the availablity of a reasonably large number of commands, some more esoteric than others, can help meet the requirements of users with different skills.

With an extensive command repertoire, a novice may require several commands to perform an operation that an expert performs with only one command. However, the casual user needs to know a much smaller number of commands than the experienced user.

Other research at AHIRC has suggested the desirability of constructing a metalanguage in which a user could specify his own command language. Such a high-level programming language could also be incorporated into the NIC.

The researchers who will use the NIC will be investigating

many different areas. The NIC must be flexible enough to serve these different needs without burdening the user with catalogs and information that are of no interest to him.

Text in the on-line system currently used at AHIRC, which will be the core of NIC services, is organized hierarchically. The commands that operate on the text provide powerful tools for obtaining different selective views of a single body of information.

The user will be able to create his own subsets of catalogs and information, and to modify his own catalogs at will.

We hope to encourage the use of the Network as a new tool for interactive research. The ability to retain comments on line, to send messages and letters, and to establish links among various documents in the collection is planned to be an integral part of NIC services.

# BIBLIOGRAPHY

The following is a selected bibliography of AHIRC publications, in chronological order. Reports with AD numbers are available from Defense Documentation Center, Building 5, Cameron Station, Alexandria, Virginia 22314.


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), AD289565.

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), AD640989.

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), AD622520.

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).

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).

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).

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, W. K. English, and D. A. Evans, "Study for the Development of Computer Augmented Management Techniques," Interim Technical Report RADC-TR-69-98, Contract F30602-68-C-0286, SRI Project 7101, Stanford Research Institute, Menlo Park, California (March 1969).

## Appendix A
## NLS CALCULATOR PACKAGE USER'S GUIDE

The NLS calculator functions internally like a desk calculator with four operations -- add, subtract, multiply, and divide -- and five storage locations: an accumulator and four registers called w, x, y, and z. The contents of the accumulator are displayed in the feedback area while the calculator is in operation.

There is an NLS text entity called "number." The syntax of a "number" is

```
('- / .EMPTY) ('$ / .EMPTY) 1$3D (', DDD $(', DDD) / $D) ('. 1$D
/.EMPTY)
```

Thus a number includes any minus sign and/or dollar sign, followed by a string of digits and/or commas, with (optionally) one decimal point followed by at least one digit.

Note that commas are only allowed where they belong.

There are two modes of usage, direct and indirect.

### Direct Operation

In direct mode the user specifies operations and operands step by step.

Operations are specified by one-letter mnemonics (a, s, m, d).

Operands may be input by any of three means:

Selecting numbers from the text

Typing numbers in as literals

Calling numbers from registers.

To accomplish all this the user first gives the command "Execute Calculator" (syntax: ex CA). The operations are then specified as subcommands. Other subcommands will clear or negate the accumulator or write its contents into one of the registers.

At any point in the calculation, the user may interrupt by returning to NLS proper, without losing the data in the accumulator and registers. This is useful in case the user wishes to jump to another file location in order to find a number. He may then start up the calculator again and resume

where he left off.

When the desired result is in the accumulator, the user may return from the calculator to NLS proper and use the "Insert" subcommand. This causes the contents of the accumulator to be copied into the desired location in the file text.

## Indirect Operation

In indirect mode, the user writes a function in infix notation as part of the file text, much as a content-analyzer pattern is written.

This is interpreted by the subcommand "Function," but not run immediately.

The function is run by giving a second subcommand, called "Execute Function." This causes the system to start demanding values for the variables mentioned in the function; the user gives the values either by bug selection of numbers or by literal entry from the keyboard or keyset.

The function, where it appears in text, ends either with a semicolon or with the end of the statement. Like a content analyzer pattern, it is selected by pointing to its beginning. Any syntax that cannot be interpreted will produce an error message.

Up to nine variables are allowed in a function. A variable is written in the form "bn", where the letter "b" identifies it as a variable and "n" is a number from one to nine.

As soon as the "Execute Function" command is given, the system demands the variables by displaying messages in the feedback area of the screen.

The variables are demanded in the order in which they have been numbered, i.e. b1, b2, b3, etc. As in the case of direct operation, the values may be given as literals or by bug selection.

The user may interrupt by returning to NLS proper so as to find a value in another part of the file; then, when he gives "Execute Function" again, the system takes up demanding variables where it left off.

When all variables have been defined, the function is evaluated and the value placed in the accumulator. From

there it may be inserted in the text, just as in the case of direct operation.

## Calculator Subcommands

### Direct Operation

(a) Add

Syntax: a ([n] / LIT / (w/x/y/z)) CA

Semantics: A number supplied by the user is added to the accumulator.

(s) Subtract

Syntax: p ([n] / LIT / (w/x/y/z)) CA

Semantics: A number supplied by the user is subtracted from the accumulator.

(m) Multiply

Syntax: m ([n] / LIT / (w/x/y/z)) CA

Semantics: The accumulator is multiplied by a number supplied by the user.

(d) Divide

Syntax: d ([n] / LIT / (w/x/y/z)) CA

Semantics: The accumulator is divided by a number supplied by the user.

(c) Clear Accumulator

Syntax: c CA

Semantics: The accumulator is set to zero.

(n) Negate Accumulator

Syntax: n CA

Semantics: The accumulator is set to the negative of its initial value.

(t) Transfer Accumulator to ...

Syntax: t w/x/y/z CA

Semantics: The contents of the accumulator are transferred to the specified register.

Indirect Operation

(f) Function

Syntax: f [f] CA

Semantics: This subcommand is used to interpret a function from the infix notation written in file text.

(e) Execute Function

Syntax: e CA

Semantics: This executes the function specified by the "Function" subcommand.

Special

(i) Insert

Syntax: i [w] CA

Semantics: The contents of the accumulator are inserted into the file text after the specified word.

(p) Parameter Set

Semantics: The subsubcommands under "Parameter Set" allow the user to set two parameters controlling the format of numbers output from the calculator.

(pd) Digits After Decimal Point

Syntax: p d <number> CA

Semantics: The user enters a number from 0 to 63. The fractional part of numbers in the calculator will be truncated to this many digits. The default value of this parameter is 4.

(pc) Commas

Syntax: p c y/n CA

Semantics: This allows the user to specify whether commas are to be left in output numbers (y for "yes") or deleted (n for "no"). The initial default is to leave them in.

Note: The following is excerpted from a very preliminary version of the User's Guide for an experimental version of TODAS. It is neither complete nor fully up-to-date, and is presented here only to give a rough picture of the nature of TODAS operation.

TODAS can be used from any existing 940 terminal, including Model 33 Teletypes. It accepts and produces NLS files -- i.e., it is totally compatible with NLS. It also produces CHECKPOINT and WORKING COPY files in exactly the same manner as NLS.

TODAS looks as much like NLS as possible. This means that all structure conventions are exactly the same and most of the commands that do not involve editing within statements are much the same. The main differences arise from the absence of the mouse.

At the structural level, this means that one must give addresses by typing in statement numbers. Please see section at end of this document, explaining TODAS addresses.

When editing within a statement, it means that everything is completely different from NLS. Most of the text-editing commands are essentially the same as QED.

## I/O Commands

(Except as noted, all command semantics are similar to NLS.)

### Load Checkpoint

Syntax: 1 .

### Load File

Syntax: 1 f <FILENAME> CA CA

Note: Enough of the <FILENAME> must be typed out so that it is uniquely identified. The system does not type back the rest of the <FILENAME> until the first CA is hit.

### Output Checkpoint

Syntax: o .

### Output File

Syntax: o f (<FILENAME>/.empty) CA CA

## Print commands

These replace the Jump commands of NLS, causing a statement, branch, plex, or group to be printed out. (See section on addresses at end of this appendix.) Note that an address string takes the place of a bug selection of a statement in NLS.

### Print S tatement

Syntax: p s <ADDRESS STRING> . VIEWSPEC (SPACE /./CA)

### Print Branch

Syntax: p b <ADDRESS STRING> . VIEWSPEC (SPACE /./CA)

### Print Plex

Syntax: p p <ADDRESS STRING> . VIEWSPEC (SPACE /./CA)

### Print Group

Syntax: p g <ADDRESS STRING> , <ADDRESS STRING> . VIEWSPEC (SPACE/./CA)

## Structural Editing Commands

### Insert Commands

#### Insert Statement

Syntax: i s <ADDRESS STRING> . LEVADJ SPACE LIT CA

### Delete Commands

#### Delete Statement

Syntax: d s <ADDRESS STRING> .

#### Delete Branch

Syntax: d b <ADDRESS STRING> .

#### Delete Plex

Syntax: d p <ADDRESS STRING> .

### Delete Group

Syntax: d g <ADDRESS STRING> , <ADDRESS STRING> .

## Replace Commands

### Replace Statement

Syntax: r s <ADDRESS STRING> . LIT CA

### Replace Branch

Syntax: r b <ADDRESS STRING> . LIT CA

### Replace Plex

Syntax: r p <ADDRESS STRING> . LIT CA

### Replace Group

Syntax: r g <ADDRESS STRING> , <ADDRESS STRING> . LIT CA

## Move Commands

### Move Statement

Syntax: m s <ADDRESS STRING> . <ADDRESS STRING> . LEVADJ CA

### Move Branch

Syntax: m b <ADDRESS STRING> . <ADDRESS STRING> . LEVADJ CA

### Move Plex

Syntax: m p <ADDRESS STRING> . <ADDRESS STRING> . LEVADJ CA

### Move Group

Syntax: m g <ADDRESS STRING> . <ADDRESS STRING> , <ADDRESS STRING> . LEVADJ CA

## Copy Commands

### Copy Statement

Syntax: c s <ADDRESS STRING> . <ADDRESS STRING> . LEVADJ CA

Copy Branch

Syntax: c b <ADDRESS STRING> . <ADDRESS STRING> . LEVADJ CA

Copy Plex

Syntax: c p <ADDRESS STRING> . <ADDRESS STRING> . LEVADJ CA

Copy Group

Syntax: c g <ADDRESS STRING> . <ADDRESS STRING> , <ADDRESS STRING> . LEVADJ CA

## Statement Editing Commands

### Alter

Syntax: a <ADDRESS> .

Semantics: This command works very much like the "modify" command in QED.

Whatever QED does with a line, "Alter" does with a statement. Therefore, a carriage return is a legal character.

The "Alter" process should be visualized as involving an old statement and a new one; when the "Alter" is terminated with a CA or control d the old statement is replaced by the new one. If the command is terminated with a CD, the new statement is thrown away and the old one remains.

(conchar) Editing is controlled by special "control characters." These are mostly the same as the QED ones; exceptions to this are marked with asterisks (*). Some of the control characters apply to the LIT part of the "Insert" and "Replace" commands; these are marked with pound-signs (#). The control characters are as follows:

#*A slash (/) causes the next character to be taken as a capital. When using a device that has uppercase characters only, the capital letters are always typed with slashes in front of them; however, the combination of a slash and a letter is treated as one character for purposes of the "Alter" command.

#*An exclamation point (←) causes the next character to be taken literally, even if it is a control character.

#*Control a copies the remainder of the old statement to the new one (printing it out), but does not terminate the "Alter" command.

*Control b works like a centerdot in NLS.

(This is not applicable in an "alter" command.)

Control c copies next character from old statement to new one.

#Control d (CA on a console) copies remainder of old statement to new one (printing it out) and terminates the "Alter" command.

#In an "Insert" or "Replace" command, this simply ends the LIT and terminates the command.

Control e is used as a delimiter for inserting text. When the first control e is hit, a < is printed; then the text to be inserted is typed; when the next control e is hit, a > is printed and the insertion is complete.

Control f copies remainder of old statement to new one (without printing it out) and terminates the "Alter" command.

*Control g, followed by some character, skips characters in the old statement up to and including the next occurrence of that character, printing a % for each character skipped.

#*Control h (BACKSPACE on a Model 37 Teletype) deletes last input character from new statement (not old one) and prints uparrow.

#In an "Insert" or "Replace" command, this simply throws away the last character of the LIT and prints an uparrow.

#Control i is a tab.

#Control j (LINE SPACE or LINE FEED on Teletypes) is a line feed.

*Control k -- not used.

*Control l -- not used.

#Control m is a carriage return.

Control n is the one-character "restorative" backspace -- it throws away the last character in the new statement and moves back one character in the old one.

Control o, followed by some character, copies characters from the old statement to the new one up to but not including the next occurrence of that character.

Control p, followed by some character, skips characters in the old statement up to but not including the next occurrence of that character, printing a % for each character skipped.

#*Control q is "backspace statement" -- it throws away all of the existing new statement, going back to the beginning of the old statement.

#In an "Insert" or "Replace" command, this simply throws away all of the LIT and leaves the user ready to start the LIT over.

Control r causes the existing portion of the new statement to be typed out fresh, allowing the user to figure out where he is at after using backspaces.

Control s skips the next character in the old statement, printing a %.

*Control t causes the remainder of the old statement to be thrown away and terminates the "Alter" command.

*Control u -- not used.

*Control v -- not used.

#Control w is "backspace word" -- it causes the last input word to be thrown away, back to but not including the last gap character. A backslash is printed.

#In an "Insert" or "Replace" command, this simply throws away the last word of the LIT and prints a backslash.

*Control x works like the CD in NLS.

*Control y -- not used.

Control z, followed by some character, copies characters from the old statement to the new one, up to and including

the next occurrence of that character.

Addresses

An address string is a string which is evaluated by TODAS to form a single address, i.e., a pointer to some specific statement.

Note on Syntax of an Address: Address strings consists of four kinds of elements -- locnums, names, strucrels, and cspcalls. Any number of these may be strung together in any order to form an address string, with the constraint that a locnum must be separated from a succeeding strucrel or cspcall by a space.

Syntax: address string = <a string of address elements, with spaces as needed>

address element = locnum / name / strucrel / cspcall

locnum = statement number

name = statement name in parentheses

strucrel = u/d/h/t/p/s/e/c

These characters have essentially the same meanings as the NLS entity-names UP, DOWN, HEAD, TAIL, PREDECESSOR, SUCCESSOR, and END, respectively.

cspcall = c

The character c means CURRENT; it refers to the current value of the "current statement pointer" (csp).

The csp points to one of the following, whichever is most recent:

The statement indicated by the last complete address string to be evaluated in a command

The last statement printed by a "Print" command.

Semantics: An address string is evaluated from left to right; each element causes a change in the "temporary statement pointer" (tsp).

At the beginning of evaluation of an address string, the tsp is set equal to the csp.

Each time a locnum or name is found in the address string, the

tsp is reset to the indicated statement.

When a strucrel (except c) is found, it is used to calculate a new tsp value from the old one.

When a c is found, the tsp is again set equal to the csp.

When the whole address string has been evaluated, the csp is set equal to the tsp.

Example 1: A "Print" command has just been executed; the last statement printed was number 1c3. The address string given in the next command is "ds".

The tsp points initially to statement 1c3.

TODAS recognizes the "d" in the address string and changes the tsp to point to 1c3a.

Then it recognizes the "s" and changes the tsp to point to 1c3b.

Finally the csp is set equal to the tp so that it also points to 1c3b.

Example 2: The address given in a command is "4b2 uupd".

TODAS recognizes the locnum and the tsp value is changed to point to statement 4b2.

TODAS recognizes the first "u" and changes the tsp to point to 4b.

Then it recognizes the second "u" and changes the tsp to point to 4.

Then it recognizes the "p" and changes the tsp to point to 5.

Next it recognizes the "d" and changes the tsp to point to 5a.

Finally the csp is set equal to the tp so that it also points to 5a.

Example 3: The address given in a command is "(logos) ddcu". The previous command was an "Alter" command on statement 7g.

The tsp points initially to statement 7g.

TODAS recognizes the name "logos" and sets the tsp to point to the statement with that name.

Then it recognizes the first "d" and sets the tsp to point to the first substatement of "logos".

Next it sees the second "d" and sets the tsp to point to the first substatement of the first substatement of "logos".

Now it comes to the "c" and resets the tsp to 7g again.

On recognizing the "u", TODAS sets the tsp to 7.

Finally, the csp is set so that it also points to statement 7.

Comment: This example shows how the user may change his mind in the middle of typing an address string; the effect of the cspcall "c" is to throw away the preceding part of the string.

Submitted by:

D. C Engelbart

D. C. Engelbart, Principal Investigator

Approved by:

David R. Brown

David R. Brown, Director
Information Science Laboratory