ED 186 025 IR 008 391

AUTHOR Montgomery, Ann D.; Judd, Wilson A.
TITLE Computer-Assisted Instruction in the Context of the Advanced Instructional System: Authoring Support Software. Final Report.
INSTITUTION McDonnell Douglas Astronautics Co. - East, St. Louis, Mo.
SPONS AGENCY Air Force Human Resources Lab., Brooks AFB, Texas.
REPORT NO AFHRL-TR-79-12
PUB DATE Dec 79
CONTRACT F33615-78-C-0022
NOTE 88p.: Parts may not reproduce clearly.
AVAILABLE FROM National Technical Information Service, Springfield, VA 22151

EDRS PRICE MF01/PC04 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction: Computer Managed Instruction: Instructional Design: *Instructional Materials: *Instructional Systems: Man Machine Systems: *Material Development: Program Descriptions
IDENTIFIERS *Authoring Languages: *Computer Software

ABSTRACT

This report details the design, development, and implementation of computer software to support the cost-effective production of computer assisted instruction (CAI) within the context of the Advanced Instructional System (AIS) located at Lowry Air Force Base. The report supplements the computer managed Air Force technical training that is currently supported by AIS, giving the Air Force a full function computer based instructional system. In describing the interactive authoring editor, presentation program, data collection, and data print software components of the CAI system, this report indicates that the editor simplifies the authoring task by (a) eliminating the need for the author to use a computer language, (b) structuring the task, (c) providing computer-aided input, and (d) providing extensive formatting and editing capabilities. The software also provides conditional and unconditional branching that can be specified from the editor. (Author)

AFHRL-TR-79-12

AIR FORCE

HUMAN RESOURCES

COMPUTER-ASSISTED INSTRUCTION IN THE CONTEXT OF THE ADVANCED INSTRUCTIONAL SYSTEM: AUTHORING SUPPORT SOFTWARE

By

Ann D. Montgomery
Wilson A. Judd
McDonnell Douglas Astronautics Company – St. Louis
P.O. Box 516
St. Louis, Missouri 63166

TECHNICAL TRAINING DIVISION
Lowry Air Force Base, Colorado 80230

December 1979

Final Report

Approved for public release; distribution unlimited.

LABORATORY

AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFHRL-TR-79-12	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) COMPUTER-ASSISTED INSTRUCTION IN THE CONTEXT OF THE ADVANCED INSTRUCTIONAL SYSTEM: AUTHORING SUPPORT SOFTWARE		5. TYPE OF REPORT & PERIOD COVERED Final
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Ann D. Montgomery; Wilson A. Judd		8. CONTRACT OR GRANT NUMBER(s) F33615-78-C-0022
9. PERFORMING ORGANIZATION NAME AND ADDRESS McDonnell Douglas Astronautics Company - St. Louis, P.O. Box 516, St. Louis, Missouri 63166		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62205F 11210226
11. CONTROLLING OFFICE NAME AND ADDRESS HQ Air Force Human Resources Laboratory (AFSC) Brooks Air Force Base, Texas 78235		12. REPORT DATE December 1979
		13. NUMBER OF PAGES 86
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Technical Training Division, Air Force Human Resources Laboratory, Lowry Air Force Base, Colorado 80230		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Advanced Instructional System (AIS); Authoring - Instructional Materials; CAI - Software; Computer Assisted Instruction (CAI); Computer Based Instruction (CBI); Computer Managed Instruction (CMI); technical training		

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report details the design, development and implementation of computer software to support the cost effective production of Computer-Assisted Instruction (CAI) within the context of the Advanced Instructional System (AIS) located at Lowry AFB. This report supplements the computer managed Air Force technical training that is currently supported by AIS, giving the Air Force a full function Computer Based Instructional system. The interactive Authoring Editor, Presentation Program, Data Collection, and Data Print software components of the CAI system are detailed. The Editor simplifies the authoring task by (a) eliminating the need for the author to use a computer language, (b) structuring the task, (c) providing computer aided input, and (d) extensive formatting and editing capabilities. The software also provides conditional and unconditional branching that can be specified from the Editor.

DD FORM 1 JAN 73 1473

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

PREFACE

Throughout this report, reference is made to Part I and Part II of a two-part report. Part I describes the development of computer software to facilitate authoring, presentation, and evaluation of computer-assisted instruction materials and is presented in this report. Part II defines a procedural model for computer-assisted instruction production and describes the evaluation of the complete authoring system. Part II is actually a separate contractual effort and is discussed in the following technical report:

Lewis, W.E., Lovelace, D.F., Mahany, R.W., & Judd, W.A. *Computer-assisted instruction in the context of the Advanced Instructional System: Materials development procedures and system evaluation.* AFHRL-TR-79-74. Lowry AFB, CO: Technical Training Division, Air Force Human Resources Laboratory, 1980.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Continued)

LIST OF ILLUSTRATIONS (Continued)

SUMMARY

Objective

The overall objective of this project was to design, develop, implement, and evaluate an authoring system which would provide a basis for cost effective production of computer-assisted instruction (CAI) materials for use in the context of computer-managed Air Force technical training. The specific target application was the Advanced Instructional System (AIS) located at Lowry Air Force Base, Colorado, and the software developed was to be integrated into this system. The project work was conducted through two parallel efforts which are described in Parts I and II of this two-part report. The first of these efforts, described here in Part I of the report, addressed the development of computer software to facilitate the authoring, presentation, and evaluation of CAI materials. The second effort, described in Part II, concerned the definition of a procedural model for CAI production and evaluation of the complete authoring system.

Approach

The design activities began with analysis of the probable functions of CAI within the AIS, review of prior approaches to supporting CAI materials development, examination of previous military CAI development experiences, analysis of the characteristics (training, prior experience and work environment) of the Air Training Command (ATC) personnel who would be developing CAI materials, and re-examination of the available AIS software and operational experience with this software. A major conclusion resulting from these analyses was that there are a number of factors in the military technical training environment which are incompatible with the typical approach to CAI production (authoring) and that prior attempts to utilize CAI in this environment had not taken these factors into sufficient consideration. Typically, it appeared that too much was expected of the CAI authors. It was decided, therefore, that it was preferable to adapt the authoring system to the existing environment rather than expect the environment to change to meet the requirements of the system, even when this approach limited the sophistication of the CAI materials which could be produced. For example, it was considered preferable to avoid author use of a programming language even though this would limit the author's flexibility. The AIS employs editors which engage the user in an interactive, English language dialog to control the system's data base. Experience with these interactive editors suggested that they could provide a model for the author/computer interface. It was also concluded that the system should structure the author's task, promote principles of good instructional design, require very little training for its use, and provide aids for managing a materials development project.

The heart of the software developed is an interactive CAI Authoring Editor. The Editor structures the author's task while providing options

as to CAI presentation strategy details. As defined by the Editor, a CAI module is divided into objectives. The system is frame oriented and each objective can contain up to 100 frames. Three classes of frame types are supported: textual content frames; question frames and special purpose frames. The author enters the frame content in exactly the format in which it will be seen by the student. All formatting of question frames is done automatically and the author is prompted to supply a feedback statement for each alternative or constructed response and a prompt statement for each attempt at answering the question. To individualize instruction, the author can define branches from any frame to any other frame. The decision of whether or not to branch can be based on a specified number of questions being answered correctly or incorrectly, on a set of frames having or not having been presented, or on a specific student response. Branching logic is entered in a highly prompted, multiple choice format, and the result is then displayed in English.

CAI materials are delivered by a CAI Presentation Program. A standard presentation program was first written for CAI as an alternative instructional module. Two modified versions of this program provide review and remediation over the specific objectives with which the student had problems. The presentation programs contain student performance data collection routines which can be turned on or off by the author. Standard data analysis reports are requested by means of a second interactive editor which prompts the user as to the options available.

A variety of printouts of program content and logic are provided. Module content can also be printed out in a format suitable for student use as a programmed text.

Evaluation Procedures and Results

The software developed was evaluated as part of the total authoring system through the production and implementation of a set of CAI lessons in one of the courses supported by AIS and by training a number of course personnel in the use of the authoring system. The evaluation procedures and results are described in detail in Part II of this report and are only summarized here, in Part I.

Under the materials production effort, CAI modules were developed for six lessons in the AIS Weapons Mechanic course. None of the three members of the authoring team had prior CAI experience; although, all were experienced technical training authors. Approximately 2200 man hours were required for development of the six modules. The modules accounted for a total of approximately 25 Plan of Instruction (POI) hours and resulted in an average student contact time of 18.7 hours. Thus, development efforts required an average of 88 man hours per POI hour and 118 man hours per student contact hour. This compares very favorably with the figures of 222 and 246 man hours per contact hour reported by Himwich (1977) for military technical training CAI.

Three ATC instructors were trained in use of the Authoring Editor during 15 one-half day sessions. None of the trainees were computer programmers or had any prior CAI development experience. There was no formal training after the first session. Given a CAI Author's Handbook to use as a reference manual, each trainee used the Authoring Editor to develop a CAI module. Contractor personnel were available to answer questions and review the trainees' work. At the end of the training period, each had developed a module through the stage of revision following single student tryouts. The trainees asked relatively few questions and the modules produced were of generally good quality and capitalized fairly well on the capabilities of CAI. The trainees were quite satisfied with the Editor and all expressed an interest in implementing CAI in their courses.

Conclusions

The approach taken to facilitating CAI development appears very promising. Experience to date, while limited, has demonstrated that contractor personnel were able to produce effective CAI with a level of effort that is comparable to the effort required to produce paper and penci materials. ATC personnel were able to learn to use the Authoring Edito and to produce CAI materials within a very short period of time. These trainees expressed highly favorable attitudes about the approach and found no serious faults with the Editor.

The major factors which contribute to simplifying the authoring task are probably elimination of any need for the author to use a computer language and the extent to which the task is structured. The human engineered, computer-aided input, formatting, and editing capabilities provided by the Editor are undoubtedly also important.

The authoring system is ready for use by ATC instructional development personnel in its current form but it should not necessarily be considered to be a finished product. There are a number of features which could be added to increase its utility. As the system is used, it can be expanded and refined on the basis of accumulating experience to become a powerful tool for instructional development.

I INTRODUCTION

Computer-based instruction has the potential of achieving significant savings in costs related to Air Force technical training. A number of steps toward realizing this potential have already been taken, e.g., use of the PLATO system and development of the Advanced Instructional System (AIS). The former is an application of computer-assisted instruction (CAI) while the latter represents a comprehensive computer-based instructional (CBI) system supporting both computer-managed instruction (CMI) and CAI.

CMI can be defined as a situation in which the majority of the student's instructional activities are completed off-line. The computer's role is that of evaluator, diagnostician, prescriber, and manager of instructional events. In CAI, by contrast, all of the student's instructional activities are conducted on-line, at an interactive computer terminal. CMI can be characterized as being extensive, managing instruction for a large number of students throughout a large body of course content. CAI, on the other hand is typically intensive, concentrating on detailed and highly interactive instruction for a limited segment of course content and a relatively small number of students. Extensive application of CAI has, to date, been limited by its high cost, in terms of both materials production and terminal costs, and by the limited utility of insular segments of individualized instruction in a group-paced environment.

The work reported herein addresses the concept of a comprehensive instructional system in which CAI is embedded in the context of CMI. Such an integrated system has a number of distinct advantages. Student pacing, being individualized, is compatible with efficient use of CAI. Student performance on CAI lessons can be recorded directly by the CMI system. Most importantly, the extensive student performance records maintained as part of CMI can be readily accessed to provide truly individualized CAI, when and where it is most needed.

The relatively expensive CAI must, however, be employed judiciously in the CMI context to be cost effective. If one assumes that the off-line instructional materials being managed by CMI are reasonably effective, the use of CAI is only appropriate when (a) the concepts or facts to be taught are uniquely difficult and existing instructional materials and methods are inadequate for a large proportion of the students, or (b) logistical problems can be resolved through the use of CAI. The cost of producing and delivering CAI for specific applications is, of course, also an important consideration.

Project Purpose

There are two basic reasons why CAI production costs remain high. With few exceptions, the programming languages and production methods employed require extensive and very detailed effort. Secondly, relatively sophisticated instructional design skills are required as well as, in most cases, computer programming skills.

A basic premise of this project was that in the military technical training environment, CAI development problems can be alleviated and production costs reduced through structuring the authoring task and providing software tools to facilitate the authoring process. The term "authoring," as it is used in this report, refers to the process of generating, evaluating, and revising an individualized, interactive CAI module. Such a module consists of information to be assimilated by the student, questions and/or practice exercises, decision rules for individualizing the amount and nature of the instruction on the basis of student performance, and questions or exercises to measure the student's mastery of the module's objectives. It was hypothesized that appropriate software, designed specifically to support CAI authoring in this environment, could directly and substantially reduce the cost of CAI development and evaluation. It was also thought that skill level requirements and production time could be further reduced if the design, development, evaluation and revision tasks were highly structured through the use of specific procedures and software which supported and enforced these procedures.

The software tools developed under this project are intended to support a broad spectrum of tutorial and drill and practice tasks. They would not be as useful, for example, in developing CAI to simulate equipment or processes. Similarly, the authoring system developed is text oriented as opposed to supporting extensive production of computer-generated graphics. Unless sophisticated graphics production software and/or equipment is available (a requirement beyond the scope of this project), the production and delivery of computer graphics is quite expensive. Additionally, when use of supplementary hard copy visuals (e.g., schematics and photographs) is considered, there are relatively few tutorial or drill-and-practice applications in which computer generated graphics can be shown to be cost effective.

Project Context: The Advanced Instructional System

The authoring support software was designed for application in the context of the AIS located at Lowry Air Force Base, Colorado. The AIS is ideal for this purpose since it is a large scale computer-based instruction (CBI) system supported by hardware and software designed to support both CMI and CAI. The system was designed to improve the effectiveness and efficiency of Air Force technical training and to provide an operational research facility for assessing innovations in instructional technology. The system supports four technical training courses representative of the range of cognitive and performance skills required by enlisted Air Force personnel. An adaptive instructional decision model employs state-of-the-art computer hardware, software, statistical methodologies and instructional procedures to provide instructional management and individualized assignments to alternative instructional materials.

AIS Course Structure. Each AIS course is divided into "blocks" of instruction which may require from 1 to 10 days to complete. Each block contains a number of lessons and a comprehensive, end-of-block test. Within a block, lessons are arranged in a hierarchy based on their prerequisite relationships. A typical hierarchy resembles a set of parallel chains diverging and converging on certain pivotal lessons, and a student may alternately work on lessons in two or more parallel chains.

The basic unit of instruction is the lesson. Each lesson consists of a set of objectives, two or more parallel forms of a criterion referenced test, a criterion defining adequate mastery on the test, and, typically a self-test by which students can evaluate their understanding of the lesson before taking the criterion test.

A lesson's instruction is provided by one or more modules, each of which teaches the complete lesson content. When two or more modules are present, they represent alternative instructional treatments. Depending on the lesson content and the nature of the treatment, a module may be a programmed text, an elaborated technical order, an audio-visual presentation, or given the results of this project, an interactive CAI session.

An AIS Student Scenario. A student's first experience with AIS is to complete a preassessment battery consisting of a number of scales which assess cognitive and affective factors considered to be predictive of students' performance in the course. The student then requests an initial assignment by submitting a Forward-Going Assignment request at a management terminal, which consists of an optical scanner and medium speed printer. At this point, the student is enrolled in the course but has not yet entered a block containing actual course content. First, therefore, the system selects the block in which the student is to start work. Since the student has not yet completed any course work, only blocks which have no prerequisites are considered. If there is more than one such block, the one containing the fewest students relative to the desired number in that block is selected. The student is then assigned to an appropriate learning center and home carrel and to a specific lesson, module, and criterion test.

Lesson assignment decisions are made jointly by two major components of the System--the Adapter and the Resource Allocation Model. The Adapter attempts to select, for each assignable lesson, the one module that is most appropriate for that student. This decision can be based on a variety of rules, e.g., select the module which the student is predicted to complete in the shortest time (assuming the student is also predicted to pass the criterion test). Each alternative module is given a weight indicating its relative preference. The Resource Allocation Model assigns preference weights to modules on the basis of minimizing the impact of the assignment on the availability of instructional resources. The final lesson and module selection is based on a compromise between the two sets of preference weights. The form of the criterion test is chosen at random.

The student, after receiving the first assignment printout (called a Student Status Report) at the management terminal, reports to the learning center instructor, obtains the instructional resources required for the assigned module, and begins work, normally at a home carrel.

After studying the lesson materials, the student completes a multiple-choice self-test and reviews the material pertaining to any questions answered incorrectly. The student then completes the lesson criterion test and submits the test form to a management terminal. The resulting Student Status Report details the student's performance on the criterion test (percentage total score, items missed, objectives failed, and pass/fail decision) and the next assignment. If the test criterion was not met, the student is reassigned the same lesson and an alternate form of the test. Otherwise, the lesson, module, and test selection procedures are repeated, and the student is assigned a new lesson.

If the student's assignment is a CAI module, there is only a slight variation in these procedures. The function of the self-test is assumed by questions embedded in the CAI presentation. The criterion test is also administered on-line and the results submitted automatically to the CMI system. The Student Status Report is displayed on the terminal and a printed copy of the report is also available from the management terminal.

When the student has completed all content lessons in the block, a Block Review lesson is assigned. Following review, the student is randomly assigned one of the alternate forms of the block test. While lesson tests can be viewed as diagnostic tools, end-of-block tests serve a certification function. That is, since there is no end-of course test, block test performance serves as the basis for certifying mastery of the objectives contained in the block. A student not meeting the block test criterion is reassigned to the block in a status whereby assignments are made by the instructor rather than by the system. If the block decision is "Go," the block selection logic is repeated, and the student is assigned to the next block of study. The student's continued progress through the course is essentially a repetition of these events.

II AUTHORING SYSTEM DESIGN CONSIDERATIONS

Design of a cost-effective CAI component for the AIS began with an analysis of the AIS environment: the appropriate role of CAI within a CMI system supporting military technical training; the characteristics of the personnel who, it was anticipated, would be developing CAI materials; and the software tools currently available within the AIS. Lessons learned from prior approaches to CAI development were also considered.

The Role of CAI within the AIS

First, it must be recognized that the prototype AIS is primarily a computer-managed instructional system. This is not to imply, however, that the system was not designed to accommodate CAI. The System language, CAMIL (Computer Assisted/Managed Instructional Language), was specifically developed to support both CAI and CMI. Rather, within the context of the AIS, CAI was seen as one of several possible media available for instructional purposes. Management and monitoring of students' progress through a course, assignment to specific instructional treatments, and evaluation of instructional effectiveness are all supported by the CMI component of the system.

Given the nature of the AIS form of CMI, defining the role of CAI within this structure was relatively straightforward. Recall that an AIS course is divided into blocks of instruction which conclude with certification tests. Blocks are divided into lessons, and each lesson is supported by one or more modules, each of which addresses all of the lesson's objectives. When more than one module is available for a lesson, they are treated as alternative instructional treatments for that lesson. The Adaptive Model assures that a student is not assigned a lesson until all prerequisites have been completed. Assignment of a specific module is also a CMI function. Thus, CAI was seen as providing one of two or more alternative instructional treatments for teaching a lesson, and as such, CAI materials would be packaged and assigned as modules. Student terminals required for CAI would be treated as instructional resources managed and assigned by the Adaptive Model.

It was assumed that if a student was assigned and completed a CAI module, it would be desirable that the lesson test also be administered on-line rather than via a management terminal interaction. While the AIS does support an on-line, computer-assisted testing (CAT) capability, it was designed primarily for block tests and was not totally suited for administration of lesson tests. Therefore, a lesson-level testing capability was incorporated into the CAI component.

The next major question concerned the types of applications for which these relatively expensive modules would be most effective. While the use of CAI for normal first-pass instruction was one obvious answer, it was hypothesized that the branching capabilities and moment-to-moment control over student behaviors afforded by CAI would be particularly useful for the functions of review and remediation.

As AIS courses are currently structured, each block ends with a review lesson which is assigned immediately prior to the block test. No specific instructional modules had been developed to support these lessons. Rather, it was intended that students would use this assignment to review those block objectives on which they had encountered problems. The CMI system provides a "Block Report" which flags the objectives which the student failed on the first attempt at the lesson test. Instructional activities during this period are, however, determined primarily by the student and the instructor, and student performance data indicated that this time was often not being used effectively. CAI was seen as an excellent way of remedying this situation.

A procedure was envisioned in which, when a student was being assigned to block review, the Adaptive Model would assess the student's prior performance in the block and, if performance was found to be marginal, assign a CAI module. The CAI module would, in turn, determine the specific objectives on which the student had encountered problems, review the student on these objectives, administer and evaluate objective-level diagnostic tests, provide further remediation as necessary, and issue a Status Report suggesting further review or assignment to the block test as appropriate.

Block remediation presented a similar situation. A student who fails a block test is placed in "block remediation" mode. In this mode, the System accepts any lesson tests input by the student or instructor input of a second attempt block test. The System does not, however, make any specific assignments. This role is delegated to the instructor. To guide the instructor in making appropriate assignments, the objectives which the student failed on the block test are listed on the Student Status Report printed when the block test is scored. Again, student performance data suggested that this remedial time could be employed much more effectively.

CAI block remediation modules were envisioned which would differ only slightly from the review modules. Student assignments to CAI remediation would be made by the instructor, and selection of specific objectives for remediation would be based on the student's block test performance rather than on performance in the block.

Although block review and remediation were seen as two prime targets for CAI, it was assumed that CAI would also be used for alternative modules for first pass instruction. While CAI might occasionally be used for the first module developed for a lesson, as long as CAI costs remain high, it was expected that it would more often be used as an alternative treatment designed to remedy specific problems detected in existing modules.

Anticipated Characteristics of CAI Development Personnel

In considering the characteristics of the personnel who would be developing CAI modules, it was first thought that a team approach, such as advocated by Bunderson (1973), would be appropriate. Such an approach would specify differing roles for (at least) subject matter experts, instructional technologists, and program coders. However, further analysis of the military technical training environment strongly suggested that, while desirable, the team approach was very likely to encounter serious problems in practice.

During the 5 years in which the AIS has been operational, there have been repeated efforts to define specific roles for the specialists who are so important to effective operation of CMI (e.g., materials writers, evaluators, and data base managers). To date, these efforts have had only limited success. It was concluded, therefore, that the CAI authoring system should be structured so as to allow a team approach to CAI development but should not be dependent on it.

For the immediate future, at least, it was reasonable to expect that the personnel who would be developing, evaluating, and revising CAI materials would, for the most part, be classroom instructors. It was assumed that such authors would typically be expert in their subject matter area, would have limited training or experience in instructional systems design, would have no prior exposure to CAI, and would have no computer programming experience. Additional pertinent characteristics included a relatively high turnover rate for military instructors, little or no opportunity for formal training, and limited, fragmented periods of availability (e.g., 60 to 90 minutes following a normal instructional day).

An obvious problem raised by this profile of the typical CAI author is the lack of computer programming expertise and the strong indication that attempting to train relatively transient authors to program would not be practical. Therefore, an approach was sought which would eliminate, or at least substantially reduce, the need for computer programming on the part of the author.

Another implication of the analysis of authoring-personnel characteristics was a need for procedures which would structure the authoring task. At the same time, it was recognized that excessive structuring could be perceived as being undesirable, even offensive, and could consequently detract from the authors' motivation. It was reasoned that they would be more amenable to task structuring once CAI was established and authoring problems had been recognized. It was concluded that the authoring system should provide a degree of task structuring necessary to prevent a novice author from becoming hopelessly bewildered and should also provide for implementing more extensive structuring in the future.

The high turnover rate of military personnel had the obvious implication that the authoring system and procedures should be such that a novice author could quickly be brought to a productive level of proficiency. Further, it was apparent that it would often be necessary for new authors to complete CAI modules that had been designed and partially developed by others.

Coupled with the high turnover rate was the expectation that there would be little or no opportunity for formal training in either the mechanics of developing CAI materials or the instructional principles underlying their design. It was assumed that most training would have to be conducted on the job. Thus, self-instructional methods, with some minimal assistance from experienced authors, appeared necessary. It was also desirable that the software tools themselves be self-documenting and self-instructional.

Since it was anticipated that authors would often have to work in short segmented periods or in an environment subject to frequent interruptions, it was desirable that they have rapid access to the portion of the module on which they were working, that any work accomplished be captured immediately rather than requiring a lengthy storage process, and that the current status of work accomplished be clearly summarized.

Finally, experience gained from various DOD projects indicates that management of instructional materials development is an area that has been generally problem prone because of its complex, unstructured nature. Few effective management procedures have been successfully implemented. Consequently, little monitoring of individual authors during materials development has been possible, especially with respect to productivity and quality control, thus, there was a need for software tools that would facilitate management of the development process. As was the case for structuring the authoring task itself, it was necessary that these management tools be fairly open-ended and allow for further development as management procedures evolved. The review of completed materials by other subject matter experts was known to be particularly time consuming. Software tools which would structure and accelerate this stage were considered especially important.

CAI materials evaluation had potential for problems which were at least equal in severity to those of authoring per se. It was assumed that the CAI authors would have primary responsibility for formative evaluation of their own materials but would have little, if any, prior evaluation experience. As a result, there was a need to structure student performance data collection, retrieval, and reporting. In this case, it was thought that there would be little negative reaction to over-structuring and that the data collection process should be almost totally predetermined and should result in standard reports tailored for formative evaluation.

Software Considerations

The two aspects of the AIS software which most strongly influenced authoring system design decisions were availability of the CAMIL programming language and past experience with the use of interactive data base editors.

The CAMIL Language. CAMIL (Computer Assisted/Managed Instructional Language) (Pflasterer, 1973) is a higher level, general purpose programming language developed as an integral part of the AIS. Several hundred CAMIL programs are currently operational and are used for all on-line phases of the system.

In specifying a language for AIS, it was determined that it would need to be both CAI and CMI oriented and usable by personnel with a wide range of experience. If support requirements had been limited to CAI, a language such as TUTOR (Sherwood, 1974) as implemented at the University of Illinois, could have been considered while traditional languages such as PL/I, FORTRAN and COBOL could have been considered if CMI had been the only target application. Since both applications were to be supported and no known language contained specific capabilities for both, a new language was considered necessary.

To accomplish its diverse goals, CAMIL is divided into two sublanguages: Basic CAMIL and Extended CAMIL. Basic CAMIL is oriented primarily toward systems and applications programming tasks. It provides a full range of standard data types including INTEGER, NUMBER, LOGICAL, CHARACTER, STRING, TEXT, CLASS and SET. The data structuring methods include ARRAY and RECORD types. The flavor of the procedural component is similar to other ALGOL-based languages: procedures are easily defined; the scope of variables is controlled through the use of blocks; and statements can be combined into logically meaningful groups through the use of compound statements.

Extended CAMIL is oriented toward the rapid development of large programs. Its most important feature is the sentence facility which allows coding in natural language-like statements. Also, fewer programmer comments are required because of the level of documentation which the sentence facility provides. By using sentence declaration statements, new sentences can be created by defining new words--nouns, verbs, adverbs, adjectives, and prepositions. These words can be combined in any meaningful way to form a large set of sentences. New data types and operators can also be defined. Sentences, extended data types, and extended operators can then be placed in a CAMIL library. Extended CAMIL Programs can also utilize Basic CAMIL, allowing the user as much control as experience permits.

A total system approach was adopted to make the language reliable, efficient, and easy to use. This system consists of the language, a compiler, loader, interpreter, source program editor, data base file manager, and program archiver. The combination of language, compiler, and source program editor facilitates efficient creation and maintenance of reliable CAMIL programs and reliability is enforced during execution by the interpreter. Programs interface easily with the file manager to rapidly perform the extensive data base management required for CMI applications.

Due to the nature of the CAMIL system, large amounts of error-free code can be written and checked out in a relatively short time. Given the structured nature of the language, the enforced modularity, and the interactive environment, relatively few logic errors are encountered. Once a program has been compiled, it is almost certain to run properly. Finally, the Extended CAMIL sentence facility provides a powerful programming and documentation tool for relatively inexperienced programmers.

<u>AIS Experience with Interactive Editors</u>. While the AIS CMI functions are supported by CAMIL software, the day-to-day operation of the CMI system is controlled through a set of data base editors. The intent of these editors was to allow course personnel with no programming language skills to define the characteristics of their courses and establish the rules by which student assignments are made.

The approach taken was that of a totally table-driven system. At each point where an instructional decision is required, an attempt was made to allow for a variety of decision rules. The indication of which rule was to be used (and in some cases, the rule itself) was then treated as a data item rather than being coded in the software. Thus, both the decision rule and the information to be processed by the rule (typically a combination of student performance data and course characteristic parameters) are considered to be data. With this approach, software changes are only required when the basic operating philosophy of the system is altered. Normal operational changes in course content and configuration, resource inventories, and student assignment selection rules are made by changing the course data base via the interactive data base editors.

Operational experience with this approach has been quite positive. Relatively few software changes have been necessary to meet the system's evolving instructional requirements. Despite the complexity of the data base, course personnel have been able to use the editors to institute data base changes appropriate to their needs. This is not to say, however, that the use of the editors is totally simple and straightforward. The data base and the interactions among its components are complex. In some cases, this complexity was not adequately taken into consideration in the human engineering of the editors. While all of the editors prompt the user's input to some degree, extensive prompting was sometimes sacrificed in favor of effi iency. In retrospect, this trade-off was often inappropriate.

In general, however, the interactive editor approach was thought to hold considerable promise for facilitating at least some aspects of the CAI development process. It was also recognized that where such an approach was adopted, it would be desirable to provide extensive prompting.

Prior Approaches to the Authoring Problem

A growing recognition of the problems associated with CAI development has resulted in a substantial literature addressing these problems and proposing alternative solutions. The following sections provide a brief introduction to the concept of authoring systems, an overview of the types of existing systems, and a summary of two Air Force experiences in CAI development.

Authoring System Considerations and Criteria. Zinn (1974) lists four criteria for assessing the effectiveness and utility of CAI authoring languages: reliability, efficiency, flexibility, and convenience. These same criteria can be applied to the more general concept of authoring systems.

Under reliability, Zinn includes automatic recovery for both author and student following system failure, limiting the loss of authored material and limiting the domain of author errors, i.e., an error in one part of a program should not impact other parts of the program or other programs. Efficiency refers to both the time required for authoring and the computer time required to translate author language statements into executable code. Flexibility considerations include access to a variety of devices, access to alternate modes of execution or conventions and the capability of adding new operators, statements, and subroutines.

Author convenience is treated as a major consideration. Zinn suggests that the language (or system) should have a minimum of redundancy and irrelevant syntax, e.g., the program listing should be no more complex than the author's actual task. There should be provision for alternative authoring styles, e.g., while many authors indicate a preference for interactive entry, others prefer to work with paper forms which remind the author of system capabilities and requirements. With respect to revising an existing program, Zinn notes the advantages of on-line editing and suggests that the system should provide access to the original file, use straightforward notation for determining changes to be made, and confirm that changes were accomplished. In testing a program, the author should be able to begin execution at any point and trace through the program using labels as indicators of location. The language notation should help a reviewer understand the intent of the instructional content and strategy. Finally, access to system capabilities should increase with experience.

Kaplow (1975) states that in order to maximize assistance to the author, it is not sufficient to simply add authoring aids to a programming language. Rather, the total CAI system must be organized around this goal. He then describes what he considers to be the basic features of such a system.

First, the system should provide a structured format to help authors organize their concepts. The structural units should match the author's conceptual units and impose a degree of modularity. The current working unit should always be identified and the author's statements should refer only to this unit. Kaplow suggests that the operational aspects of the authoring and CAI delivery components should be separated. A CAI program should be treated as a data base containing the content and structure of the program as well as the computer commands to be executed. The system design should make it explicit that the program is a collection of information organized so as to be amenable to understanding. The computer itself should automatically perform many programming functions, such as checking structural completeness and finding cross-reference errors. Finally, the system should not require that a program be complete before it can be tried out.

In his subsequent discussion, Kaplow makes a number of additional critical points. The system should be tolerant of user errors and pinpoint errors at the time they are made. The detailed actions to be taken at student run time should be defined on the basis of the implications of the author's instructions rather than having to be spelled out. Given a modular program structure, the system should help the author keep track of the interrelationships between the various parts. The fact that it is often easier to write a new program rather than to modify an existing one is particularly unfortunate considering the opportunity, even requirement, for CAI revisions based on student performance. Since the source of this problem is usually one programmer's difficulty in understanding the structure and logic used by another, the system should place particular emphasis on the ease with which one can build on existing material.

Example Authoring Systems. The earliest CAI programs were written in the available general purpose languages. Although this is still a popular approach (e.g., the extensive use of BASIC), there was early recognition that authoring cou be facilitated by languages tailored to the particular requirements of CAI. Consequently, there was a proliferation of CAI authoring languages--at least 30 by 1973. The last decade has also seen experimentation with various author entry systems--approaches which are relatively independent of a specific language. The following paragraphs briefly describe an example of a modern CAI authoring language (TUTOR) and some approaches to developing authoring systems.

TUTOR (Sherwood, 1974) needs to be considered in the context of the complete PLATO (Programming Logic for Training Operations) system. The system provides for interactive entry, easy trial and revision of code, and is quite responsive to authors' needs in terms of display time, compilation time, and diagnostics. The author's task is facilitated by a number of aids, such as on-line access to reference materials, sample program routines, and files of current documentation and comments. To a large extent, the PLATO approach to authoring is based on the model of an author who is a versatile professor, an expert in the subject matter, an experienced teacher with sound but innovative ideas about instructional presentation, and a capable programmer. Within the university environment, this approach has resulted in a large number of excellent CAI lessons. In other environments, where the authors have been less experienced, less skilled, and/or less motivated, the approach has not proven as satisfactory.

Dowsey (1974) describes five categories of approaches to building easy author-entry systems: separation of logic and content, avoidance of any authoring language, use of lesson planning guides, conversational materials generation, and macro systems. The approaches in each category tend to build on the concepts of the prior categories.

Almost all of these approaches employ the tactic of separating instructional content from program logic. This divides the authoring task in a way that is particularly amenable to a team approach. Dowsey notes that to be effective, such separation requires similarity of structure between the two components.

The no-author-language approach not only separates content and logic but does not require the author to define the logic. Only the content is specified, in the form of frames or problem categories. This is then acted upon by a lesson generation program which treats the content as data to produce instructional materials.

The use of lesson planning guides, while requiring the services of a coder, permits the author to communicate with the computer in English. Typically, the author defines the material to be presented, questions, expected answers, and corresponding courses of action on a standard form which specifies the categories of information required. One of the more sophisticated examples of this approach is Dowsey's own COURSEMAKER, designed for use with the COURSEWRITER III language. It is based on use of a paper form which includes a presentation section containing the material to be displayed, a question section indicating the student response(s) expected, a decision section defining whether a branch is to be taken (and if so, the type of branch), and an analysis section containing response judging rules.

Conversational materials generation represents a quite different approach. The interactive system assists the author by eliciting the content and logical structure of the lesson through a natural language conversation. Paloian (1974) describes one example of this approach. The author defines the program structure by entering a sequence of action verbs. The system checks the accuracy of the sequence and, if it is correct, prompts the author to supply the text, anticipated responses, counter names, etc.

Dowsey sees the use of macro routines as potentially being the most powerful approach. In using such a system, the author retrieves a preprogrammed sequence of code and specifies arguments which complete the routine. One characteristic of this approach is that it imposes a definite structure on the material produced. The Time-shared, Interactive, Computer-Controlled, Information Television (TICCIT) system (Stetton, Volk & Bunderson, 1973) is probably the best known example of a macro system. TICCIT employs a single instructional strategy, oriented toward concept learning, which assumes learner control over sequence. The strategy specifies how the subject matter should be structured (e.g., rules, examples, practice items) and even suggests the appropriate number of items in each category. Thus, much of the author's task consists of molding the content to fit the strategy. Planning guides are used to format the content and define presentation tactics by selecting among available options. A macro processor then converts this information (content, content format, and strategy options) into computer language.

One other macro-oriented approach which deserves mention is the use of Monoforms (Schulz, 1975) developed for use by military authors on PLATO IV. The Monoform macros, written in TUTOR, were intended to facilitate preparation of frequently used question types by providing a question format and eliminating the need for the author to understand TUTOR. A total of nine macros were developed for multiple choice, constructed response, and matching questions. Within each question type, the Monoforms differ with respect to variations in format, type of feedback and (for multiple choice questions), order of alternative presentation. There are also a number of options within each Monoform. The author copies the desired Monoform and follows the instructions (supplied in the form of program comments) to supply the question content and tailor the TUTOR commands to his/her specific requirements. Schulz reports that use of Monoforms reduced the 2 to 6 hour question development time to only 10 to 15 minutes.

<u>Air Force CAI Authoring Experience</u>. Himwich (1977) reports a comparison of TICCIT and PLATO authoring efficiency conducted at Maxwell AFB. The results of the comparison were inconclusive with little difference found between the two approaches. In the production of 32 student contact hours of CAI, the PLATO team required an average of 222 manhours per hour while the TICCIT team required 246. Of greater interest is Himwich's description of the procedures followed and problems encountered, particularly by the PLATO team. The training provided for PLATO team members consisted of an intensive 2 week session and subsequent continuous consulting support by the PLATO staff. TICCIT team training began with 7 weeks of familiarization followed, some months later, by 3 weeks of intensive training. In both cases, the training required appears excessive to the authors of this report. Further,

Himwich reports that in many instances the authors did not capitalize on PLATO's flexibility with respect to instructional strategies and, consequently, did not demonstrate the system's full capabilities.

Dallman, DeLeo, Main and Gillman (1977) provide a comprehensive description of an experiment in the use of PLATO IV for Air Force technical training. At the beginning of the test, conducted at Chanute AFB, the typical PLATO authoring model was adopted with each author acting independently. Learning TUTOR was found to occupy a major portion of authors' time. Authors had varying styles and quality control standards, and as a result, the curriculum was fragmented with little continuity between lessons. Dallman et al. concluded that a basic flaw in this approach was the unrealistic assumption that all materials authors were experts in both subject matter and instructional practices.

A team approach was subsequently adopted in which the team consisted of an author, subject matter expert, instructional programmer, and coder. Although this was a distinct improvement over the prior approach, a number of problems were still encountered. No written procedures were defined, only informal understandings among team members. This resulted in time consuming coordination problems and inefficient use of team specialists. Administrative and management procedures were never well defined and there was a continuous need, never resolved, for better, more extensive author training.

A number of PLATO features were found to be quite useful. On-line data collection routines supplied by the PLATO staff and the capability for on-line text editing were both considered important. TUTOR was adequate for the site's needs with authors handling the simpler aspects of programming and coders required for only the more complex portions. Only a few of the more experienced authors, however, capitalized on PLATO's instructional flexibility. Almost all of the lessons produced employed the same simple tutorial model. The report authors suggest that this approach was followed because the materials were easy to prepare, the subject matter was not that complex, and student comprehension and retention requirements were low. Branching was used mainly for forced review and TUTOR's response judging capability was seldom utilized. Only about 20 percent of the questions developed employed a constructed response format. Not only were constructed response questions more time consuming to code, students disliked them because of unfamiliarity with the typewriter keyboard.

Dallman et al. drew a number of conclusions relevant to the current project. The team process was found to be more efficient and effective than individual authors. Authors did not exploit PLATO's full capabilities due to resource constraints, lack of CAI expertise, and inadequate training in instructional programming techniques. Finally, sophisticated CAI capabilities may not be necessary for effective learning of the type of tasks and level of knowledge required for adequate military technical training.

The types of problems encountered at Maxwell and Chanute appear typical for military technical training. Kimberlin (1977), describing the status of Project ABACUS at Ft. Gordon, reports 5 to 6 month slips in the full implementation of CAI courses. The major problems were reported to be changes in the Plans of Instruction during CAI development and the fact that the project was never assigned an adequate number of instructional programmers.

Design Conclusions

Although many aspects of the military technical training environment are not amenable to efficient development of instructional materials, particularly CAI, it was concluded that it was preferable, at this time, to adapt the system to the environment rather than to develop an authoring system which assumed a more responsive environment. It is hoped that the authoring tools would themselves act as change agents. For example, it was decided that the authoring system should be designed so as to promote the concept of an authoring team but not to rely on its existence. Given a trade-off between authoring flexibility and a structured presentation format, structure was almost always selected. Difficulties in assuring adequate author training was accepted as a major problem. Despite the fact that many features of CAMIL were designed expressly for this purpose, it was recognized that any approach which relied on an author/programmer had little chance of success. Further, it was considered unlikely that personnel would be assigned to such a specialized function as CAI computer programming in sufficient numbers to adequately support many authors.

This assessment suggested either a no-author-language approach or a macro system approach. The former was considered to be too rigid for the evolving computer-based technical training environment. A macro system modeled on the TICCIT approach was rejected because a simpler authoring process was desired; the same objection was encountered for approaches similar to Monoforms. The adopted design combined the best aspects of several prior approaches, relied heavily on characteristics of the existing CAMIL system, and capitalized on prior AIS experience with interactive data base editors. As suggested by Kaplow (1975), the total system was designed with the major goal of facilitating CAI materials development.

To alleviate the need for computer programming, the approach adopted was to provide a small number of flexible "template" CAI presentation programs, each of which would support a class of CAI modules. While this approach still requires the services of a computer programmer to expand, the extent of these services was reduced in two ways. First, the template programs were designed to be as flexible as possible without being unduly complex. Principles which had been employed in developing the AIS Adaptive Model and its data base were used to assure that a single program structure would support a variety of modules addressing different topics. Second, given the characteristics of the

CAMIL language and a template program which was well structured and documented, it was anticipated that a relatively inexperienced programmer could modify the basic program to meet the requirements of new applications.

The major emphasis was placed on designing an interactive authoring editor by means of which an author could define CAI content and branching logic. The design goals were to provide authors with a variety of options within a structured framework, to appropriately guide and prompt authors' decisions with respect to these options so as to minimize the need for formal training, to automatically accomplish as much of the programming detail as feasible, and to eliminate any need for authors to be aware of the computer language. The format of the authors' input was to be as similar as possible to what students would actually see. Author errors were to be detected and identified at the time they were committed. Standard routines and reports were to be provided for student performance data collection and analysis. Finally, the system was to provide convenient access to meaningful information which would assist in managing CAI development.

III APPROACH

The approach taken to supporting efficient CAI production in the technical training environment centered around development of a CAI Authoring Editor and a template Presentation Program. Student performance data acquisition routines were built into the presentation program and reports developed which focus on formative evaluation requirements. A CAI Materials Print Program was developed to provide hard copy listings of CAI materials for author and student use. All but one of the supporting programs were written in CAMIL. The Editor and Presentation Program were designed to operate on the current AIS interactive terminals, which are a modification of the PLATO terminal, but provisions were made for easy transition to less expensive terminals. The Authoring Editor, Presentation Program, Data Acquisition and Analysis Programs, the Materials Print Program, and documentation of the authoring system are described in detail in the following sections.

The CAI Authoring Editor

The heart of the software supporting the AIS CAI authoring system is an interactive Authoring Editor. To introduce the Editor, the structure and characteristics of the CAI modules produced with it are first described. The Editor itself is then discussed, beginning with various summary displays which provide central reference points for the author's work. The mechanics of generating text and question frames and of defining branching logic are then described and illustrated in some detail.

Module Structure, Frame Types and Frame Flags. As structured by the Authoring Editor, a CAI module is segmented into objectives. Through the Authoring Editor, the author can define only the sequence in which objectives are to be presented. Any branching between objectives is a function of the Presentation Program. Each module must include an Objective 0 that contains material (usually an overview) pertaining to the lesson as a whole. The author may then define up to 100 numbered objectives containing the module's instructional content. Each objective can contain up to 100 frames. Ten frame types have been defined which can be classified into three categories: textual content frames; question frames; and special purpose frames. New types can be defined as requirements arise. The frame types in each category are described briefly below.

There are six different types of textual content frames. These are all similar in that they simply present textual information to the student and require no response other than an indication that the student is ready to proceed. The six types are differentiated primarily for management reasons. In some cases, a specific frame type may be required at a certain point in the frame sequence and listing frames by type provides a quick overview of the instructional sequence.

The basic "Text" frame is, as its name implies, the primary medium for presenting instructional content. A Text frame can contain up to four "pages" or screen displays. There is no provision for branching within frames, i.e., between pages. Once a frame has been selected, all of the pages in that frame are presented. Authors are instructed that a Text frame should present a discrete chunk of information pertaining to a single concept.

The second textual frame type is the Elaboration frame. It differs from a Text frame only by name and intended function--to present information to students whose performance indicates that they need further, more detailed explanation of a concept presented in a Text frame. The Title frame, when used for the purpose implied by its name, can be recognized as a dividing point in the frame sequence. The Statement of Objective frame is required at the start of each objective other than Objective 0. Similarly, an Overview frame is mandatory within Objective 0. Finally, a Materials List frame, listing any reference materials which the student needs to complete the module, is required in Objective 0. Any of the mandatory frame types can also be used, at the author's discretion, at other points in the module.

The Editor currently supports two types of Question frames: multiple choice and constructed response. These frame types provide the primary means of evaluating students' understanding of the material being presented. Question frames consist of the question stem, up to five multiple-choice alternatives or nine anticipated constructed response alternatives, feedback statements for each alternative and prompt statements associated with successive attempts to answer the question.

There are two special purpose frame types, neither of which is presented to students. The first is Documentation frame required at the beginning of each objective to provide a means of documenting the authoring/evaluation/revision process. The author is instructed to provide information such as the dates during which the module was being developed or revised, the names of additional authors who have worked on the module, the learning strategies employed, the nature of and reason for any revisions, etc. It is anticipated that the information requested will become more structured as a function of experience. The second special purpose frame type is the "Branching Decision" frame. It is a dummy frame, containing no material, which allows for a branching point without anything being presented on the screen.

There are eight varieties of "flags" which can be set against a frame to indicate to the CAI Presentation Program that a particular action is to be taken when that frame is encountered. Like the frame types, flags have one to two character code names which, when displayed on the frame listing, contribute to providing a summary of the instructional strategy employed.

Four of the flags (Objective Passed, Objective Failed, Lesson

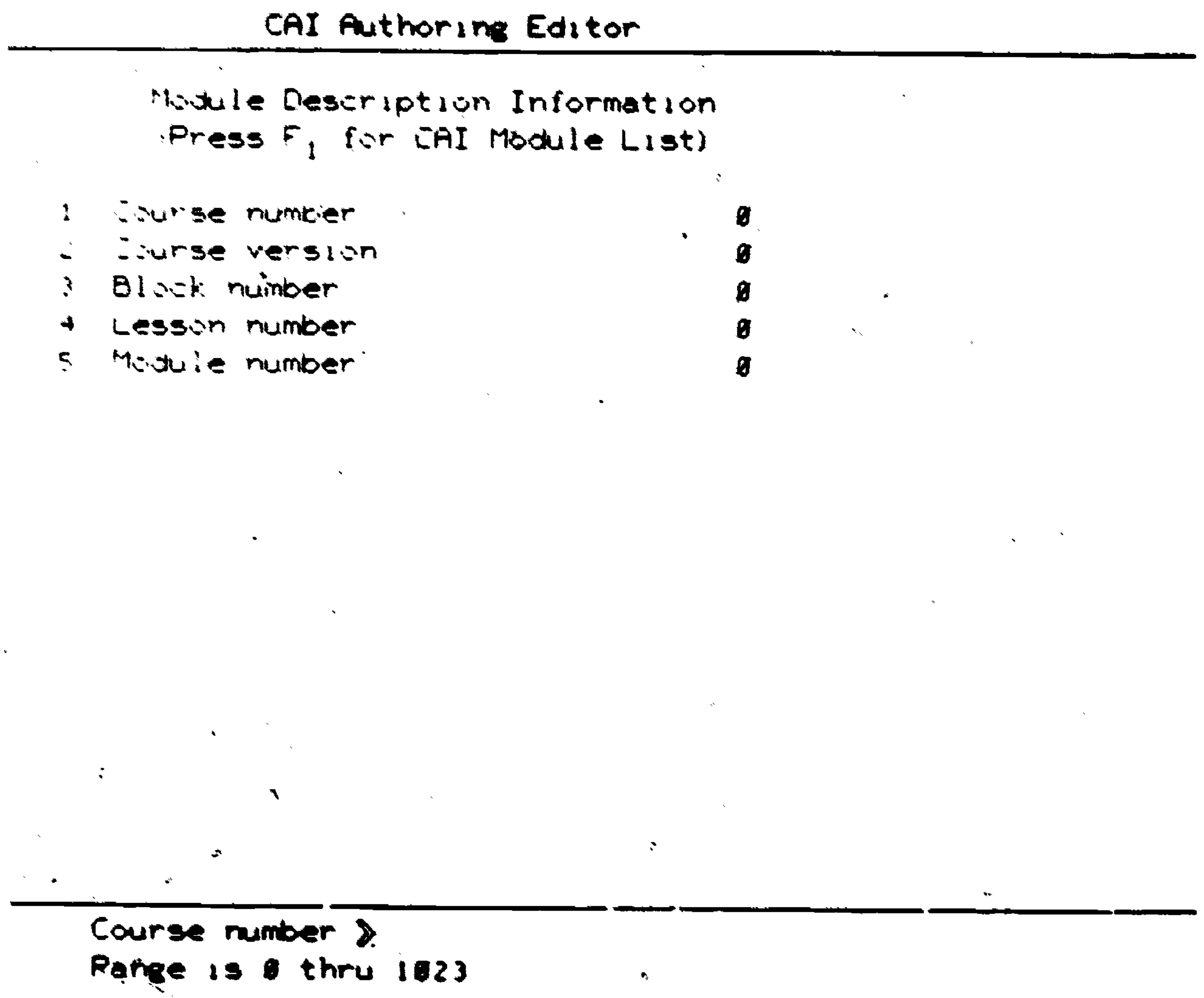
Passed, and Lesson Failed) notify the Presentation Program that, if this frame is encountered due to author-defined branching, the student has or has not completed the objective or lesson satisfactorily. The Skip This Frame flag indicates that the frame, even when encountered, is not to be presented to students. This allows the author to store alternative material in the body of the module itself. The Fix Answer List flag, which can only be set against a multiple-choice question frame, indicates that the list of alternatives is to be presented in the order input by the author rather than being randomized. A Student Break flag elicits a display generated by the presentation program suggesting that the student take a short break from working on the module. It allows the author to alleviate student fatigue on long, complex modules. Finally, the Decision Point flag indicates that the presentation program is to collect Decision Point student performance data following presentation of that frame.

Module, Objective and Frame Summary Displays. Throughout the Editor, a philosophy was adopted of providing extensive prompting and "fail-safe" mechanisms. Each author entry is prompted to the extent possible. Commands which will result in the deletion of materials or branching logic require a second "Yes, I want to do it," response on the part of the author and it is always possible to back out of an error situation without the Editor taking any action.

Whether an author wishes to create a new CAI module, revise an existing module or display a module, the same basic procedures apply. After accessing the Authoring Editor from an interactive terminal, the author encounters a display, illustrated in Figure 1, requesting the sequence of numbers (Course Number through Module Number) identifying the particular module to be accessed or created. Entry of each value is prompted and the range of allowable values is specified. The author may also request a list of existing modules (by pressing Function Key F1). If this option is selected, the author is asked whether all modules or only those belonging to a particular author are to be listed and whether the list is to be displayed on the terminal or printed on the central line printer.

If the identifier of an existing module is entered, the author is asked whether that module is to be displayed, changed (revised or added to), or deleted. Whereas any module can be displayed, only those created by that author can be changed or deleted. If the author enters the identifier of a module which does not exist, the author is asked whether a new module is to be created, and if so, whether the module is to be a copy of an existing module. This copy feature allows an author to revise a module which has already been implemented in the classroom, to revise another author's module, etc. If the new module is to be created from scratch, the author is required to enter the title of the module and the number of objectives which it is to contain.

If the author is creating a new module or revising an existing



```
                 CAI Authoring Editor
        ------------------------------------------
           Module Description Information
            (Press F1 for CAI Module List)

   1  Course number                       0
   2  Course version                      0
   3  Block number                        0
   4  Lesson number                       0
   5  Module number                       0

        ------------------------------------------
    Course number >
    Range is 0 thru 1023
```

Figure 1. Authoring Editor Module Selection Display.

module, the next display asks which data collection routines are to be activated: response data, decision point data, and/or student comments. None, any, or all of the three types can be selected.

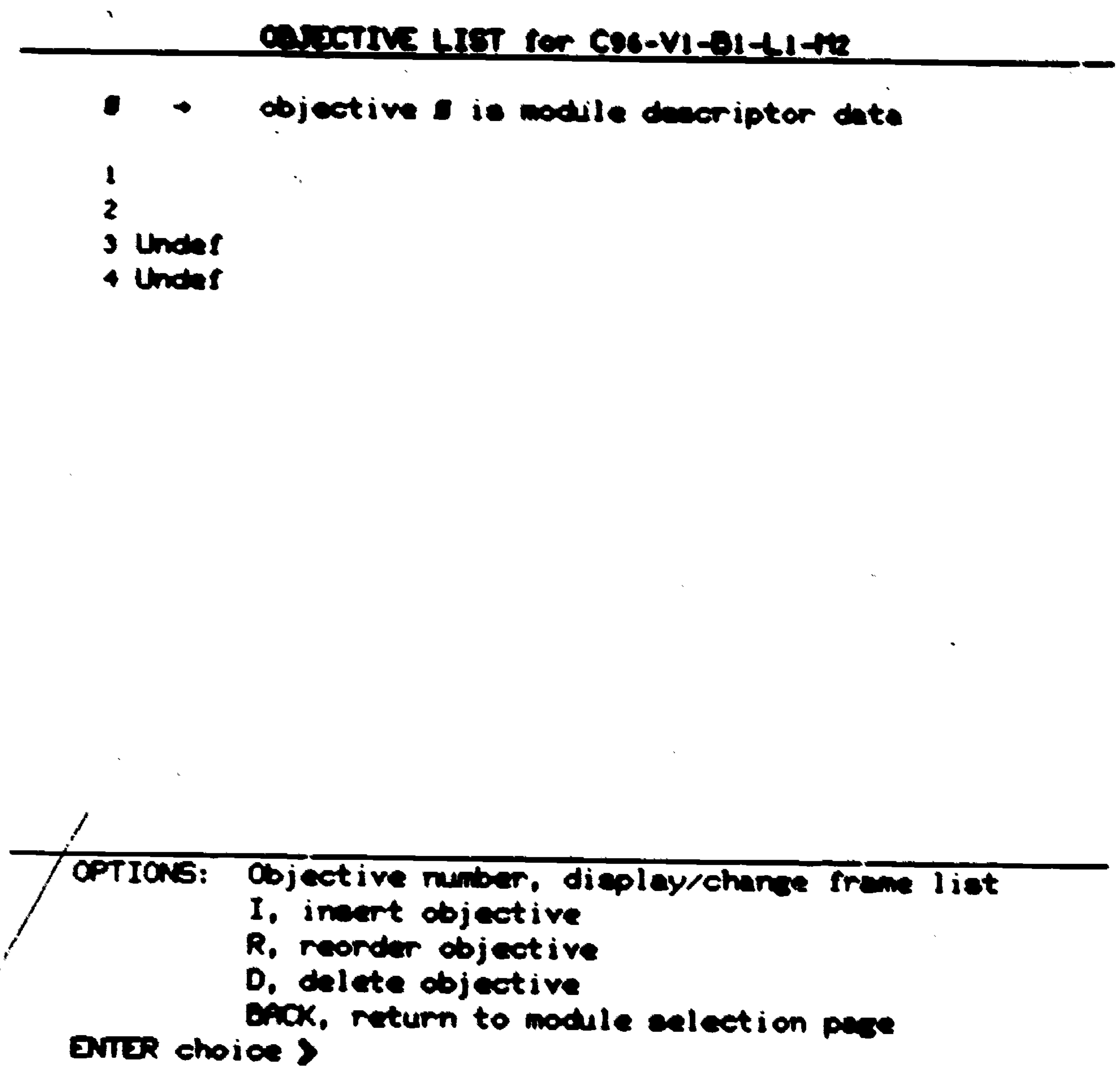
The author next encounters the Objective List display illustrated in Figure 2. This display provides a central reference point for the author--a top level overview of the module's content and the control point for creating, changing, or listing complete objectives. Objectives against which no material has been entered are shown as being undefined (Undef). For the display illustrated in Figure 2, the module includes the mandatory Objective 0 and four additional objectives, two of which are as yet undefined.

Actions which can be taken at this point are listed as Options at the bottom of the display. If the author enters the number of an objective, the list of the frames defined for that objective is displayed. One can also add or delete an objective, reorder the sequence in which the objectives are to be presented, or request a printer listing of one or more objectives. As is the case for all of the Authoring Editor displays, the author can back out without taking any action by pressing the "BACK" key.

If a new module is being created, all of the objectives would be shown as undefined, and the author would first be required to define Documentation, Overview, and Materials List frames for Objective 0. The author could then define additional Objective 0 frames or return to the Objective List display and proceed to define content and branching within the individual, numbered objectives.

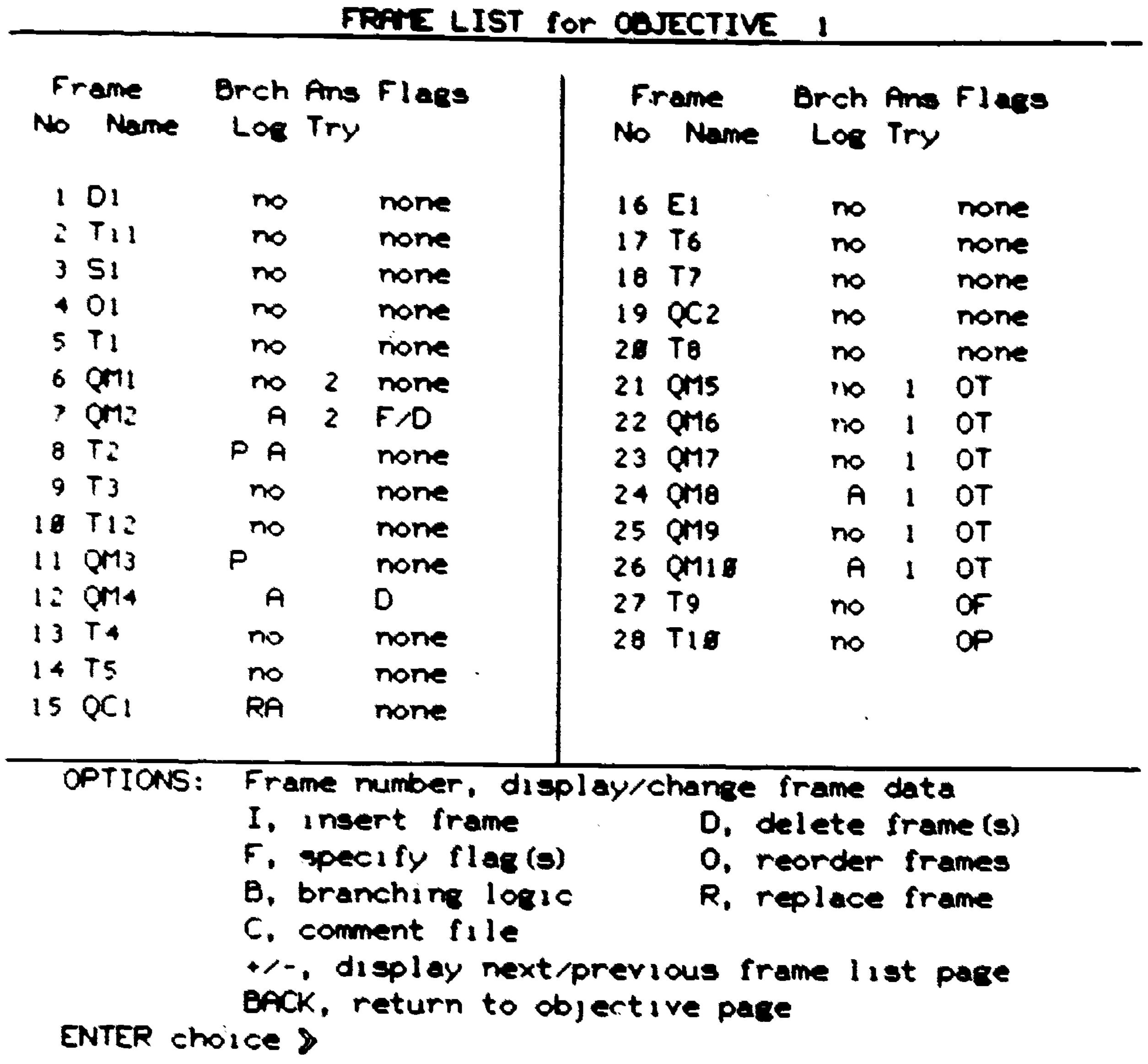
When the author accesses a specific objective, the first display encountered is the Frame List illustrated in Figure 3. The Frame List provides a central point for work within an objective. The list defines the sequence and types of frames comprising the objective and provides an overview of the branching logic and any special conditions imposed on specific frames. Actions which can be taken are listed as Options at the bottom of the display.

A Frame List consists of one to four pages with up to 30 frames listed per page. The objective represented by the Frame List shown in Figure 3 is relatively short, only 23 frames. The display provides a substantial amount of information about the content and instructional strategy employed in this objective. Each frame has both a number (1 through 23) and a name consisting of one or two letters (e.g., D for Documentation, S for Statement of Objective) and one or more digits indicating whether this was the first, second, etc. frame of this type to be defined. If there is branching logic associated with a frame, the class of logic (Pre-frame, After-frame, or Response contingent) is indicated, and limits on the number of attempts allowed to answer questions are shown, as are Frame Flags.



```
            OBJECTIVE LIST for C96-V1-B1-L1-M2

#   ->    objective # is module descriptor data

1
2
3 Undef
4 Undef



OPTIONS:  Objective number, display/change frame list
          I, insert objective
          R, reorder objective
          D, delete objective
          BACK, return to module selection page
ENTER choice >
```

Figure 2. Authoring Editor Objective List Display.



```
                        FRAME LIST for OBJECTIVE  1
 Frame      Brch Ans Flags          |   Frame      Brch Ans Flags
No  Name    Log  Try                |  No  Name    Log  Try
                                    |
 1  D1       no       none          |  16  E1       no       none
 2  T11      no       none          |  17  T6       no       none
 3  S1       no       none          |  18  T7       no       none
 4  O1       no       none          |  19  QC2      no       none
 5  T1       no       none          |  2Ø  T8       no       none
 6  QM1      no   2   none          |  21  QM5      no   1   OT
 7  QM2       A   2   F/D           |  22  QM6      no   1   OT
 8  T2      P A       none          |  23  QM7      no   1   OT
 9  T3       no       none          |  24  QM8       A   1   OT
1Ø  T12      no       none          |  25  QM9      no   1   OT
11  QM3     P         none          |  26  QM1Ø      A   1   OT
12  QM4       A       D             |  27  T9       no       OF
13  T4       no       none          |  28  T1Ø      no       OP
14  T5       no       none          |
15  QC1      RA       none          |

 OPTIONS:  Frame number, display/change frame data
           I, insert frame           D, delete frame(s)
           F, specify flag(s)        O, reorder frames
           B, branching logic        R, replace frame
           C, comment file
           +/-, display next/previous frame list page
           BACK, return to objective page
 ENTER choice ❯
```

Figure 3. Authoring Editor Frame List Display Showing Set of Initial Options.

The objective illustrated in Figure 3 begins with the required Documentation frame (D1), a Statement of Objective (S1), a Title (Ti1) and an Overview frame (O1). Following a single Text frame (T1), the author tests the student on a prerequisite concept, using two multiple choice questions (M1 and M2). Only two attempts are allowed to answer each question. The second question is followed by (After-frame) branching logic which routes the student around a review of the concept (frames 7 through 12) if performance on the two questions is adequate. Only the existence of the logic is shown, not its detail. The review consists of three Text frames and two more multiple choice questions. Note that the name of frame number 10 is T12. This indicates that the frame was added later, after frame T11 had been defined. The frame 12 After-frame logic determines whether the student has now mastered the concept. If not, the student is returned to frame number 8 and to review the concept a second time. When frame number 11 is encountered the second time, the Pre-frame logic bypasses the multiple choice questions and routes the student to frame number 13. To determine how many students are being routed through these review frames and the time required for review, the author has defined Decision Point Flags (D) prior to (frame number 7) and following (frame number 12) the review sequence.

The main body of instructional content is presented by frame numbers 13 through 20, consisting of Text, Elaboration (E), and constructed response (CR) frames. Since no specific limit has been placed on the number of attempts for answering these questions, the default value of three attempts will be allowed. Response Contingent branching logic has been defined for frame CR1. If the response matches a particular anticipated incorrect response, the student will be routed to Elaboration frame E1 designed to correct the misconception. Otherwise, frame E1 is skipped via the CR1 After-frame branch.

Mastery of the objective is evaluated by six multiple choice questions (frame numbers 21 through 26). Only one attempt is allowed on each question. The criterion for passing the objective is four out of six correct, and the author has defined After-frame branching logic following M8 to skip the last two questions if the first four were all correct. Otherwise, the student's performance is evaluated following frame M10. If the criterion was not met, frame T9 is presented to inform the student of this fact. The Frame Flag (OF) indicates the occurrence of an objective failure to the presentation program. If the criterion was met, after either four or six questions, frame T10 is presented which congratulates the student and the OP Flag notifies the Presentation Program that the objective has been passed.

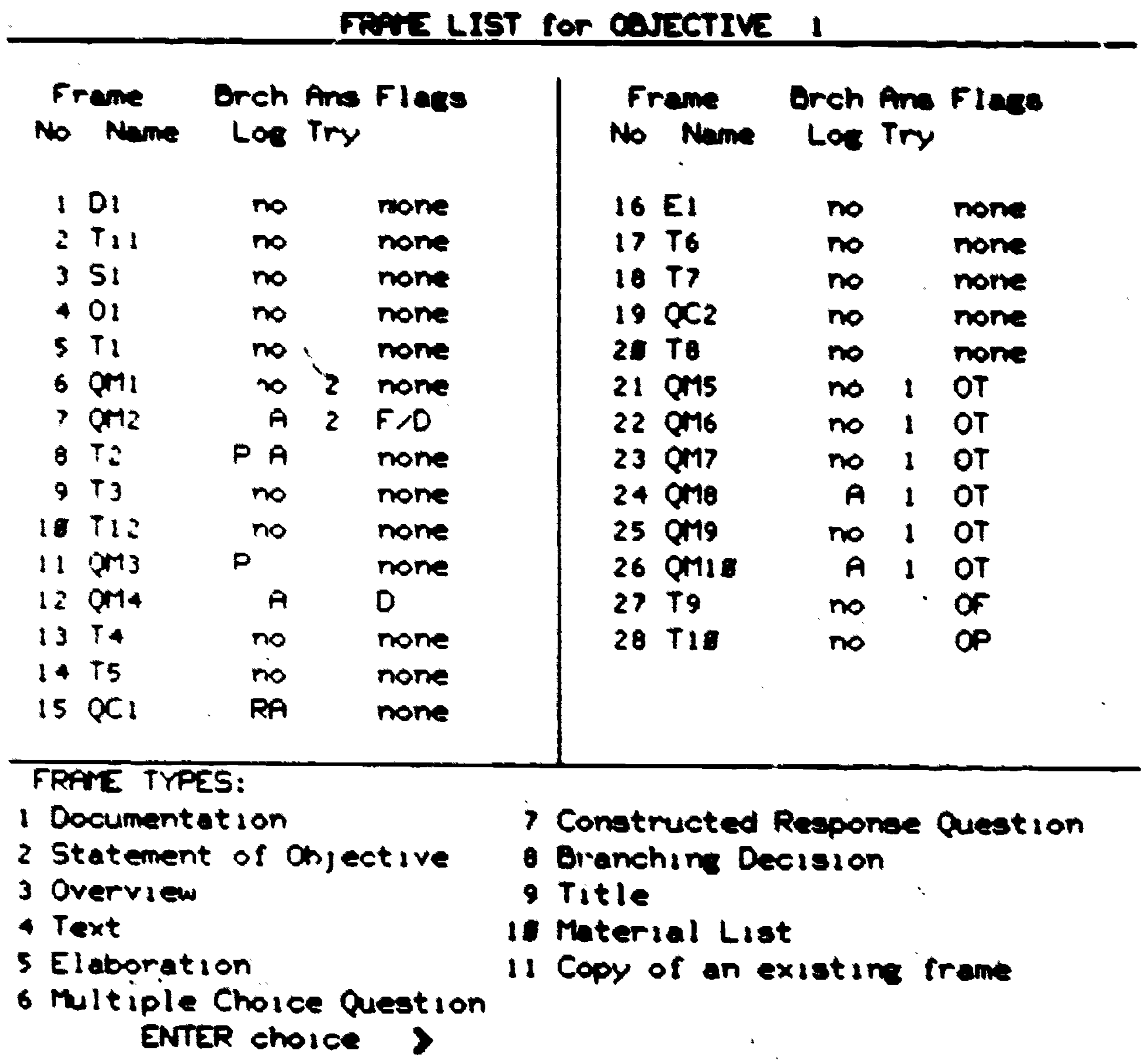
The actions which the author can take are listed as Options at the bottom of the display (see Figure 3). Entry of a specific frame number accesses that frame for the author's review or revision. Option I (Insert) allows the author to define a new frame and insert it at any point in the frame sequence. If the Insert Option is selected, the author is asked to provide the number of the frame after which the new frame is to be inserted and to select the Type of the new frame. The Frame Type selection menu is shown in Figure 4. Once Frame Type has been selected, the author is placed in the frame creation mode for that type. Note that Option 11 on the Frame Types menu allows the author to copy an existing frame. Any frame, including one from another author's program can be copied. Further, the author has the choice of making an actual copy of the frame, which can then be modified to suit the author's purposes, or of simply "aliasing" an existing frame. With the latter option, the frame is not actually copied but rather, when an "alias" frame is encountered by the Presentation Program, the frame content is retrieved from its original location for display.

The third initial Frame List Option, Specify Flags (F), allows the author to set the previously discussed Frame Flags against specific frames. Specify Branching Logic (B), permits the author to define branching logic for specific frames. Option C, Comment File, accesses the list of student comments which have been made against specific frames. The existence of such comments is indicated by the occurrence of a "C" in the Flags column of the Frame List display. The Frame List may consist of up to four pages, and the control characters "+" and "-" access display of the next or previous page of the List, respectively. The Delete Frame (D) Option allows deletion of one or more frames which are no longer required. The Reorder Frames (O) Option permits the author to move frames to different positions in the frame sequence. If the author decides to replace all of a frame's content, the Replace Frame (R) Option is available. Finally, the author can always return to the Objective List display by pressing the BACK key.

Text Generation. The process by which all six types of textual content frames are produced and edited is essentially the same. When the author elects to create one of these frame types via the Frame Types menu (see Figure 4), a blank template, such as is illustrated in Figure 5, is displayed. Note that the frame number, frame name, and number of this page within the frame are shown at the top of the display as is the fact that the author is in Insert mode and has already entered one line. Typed content appears on the line next to the arrow-shaped cursor, and the NEXT key is pressed to end each line. The author is free to enter the content in any format within the constraints of the margins, indicated by the cursor on the left and the vertical line on the right. Each page can contain up to 21 lines of 50 characters each.

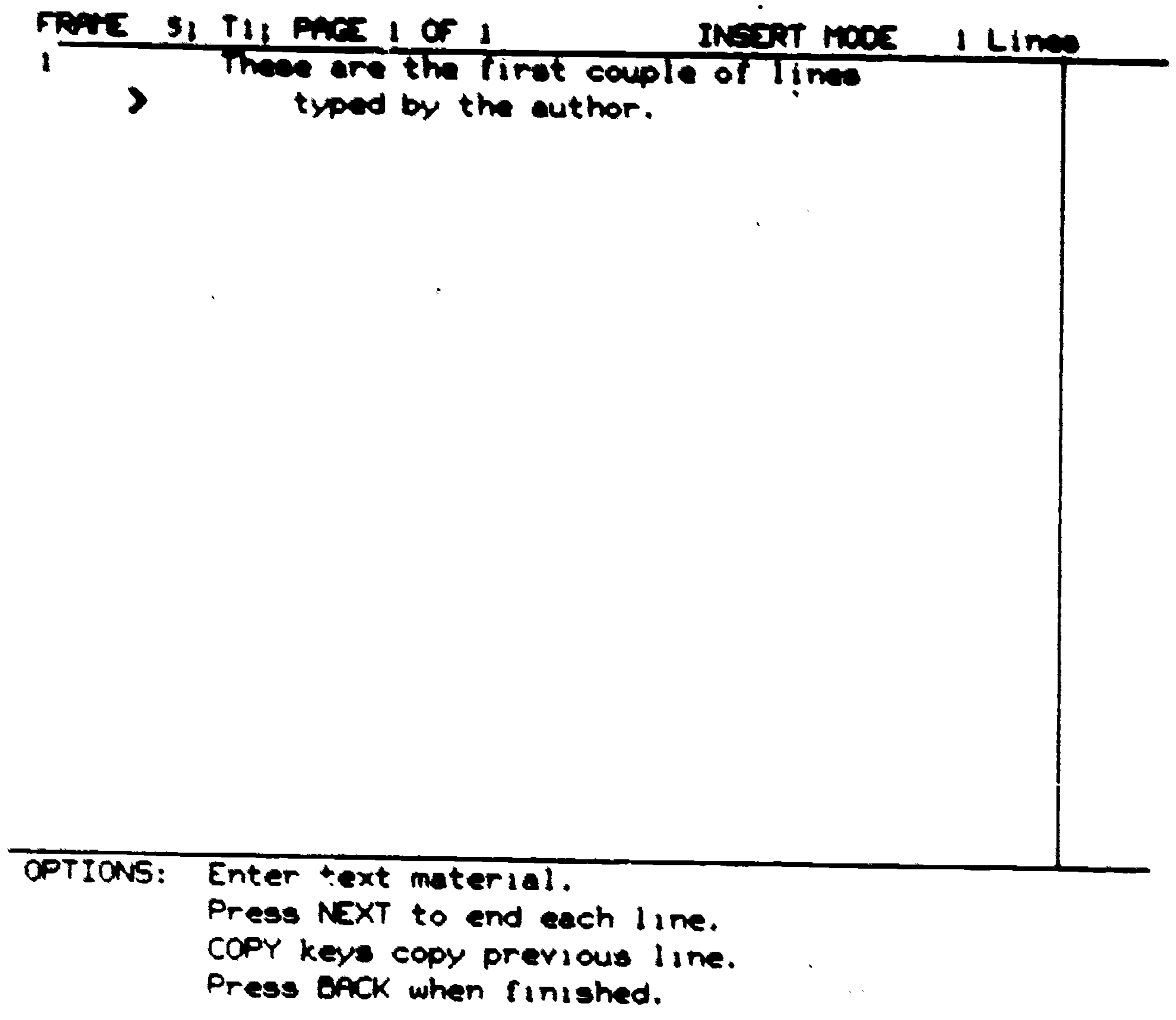
After entering one or more characters, the author can press BACK to access a collection of editing functions. This situation is illustrated in Figure 6 and is the same as the display which would be presented if the author accessed an existing Text frame to modify it. The Editing Options are listed at the bottom of the display.

The Insert Line(s) (I) Option allows the author to define additional lines of text following or between lines of existing text. While insert-



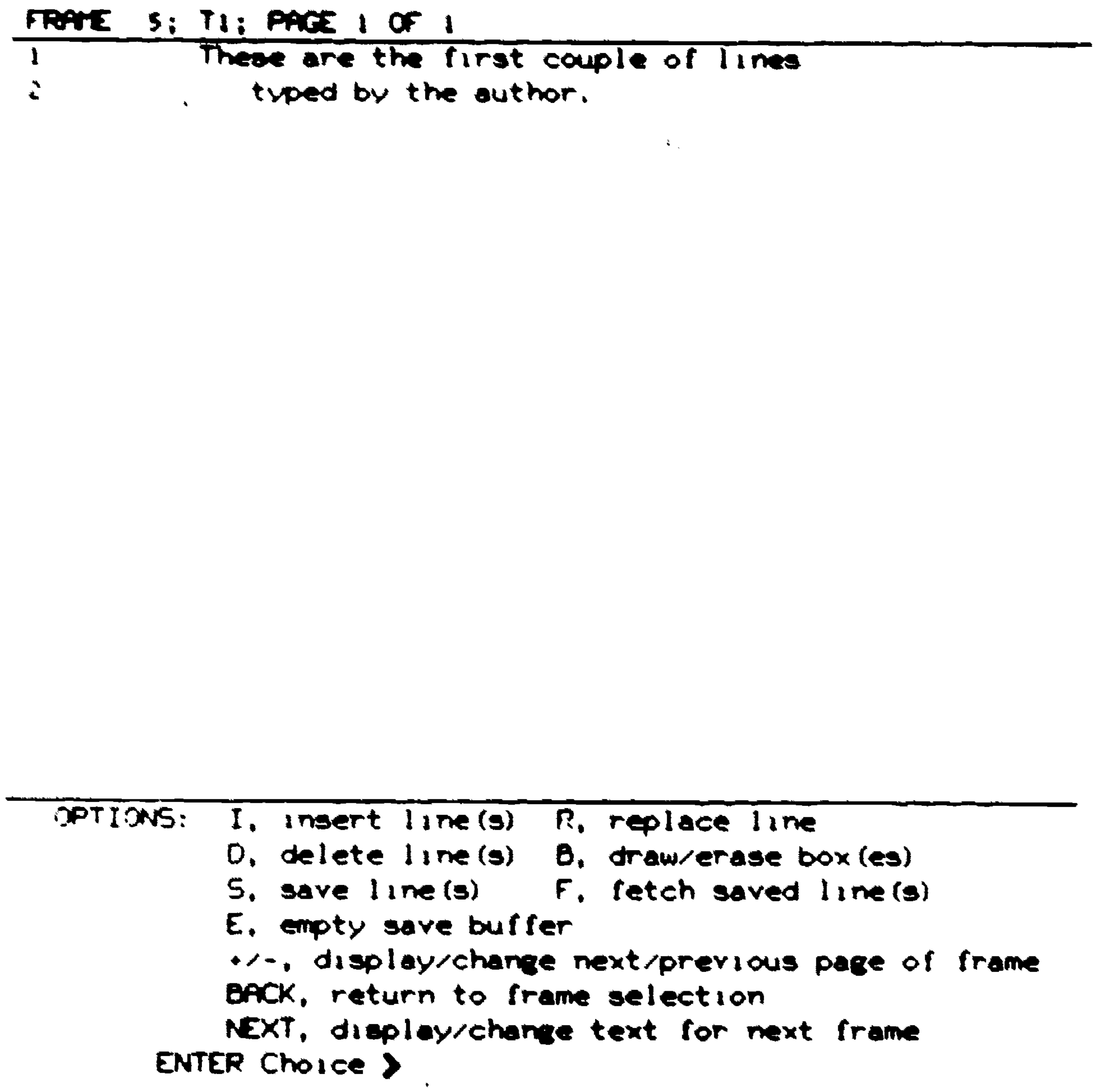
```
                      FRAME LIST for OBJECTIVE  1
 Frame        Brch Ans Flags        |   Frame        Brch Ans Flags
No  Name       Log  Try             |  No  Name       Log  Try

 1 D1           no       none       |  16 E1          no        none
 2 T11          no       none       |  17 T6          no        none
 3 S1           no       none       |  18 T7          no        none
 4 O1           no       none       |  19 QC2         no        none
 5 T1           no       none       |  2Ø T8          no        none
 6 QM1          no   2   none       |  21 QM5         no   1    OT
 7 QM2           A   2   F/D        |  22 QM6         no   1    OT
 8 T2          P A       none       |  23 QM7         no   1    OT
 9 T3           no       none       |  24 QM8          A   1    OT
1Ø T12          no       none       |  25 QM9         no   1    OT
11 QM3         P         none       |  26 QM1Ø         A   1    OT
12 QM4           A       D          |  27 T9          no        OF
13 T4           no       none       |  28 T1Ø         no        OP
14 T5           no       none       |
15 QC1          RA       none       |
FRAME TYPES:
1 Documentation                    7 Constructed Response Question
2 Statement of Objective           8 Branching Decision
3 Overview                         9 Title
4 Text                            1Ø Material List
5 Elaboration                     11 Copy of an existing frame
6 Multiple Choice Question
       ENTER choice  >
```

Figure 4. Authoring Editor Frame List Display
Showing Frame Type Selection Options.



```
FRAME  5; T1; PAGE 1 OF 1            INSERT MODE   1 Lines
1         These are the first couple of lines
    >          typed by the author.

OPTIONS:  Enter text material.
          Press NEXT to end each line.
          COPY keys copy previous line.
          Press BACK when finished.
```

Figure 5. Authoring Editor Text Frame Display as Seen When Creating a Frame.



```
FRAME  5; T1; PAGE 1 OF 1
1          These are the first couple of lines
2             typed by the author.




OPTIONS:  I, insert line(s)  R, replace line
          D, delete line(s)  B, draw/erase box(es)
          S, save line(s)    F, fetch saved line(s)
          E, empty save buffer
          +/-, display/change next/previous page of frame
          BACK, return to frame selection
          NEXT, display/change text for next frame
      ENTER Choice ❯
```

Figure 6. Authoring Editor Text Frame Display Showing Frame Editing Options.

ing material between lines, the two lines preceding and one line following the entry are also displayed.

The Replace Line (R) Option allows the author to edit one or more existing lines. The original version of the line is displayed, the author's new version is shown below it (to the right of the cursor) and the preceding and following lines are also displayed to provide the context of the change. A situation in which line 3 of a display is being edited is illustrated in Figure 7. A set of Edit keys facilitate the editing process by allowing the author to erase, copy, or temporarily store the complete line, specific words or individual characters. After the line has been changed, the Editor redisplays the edited text and moves down to the next line to allow the author to edit it. As in the Insert mode, the author can BACK out of Replace mode at any time.

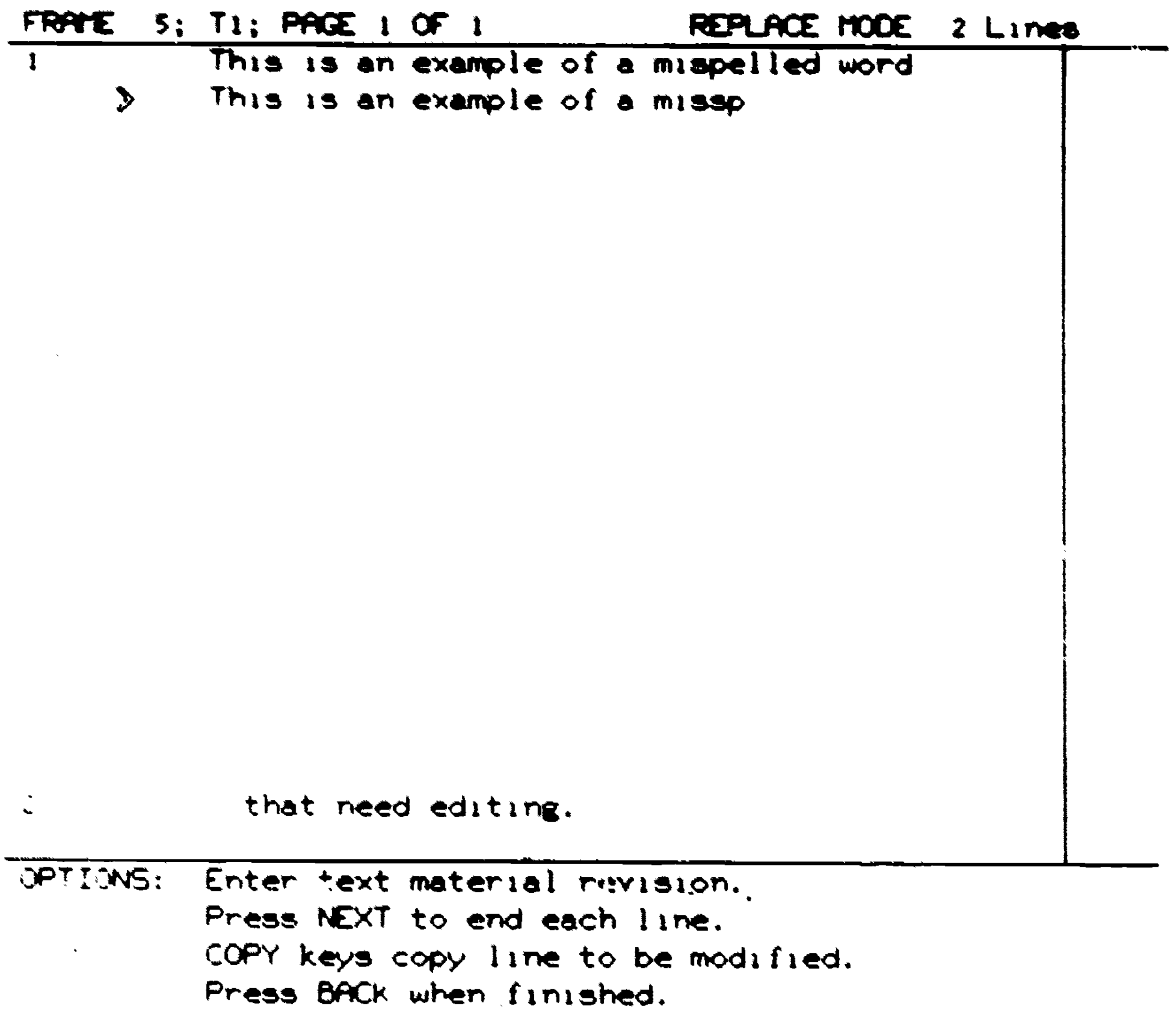
The Delete Line(s) (D) Option (see Figure 6) allows the author to erase one or more lines of text. The Editor prompts the author to indicate the lines to be deleted (the numbers of the first and last lines), surrounds the indicated lines by a box for visual emphasis and asks if the author wishes to complete or cancel the deletion request. Any text on the page following the deletion is moved up to replace the deleted material.

The Save Line(s) (S) Option provides a capability for saving up to 21 lines of material in a "Save" buffer. The saved material can then be inserted at any point in any textual content frame in any CAI module. This is useful in editing material, copying pages of frames without retyping and moving material to other modules. Once an author has saved one or more lines, they can be retrieved at any time until the Save buffer is emptied.

The Fetch Saved Line(s) (F) Option provides the means by which material is retrieved from the Save buffer. If, when the material is fetched to be inserted on a page already containing text, the inserted material will result in a total of more than 21 lines, the author is warned of this fact and given the option of canceling the Fetch command or allowing the Editor to reformat the text to overflow onto subsequent pages of the frame.

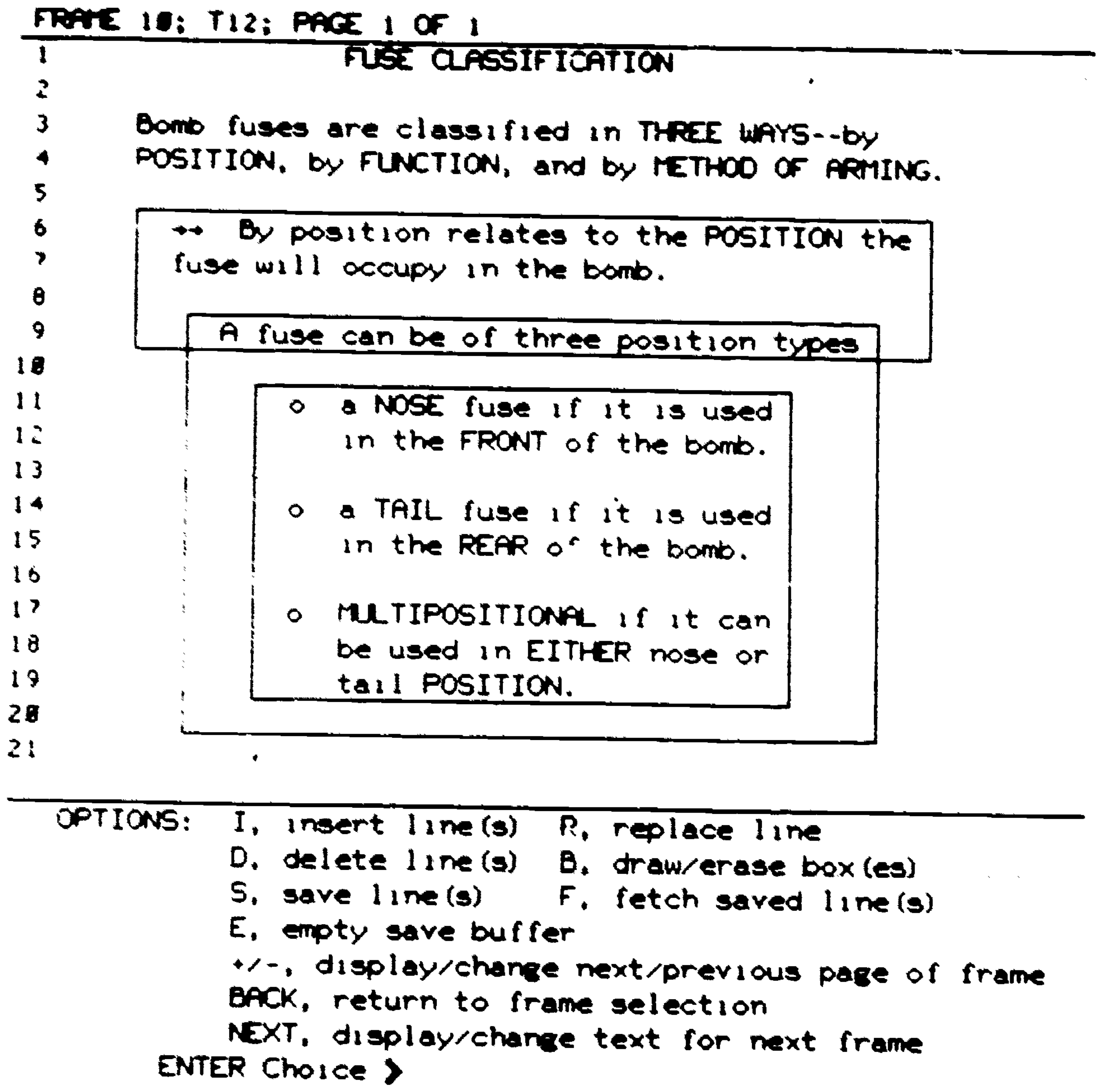
The Empty Save Buffer (E) Option simply erases the contents of the Save buffer prior to saving new material.

The Draw/Erase Box(es) (B) Option allows the author to graphically surround one or more lines of material for emphasis. An example is shown in Figure 8. Boxes can be drawn around any single line or group of lines. The author enters the number of the first and last lines to be surrounded and the horizontal width of the box is determined by the position of the first and last characters in the lines. Up to sixteen boxes can be defined in a single frame.



```
FRAME  5; T1; PAGE 1 OF 1            REPLACE MODE  2 Lines
1          This is an example of a mispelled word
      >    This is an example of a missp




2            that need editing.
OPTIONS:   Enter text material revision.
           Press NEXT to end each line.
           COPY keys copy line to be modified.
           Press BACK when finished.
```

Figure 7. Authoring Editor Text Frame Display In Replace Mode.



```
FRAME 10; T12; PAGE 1 OF 1
                    FUSE CLASSIFICATION

    Bomb fuses are classified in THREE WAYS--by
    POSITION, by FUNCTION, and by METHOD OF ARMING.

     ++  By position relates to the POSITION the
     fuse will occupy in the bomb.

       A fuse can be of three position types

            o  a NOSE fuse if it is used
               in the FRONT of the bomb.

            o  a TAIL fuse if it is used
               in the REAR o^ the bomb.

            o  MULTIPOSITIONAL if it can
               be used in EITHER nose or
               tail POSITION.


  OPTIONS:  I, insert line(s)   R, replace line
            D, delete line(s)   B, draw/erase box(es)
            S, save line(s)     F, fetch saved line(s)
            E, empty save buffer
            +/-, display/change next/previous page of frame
            BACK, return to frame selection
            NEXT, display/change text for next frame
      ENTER Choice >
```

Figure 3. Authoring Editor Text Frame Display
Showing Use of Boxes for Emphasis.

The final three options pertain to the mechanics of page and frame selection. To move forward to the next page of the current frame or back to the previous page, the author enters "+" or "-" respectively. Pressing the NEXT key routes the author to the first page of the next frame in the sequence. The BACK key returns the author to the page of the Frame List display listing the current frame.

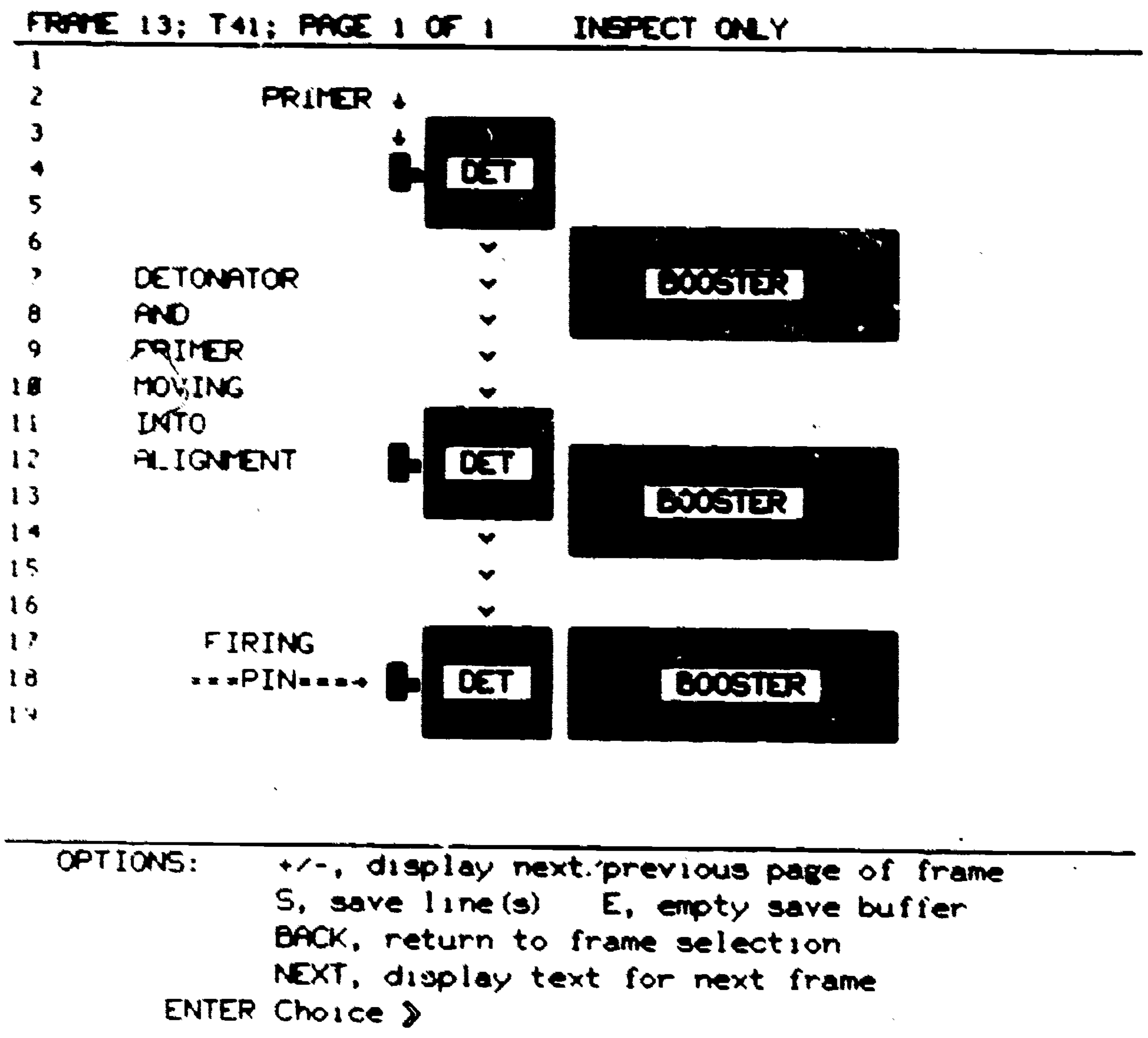
Although the Authoring Editor does not currently provide any extensive graphics capability, the author can, with some ingenuity, create simple graphics figures by using the standard keyboard characters and any "unrecognized" character, which creates a solid, character-sized rectangle. An example is provided in Figure 9. Note that the display is shown in Inspect Only mode with an abbreviated list of author options.

Question Generation. Multiple choice and constructed response question frames provide the author's primary means of interacting with and evaluating the student's understanding of the material being presented. As structured by the Authoring Editor, a multiple choice question consists of the question stem and two to five alternatives. After selecting this frame type (QM) from the Frame Types menu, the author is shown a template display and prompted to enter the question stem. The text of the stem is displayed as it is entered and a double press of the NEXT key notifies the Editor that the stem has been completed. The author is then prompted to enter the first alternative. A double NEXT indicates the end of each alternative and elicits a prompt for the next alternative. This process continues until the author indicates that all alternatives have been entered by pressing BACK or until the maximum of five alternatives has been entered. An example of a partially completed multiple choice question is shown in Figure 10.

Once the question stem and alternatives have been entered, the author is prompted to specify the correct alternative or alternatives. At least one correct answer must be defined before the author is allowed to exit the question frame. In addition, the author is prompted to specify the maximum number of attempts allowed in answering the question. The author may select a number or indicate that the default value, currently three, be used.

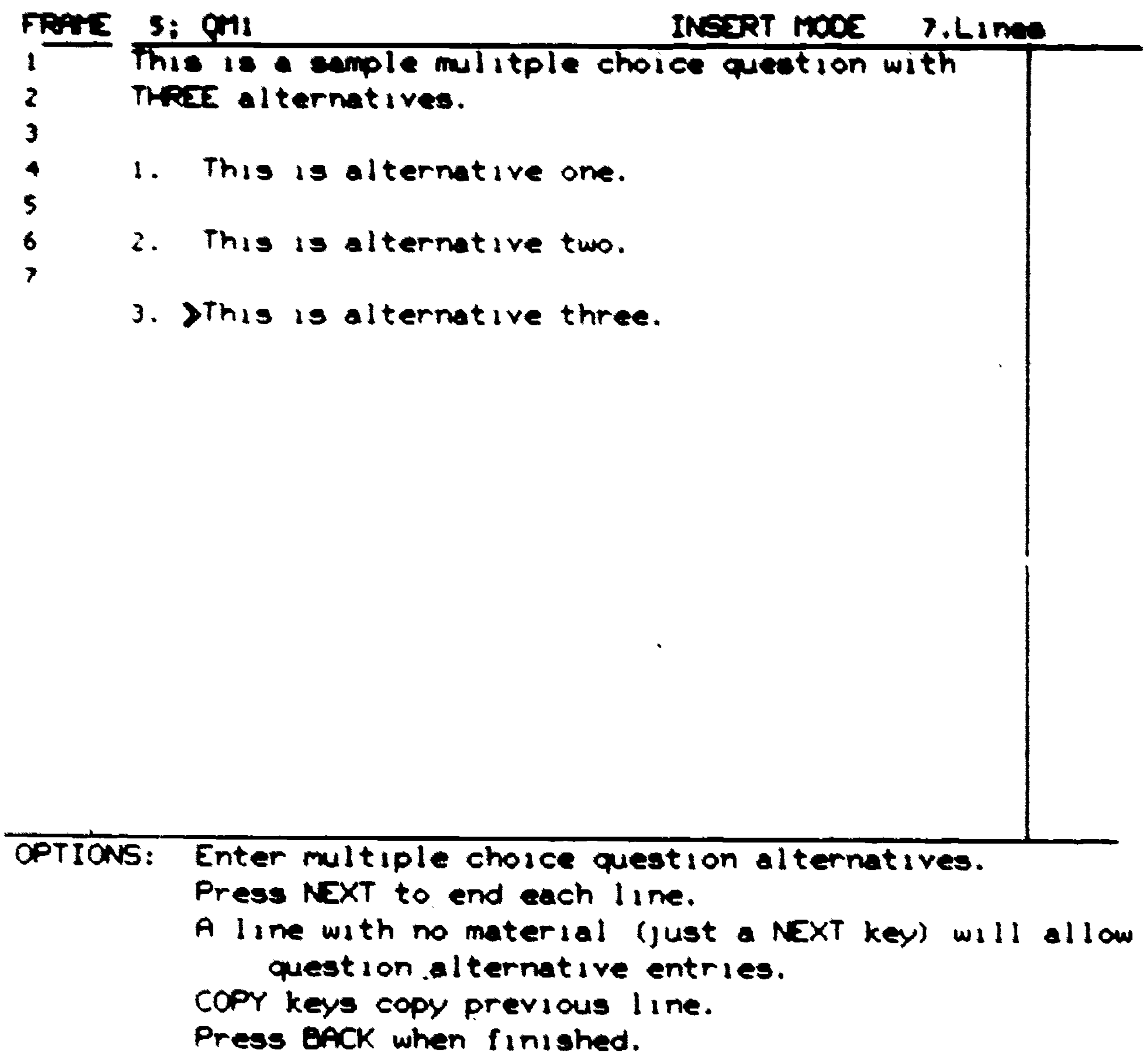
A constructed response question consists of the question stem and one to nine anticipated responses. When this frame type (QC) is selected, the procedure followed is essentially the same as for a multiple choice question. The author is prompted to enter the question stem and successive anticipated responses up to a maximum of nine. The correct response(s) and maximum number of allowable attempts must then be specified. A completed constructed response question is illustrated in Figure 11. The asterisks to the left of four of the alternatives indicate that any of these responses will be accepted as being correct.

Currently, the AIS CAI software supports only exact matches on constructed response questions. That is, other than variations between



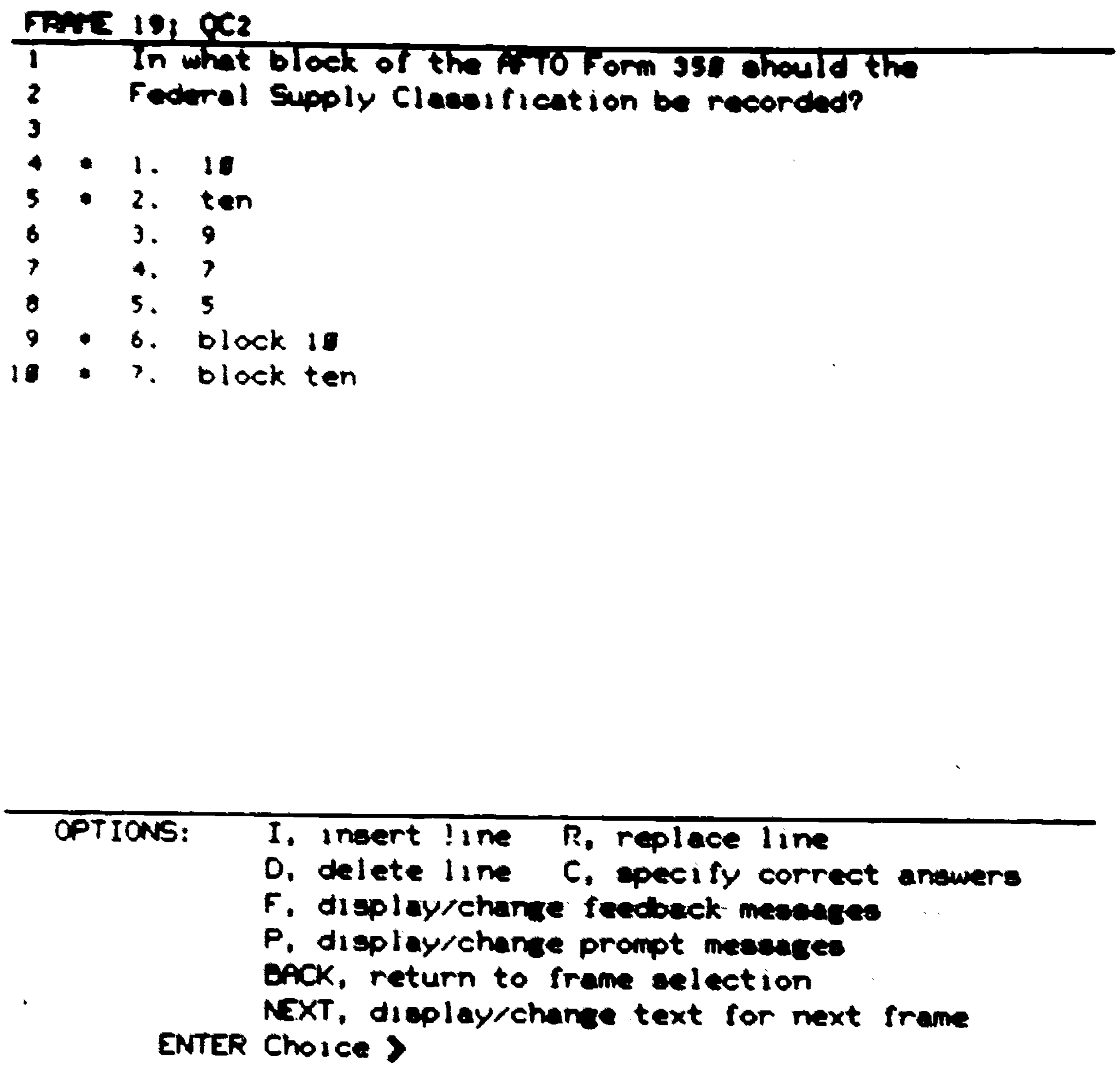
```
FRAME 13; T41; PAGE 1 OF 1     INSPECT ONLY

        DETONATOR
        AND
        PRIMER
        MOVING
        INTO
        ALIGNMENT

OPTIONS:    +/-, display next/previous page of frame
            S, save line(s)    E, empty save buffer
            BACK, return to frame selection
            NEXT, display text for next frame
      ENTER Choice >
```

Figure 9. Authoring Editor Text Frame Display in Inspect Only Mode Showing Use of Simple Graphics.



```
FRAME  5; QM1                               INSERT MODE   7.Lines
       This is a sample mulitple choice question with
       THREE alternatives.

       1.   This is alternative one.

       2.   This is alternative two.

       3. >This is alternative three.

OPTIONS:  Enter multiple choice question alternatives.
          Press NEXT to end each line.
          A line with no material (just a NEXT key) will allow
              question alternative entries.
          COPY keys copy previous line.
          Press BACK when finished.
```

Figure 10. Authoring Editor Display of Partially Completed Multiple Choice Question.



```
FRAME 19; QC2
 1        In what block of the AFTO Form 350 should the
 2        Federal Supply Classification be recorded?
 3
 4   *    1.   10
 5   *    2.   ten
 6        3.   9
 7        4.   7
 8        5.   5
 9   *    6.   block 10
10   *    7.   block ten

   OPTIONS:      I, insert line    R, replace line
                 D, delete line    C, specify correct answers
                 F, display/change feedback messages
                 P, display/change prompt messages
                 BACK, return to frame selection
                 NEXT, display/change text for next frame
            ENTER Choice >
```

Figure 11. Authoring Editor Display of Completed Constructed Response Question.

upper and lower case, the student's response must exactly match one of the author's anticipated responses. Otherwise, the student's answer is treated as an unanticipated response. Work is currently in progress to support recognition of key words and strings and misspelled words which phonetically match an anticipated response.

To provide students with substantive information about their responses, provision has been made for author entry of feedback and prompt statements for both multiple choice and constructed response questions (Options F and P in Figure 11). The intent of a feedback message is to tell the student that his or her answer was correct or, if it was incorrect, why it was wrong. For each alternative or anticipated response defined, the author can write a feedback statement which will be presented whenever a student selects that alternative. For constructed response questions, a feedback message can also be defined for the occurrence of unanticipated responses. If the author does not define a feedback message for an alternative, a standard statement, matched to whether the response is correct or incorrect, will be selected at random and presented. The author can suppress a feedback message by entering one or more blank characters.

The intent of a prompt message is to guide the student toward the correct answer. Whereas feedback messages are alternative-specific, prompts are selected and presented as a function of the number of the student's attempt at answering the question. That is, prompt message number one will be presented following the student's first incorrect attempt, prompt two following the second incorrect attempt, etc. Prompts are not presented following a correct response but are otherwise independent of the specific alternative chosen.

Both feedback and prompt messages may consist of up to three lines of 60 characters each. Depending on the author's actions, one or both messages will be displayed at the bottom of the screen following a student's response. The prompt statement immediately follows the feedback statement, giving the appearance of a single informative message. This concatenation of feedback and prompt provides the author with a powerful tool for responding appropriately to students' errors.

Question generation mode provides the author with some of the text editing features (insert, replace and delete) described under Text Generation. The use of these features is, however, somewhat restricted due to the structured nature of question entry. The save buffer and box options are not available.

Definition of Branching Logic. In preparing a CAI module, definition of effective branching logic may require the greatest thought and be the most difficult part of the process for an author to understand. The effectiveness of a lesson, however, can hinge on how well this feature is used. Therefore, particular attention was given to facilitating this aspect of the authoring process.

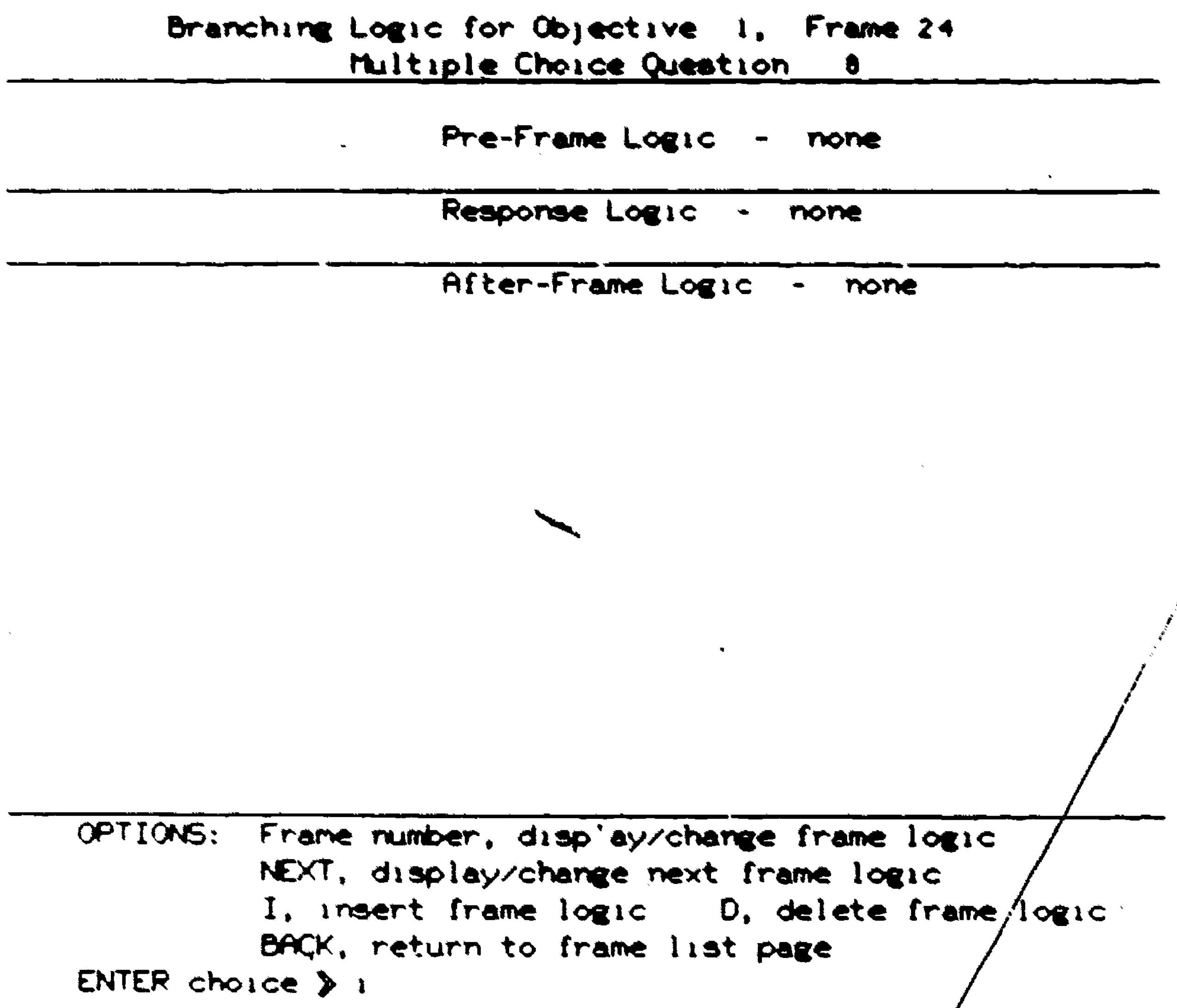
Within the AIS CAI scheme, all branching decisions are associated with individual frames and can be evaluated at three different points in the presentation of the frame. Pre-frame logic is evaluated at the point at which the Presentation Program first encounters the frame, before it is displayed. Response logic, which can only be defined for a question frame, is evaluated as the student answers the question with the branch being taken if a particular response is made. After-frame logic is evaluated after the student has indicated a readiness to advance to the next frame. A branch can be made to any frame within the objective which has been defined in the Frame List display.

Currently, four types of conditions can be evaluated by Pre- or After-frame branching logic: (a) Given a set of question frames, take the branch if at least a specified number of these questions were answered correctly. (b) Given a set of question frames, branch if at least a specified number of these questions were answered incorrectly. (c) Branch if at least a specified number of a given set of frames (of any type) have been presented. (d) Branch if at least a specified number of a given set of frames have not been presented. In addition, unconditional branches can be defined where the branch will always be taken if the branching point is reached. As mentioned above, Response logic presents a different situation in which the branch is taken if a particular response is made.

Any number of branching logic instructions, of any or all of the three types, can be entered against a single frame. The only restriction is the total storage space required for all branching instructions within an objective. At least 300 instructions can be defined for a single objective.

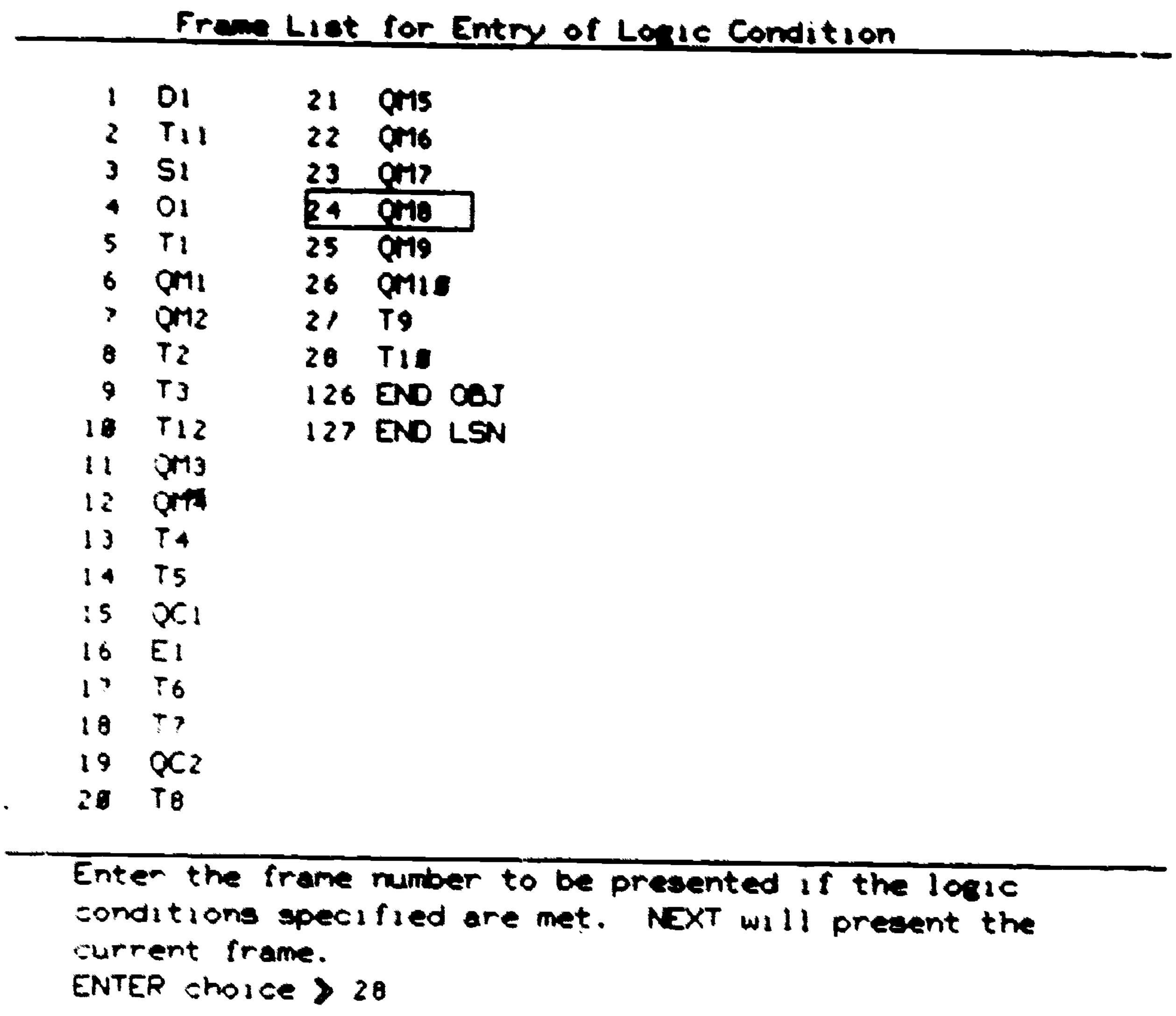
Branching instructions are evaluated in the order in which they are encountered. Thus, whether or not a subsequent instruction will be evaluated is dependent on the result of evaluating the current instruction. If none of the conditions of Pre-frame logic are met (or if no Pre-frame logic was defined), the frame will be presented. If none of the conditions of Response or After-frame logic are met (or if no such logic was defined), the next frame in the sequence will be presented.

The process by which an author defines branching logic for a frame is illustrated by Figures 12 through 20. Like most aspects of an author's work within an objective, the process begins on the Frame List display where the author selects the Branching Logic Option (B) and enters the number of the frame for which logic is to be defined. This results in the Branching Logic display shown in Figure 12. In the example shown, no logic has yet been defined for the frame. The author selects the action to be taken from the Options list--in this case, to Insert Frame Logic (I). The author is then asked whether Pre-frame, Response, or After-frame logic is to be defined and a brief description of each of the three types is provided. Assume that After-frame logic is to be defined, for which the Author enters an "A." This results in



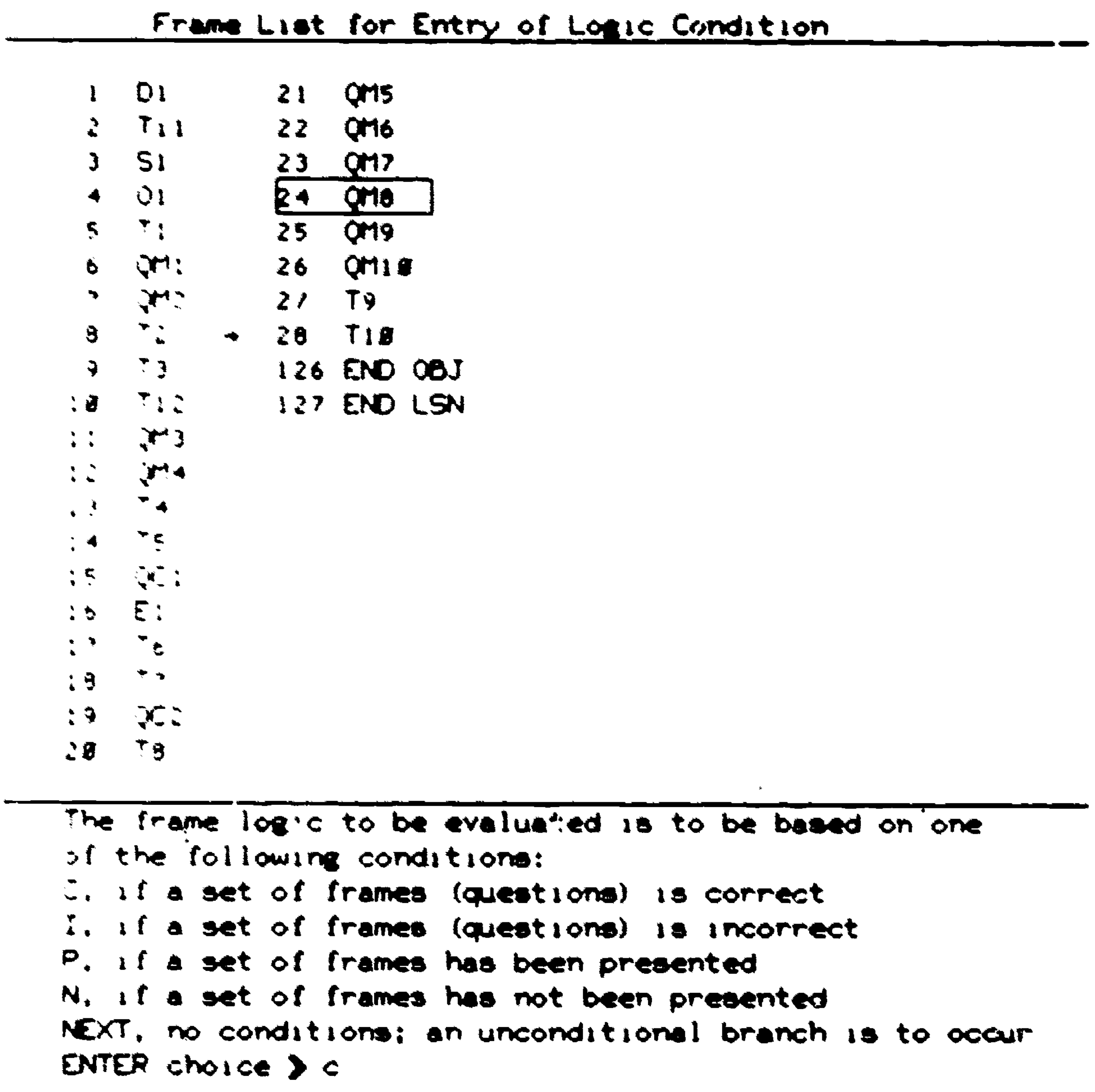
```
        Branching Logic for Objective  1,  Frame 24
                   Multiple Choice Question   8
_______________________________________________________________

                    Pre-Frame Logic  -  none
_______________________________________________________________
                    Response Logic  -  none
_______________________________________________________________
                    After-Frame Logic  -  none









_______________________________________________________________
  OPTIONS:  Frame number, disp'ay/change frame logic
            NEXT, display/change next frame logic
            I, insert frame logic    D, delete frame logic
            BACK, return to frame list page
  ENTER choice ❯ i
```

Figure 12. Authoring Editor Display for Defining Branching Logic.



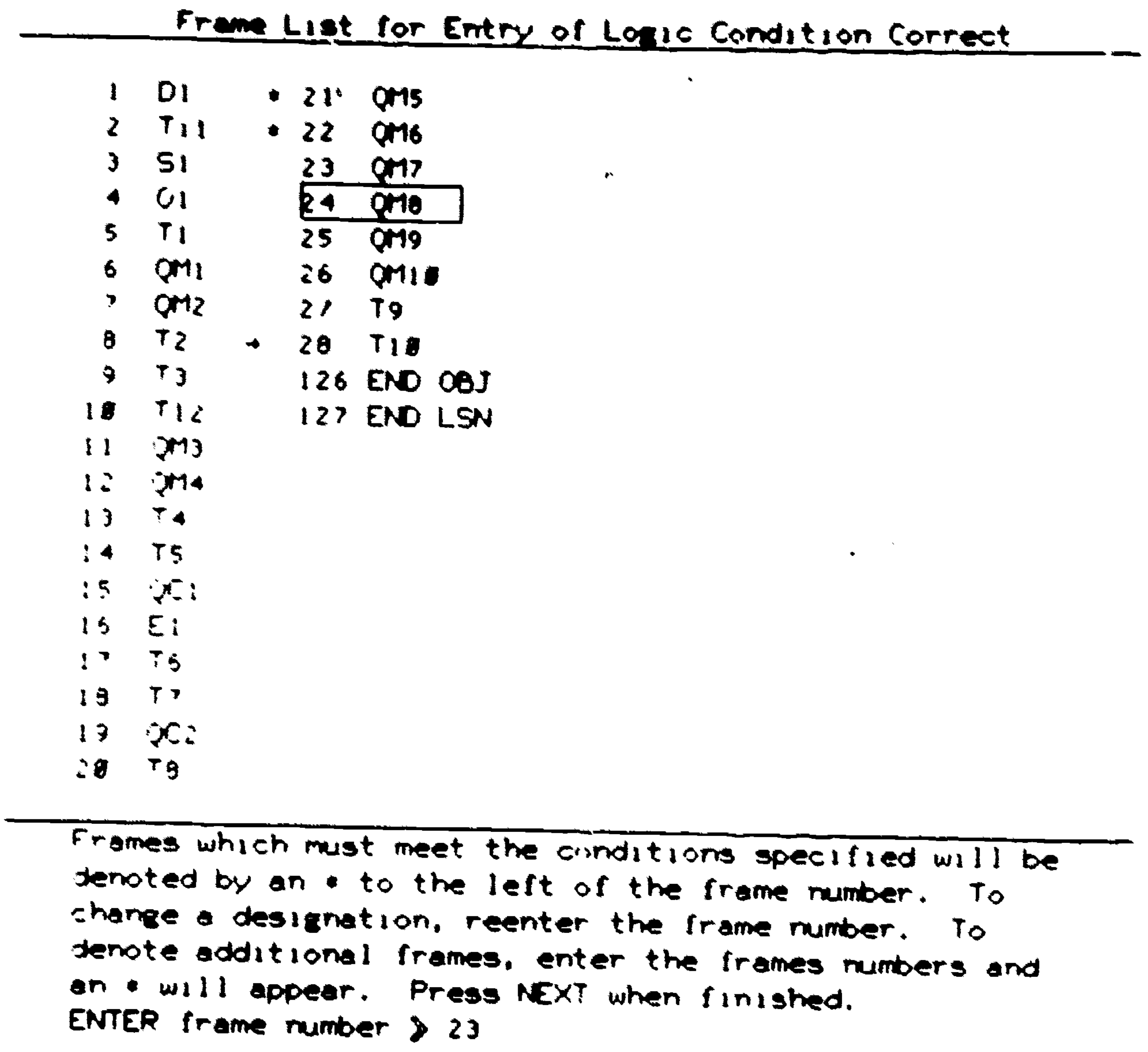
```
              Frame List for Entry of Logic Condition

     1  D1          21  QM5
     2  T11         22  QM6
     3  S1          23  QM7
     4  O1          24  QM8
     5  T1          25  QM9
     6  QM1         26  QM10
     7  QM2         27  T9
     8  T2          28  T10
     9  T3          126 END OBJ
    10  T12         127 END LSN
    11  QM3
    12  QM4
    13  T4
    14  T5
    15  QC1
    16  E1
    17  T6
    18  T7
    19  QC2
    20  T8

  Enter the frame number to be presented if the logic
  conditions specified are met.  NEXT will present the
  current frame.
  ENTER choice > 28
```

Figure 13. Authoring Editor Display for Selecting To-be-branched-to Frame for Pre- or After-frame Branching Logic.



```
        Frame List for Entry of Logic Condition
_____________________________________________________________

  1   D1          21   QM5
  2   T11         22   QM6
  3   S1          23   QM7
  4   O1          24   QM8
  5   T1          25   QM9
  6   QM1         26   QM1Ø
  7   QM2         27   T9
  8   T2     →    28   T1Ø
  9   T3          126  END OBJ
 1Ø   T12         127  END LSN
 11   QM3
 12   QM4
 13   T4
 14   T5
 15   QC1
 16   E1
 17   T6
 18   T7
 19   QC2
 2Ø   T8
_____________________________________________________________
 The frame logic to be evaluated is to be based on one
 of the following conditions:
 C, if a set of frames (questions) is correct
 I, if a set of frames (questions) is incorrect
 P, if a set of frames has been presented
 N, if a set of frames has not been presented
 NEXT, no conditions; an unconditional branch is to occur
 ENTER choice > c
```

Figure 14. Authoring Editor Display for Selecting Class of Pre- or After-frame Branching Conditions.

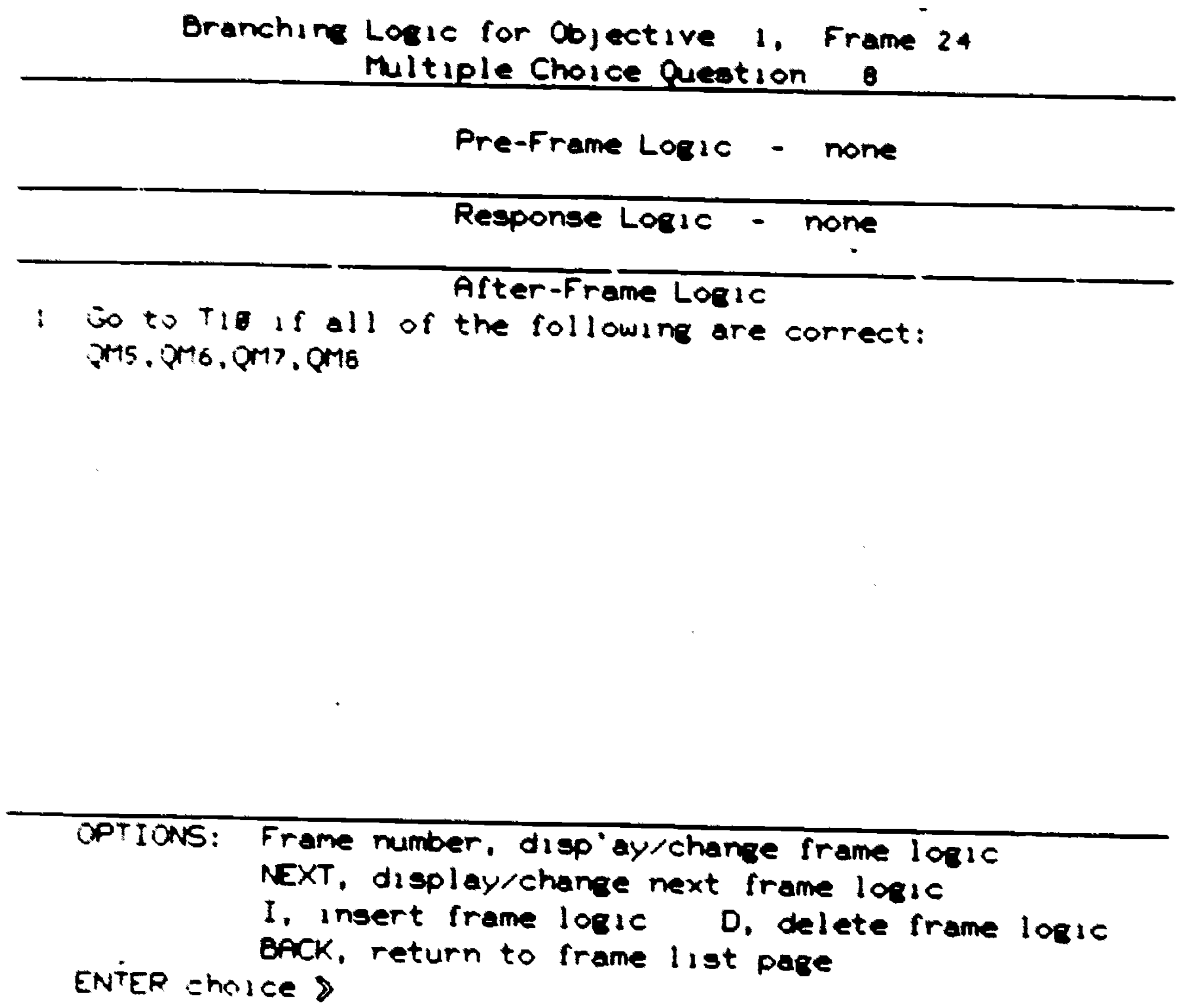


```
          Frame List for Entry of Logic Condition Correct
_______________________________________________________________
    1  D1      * 21  QM5
    2  T11     * 22  QM6
    3  S1        23  QM7
    4  O1        24  QM8
    5  T1        25  QM9
    6  QM1       26  QM1Ø
    7  QM2       27  T9
    8  T2     →  28  T1Ø
    9  T3        126 END OBJ
   1Ø  T12       127 END LSN
   11  QM3
   12  QM4
   13  T4
   14  T5
   15  QC1
   16  E1
   17  T6
   18  T7
   19  QC2
   2Ø  T8
_______________________________________________________________
    Frames which must meet the conditions specified will be
    denoted by an * to the left of the frame number.  To
    change a designation, reenter the frame number.  To
    denote additional frames, enter the frames numbers and
    an * will appear.  Press NEXT when finished.
    ENTER frame number ≫ 23
```

Figure 15. Authoring Editor Display for Selecting Question Frames to be Evaluated for Correctness for a Pre- or After-frame Branch.

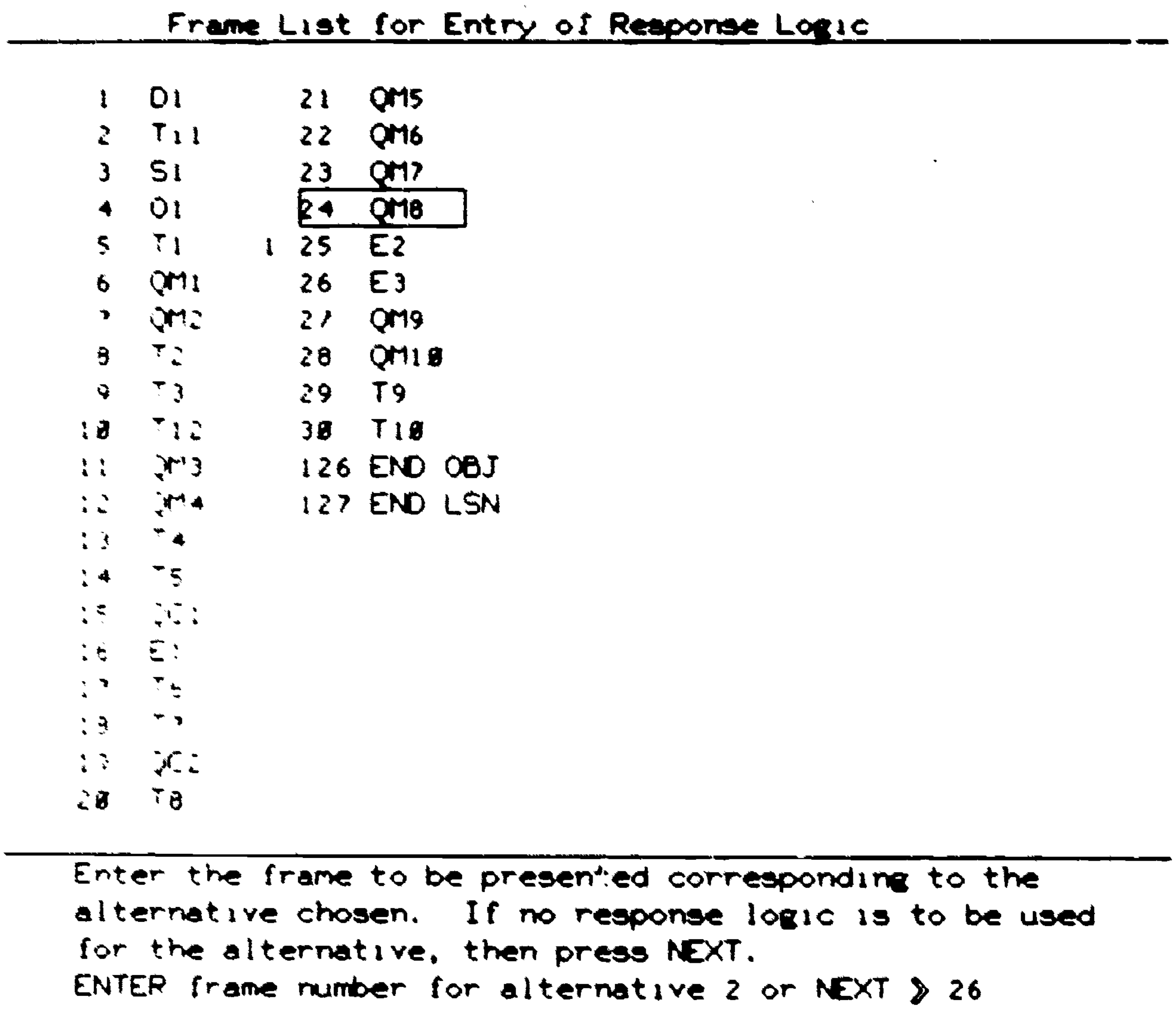
```
          Frame List for Entry of Logic Condition Correct
__________________________________________________________________

     1   D1        * 21   QM5
     2   T11       * 22   QM6
     3   S1        * 23   QM7
     4   O1        * 24   QM8
     5   T1          25   QM9
     6   QM1         26   QM1Ø
     7   QM2         27   T9
     8   T2      →   28   T1Ø
     9   T3          126  END OBJ
    1Ø   T12         127  END LSN
    11   QM3
    12   QM4
    13   T4
    14   T5
    15   QC1
    16   E1
    17   T6
    18   T7
    19   QC2
    2Ø   T8
__________________________________________________________________
    Enter the number of frames which must meet the
    condition specified in order to present the desired
    frame.
    ENTER choice > 4
```

Figure 16. Authoring Editor Display for Indicating Number of Correctly Answered Questions to Meet Pre- or After-frame Branching Condition



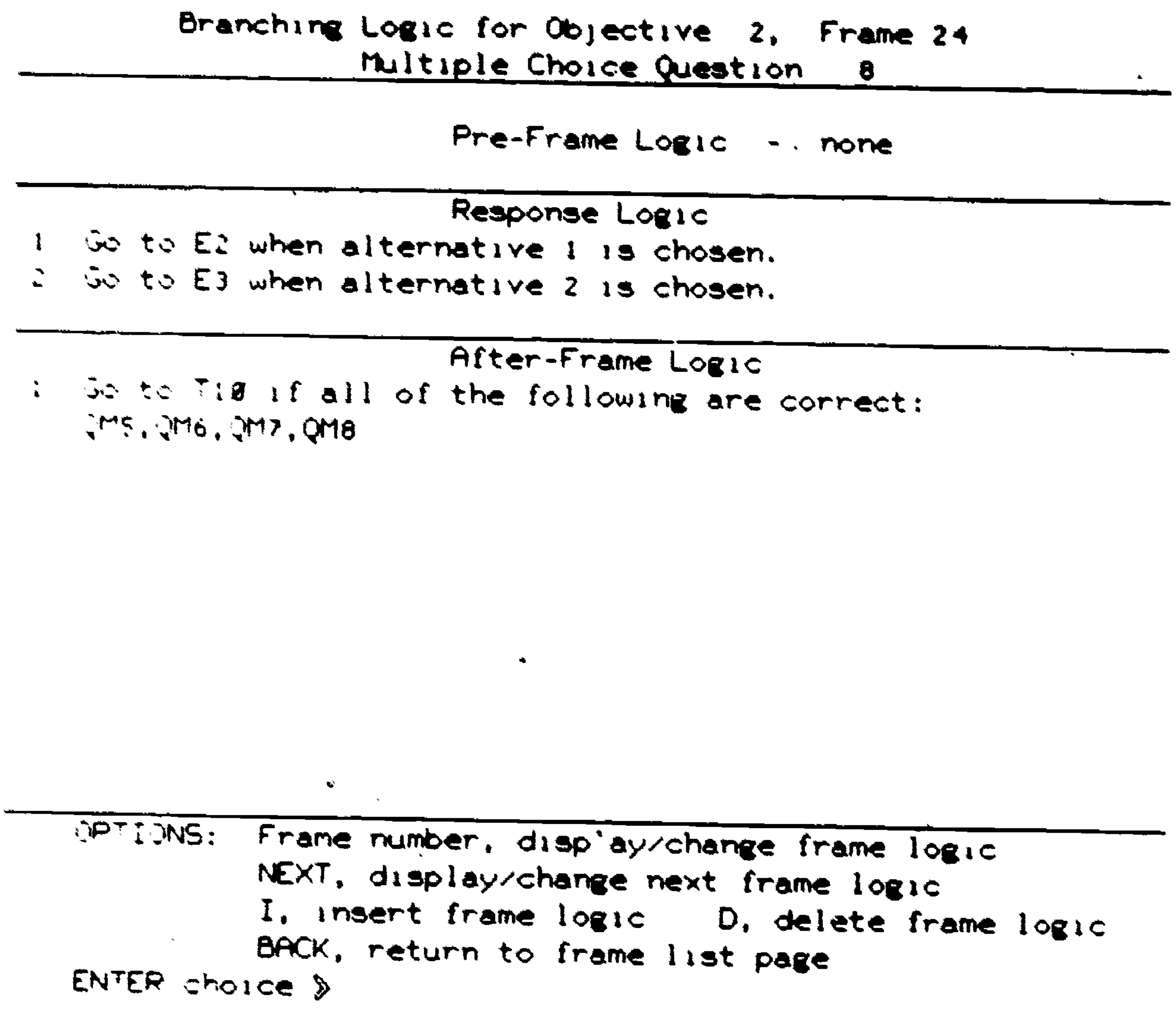
```
        Branching Logic for Objective  1,  Frame 24
                  Multiple Choice Question   8
                        Pre-Frame Logic  -  none
                        Response Logic  -  none
                        After-Frame Logic
 1  Go to T10 if all of the following are correct:
    QM5,QM6,QM7,QM6

   OPTIONS:  Frame number, disp'ay/change frame logic
             NEXT, display/change next frame logic
             I, insert frame logic    D, delete frame logic
             BACK, return to frame list page
   ENTER choice »
```

Figure 17. Authoring Editor Display Showing Completed After-frame Branching Instruction.



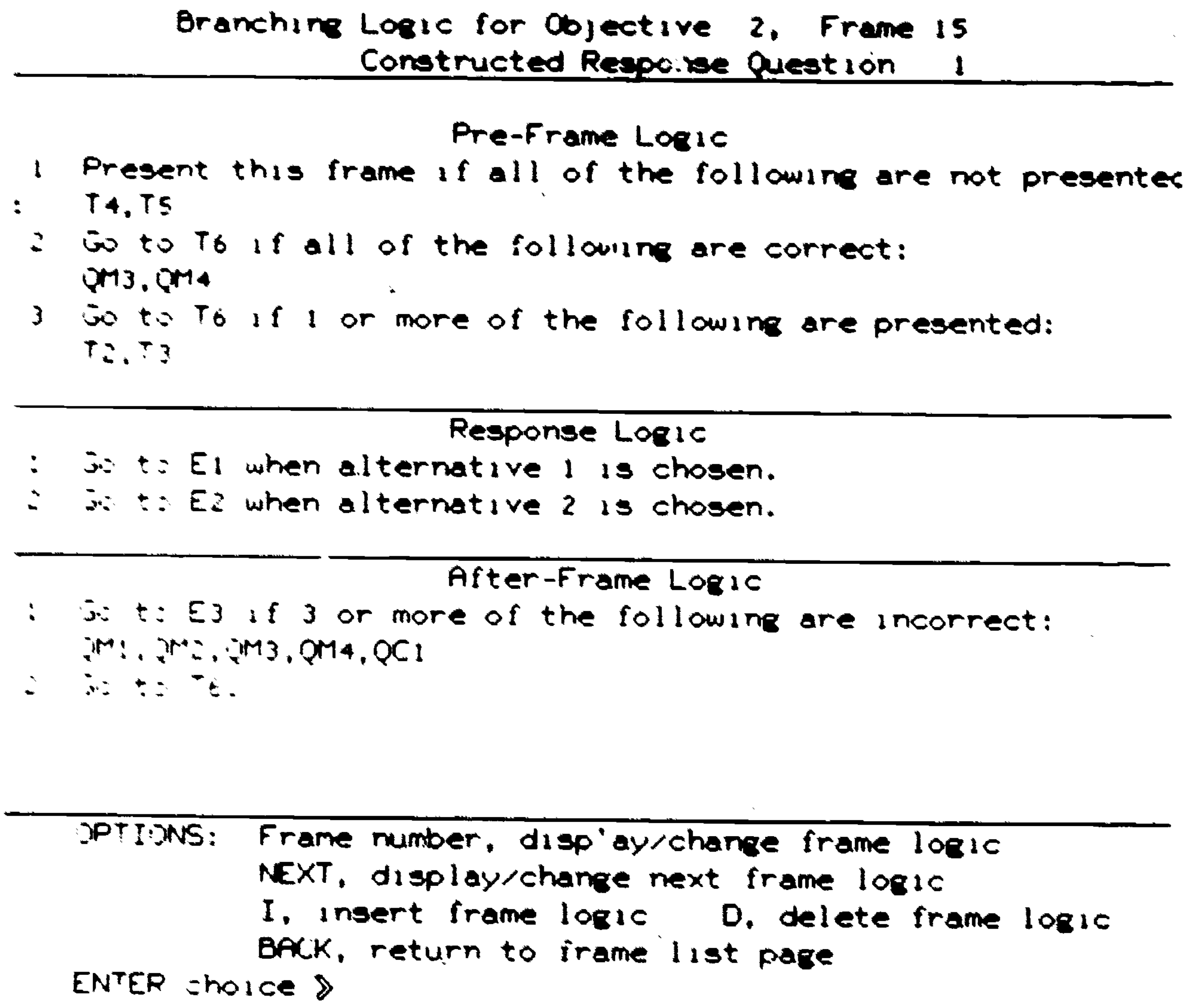
```
        Frame List for Entry of Response Logic

   1  D1           21  QM5
   2  T11          22  QM6
   3  S1           23  QM7
   4  O1           24  QM8
   5  T1         1 25  E2
   6  QM1          26  E3
   7  QM2          27  QM9
   8  T2           28  QM10
   9  T3           29  T9
  10  T12          30  T10
  11  QM3         126  END OBJ
  12  QM4         127  END LSN
  13  T4
  14  T5
  15  QC1
  16  E1
  17  T6
  18  T7
  19  QC2
  20  T8

   Enter the frame to be presented corresponding to the
   alternative chosen.  If no response logic is to be used
   for the alternative, then press NEXT.
   ENTER frame number for alternative 2 or NEXT > 26
```

Figure 18. Authoring Editor Display for Defining Response Contingent Branching Logic.



```
            Branching Logic for Objective  2,  Frame 24
                      Multiple Choice Question   8
________________________________________________________________
                        Pre-Frame Logic  -- none
________________________________________________________________
                        Response Logic
1  Go to E2 when alternative 1 is chosen.
2  Go to E3 when alternative 2 is chosen.
________________________________________________________________
                        After-Frame Logic
1  Go to T10 if all of the following are correct:
   QM5,QM6,QM7,QM8




________________________________________________________________
   OPTIONS:  Frame number, disp'ay/change frame logic
             NEXT, display/change next frame logic
             I, insert frame logic    D, delete frame logic
             BACK, return to frame list page
   ENTER choice 》
```

Figure 19. Authoring Editor Display Showing Completed Response Contingent and After-frame Branching Instructions.



```
        Branching Logic for Objective  2,  Frame 15
                      Constructed Response Question   1
-----------------------------------------------------------------
                          Pre-Frame Logic
 1   Present this frame if all of the following are not presentec
     T4,T5
 2   Go to T6 if all of the following are correct:
     QM3,QM4
 3   Go to T6 if 1 or more of the following are presented:
     T2,T3
-----------------------------------------------------------------
                          Response Logic
 1   Go to E1 when alternative 1 is chosen.
 2   Go to E2 when alternative 2 is chosen.
-----------------------------------------------------------------
                          After-Frame Logic
 1   Go to E3 if 3 or more of the following are incorrect:
     QM1,QM2,QM3,QM4,QC1
 2   Go to T6.

-----------------------------------------------------------------
   OPTIONS:  Frame number, disp'ay/change frame logic
             NEXT, display/change next frame logic
             I, insert frame logic    D, delete frame logic
             BACK, return to frame list page
  ENTER choice 》
```

Figure 20. Authoring Editor Display Showing Three Classes of Branching Instructions.

the display of the list (shown in Figure 13) of all of the frames which have been defined for the objective. The current frame (frame number 24 in this case) is indicated by being boxed in. The author is to enter the number of the frame to which the branch is to be made if the logic conditions are met--frame number 28 in this example.

Once the to-be-branched-to frame has been selected, the display changes to that shown in Figure 14. The arrow pointing to frame number 28 indicates the frame to which the branch will be made if the conditions are met. The author is now requested to select the class of conditions to be evaluated. Assume that the author wishes to branch if some number of questions was answered correctly and, therefore, enters "C." The author is then requested (Figure 15) to enter the numbers of the frames to be evaluated. As a frame number is entered, an asterisk is shown to the left of the frame in the list. When this step is completed (Figure 16), the author is asked to enter the minimum number of questions which must be answered correctly for the branch to be taken--in this case, all four. This completes the author's entry of this logic instruction and the logic defined for the frame thus far is summarized as shown in Figure 17. At this point, the author can enter another logic instruction, go on to the next frame, or return to the Frame List page.

Assume that the author wishes to enter a Response logic instruction. After Options I (Insert logic) and R (for Response logic) have been selected, the text of the question is displayed with the correct answer(s) indicated to refresh the author's memory as to its content. Next, the author is shown a variation of the Branching Logic Frame List, as illustrated in Figure 18. For each alternative, the author is to choose a frame number from the list or simply press the NEXT key if no branch is to be taken when that alternative is selected. As they are entered, the numbers of the alternatives are displayed to the left of the corresponding frames in the list. In this case, the author has indicated a branch to frame E2 if alternative 1 is selected and is in the process of indicating a branch to E3 if alternative 2 is selected. After all alternatives have been considered, the logic which has been defined is summarized as shown in Figure 19. When the author returns to the Frame List display, the existence of Response and After-frame logic will be indicated for this frame.

Some of the more extensive branching possibilities are illustrated by Figure 20.

The CAI Presentation Programs

While Authoring Editor provides the vehicle by which CAI materials are developed, there must also be software to support presentation of these materials to students. CAI presentation is accomplished through the use of a general program structure and set of support routines driven by the CAI module description, decision logic, and text records created by an author using the Authoring Editor. Through the use of this generalized program structure and a table-driven approach, a wide range of computer-assisted tutorial and drill and practice instruction can be presented with minimal programming effort. To date, three different CAI Presentation Programs have been developed from the basic skeletal structure and support routines. One, the first developed, presents standard modules in support of lessons assigned on the student's first pass through a block. The second supports block review modules, assigned just prior to a student's block test, which present material for just those objectives which the student failed while studying in the block. The third supports block remediation modules, assigned after a block test failure, and presents material for just those objectives that the student failed on the test.

The basic skeleton and support routines were written during the development of the first-pass module program. This program required the most extensive design and development since it was to form the basis for subsequent programs. The block review and remediation programs were then created by slightly modifying the main loop code. Development time for variations to the basic program has proven to be minimal and can be done by entry level programmers. It should be noted that the first-pass module presentation program is sufficiently general to support the presentation of all such CAI modules written for any AIS course. Similarly, the review and remediation programs are general enough to handle the presentation of any cognitive objectives for AIS block review and remediation.

In addition to supporting CAI presentation for student study, the same presentation programs can be used by subject matter experts and instructional designers who wish to view the module from the student's perspective.

The following section describes the typical sequence of events which occurs when a student interacts with a CAI module of the type developed; specifically, assignment to a first-pass module. The subsequent section describes the use of the presentation program by an author or reviewer. The student performance data collection actions which take place during the presentation of a CAI module are mentioned only briefly. A full description of the data acquisition and analysis process is presented in the subsequent section.

<u>Student CAI Scenario.</u> As was discussed in the Introduction to this report, when the AIS makes a lesson assignment, it determines whether there are two or more alternative modules, including any CAI modules, available for teaching that lesson. If so, there are a variety of decision rules for determining which module should be assigned. As a general rule, CAI modules are assigned to that proportion of the students for whom they are considered to be the most appropriate, assuming that the required instructional resources (in this case, an interactive terminal) are available.

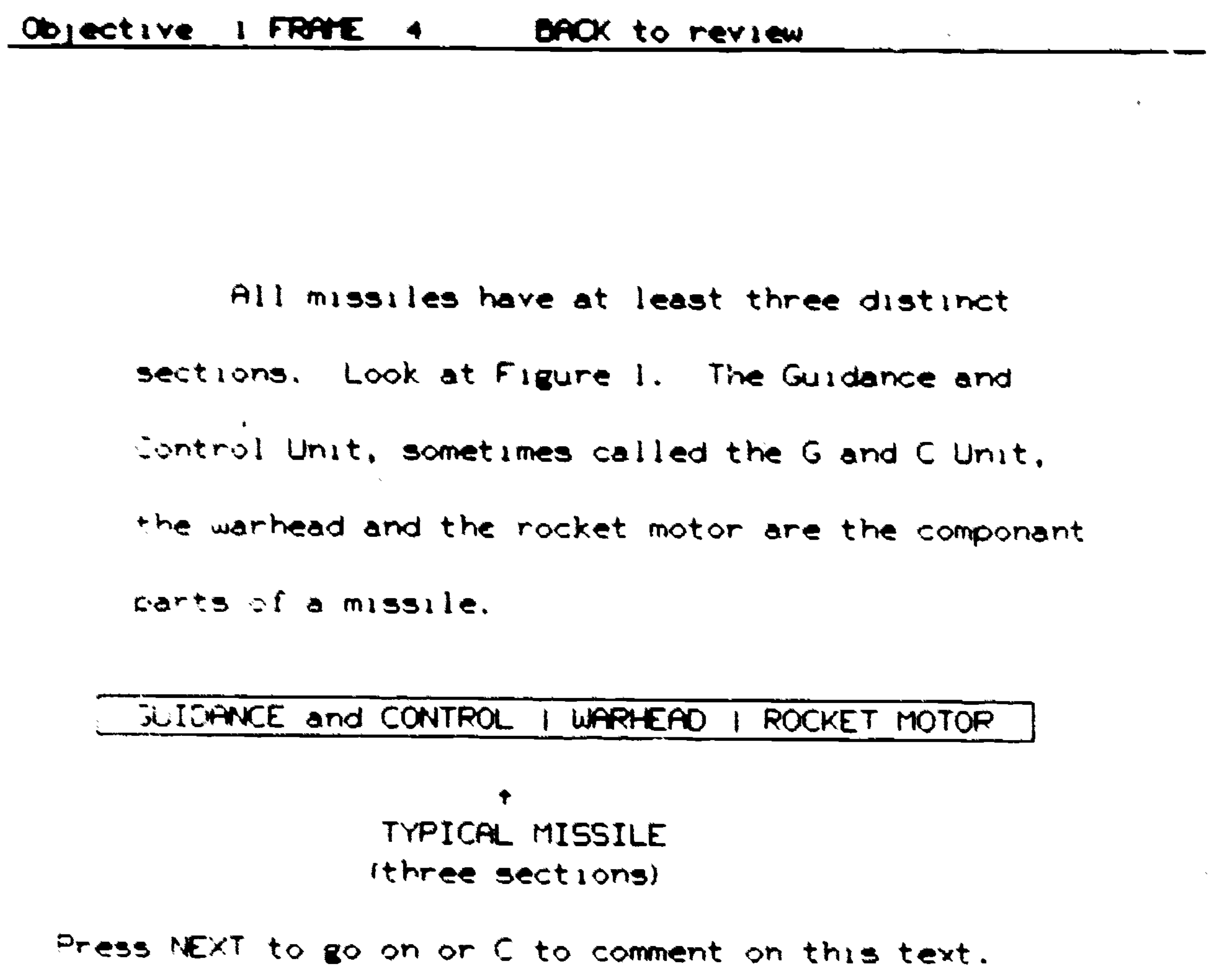
When assigned a CAI module, the student signs onto any available interactive terminal by typing in his or her student account number. If the AIS does not recognize the number, the user will be told that the account number is not registered and will be logged off. If the number is recognized, the student's Student Data Profile record is checked to determine whether the student's current assignment is indeed a CAI module. If not, the student is so informed and logged off.

If the student is assigned a CAI module, the particular module is determined from the Student Data Profile record. The Presentation Program then reads the CAI module description record, created by the Authoring Editor, to determine whether or not student response and decision point data are to be recorded and whether student comments are to be elicited. If so, data recording is activated for those classes of data which are to be collected.

The Presentation Program begins by presenting the material in Objective 0, containing at least an overview of the lesson and a required materials list, and then begins the instruction contained in the first numbered objective in the series. Frame descriptions, branching logic, and text records, all created by the Authoring Editor, determine the sequence and content of the presentation. While authors can define a variety of different instructional strategies, a single approach will be described here for explanatory purposes. First, the statement of the objective is presented and briefly elaborated. This could be followed by one or more pages of text and a set of practice questions. The number of practice questions could, and should, be a function of the student's performance on prior questions. Additional frames, elaborating specific problem areas will be presented, as necessary, on the basis of the student's responses. A typical page from a text frame, as seen by the student, is presented in Figure 21.

At any point in the objective, the student may opt to review the material that has already been presented by pressing the BACK key. In review mode, text is displayed in the normal manner and questions are displayed with the student's answers indicated.

Questions and the feedback and prompts following incorrect responses form a critical part of the instructional process. For both constructed response and multiple choice questions, students are required to continue answering until correct or until reaching the specified maximum number of attempts. For multiple choice questions, the student's last response is indicated by an arrow pointing to the alternative selected while prior responses are indicated by asterisks. Figure 22 presents an example in which the student is about to make a third attempt to answer the question. Note the author-supplied feedback (first three lines) and prompt (last three lines) at the bottom of the display. On constructed response questions, the student's prior incorrect responses are listed beneath the question stem. Figure 23 presents an example in which the student made two incorrect responses followed by the right answer, has



Objective 1 FRAME 4 BACK to review

All missiles have at least three distinct sections. Look at Figure 1. The Guidance and Control Unit, sometimes called the G and C Unit, the warhead and the rocket motor are the componant parts of a missile.

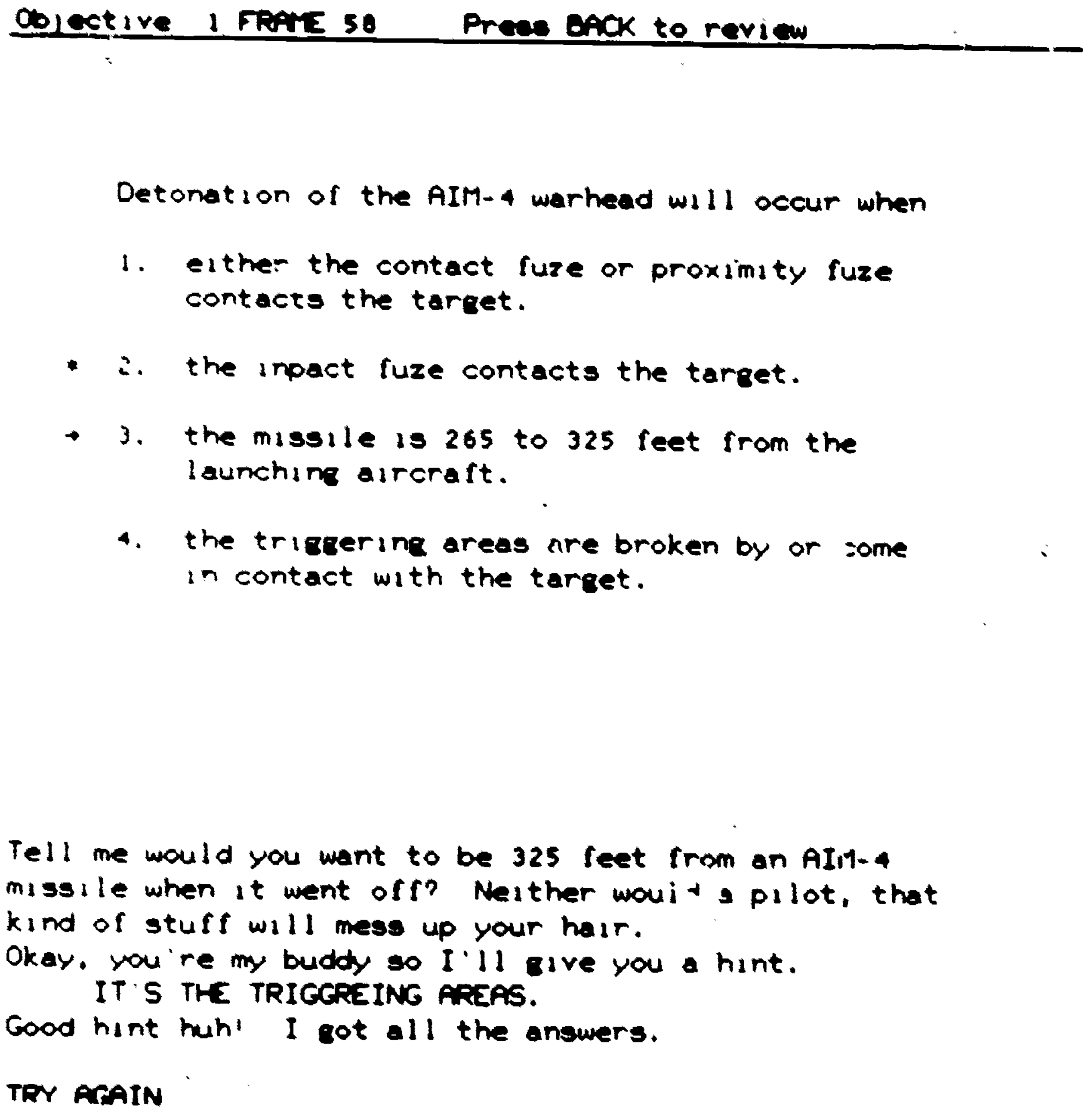
GUIDANCE and CONTROL	WARHEAD	ROCKET MOTOR

↑

TYPICAL MISSILE
(three sections)

Press NEXT to go on or C to comment on this text.

Figure 21. Presentation Program Display of a Text Frame.



Objective 1 FRAME 58 Press BACK to review

Detonation of the AIM-4 warhead will occur when

1. either the contact fuze or proximity fuze contacts the target.

* 2. the impact fuze contacts the target.

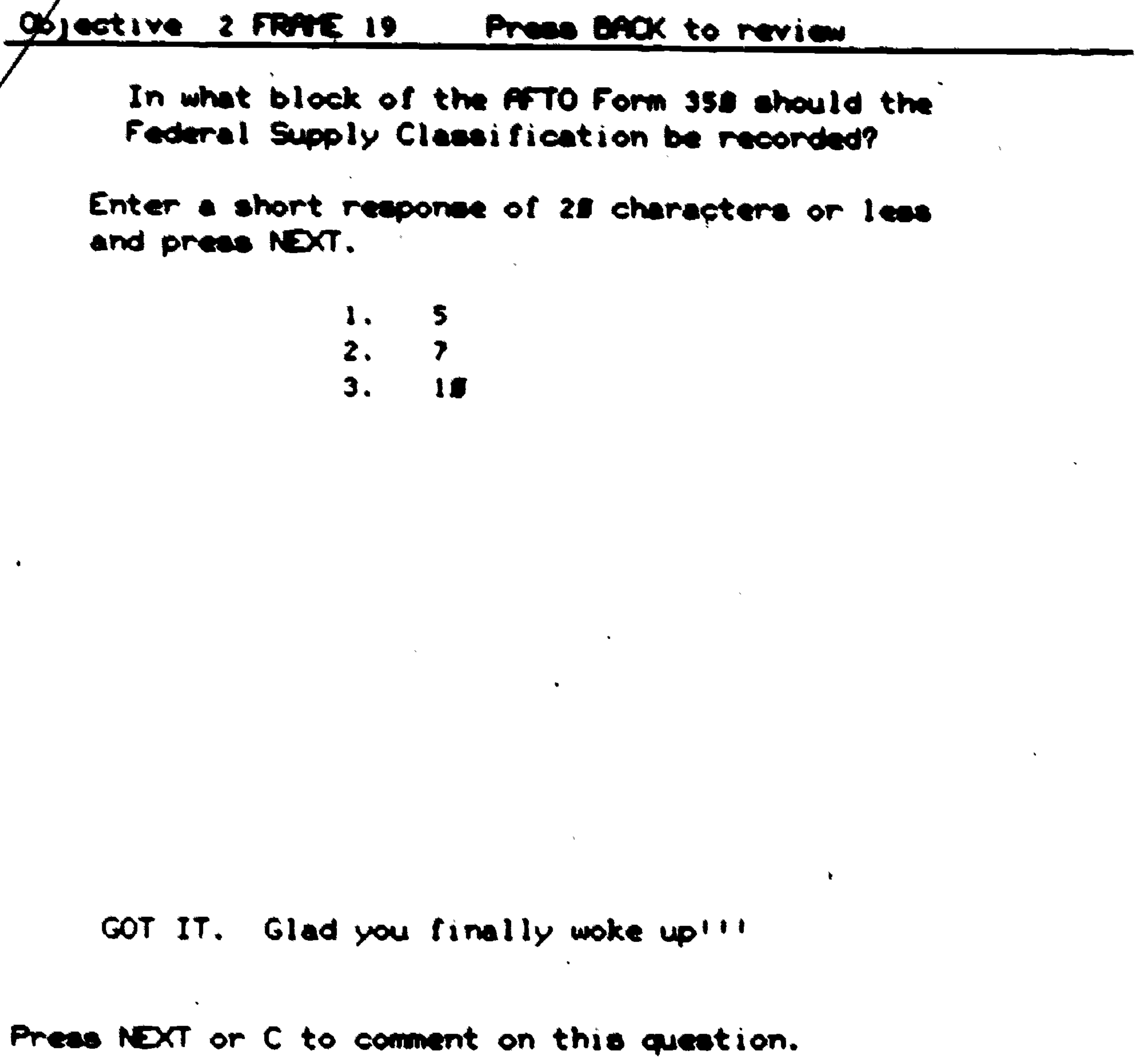
→ 3. the missile is 265 to 325 feet from the launching aircraft.

4. the triggering areas are broken by or come in contact with the target.

Tell me would you want to be 325 feet from an AIM-4 missile when it went off? Neither would a pilot, that kind of stuff will mess up your hair.
Okay, you're my buddy so I'll give you a hint.
IT'S THE TRIGGREING AREAS.
Good hint huh! I got all the answers.

TRY AGAIN

Figure 22. Presentation Program Display of Multiple Choice Question Showing Feedback and Second Attempt Prompt.



```
Objective  2 FRAME 19     Press BACK to review

      In what block of the AFTO Form 35Ø should the
      Federal Supply Classification be recorded?

    Enter a short response of 2Ø characters or less
    and press NEXT.

                  1.   5
                  2.   7
                  3.   1Ø




     GOT IT.  Glad you finally woke up!!!


  Press NEXT or C to comment on this question.
```

Figure 23. Presentation Program Display of Constructed Response Question Answered Correctly on the Third Attempt.

received a standard feedback message for a correct response and is ready to continue.

Over the course of a long module, the author may want to encourage the student to take one or more breaks; to exit the module and leave the terminal for a short rest period. Other forms of module interruptions can occur due to computer failure, end of shift, or breaks for meals. In each case, the student can log off or, if the Presentation Program does not receive a keypress for a specified period (currently 15 minutes), it is assumed that the student has left the terminal and, after displaying an inquiry as to whether anyone is there, the student is logged off automatically. If a module is interrupted for any reason, the Presentation Program automatically restarts the module at the frame on which the interruption occurred when the student logs back onto the terminal.

After a period of instruction and practice, an objective typically ends with a series of test questions. Given the criterion that a certain number of the questions must be answered correctly, the author-defined branching would normally route the student to the end of the objective as soon as the criterion has been met. If, on the other hand, the student's performance is below criterion, the student would normally not exit the objective until troublesome points have been reviewed and retested with additional test items.

Upon exiting the objective, the student will pass through a frame against which either an Objective Passed or Objective Failed Flag has been set. If the objective is considered to be prerequisite to a subsequent objective, a Lesson Failed Flag would normally be set against the same frame as the Objective Failed Flag. The student will continue through the objectives, in sequence, until all of the objectives have been presented or until the Presentation Program encounters a Lesson Passed or Lesson Failed Flag.

When a Lesson Passed or Lesson Failed Flag is encountered, the Presentation Program generates a module test form containing a list of any objectives failed and a lesson passed or lesson failed designator. The Program passes this form to the main AIS management program, the Adaptive Model. The Adaptive Model records the student's performance on the lesson and generates the student's next assignment. This assignment is displayed on the terminal for the student to copy. The student is then logged off the terminal and the module's instructional resource (the terminal) is returned to the resource pool for reassignment. The student can then obtain a hard copy printout of the next assignment by submitting a request form to a management terminal.

Author/Reviewer Mode. Access to the CAI Presentation Program is not limited to students. Lesson authors and reviewers may wish to run the Presentation Program to verify accuracy of module content and to view the module from the student's perspective. Author and reviewer access to the Presentation Programs is, however, handled by standard AIS program access methods rather than being under the control of the Adaptive Model. The CAI Presentation Program differentiates between students and registered authors and reviewers. The author/reviewer does not have a Student Data Profile record containing a current assignment, and therefore is required to enter the identifier of the CAI module to be presented.

Having accessed a particular module, the author/reviewer may, with a few exceptions, take the module like a student. Unlike a student, an author/reviewer may override the frame control logic of the Presentation Program and request the presentation of any frame within an objective through the use of a special function key. In addition, the author/reviewer may always enter comments about the material that is being presented. Student comments are only elicited and accepted if the Student Comment Module Flag has been set to True. Completion of a module in author/reviewer mode does not result in submission of a lesson completed or failed form to the Adaptive Model. In all other respects, the user's interaction with the Presentation Program is identical to that of a student.

Student Performance Data Acquisition and Analysis

The CAI Data Acquisition and Analysis system consists of four major components: data recording routines in the Presentation Programs; a Data Collection Program which moves student performance data from disk to tape; a Data Analysis Report Program which generates three different types of reports; and a Report Submittal Program which facilitates users' requests for specific reports.

The student performance data collection and analysis process begins with data recording routines embedded in the Presentation Programs. These routines operate interactively with data being recorded directly onto disk. The information recorded is segregated into two files: Response Point data, which are recorded following every frame; and Decision Point data, recorded only at specified Decision Points. Periodically, the data are dumped from disk to tape by the batch CAI Data Collection Program. The CAI Response and Decision Point history tapes provide the primary data source for the CAI Data Analysis Report Program. The Report Program can, however, also retrieve recent data stored in the disk files for applications requiring small, current student samples. There are both background (CAMIL) and batch (PASCAL) versions of the Data Analysis Report Program. The former is used to access data on disk while the latter accesses only the tape data. A single CAMIL program provides for setup and submittal of both the background and batch versions of the Report Program.

The data collection process and the reports available are discussed in greater detail in the following subsections. Although the process employed is somewhat different than for performance data collection, the storage and retrieval of student and reviewer comments are also described here.

Data Collection and Storage. Whether student performance data and comments are collected during the presentation of a CAI module is dependent on whether the module author has set the appropriate data collection flags for that module via the Authoring Editor. That is, for data within one of the three categories (Response, Decision Point, and Comments) to be saved, the author must have set the Data Collection Flag for that category to True. This philosophy of limited data collection was adopted to avoid generation and storage of the immense amounts of data which would otherwise occur. The intent is that data be collected for formative and summative evaluation purposes but not during normal operations except for consciously initiated sampling.

Response data represents the most detailed data category. The data are collected at the end of each frame presented to the student, regardless of frame type, and include the following items:

1. The student's Student Account Number.

2. The frame identifier.

3. The number of the student's pass through this frame.

4. The current date.

5. The current time in minutes after midnight.

6. The total time, in seconds, spent on the frame.

7. The time, in seconds, spent in review mode if review was initiated from this frame.

If the frame is a question frame, the following data are also collected:

8. The number of attempts made to answer the question.

9. The number of the alternative selected, by attempt number (where alternative numbers are also assigned to the various anticipated responses and the category of unanticipated response for constructed response questions).

10. The response latency, by attempt number.

11. The number of unanticipated responses.

12. The text of up to five unanticipated responses.

Decision Point data are collected at the end of the module, at the end of each objective, and at the end of each frame against which a

Decision Point Flag has been set. The data collected include the following:

1. The student's Student Account Number.
2. The type of decision point (frame, objective or module).
3. The frame identifier.
4. The number of the student's pass through this frame.
5. The current date.
6. The current time in minutes after midnight.
7. The elapsed time since the last decision point.
8. The number of questions presented since the last decision point.
9. The number of questions answered correctly since the last decision point.
10. The number of branching logic decisions processed for the current frame; and for each branching logic instruction processed:
11. The branching type (Pre-frame, Response or After-frame).
12. The number of the instruction within its type.
13. The branch actually taken, if any.

Both Response Point and Decision Point data are stored on disk as they are collected. Periodically (e.g., once a week), the CAI Data Collection Program is run to delete the oldest data from the disk Response and Decision Point Files and transfer the data to the CAI History tapes.

Collection and storage of student (and reviewer) comments is handled somewhat differently. First, of course, the student or reviewer must take overt action to enter a comment against a frame as opposed to data collection being transparent to the user. Up to eight comments are stored for each frame in a "circular" file on disk. That is, when a ninth comment is entered against that frame, its content replaces that of the first (oldest) comment. The content of the Comment files is never transferred to tape.

<u>Data Analysis Reports</u>. There are four different CAI Data Analysis Reports available to authors and evaluators: the Decision Point Data

Report, the Response Analysis Data Report, the Unanticipated Response Report, and the Comments Listing. In addition, standard AIS CMI reports can be used to provide a description of overall module performance.

All reports are requested from an interactive terminal. For the three student performance reports, requests are submitted via a CAMIL program (CAI Reports Program) which prompts the user to enter the various report request parameters. Thus, the user does not need to learn how to set up job control and input data cards. The report request parameters include:

1. The module identifier (Course, Course Version, Block, Lesson and Module numbers).

2. The type of report.

3. The date constraints (the period from which data are to be drawn).

4. The input medium (disk or tape).

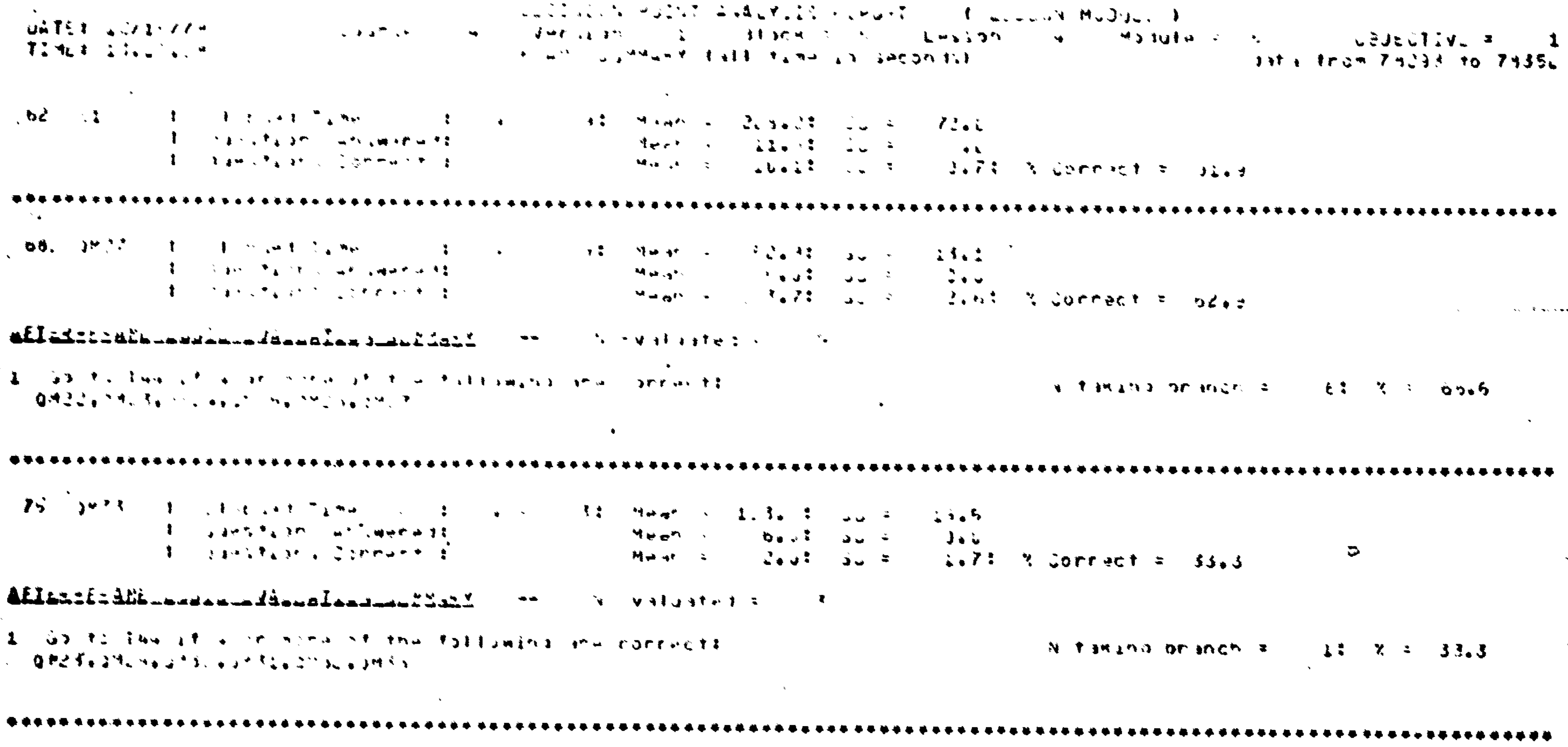
5. The objectives within the module which are to be reported.

6. If desired by the user, the numbers of the specific frames within each objective to be reported (where a set of frames is defined by the numbers of the first and last frames in the set).

Once the request parameters have been defined, the report submittal program sets up the necessary files, submits the job to generate the report on the central line printer, and gives the user the job number which will appear on the report printout.

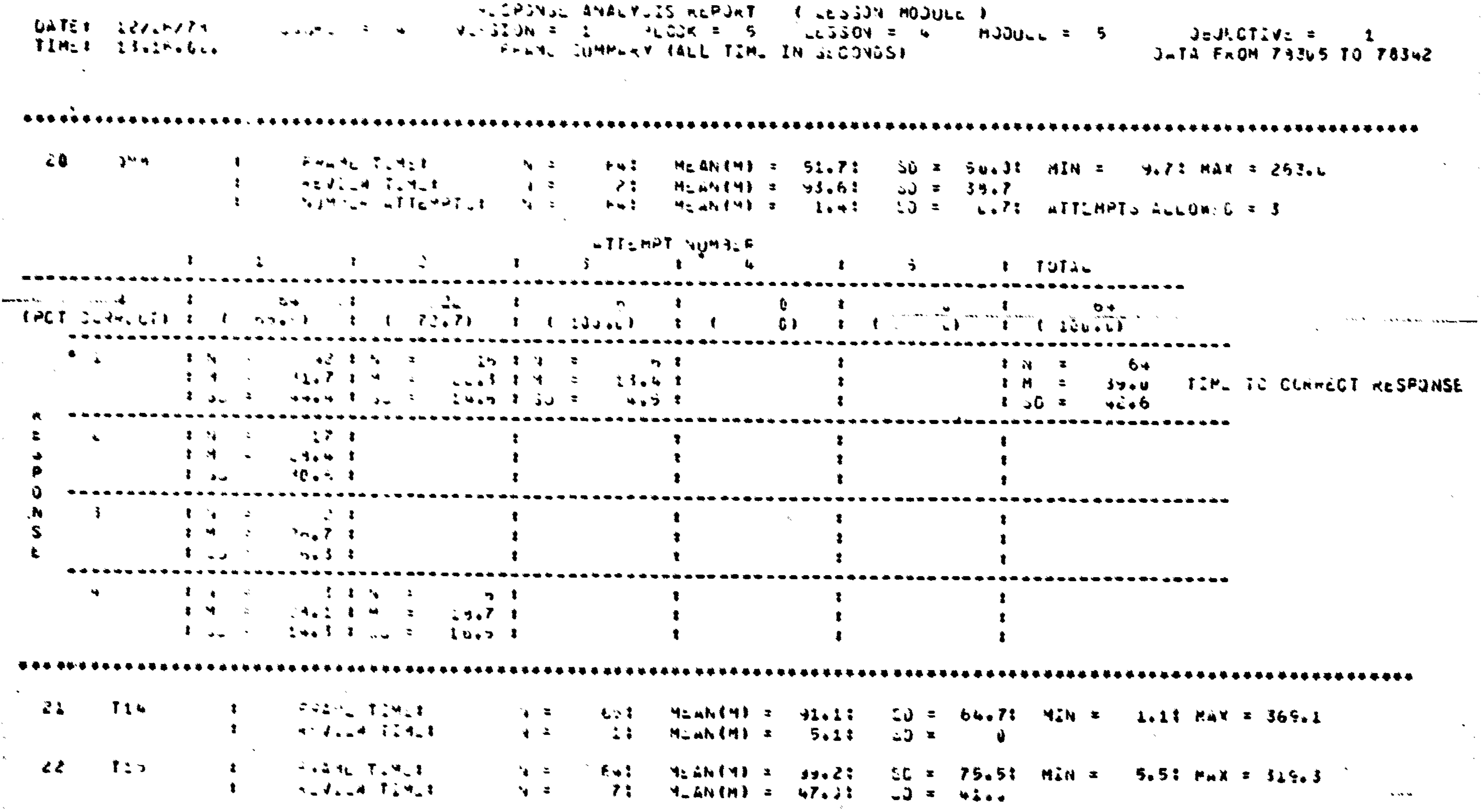
The Decision Point Report is generated from data, stored either on disk or tape, in the Decision Point Data File. This report provides a summary of student performance within each objective and within those intraobjective segments (sets of frames) which the author has defined by setting Decision Point Flags at the beginning and end of each segment. An example page from the Decision Point Report is presented in Figure 24. Each component of the Report contains the number and name of the frame representing that Decision Point; whether the data reported pertains to students' first, second, or subsequent passes through that point; the elapsed time, number of questions answered and number answered correctly since the last Decision Point; the branching logic evaluated at this point; and the number and percentage of students taking each branch.

An example page from the Response Analysis Report, is presented in Figure 25. Each component of the report is identified by the number and name of the relevant frame and whether the data reported pertains to students' first, second, etc., pass through this frame. For frames



```
DATE: [illegible]              [illegible]     DECISION POINT ANALYSIS REPORT     ( [illegible] MODULE )
TIME: [illegible]                              Version   1    Block [illegible]   Lesson   [illegible]   Module [illegible]          OBJECTIVE =    1
                                               [illegible] SUMMARY (ALL TIME IN SECONDS)                                    Data from 74238 to 74356

62  [illegible]     !  [illegible] Time        :   [illegible]     :  Mean =  2[illegible]:  SD =   72.[illegible]
                    !  [illegible] Answered:                          Mean =   11.[illegible]:  SD =     [illegible]
                    !  [illegible] Correct :                          Mean =   16.[illegible]:  SD =    3.7:  % Correct =  [illegible]
*****************************************************************************************************************************

68. [illegible]     !  [illegible] Time        :   [illegible]     :  Mean =  [illegible]:  SD =   15.[illegible]
                    !  [illegible] Answered:                          Mean    [illegible]:  SD =    [illegible]
                    !  [illegible] Correct :                          Mean =   [illegible].7:  SD =    2.6:  % Correct =  62.[illegible]
AFTER-[illegible]                    --    % Validated =    %

1  Go to [illegible] if [illegible] of the following are correct:                              N taking branch =     6:  % =  66.6
   QM22,[illegible]

*****************************************************************************************************************************

75  QM73            !  [illegible] Time        :   [illegible]     :  Mean =  1.3[illegible]:  SD =   15.5
                    !  [illegible] Answered:                          Mean =    6.[illegible]:  SD =    3.6
                    !  [illegible] Correct :                          Mean =    2.[illegible]:  SD =    1.7:  % Correct =  33.3
AFTER-[illegible]                    --    % Validated =    %

1  Go to [illegible] if [illegible] none of the following are correct:                         N taking branch =     1:  % =  33.3
   QM25,[illegible]

*****************************************************************************************************************************
```

Figure 24. Example Page from Decision Point Analysis Report.



```
                                        RESPONSE ANALYSIS REPORT   ( LESSON MODULE )
DATE: 12/[illegible]/7[illegible]      [illegible] = 4    VERSION = 1    BLOCK = 5    LESSON = 4    MODULE = 5         OBJECTIVE =    1
TIME: 13.1[illegible].6[illegible]                 FRAME SUMMARY (ALL TIME IN SECONDS)                      DATA FROM 79305 TO 78342

*************************************************************************************************************

 20    [illegible]        :     FRAME TIME:        N =    64:   MEAN(M) =  51.7:   SD =  50.3:  MIN =    9.7: MAX = 263.[illegible]
                          :     REVIEW TIME:       N =     7:   MEAN(M) =  93.6:   SD =  39.7
                          :     NUMBER ATTEMPTS:   N =    64:   MEAN(M) =   1.4:   SD =   [illegible].7:  ATTEMPTS ALLOWED = 3

                                                  ATTEMPT NUMBER
                 :     1      :     2      :     3      :     4      :     5      : TOTAL
--------------------------------------------------------------------------------------------------
         [illegible]    :      64    :     [illegible]    :      [illegible]     :      0     :     [illegible]    :     64
(PCT CORRECT)  :  ( [illegible])  :  ( 72.7)   :  ( 100.0)  :  (     0)   :  (     0)   :  ( 100.0)
   -----------------------------------------------------------------------------------------------------
   * 1         : N  =    42 : N  =    15 : N  =     [illegible] :            :            : N  =     64
               : M  =  41.7 : M  =  [illegible].3 : M  =  13.4 :            :            : M  =   39.0   TIME TO CORRECT RESPONSE
               : SD =  44.4 : SD =  14.[illegible] : SD =   4.5 :            :            : SD =   42.6
 R -------------------------------------------------------------------------------------------------
 E   2         : N  =    17 :            :            :            :            :
 S             : M  =  [illegible].4 :            :            :            :            :
 P             : SD =  40.[illegible] :            :            :            :            :
 O -------------------------------------------------------------------------------------------------
 N   3         : N  =     2 :            :            :            :            :
 S             : M  =  [illegible].7 :            :            :            :            :
 E             : SD =   [illegible].3 :            :            :            :            :
   -------------------------------------------------------------------------------------------------
     4         : N  =     [illegible] : N  =     [illegible] :            :            :            :
               : M  =  [illegible].1 : M  =  [illegible].7 :            :            :            :
               : SD =  [illegible].1 : SD =  10.[illegible] :            :            :            :

*************************************************************************************************************

 21    T14        :     FRAME TIME:        N =    6[illegible]:   MEAN(M) =  91.1:   SD =  64.7:  MIN =    1.1: MAX = 369.1
                  :     REVIEW TIME:       N =     1:   MEAN(M) =   5.1:   SD =     0

 22    T15        :     FRAME TIME:        N =    64:   MEAN(M) =  [illegible].2:   SD =  75.5:  MIN =    5.5: MAX = 319.3
                  :     REVIEW TIME:       N =     7:   MEAN(M) =  47.[illegible]:   SD =  41.[illegible]
```

Figure 25. Example Page from Response Analysis Report.

other than question frames, only time data are reported. For question frames, a matrix format is used to present student performance and response latency data as a function of the response (multiple choice alternative or constructed response) selected on successive attempts. The margins of the matrix provide a summary of student performance on the question (total percentage correct, percentage correct by attempt, and total time to correct response) while the matrix cells provide a more detailed picture of how students reacted to the question.

An example page from the Unanticipated Response Report is presented in Figure 26. For each constructed response question, each unique unanticipated response is listed in order of frequency of occurrence together with the number of times which that particular response has been entered.

Comments Listings are requested via the Authoring Editor rather than the CAI Reports Program. Frames against which comments have been made are indicated on the Editor's Frame List display for each objective. The user can request that comments made on a particular frame be displayed at the terminal or that comments on one or more frames be listed on the central line printer. The user also has the option of having the comments purged from the file as they are displayed or listed.

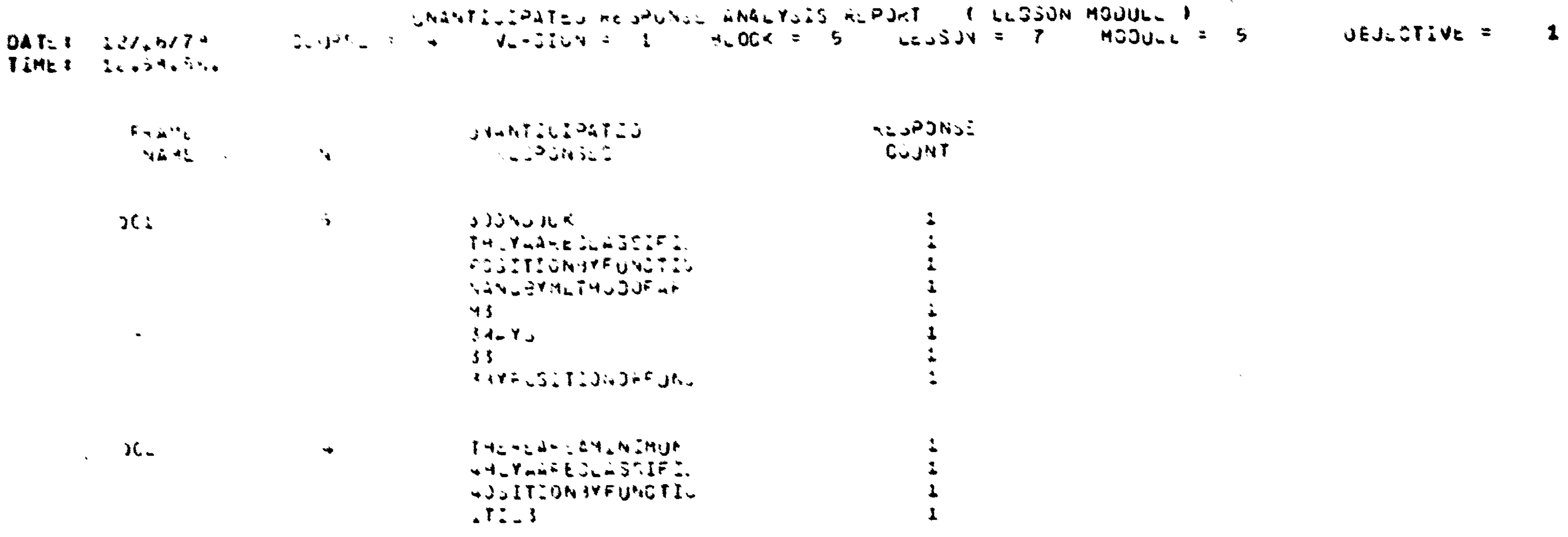
CAI Material Print Program

As authors create, review, and revise CAI modules, it is often useful to work from a hard copy printout of the module's content in addition to, or in place of, the displays provided by the Authoring Editor. There are also instances in which hard copy printout is desirable for student use. A feature of the Authoring Editor is the capability to request a variety of printed listings of CAI module's content and branching logic. The Editor queries the author for the desired print options and then initiates a special background (non-interactive, low priority) program to produce the printouts.

Author Listings. In addition to listings of student comments, four different types of printer listings are available to authors, ranging from summary information to detailed listings of frame contents. The option of requesting multiple copies is available for each type of listing.

At the most general level, the Module Summary Listing provides an overview of all of the CAI modules, operational or under development, currently defined in the data base. The information provided for each module includes the module identifier (Course Number through Module Number), module title, author's ID (Social Security Number), and the number of objectives defined within the module.

For a particular module, the Frame Summary Listing provides an overview of the content of individual objectives. An example of this listing



```
                                   UNANTICIPATED RESPONSE ANALYSIS REPORT    ( LESSON MODULE )
DATE: 12/16/74     COURSE =  4     VERSION =  1     BLOCK =  5     LESSON =  7     MODULE =  5          OBJECTIVE =     1
TIME: [illegible]

        FRAME                          UNANTICIPATED                         RESPONSE
         NAME             N              RESPONSES                            COUNT

        001               5            [illegible]                              1
                                       THEYARECLASSIFI                          1
                                       POSITIONBYFUNCTIO                        1
                                       NAMEBYMETHODOFAP                         1
                                       MS                                       1
                                       BRAYS                                    1
                                       33                                       1
                                       BYPOSITIONOFFUNC                         1

        002               4            THEREAREAMINIMUM                         1
                                       THEYAREECLASSIFI                         1
                                       POSITIONBYFUNCTIO                        1
                                       [illegible]                              1
```

Figure 26. Example Page from Unanticipated Response Report.

is presented in Figure 27. The information provided by this listing is essentially the same as the Editor's Frame List Display . Each frame in the objective is listed by number and frame name. The existence of any branching logic and Frame Flags is noted and the maximum number of attempts allowed to answer questions are shown. If a frame is an alias (i.e., references another frame), the referenced frame is identified.

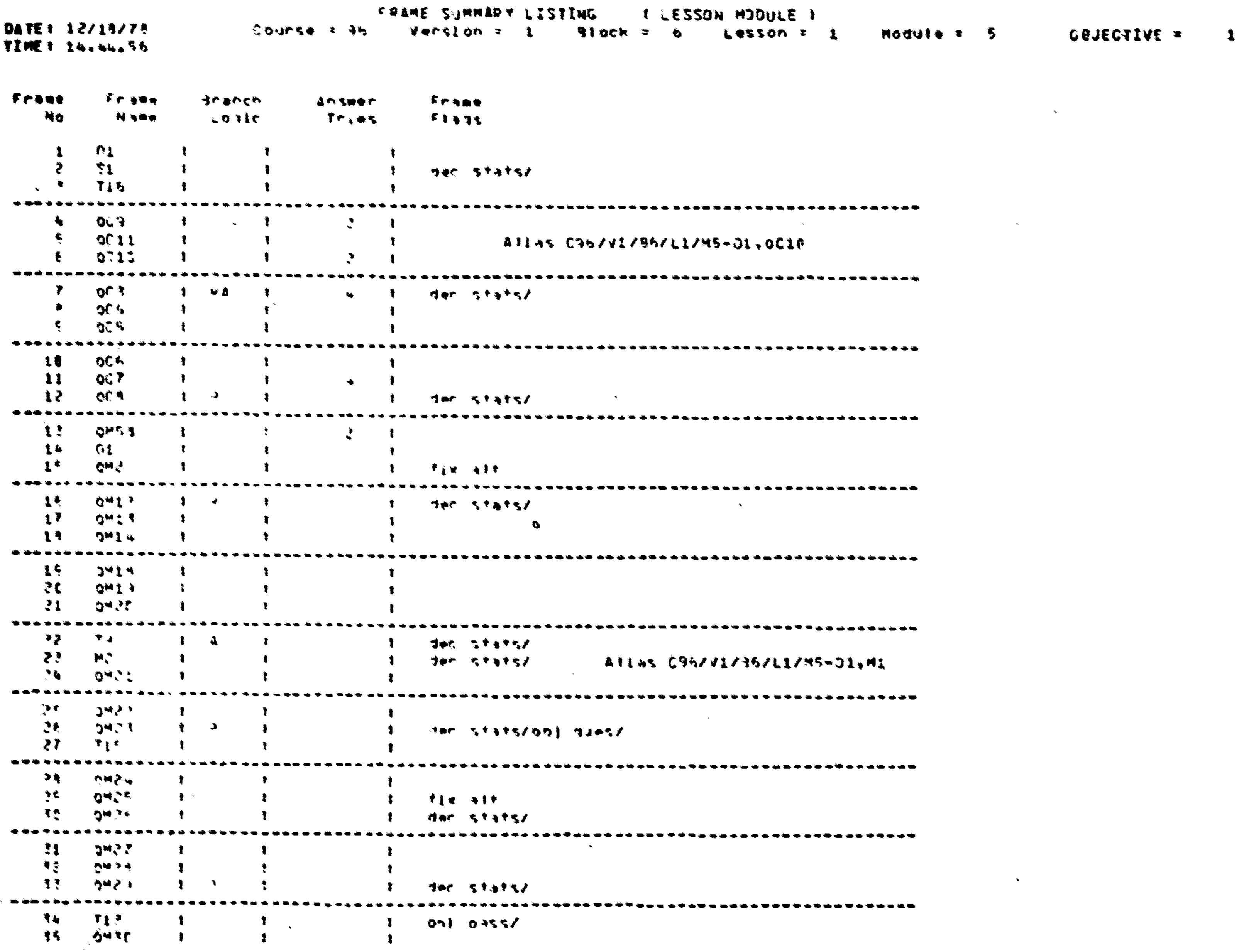
The most frequently used printout is probably the Frame Contents Listing: the complete printout, by frame, of all text and question material. Such a listing can be requested for an entire module, an individual objective, a specified set of frames or a single frame. The material contained in a textual content frame, up to the full four pages, is printed on a single printer page with appropriate headings. An example printout of a Text frame is shown in Figure 28. For a Question frame, the printout includes the question stem, the alternatives or anticipated responses with the correct answers denoted, and all author-supplied feedback and prompt messages. An example is presented in Figure 29.

Finally, the Branching Logic Listing, an example of which is shown in Figure 30, provides a hard copy listing of all of the branching logic which has been defined for frames within an objective. The format in which information is presented is similar to the Editor's Branching Logic displays.

Printed Materials for Student Use. In addition to the various author's listings, hard copy printout of a CAI module's content can be requested in a format appropriate for direct use by students as a programmed text. Special purpose (Documentation and Branching Decision) frames are automatically suppressed, and the author can elect to suppress any other frames by setting "Skip this Frame" flags via the Editor. All other frames are printed in the order in which they occur in the module. Branching logic is simply ignored.

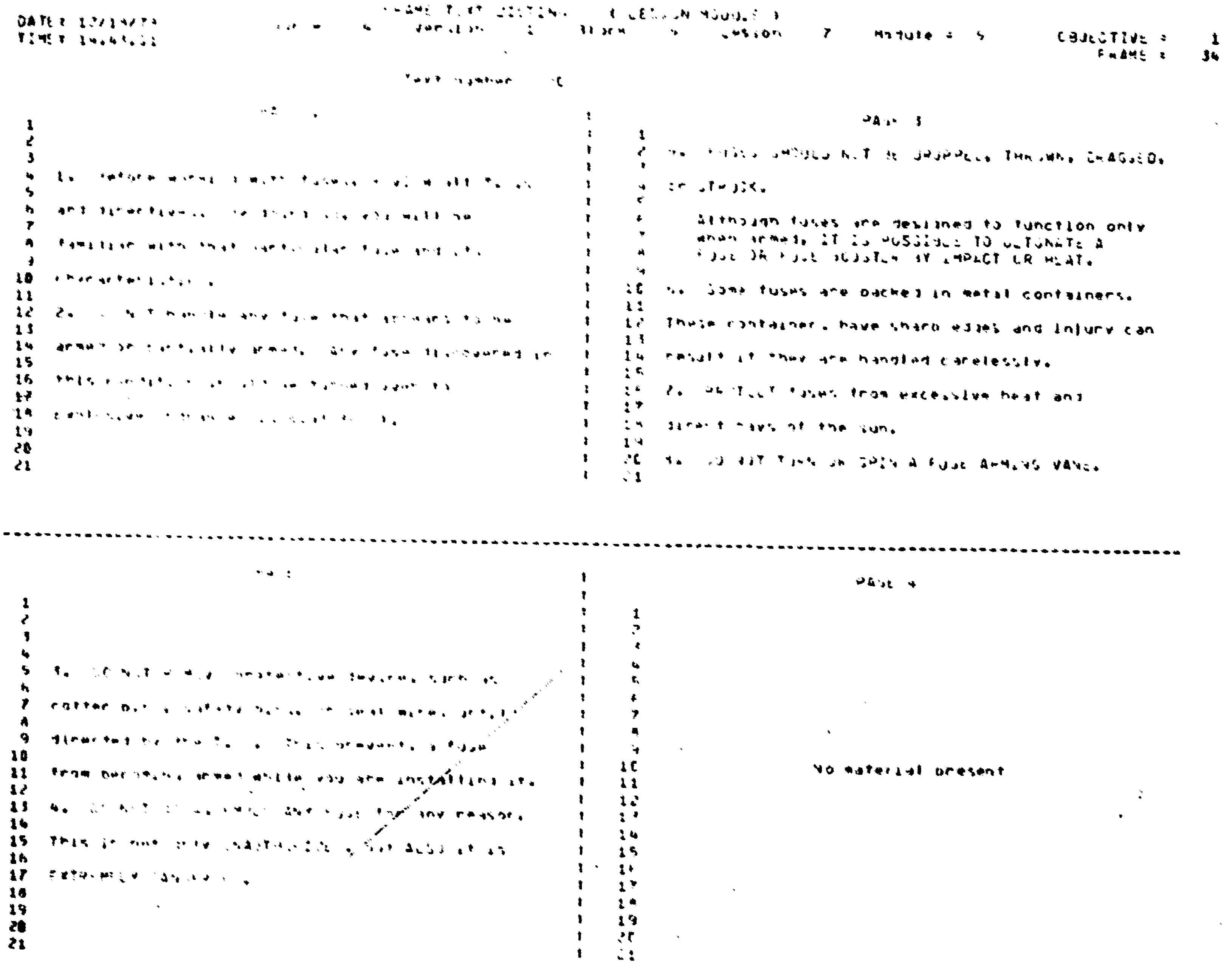
For textual content frames, one screen display (a frame page) is printed on each page. For question frames, each question is identified by a number, assigned sequentially. For constructed response questions, only the question stem is printed, while the alternatives and their numbers are printed for multiple choice questions. The answers to all questions, identified by number, are listed on a test key page following the body of the module. Page numbering is provided automatically.

Material is currently printed in a double-spaced format. Boxes which the author has used to emphasize material in the module are shown, as are any of the simple graphics which the author may have provided. For such graphics, a "%" symbol is substituted for the solid, "unrecognized character" symbol typically employed. As an option, the author may request that the material be printed on special unlined, 8 1/2" by 11" paper. The option of printing multiple copies is also available.



```
                                          FRAME SUMMARY LISTING      ( LESSON MODULE )
DATE: 12/18/78            Course = 96    Version =  1    Block =  6    Lesson =  1    Module =  5        OBJECTIVE =    1
TIME: 14.44.56

Frame    Frame     Branch     Answer     Frame
  No      Name      Logic      Tries      Flags

   1    O1      !          !            !
   2    S1      !          !            !   dec stats/
   3    T16     !          !            !
--------------------------------------------------------------------------------------------
   4    OC9     !          !       2    !
   5    OC11    !          !            !          Alias C96/V1/96/L1/M5-O1,OC10
   6    OC13    !          !       2    !
--------------------------------------------------------------------------------------------
   7    OC3     !   WA     !       4    !   dec stats/
   8    OC4     !          !            !
   9    OC5     !          !            !
--------------------------------------------------------------------------------------------
  10    OC6     !          !            !
  11    OC7     !          !       4    !
  12    OC8     !   2      !            !   dec stats/
--------------------------------------------------------------------------------------------
  13    OM53    !          !       2    !
  14    O1      !          !            !
  15    OM2     !          !            !   fix alt
--------------------------------------------------------------------------------------------
  16    OM12    !   4      !            !   dec stats/
  17    OM13    !          !            !
  18    OM14    !          !            !
--------------------------------------------------------------------------------------------
  19    OM18    !          !            !
  20    OM19    !          !            !
  21    OM20    !          !            !
--------------------------------------------------------------------------------------------
  22    T1      !   A      !            !   dec stats/
  23    M2      !          !            !   dec stats/      Alias C96/V1/96/L1/M5-O1,M1
  24    OM21    !          !            !
--------------------------------------------------------------------------------------------
  25    OM22    !          !            !
  26    OM23    !   2      !            !   dec stats/obj ques/
  27    T15     !          !            !
--------------------------------------------------------------------------------------------
  28    OM24    !          !            !
  29    OM25    !          !            !   fix alt
  30    OM26    !          !            !   dec stats/
--------------------------------------------------------------------------------------------
  31    OM27    !          !            !
  32    OM28    !          !            !
  33    OM29    !   1      !            !   dec stats/
--------------------------------------------------------------------------------------------
  34    T13     !          !            !   obj pass/
  35    OM30    !          !            !
```

Figure 27. Example Frame Summary Listing Printout.

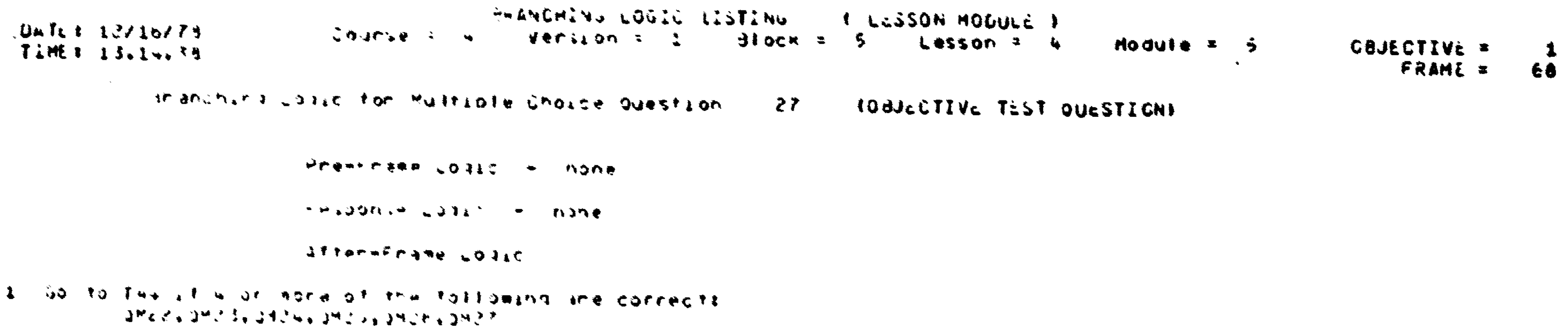


```
DATE: [illegible]          FRAME TEXT LISTING  ( LESSON MODULE )
TIME: [illegible]          [illegible]  VERSION  1  BLOCK  [illegible]  LESSON  7  MODULE = 5     OBJECTIVE =   1
                                                                                                      FRAME =  34
                                  TEXT NUMBER  [illegible]

              PAGE [illegible]                     !                  PAGE 3
                                                   !
 1. [illegible] FUSES [illegible] ALL [illegible]  !  [illegible] FUSES SHOULD NOT BE DROPPED, THROWN, DRAGGED,
 AND [illegible] WILL BE                           !  OR STRUCK.
 FAMILIAR WITH THAT PARTICULAR FUSE AND ITS        !     ALTHOUGH FUSES ARE DESIGNED TO FUNCTION ONLY
 CHARACTERISTICS.                                  !     WHEN ARMED, IT IS POSSIBLE TO DETONATE A
                                                   !     FUSE OR FUSE BOOSTER BY IMPACT OR HEAT.
 2. [illegible] HANDLE ANY FUSE THAT APPEARS TO BE !  [illegible] SOME FUSES ARE PACKED IN METAL CONTAINERS.
 ARMED OR PARTIALLY ARMED. [illegible] FUSE        !  THESE CONTAINERS HAVE SHARP EDGES AND INJURY CAN
 [illegible] IN                                    !  RESULT IF THEY ARE HANDLED CARELESSLY.
 THIS [illegible]                                  !  2. PROTECT FUSES FROM EXCESSIVE HEAT AND
 [illegible]                                       !  DIRECT RAYS OF THE SUN.
                                                   !  3. DO NOT TURN OR SPIN A FUSE ARMING VANE.
-------------------------------------------------------------------------------------------------------
              PAGE [illegible]                     !                  PAGE 4
                                                   !
 3. DO NOT [illegible] DEVICES, SUCH AS            !
 COTTER PINS, SAFETY [illegible] WIRE, [illegible] !
 DIRECTED [illegible] THIS PREVENTS A FUSE         !
 FROM BECOMING ARMED WHILE YOU ARE INSTALLING IT.  !          NO MATERIAL PRESENT
 4. DO NOT [illegible] ANY FUSE FOR ANY REASON.    !
 THIS IS NOT ONLY [illegible] BUT ALSO IT IS       !
 EXTREMELY [illegible]                             !
```

Figure 26. Example Page from Frame Contents Listing Showing a Text Frame.

```
                                 FRAME TEXT LISTING    ( LESSON MODULE )
DATE: 12/16/74        Course =  4    Version =  1    Block =  5    Lesson =  4    Module =  5      OBJECTIVE =    1
IME: 13.14.34                                                                                       FRAME =    9

                                   Multiple Choice Question number    4

              QUESTION                     !                 FEEDBACK MESSAGES
                                           !  1  No.  Both rocket and missiles have warheads.
Rockets differ from missiles in that,      !
                                           !
  1.  rockets have no warhead.             !  2  No.  rockets do not have any guidance system.
                                           !
  2.  rockets have a guidance system.      !  3  Way off!  The type of propellent doesn't matter.
                                           !
  3.  missiles use liquid propellent motors.!
                                           !
* 4.  missiles have a guidance system.     !-------------------------------------------------------
                                           !
                                           !                 PROMPT MESSAGES
                                           !
                                           !  1  There are three parts of a missile one of them is
                                           !     not used on a rocket.
                                           !
                                           !  2  The one single thing that makes a missile a MISSILE
                                           !     is the GUIDANCE SYSTEM.
```

Figure 29. Example Page from Frame Contents Showing a Question Frame.



```
                                   BRANCHING LOGIC LISTING    ( LESSON MODULE )
 DATE: 12/16/79          Course =  4    Version =  1    Block =  5    Lesson =  4    Module =  5        OBJECTIVE =    1
 TIME: 13.14.39                                                                                           FRAME =   68

          Branching Logic for Multiple Choice Question     27     (OBJECTIVE TEST QUESTION)

                      Pre-Frame Logic  =  none

                      Response Logic   =  none

                      After-Frame Logic

 1   Go to T44 if 4 or more of the following are correct:
          3M22,3M23,3M24,3M25,3M26,3M27
```

Figure 30. Example Page from Branching Logic Listing.

There are a variety of uses for such printouts. They can be used as hard copy backups for students assigned CAI modules in the case of computer failure. They are useful to instructors for answering the questions of students assigned CAI modules. Their most important function however, may well be as a first step toward the on-line development, evaluation, and revision of materials intended for off-line use.

Authoring System Documentation

A CAI Authoring Editor User's Manual was written to describe the CAI concepts supported by the Editor, the mechanics of its use, the CAI Presentation Programs, the various CAI Reports available and the procedures by which they are requested, and the availability and use of the different types of author and student printouts. The Manual was actually developed on-line as a CAI module (without branching or question frames) and is available to novice authors via any of the AIS interactive terminals. Hard copy versions of the manual are also produced as needed by the CAI Material Print Program.

This approach has assured rapid and easy revision of the manual to provide up-to-date documentation as the Editor and its various supporting programs are expanded and refined. Under a companion, Authoring Procedures contract, the Manual is being expanded to include information on the selection of content for CAI treatment, instructional strategies appropriate to CAI, and guidelines for evaluation and revision of CAI modules.

Although it had not been implemented at the time this report was prepared, it is intended that a second type of Editor documentation will be provided. "Help" routines will be imbedded in the Editor itself. That is, through the use of a special Function Key, the author will be able to access information pertaining to the use of the Editor Options which are available to him/her from the currently displayed page of the Editor. These Help routines will access the information contained in appropraite frames of the User's Manual. Thus, most required revisions to the documentation will only need to be made in one place, in the manual itself. The Help sequences will then access the updated information automatically.

Documentation to support subsequent maintenance and revisions of the Editor itself, the Presentation Programs and the various supporting programs is provided by a Part II, Product Development Specification. This Part II Spec was also produced on-line (via a different, less structured Editor), is stored on tape and is accessible for the production of hard copy documentation.

IV IMPLEMENTATION AND EVALUATION

In addition to the CAI support software effort described here, the CAI development project also contained a second authoring procedures effort. The purposes of this second effort were to (a) define and document a set of procedures for CAI materials selection, production, evaluation, and implementation in the AIS environment, (b) evaluate the utility of the total CAI authoring system by developing, implementing and evaluating a number of CAI modules in an AIS course, and (c) train a small number of ATC personnel in the use of the procedures and support software. The results of this effort are reported in full in Part II of this report (Lewis, Lovelace, Mahany and Judd, in press) and will be outlined only briefly here.

The authoring procedures effort began with a rough definition of the procedural model which was then revised and refined on the basis of experience. Following the steps outlined by the procedural model, work began by selecting candidate lessons for the CAI demonstration. Of the four AIS courses, the Weapons Mechanic course was considered the most promising because of its combination of varied subject matter, large student flow, and recent relatively poor field performance data. Standard AIS CMI reports were used to identify six lessons in two blocks of the course which demonstrated unacceptably high lesson test failure rates, relatively high failure rates on the end-of-block tests for the objectives contained in these lessons, and which contained subject matter unlikely to be modified in the near future. These lessons were targeted for three types of CAI application: (a) six modules to be used as alternative treatments on students' first pass through the block, (b) two block review modules containing CAI treatment of the objectives from these lessons, and (c) two block remediation modules, again containing CAI treatment of the objectives from the six lessons.

After examining the existing materials and tests, it was concluded that neither the lesson tests nor those portions of the block tests pertaining to the relevant objectives were adequately valid or reliable for evaluating the effectiveness of the CAI treatments. Consequently, both forms of the two block tests and two forms of each of the six lesson tests were substantially revised and expanded to more closely match the stated performance requirements of the Specialty Training Standard. Once the revised tests had been approved by course personnel, they were implemented in paper-and-pencil form for all students in the course.

Work then began on developing six CAI modules for administration during students' first pass through the block. While existing materials were available in the form of programmed text and, in some cases, audio-visual modules, the content was substantially revised to more closely match the requirements of the Specialty Training Standard. Instructional strategies were also, of course, tailored for CAI presentation.

Work started on three of the modules before the Authoring Editor was ready, but after it had been designed and the module structure defined. Therefore, the earliest authoring was done on paper display forms. When a rudimentary form of the Editor became available, text and questions which had been prepared on the forms were input by a secretary. It was thought that some of the authors might prefer the use of forms and wish to continue with this approach but, as additional Editor features became available, all of the authors found it more convenient to input and format the materials themselves. All of the work on the last three lessons was accomplished this way. Surprisingly few problems were encountered during module development, but many aspects of the Authoring Editor were shaped by frequent interactions between the authoring and software teams.

As soon as the modules were completed and their content reviewed by Weapons Mechanic course instructors, they were tried out on a one-on-one basis with a small number of student volunteers. A number of minor revisions were made on the basis of these reviews and tryouts, and the modules were implemented in the course for purposes of formative evaluation. Following revisions made on the basis of this evaluation, the modules were reimplemented for summative evaluation. The results of this evaluation are presented in Part II of this report. The objectives from the first-pass modules were then copied, revised and shortened, and combined to form two block review modules and two block remediation modules.

The CAI authoring team consisted of three members. All were experienced technical training authors (of programmed text and audio-visual materials) but none had any prior CAI authoring experience. In fact, none had even used CAI as a student. Only one of the three could be considered a Weapons Mechanic subject matter expert.

As is often the case, the team did not keep accurate records of development times but times can be estimated for the six first-pass modules. At the end of the first 6 months of the project, the first-pass modules had been revised and implemented for summative evaluation. The team leader spent relatively little time actually authoring, concentrating instead on defining the procedural model, producing the CAI Author's Handbook, interfacing with the software personnel, and attending to administrative problems. The other two team members were occasionally called upon for assistance on other on-going projects. A liberal estimate of the total time spent in developing, evaluating, and revising the six modules is 2200 manhours. This includes time spent in revising the block and lesson tests even though block test revision was technically not part of the CAI effort.

According to the course's Plan of Instruction (POI), the content taught by the CAI modules was equivalent to approximately 25 classroom hours. On this basis, CAI development required 88 manhours per POI hour. Average student completion time, totalled across the six modules, was [illegible]7 hours. This results in an estimated 11[illegible] manhours per student contact hour. These values are comparable to development times for programmed text materials in this environment.

Usability of the Authoring Editor was also evaluated through training three ATC instructors in use of the Editor. These instructors were drawn from three different courses supported by the AIS--Inventory Management, Precision Measuring Equipment, and Weapons Mechanic. None of the author trainees were computer programmers and none had prior CAI development experience. The training period lasted for 3 weeks, during which the trainees came to the contractor's facility for 3 to 4 hours each morning. The first session was spent in providing an overview of the AIS CAI system and the role of CAI within the AIS and in introducing the trainees to the Authoring Editor and the CAI Author's Handbook. No formal training took place during the subsequent 14 sessions. Using the Handbook as a reference manual, each of the trainees used the Authoring Editor to develop a CAI module in the area of their own specialty. Contractor personnel were available to answer questions and to review and comment on the trainees' work. In many cases, author trainees were also able to work on their modules during the afternoons while performing their normal classroom duties.

The author trainees asked relatively few questions after the first few sessions. Most of the suggestions made by contractor personnel pertained to the need for more frequent questions in the modules and increased individualization through branching. At the end of the 3 week period, each trainee had developed a module, had had it reviewed by the contractor and other ATC personnel, had run single student tryouts, and had made minor revisions on the basis of these reviews and tryouts. The consensus of those reviewing the modules was that they were generally of good quality and had capitalized fairly well on the capabilities of CAI. One of the modules was subsequently implemented in the Weapons Mechanic course. The time expended by the author trainees on this first module, through revision following single student tryouts, was approximately 90 hours per student contact hour.

V CONCLUSIONS AND RECOMMENDATIONS

The Authoring Editor approach to facilitating CAI development appears, at this time, to be very promising. Experience to date, while admittedly limited, has demonstrated that reasonably effective CAI modules can be produced at a very acceptable cost in terms of manhours per hour by personnel without prior CAI authoring experience. In addition, it has been shown that ATC personnel can learn to use the Editor in a reasonable period of time without formal training. While it had been anticipated that the ATC author trainees would have numerous suggestions regarding desired changes to the Authoring Editor, this was not found to be the case. They were, in general, quite satisfied with the Editor and all expressed an interest in having CAI implemented in their respective courses. As it stands, the authoring system appears ready for use by ATC instructional development and evaluation personnel.

In assessing the various features and components of the authoring system, the major contributor to simplifying the task and hence reducing costs is probably elimination of any need for the author to work in a computer language. All of the programming work has been done beforehand and provided in the form of the Editor and Presentation Programs. Future needs for programming effort will depend on how adequately this software meets the requirements of future applications. Although it certainly cannot be proven, the authors of this report think that, due to the flexibility built into it, the existing software could serve the needs of the AIS environment for some time to come. Eventually, however, it is anticipated that developing author expertise will justify increased software capability.

The second greatest contributor to facilitating the author's task is probably the extent to which the task is structured by the Editor. The overall structure of the module is determined for the author, units within this structure are matched to the requirements of the environment, and the occurrence of critical units is either forced or prompted. While the author retains a great deal of flexibility, this flexibility is exercised through selection of specific options which provide a degree of control over the authoring process, while reminding the author of the various courses of action which may be taken.

A third major factor in facilitating authoring is undoubtedly the human-engineered, computer-aided input, formatting, and editing capability provided by the Editor. Other than the approach to defining branching logic, there is little here that is radical or even novel. The work involved only the application of existing technology to a particular problem area. Given the diminishing cost of computer use, there is little reason not to provide authors with the benefits of this technology.

At this time, it is difficult to evaluate the utility of the automatic, structured student performance data collection and analysis routines. The reports did not become available until the later stages of formative evaluation of the six first-pass modules. During their limited period of use, they did appear to be easy to use and interpret. The members of this particular authoring team were, however, accustomed to collecting such data and using the data to improve instructional materials. For broader application, more extensive prompting and guidance in use of the data collection routines might be desirable.

It should be noted that the software development effort reported here was substantial. Given the extent of the various functions to be supported, the authors of this report believe that it would not have been possible to complete the project within the temporal and fiscal constraints of the contract if it had not been for the availability of the CAMIL language, the software development capability of the AIS, and the well-structured nature of the CMI system in which the CAI was embedded. The relative ease with which CAMIL code could be produced and debugged allowed the developers to experiment with a number of different approaches to various problems, obtain user feedback on these different approaches, and revise the code accordingly. The process was essentially that of formative evaluation and revision within the domain of human-engineered software. These same characteristics of CAMIL and the AIS will facilitate any future expansion of the CAI authoring system.

With respect to recommendations for future activities, the most critical action that must be taken if the effort reported here is to be justified, is that the CAI authoring system be used. The results of the evaluation described in Part II of this report, although limited, strongly indicate that, used judiciously, CAI can serve a useful function in a technical training environment such as that of the AIS. Further, usability evaluation results reported both here and in Part II demonstrate that CAI development can be made cost effective through use of the authoring system.

It is recognized that there are certain inadequacies in the authoring system, primarily in the areas of response processing and author-generated graphics. These areas provide two of the most likely candidates for future software development.

As was discussed in Section II of this report, answer processing for constructed response questions has been a problem area in CAI developed for military technical training. While sophisticated algorithms exist to aid in response recognition, the CAI systems employing these algorithms have not made the authoring process sufficiently simple for the algorithms to be used correctly. It is recommended, therefore, that a response processing dialog be developed and added to the Authoring Editor which would guide the author in defining anticipated responses to constructed response questions. The encoding algorithm employed by PLATO could be used to recognize misspellings. As currently conceived, this dialog would first prompt the author to enter an anticipated response. The Editor would break the response down into its component words and ask the author to identify the "important" words in the response. Noise words such as articles, prepositions, and auxiliary verbs would be recognized and the author would be questioned as to their importance. Next, the author would be asked if word order is important and, if so, would be prompted to indicate important word groups and any ordering restrictions within and among word groups. Finally, the author would be prompted to enter synonyms for the important words. An approach such as this would not only simplify the authoring process but would help the author to identify the critical aspects of response judging. This, in turn, should promote generation of better anticipated responses and constructed response questions of more uniform quality.

The software described in this report makes no provision for author generation of graphic displays. Although such displays can be constructed through use of the CAMIL language, it is recommended that a graphics editor be developed which would allow the non-programming author to define drawings in terms of basic geometric elements (e.g., straight lines, circles, and arcs). Such an editor would be relatively efficient in terms of storage and redisplay since only the basic elements of the drawing need to be stored and should prove adequate for preparing simple figures and almost all schematic diagrams.

If more complex drawings are required, however, it becomes much more difficult for the author to define the figures in terms of geometric shapes. While it is possible to define "freehand" shapes through keyboard control over a cursor, the process is substantially simplified through use of a light pen or digitizing device. The plasma display terminals used by the AIS cannot support light pen capability. Two forms of digitizers are available--video and tablet. A video digitizer uses a video camera to scan the drawing and convert it into a dot matrix. To use a tablet digitizer, the author overlays a special electronic tablet with the drawing to be reproduced and then uses a stylus to trace that portion of the drawing which is to be transmitted to the host computer and stored. Typically, the tracing process can be either continuous or stepwise. The major problem with a video digitizer or continuous tracing on a tablet digitizer is that the drawing is represented in terms of dots rather than geometric elements. Redisplay of single dots on a vector terminal is an extremely time-consuming process. The digitized matrix is really suitable only for redisplay on a terminal with a refresh memory which can be preloaded with the graphic. For a graphics digitizer to be feasible for graphics generation using the current AIS terminals, software would be required to transform the dot matrix into a number of basic geometric elements which could be more quickly redisplayed. Such software would require contour recognition routines which would be non-trivial to develop.

It is recommended that the use of a tablet digitizer connected directly to a graphics editor be investigated. With such an approach, the stylus of the digitizer could be used like a light pen, and the tablet could contain a menu of the geometric shapes recognized by the editor. To use such a system, the author would select the geometric element to be reproduced from the menu and note the points representing the limits of the element. For example, after the shape was defined from the menu, a line segment would be entered as the two end points of the line; a circle would be entered as the center point and a point on the circumference. This approach could combine the strong points of both the digitizer and a graphics editor.

In a totally different area, it is suggested that the utility of the authoring system could be substantially increased through provision of additional tools for managing the authoring process. The approach envisioned includes capturing relevant parameters of the development process and providing access to this information through summary displays and reports. Only a start has been made in this area. Much remains that can and should be done.

Finally, it is recommended that the authoring system's capabilities for on-line production of materials intended for off-line use be expanded. This would be particularly useful if the additional management tools mentioned above were also made available. Software development in this area would include a means of producing scrambled programmed texts from lesson materials and author-supplied decision logic and a text archiver. Authors could develop materials on-line, use the CAI Presentation Program and its embedded data collection routines for formative evaluation, make needed revisions on-line, print the number of copies needed, and allow the material to be removed to tape. When additional copies are needed or revisions are required, the author could place an archive request to move the lesson material from tape to disk for revision or printing. Such an approach would not only facilitate the process of development and formative evaluation, it could drastically reduce materials reproduction requirements. Currently, it is common practice to request many more copies of materials than are required so as to allow for normal classroom wear and tear. The materials are then often revised before many of the extra copies are ever put to use. The approach suggested here would eliminate the need for these extra copies.

REFERENCES

Bunderson, C. V. Team production of learner-controlled courseware: A progress report. In K. L. Zinn, M. Refice, & A. Romano (Eds.), Computers in the instructional process: Report of an international school. Ann Arbor, MI: Extend Publications, 1973.

Dallman, B. E., DeLeo, P. J. Main, P. S., & Gillman, D. C. Evaluation of PLATO IV in vehicle maintenance training. AFHRL-TR-77-59, AD-A052,623. Lowry AFB, CO: Technical Training Division, Air Force Human Resources Laboratory, November 1977.

Dowsey, M. W. Easy author-entry systems: A review and prototype. Internatl. Journal of Man-Machine Studies, 1974, 4, 401-419.

Hirwich, H. A. A comparison of the TICCIT and PLATO systems in a military setting. MTC Report No. 16. Computer-based Education Research Laboratory, Univeristy of Illinois, 1977.

Kaplow, R. Description of basic author aids in an organized system for computer-assisted instruction. Cambridge, MA: Massachusetts Institute of Technology, 1975.

Kimberlin, D. A. Fifth year status report: Computerized training systems project, project ABACUS. Report CTD-TR-77-1. Fort Gordon, GA: Communicative Technology Directorate, U.S. Army Training Support Center, August 1977.

Lewis, W. E., Lovelace, D. F., Mahany, R. W., & Judd, W. A. Computer-assisted instruction in the context of the Advanced Instructional System: Part II of II - Materials development procedures and system evaluation. Lowry AFB, CO: Technical Training Division, Air Force Human Resources Laboratory, in press.

Paloian, A. Y. An interrogative authoring system. Internatl. Journal of Man-Machine Studies, 1974, 6, 421-444.

Pflasterer, D. C. The CAMIL programming language. SIGPLAN Notices, 1978, Vol. 13, No. 11.

Schulz, R. E. Monoforms as authoring aids for the PLATO IV CAI system. HumRRO Technical Report 75-5. Alexandria, VA: Human Resources Research Organization, June 1975.

Sherwood, B. A. The TUTOR language. PLATO Services Organization, Computer-based Education Research Laboratory, University of Illinois, June 1974.

Stetton, K. J., Volk, J., & Bunderson, C. V. Toward a market success for CAI: An overview of the TICCIT program. In K. L. Zinn, M.

Refice, & A. Romano (Eds.), Computers in the Instructional process: Report of an international school. Ann Arbor, Mich.: Extend Publications, 1973.

Zinn, K. L. Requirements for effective authoring systems and assistance. Internatl. Journal of Man-Machine Studies, 1974, 6, 344-352.

88