THE DESIGN AND IMPLEMENTATION OF A COMPUTER-ASSISTED INSTRUCTION SYSTEM IN A UNIVERSITY ENVIRONMENT

by

A. D. FLOCKHART

Submitted for the Degree of Master of
Philosophy in Computer Science of the
University of Aston in Birmingham.

November 1973.

SUMMARY.

As a result of the increasing demands being placed on our educational system, new means have been sought to help the educator in his work. One of the most promising of these techniques is computer-assisted instruction (CAI).

This thesis describes an investigation into current CAI systems and attempts to isolate the reasons why the use of computers in education has fallen well below expectations. The major conclusion drawn from this investigation was that the disappointing performance of existing systems was at least partly due to their being too computer-orientated.

With the observations of this investigation in mind, a user-orientated CAI system was designed. This system was then implemented under the MOP multi-access system of the University's ICL 1900 series computer.

Finally, a number of trials were carried out using this system. The performance of the system and the attitudes of the users are noted and discussed.

ACKNOWLEDGEMENTS.

My very sincere thanks go to Mr. J. M. Doubleday of the University of Aston Computer Centre for his valuable help and supervision throughout the course of this work.

I would also like to thank the University of Aston in Birmingham whose financial assistance enabled me to carry out the research.

CONTENTS.

CHAPTER 1.

INTRODUCTION.

1.1) Object of Research.

Originally, the object of the research project was to design an author language and compiler as the basis of a CAI system in the environment of a technological university. An examination of existing computer-assisted teaching systems, however, revealed that a considerable knowledge of computer science and computer programming techniques was required on the part of the instructional programmers before proper use of the systems could be made. This was largely due to the method of entering instructional information onto the computer in the form of a program written in one of these author languages.

As a result of this observation, it was decided to change the aims of the research project to that of the design and implementation of a computer-assisted teaching system whose usage required virtually no computing knowledge or experience. This was to be achieved largely by the design of a system employing some alternative method of entering instructional information onto the computer. This method would ideally require no knowledge of computer science or of computer programming techniques, and would not require any form of author language whatsoever.

At all stages in the development of the system, the greatest care was taken to maintain this simplicity of usage while maximising the flexibility and incorporating as wide a range of facilities as possible into the system.

1.2) Structure of Thesis.

Chapter 1 describes the main aims of the research project and outlines the reasons for diverting from the aims as they were originally set down. The structural content of the thesis is also presented.

Chapter 2 gives a brief description of the history and principles of Programmed Instruction, and discusses the relative effectiveness of instructional programming and conventional teaching techniques.

Chapter 3 describes how computers have been applied in the field of education and suggests a possible explanation for the failure of most computer-assisted instruction (CAI) systems to attain some high quality production use.

Chapter 4 describes in detail the aims of ACATS (Aston Computer-Assisted Teaching System) and the facilities it should offer as a teaching system. Also discussed are some of the major problems encountered in the implementation of the system.

Chapter 5 contains a comprehensive description of the Aston Computer-Assisted Teaching System.

Chapter 6 describes the experiments carried out to test the performance of ACATS and its effectiveness as a teaching aid.

CHAPTER 2.

PROGRAMMED INSTRUCTION.

2.1) Introduction.

In recent years the rapid growth of science and the ever increasing complexity of the technological society in which we live have resulted in an enormous increase in the demand for education. In addition, the effects of the population explosion and current trends in education, such as greater numbers of students entering higher educational establishments and increased emphasis on personality development, have served only to aggravate the problems of our educational system.

To be able to meet these demands, without lowering the standard of education, new means have been sought to help the educator in his work. These developments have brought about the use of various new instructional techniques, including audio/visual aids, educational television and films, and programmed instruction. Of all these new instructional techniques it is programmed instruction which, potentially, offers the greatest benefits for the future.

Programmed instruction is the name given to a new development in educational technology based on principles of learning theory that derive from the research of recent experimental psychologists. Its foundation lies in the control over the interaction between the learner and the material to be learned.

2.2) History.

The principles of programmed instruction can be traced back to ancient Greece with the Socratic method of education, but more recent developments can be said to date from the work of Dr. S. L. Pressey (Pressey, 1926), an experimental psychologist at Ohio State University in the 1920's.

In 1926 Dr. Pressey developed a self-scoring testing device. This mechanical device presented a series of multiple-choice questions on a scroll of paper. The student then pushed one of four keys and, if correct, the machine would present the next question or if not correct would record the student's answer.

It was in his experiments with this device that he noticed it was capable not only of testing and scoring but also of teaching. Dr. Pressey immediately began a course of research to ascertain why his testing device was capable of producing learning on the part of the student being tested. However, his ideas did not gain acceptance with his contemporaries and he was forced, through lack of support from his colleagues, to abandon his research at an early stage.

Programmed instruction then seems to have been forgotten until as late as 1954 when there was a resurgence of interest in programmed instruction brought about by the work of yet another American experimental psychologist, Professor B. F. Skinner of Harvard University (Skinner, 1954).

2.2) contd.

Dr. Skinner's experimental work was concerned with an analysis of the effects of reinforcement in learning, and the design of educational techniques by which reinforcement could be used to greatest effect. It is largely through the work of Dr. Skinner that programmed instruction has today received broad recognition and support.

In 1960 Dr. N. A. Crowder (Crowder, 1960) suggested intrinsic programming. The most important advantage of intrinsic or branched programmes is their ability to adapt to the specific requirements of each individual student.

2.3) Criteria for Programmed Instruction.

There are several criteria for programmed instruction (Silvern and Silvern, 1966) which are now generally accepted.

i) Instruction is presented to the learner without the presence or intervention of a human instructor.

ii) Learning is self-paced by the learner.

iii) Instruction is presented in small, incremental steps requiring frequent responses by the learner.

iv) There is a two-way communication between the learner and the instructional programme.

v) The learner receives feedback from the instructional programme immediately after each response.

vi) Reinforcement is used to strengthen learning.

2.3) contd.

vii) The sequencing of lessons is controlled and consistent.

viii) The instructional programme shapes and controls the learner's behaviour.

2.4) Instructional Programming Techniques.

The stages necessary in the development of an instructional programme are varied and complex. The first stage is to specify in great detail the desired terminal behaviour of the learner after successful completion of the programme. The subject matter is then broken down into a number of small steps called "frames", consisting of a small unit of instruction and a question to which the learner must respond before being allowed to progress to the next frame. Following his response to each question, the learner is given immediate confirmation of the correctness of his response.

These frames are organized in a logical sequence that leads the learner from his pre-programme behaviour to the desired terminal behaviour. Generally, at the start of each instructional programme the learner is required to give simple answers to simple questions, but as the programme progresses the level should become more advanced, until at the end, he is giving sophisticated answers to quite complex questions.

In this way, the learner proceeds at his own pace through a logical sequence of frames of gradually increasing

2.4) contd.

complexity. The learner's mastery of the subject is thus built up in a series of small, easily comprehensible steps, resulting in a gradual accumulation of knowledge.

The major psychological principle on which programmed instruction is based is that of "reinforcement"; the providing of a reward for each correct response, so that on future occasions the correct response will tend to reoccur. With human beings, confirmation of each correct response immediately after it occurs is sufficient reinforcement in itself as the learner's self-awareness of successfully responding is inherently rewarding.

With programmed instruction, the learning situation is structured in such a way as to allow the learner to progress from one learning step to the next with minimum errors. The learner's knowledge of his capability to progress is yet another source of reinforcement, thus serving to enhance the effectiveness of the learning process.

2.5) Comparison with Conventional Teaching Methods.

Programmed instruction has many advantages over conventional teaching methods :-

i) The student's concentration is assured as he is constantly required to participate in the lesson by actively responding to the questions presented to him.

ii) As the student's responses are being continually

2.5) contd.

assessed, he is receiving feedback from the lesson and is thus fully aware of the progress he is making. This serves as an inducement to learning.

iii) Teaching sessions are self-paced by the learner.

iv) In the case of branched programmes the lesson material adapts itself to the particular needs of the student.

v) The teacher is freed from the time consuming and routine tasks, such as presentation of instruction, testing and marking, leaving him free to concentrate on the more complex aspects of teaching such as tutorial work, discussion, personality development, etc.

A great deal of time and effort has been spent in attempts to scientifically prove that instructional programming techniques are more efficient and effective than conventional teaching methods. Despite this, the superiority of one or other of the methods has still to be demonstrated convincingly.

A review of the results of such research projects up to 1963 (Schramm, 1964) was carried out. The reports of 36 research projects, comparing programmed instruction with conventional teaching, were examined. Of these 36 reports, 18 could find no difference, 17 suggested that programmed instruction was better, and 1 that conventional teaching methods were better.

Despite the obvious difficulties in reaching a

2.5) contd.

scientific proof of the superiority of instructional programming techniques, they do seem to offer some considerable advantages.

CHAPTER 3.

COMPUTER-BASED LEARNING.

3.1) Introduction.

Computer-based learning (CBL) covers all aspects of the use of computers in education. These uses were summarised as being (Lippert, 1971) :-

i) Computer-assisted instruction (CAI) for remedial and supplementary instruction and for diagnostic and criterion testing.

ii) Computer-managed instruction (CMI) where the computer monitors learning, rather than testing the student.

iii) The creation and availability of student records and curricular data records as a by-product of the CMI.

iv) Computational time-sharing for students doing homework, solving problems and simulating functional relationships using languages such as APL and BASIC.

v) Remote job entry terminals in classrooms, laboratories and areas where learning takes place.

vi) The provision of a wide variety of application programs for data reduction.

vii) Academic and career placement counselling (computer assisted guidance).

Obviously, a tool of such great flexibility as the computer would find many useful applications in the field of education. However, it was not until the advent of time-sharing systems in the late 1950's that CAI became a viable proposition.

The potential of computers in CAI was summarised by Uttal (Uttal, 1962) when he stated that the flexibility and

3.1) contd.

range of capabilities offered to the teacher by the powerful decision logic and large memory of computers transcend mere quantitative differences and suggested that there is a true qualitative difference between CAI and conventional teaching.

These qualitative differences are :-

i) Computers can continually judge the student's performance and present new material depending on his performance. His performance can be judged on the basis of a large number of individual responses.

ii) This ability to continually test, prevents the student from moving on before each learning step is thoroughly understood. On the other hand, it also prevents him from being held back by the constraints imposed by classroom teaching.

iii) Quite elaborate records can be kept of the student's progress for use by the teacher.

iv) Course material can be easily edited.

3.2) Methods of Using Computers in CAI.

Computers are employed in CAI in many different ways. The most common methods are :-

i) Drill and Practice.

This is by far the simplest method. It is used for the learning of elementary skills, where the learner practices until a certain degree of proficiency is attained.

3.2) contd.

ii) Author-Controlled Tutorial.

Here, the author (instructional programme writer) maintains control throughout. Student responses are evaluated by comparing them with expected responses. With intrinsic programming, the instruction presented depends on the student's past history of responses. The programmes are therefore adaptive in both pace and in the amount of instruction presented to each student.

iii) Socratic Tutorial.

In a Socratic tutorial, the programme permits dialogues between the student and the computer. The student is guided by the programme towards some acceptable solution.

iv) Learner Control.

In this situation, the initiative is placed with the student. The student learns by discovery, accessing only the subject matter which interests him.

v) Paired Students.

In this case, the students learn from each other. The interaction between students helps overcome the impersonality and isolation of the CAI session. Very little work, however, has been done in this area.

3.3) Advantages of CAI.

Obviously, the use of such a costly and complex machine as a computer must be justified by some quite considerable advantages. The major advantages of CAI over programmed texts and simple teaching machines are :-

i) Student participation is assured as he simply cannot progress through the lesson without answering the questions. With programmed texts, it is still possible for him to cheat.

ii) Constructed response questions can be used. Programmed texts and simple teaching devices are limited to the use of multiple-choice questions. Answering a multiple-choice question correctly does not necessarily mean that the student understands, but only that he can recognise the correct answer.

iii) Student responses can be automatically evaluated and marked.

iv) Because of its very large memory, the computer can offer almost unlimited branching capabilities.

v) Records of the student's performance can be kept for the use of the teacher.

vi) Since the student's past history is recorded, it is possible to make branching decisions on a variety of criteria.

vii) Records can also be kept on the performance of the lesson material itself. In this way, errors and ambiguities in the lesson can be easily and accurately pinpointed.

3.3) contd.

viii) The lesson material can be easily edited.

ix) There is far greater flexibility as the computer can control a wide variety of terminal equipment.

x) Finally, there is the novelty of conversing with a machine. This is not a lasting advantage, but still of major importance.

3.4) Existing CAI Systems.

Perhaps the first CAI system to be developed was at the IBM Watson Research Laboratory (Rath, 1960) in 1958. Since then a large number of projects, almost exclusively based in the USA, have been carried out.

There are approximately one hundred CAI systems currently in operation, the most successful being the PLATO, Programmed Logic for Automatic Teaching Operation, and the TICCIT systems.

The PLATO project (Avner and Tenczar, 1969) at the University of Illinois has been in operation since 1960, and now incorporates over four thousand terminals located within a wide radius of Urbana, Illinois. By 1970, 720 hours of instructional material had been developed and the system had completed 100,000 student-contact hours.

The PLATO IV system uses a wide range of advanced terminal equipment, including plasma display panels, high speed individual slide selectors and random access audio devices. Time-sharing operation of the CDC 6000 series computer allows student access to about 250 lessons at any one time. The cost of a student terminal hour is estimated

3.4) contd.

to be in the range 35 to 50 cents.

The MITRE Corporation's TICCIT system employs a small 32K, 16-bit word computer, and uses standard television receivers to provide computer-generated voice, pictures and text. The cost for the TICCIT system is 20 cents per student-contact hour.

In Europe, the first full scale application of CAI has been in operation since 1971 at St. Anna School, Augsburg, West Germany under the direction of Dr. Karl-August Keil (Computer Weekly, 1973). The LIDIA, Learning In DIAlogue, system uses a Seimens series 4004 computer with 23 Transdata display terminals. A library of over one hundred hours of dialogue study has been compiled.

As the equipment necessary and the production of instructional material requires a substantial financial outlay, research efforts in this field depend to a large extent on Government grants.

In Britain, the National Council for Educational Technology, NCET, have estimated that £2 million was needed over five years (NCET, 1969).

In the USA, the National Science Foundation, NSF, have made available 15 million dollars over the next four and a half years, to be divided between the PLATO and the TICCIT projects (Dowsey, 1973). However, despite the large number of outstanding projects being carried out in the USA, the total yearly support is decreasing. Computer-based learning has not proved conclusively its effectiveness, and the impact of computers in education has fallen well below expectations.

3.5) Author Languages.

One feature inherent in existing CAI systems is the use of an author language to facilitate the storage of instructional material on the computer and to determine the particular tutorial logic to be used during the learning sessions.

Recently, greater emphasis has been laid on the use of existing multi-purpose computing languages such as BASIC, APL and PL/1. In general, however, special purpose author languages are used which have been specially designed to meet the particular requirements of an interactive teaching system.

These languages can be distinguished from scientific and business languages by a number of factors (Zinn, 1971):

i) Convenience for display of text.
ii) Acceptance and classification of relatively short strings of text.
iii) Automatic recording of answers or other performance data.
iv) Implicit branching determined by the categorization of an answer or the contents of a counter which is part of the history of student responses.

There are currently in use approximately forty author languages (Zinn, 1969). The major author languages are :-

CATO	(PLATO System, University of Illinois.)
MENTOR	(Bolt Beranek and Newman, Cambridge, Mass.)
COURSEWRITER	(IBM Watson Research Center, Yorktown Heights, New York.)

3.5) contd.

TUTOR	(CERL, University of Illinois, Urbana.)
PLANIT	(System Development Corporation, Santa Monica, California.)
ELIZA	(Educational Research Center, MIT, Cambridge, Mass.)
COMPUTEST	(University of California, Medical Center, San Francisco.)
PILOT	(University of California, Medical Center, San Francisco.)
FOIL	(University of Michigan, Ann Arbor.)
INFORM	(Philco-Ford.)
COPI	(Univac.)
EXPER	(GE Information Systems, Schenectedy.)
CHIMP	(University of Maryland, College Park.)
XXXX	(IBM DP Educational Research, Poughkeepsie, New York.)

These author languages vary in complexity from relatively simple languages such as WRITEACOURSE (Hunt and Zosel, 1968) to languages such as COURSEWRITER (IBM, 1967) which has a very large instruction set and complex instruction format.

The use of an author language therefore necessitates that potential instructional programmers must first of all be proficient in the following areas :-

i) The particular author language being employed.

ii) Computer programming techniques.

iii) Local computing facilities.

3.5) contd.

This involves the potential author in a considerable amount of work before he can even begin his instructional programming. Certainly, in the case of a technological university, whose staff already have a full timetable, a CAI system which required such a large initial involvement would simply not be used.

One further consequence of the use of author languages has been that the production of instructional programmes has been carried out largely by persons who already possess a knowledge of computer science such as mathematicians and computer programmers. As the writing of instructional programmes is a very complex task to be undertaken only by experienced educationalists, the present situation is quite unacceptable.

CHAPTER 4.

DEVELOPMENT OF ACATS.

4.1) Aims of ACATS.

As a result of this examination of existing CAI systems, the major priority in the development of the research project has been to design a system which could be used efficiently by all the staff of the university. Only in this way could the full potential of the CAI system be realised. To achieve this, a system had to be designed whose usage required virtually no computing knowledge whatsoever and did not involve the potential author in any considerable amount of work before being able to make use of the system.

One immediate essential was that some alternative method of entering instructional material onto the computer, other than the use of an author language, would have to be employed. An alternative method was found by breaking down into its component parts, all instructional data considered necessary or desirable for interactive computer-controlled teaching sessions. Special data entry forms, to be completed by the instructional authors, were then designed. These forms were designed for both ease of completion and ease of translation onto punched cards.

The function of the author has thus been altered from that of combining the instructional material and the tutorial logic in the form of a program written in one of these author languages, to that of simply completing a set of forms. This process requires absolutely no computing knowledge whatsoever.

The main aim in the design of ACATS was to maintain this simplicity of usage while at the same time maximising

4.1) contd.

the flexibility, and hence usefulness, by incorporating as many facilities as possible into the system.

Other items of a more specific nature, which were given high priorities during the design of ACATS, were :-

i) Response times during learning sessions should be as short as possible.
ii) Retrieval of instructional information stored on the computer should be as fast and efficient as possible.
iii) Costs should be kept to a minimum.
iv) The amount of core store required by each student should also be kept to a minimum.

4.2) Facilities Offered by ACATS.

Before deciding what facilities to incorporate in the system, due consideration was given to the particular needs of the students and authors who would use the system. It has already been stated that ACATS was designed to allow members of the university staff from all disciplines to make efficient use of the system with a minimum of computing knowledge. Particular emphasis has already been placed on some of the major aspects in the development of ACATS, however, a more comprehensive list of features which were considered desirable for ACATS authors would be :-

i) Easy entry of instructional material onto the computer.
ii) The system should be suited to almost all teaching situations, and should be capable of presenting

4.2) contd.

original instruction, revision and testing material.

iii) The system should be capable of handling both linear and branching programmes.

iv) Both multiple-choice and constructed response questions should be allowed.

v) Automatic evaluation and marking of student responses.

vi) Records should be kept of each student's performance during each learning session.

vii) Records should also be kept of the performance of each frame in the lesson.

viii) Easy editing of the instructional material on the computer.

ix) A set of simple author commands to allow easy manipulation of files and records on the computer.

x) A non-technical and easily understandable author's manual.

The features of the teaching system that the student would find most desirable were considered to be :-

i) Ease of operation.

ii) Fast response times.

iii) To have a hard copy of the notes after each lesson.

iv) A knowledge of his performance during each learning session.

v) The facility to stop at any point in the lesson.

vi) The facility to restart at any point in the lesson.

vii) To be allowed a certain degree of control over the teaching process.

4.2) contd.

- viii) The facility to permit simultaneous access by a large number of students to any one lesson.
- ix) A brief, non-technical user's guide.

All facilities mentioned above have been included in the design of ACATS.

4.3) Problems in Implementation.

This section deals with the major events and problems which occured during the implementation of ACATS. Explanations of these problems together with a discussion on how each was solved are included.

4.3.1) Choice of Programming Language.

As ACATS is essentially a suite of programs to facilitate interactive computer-controlled teaching, the performance of the system as a whole depends very largely on the programming which forms its basis. A very considerable programming effort was required.

One of the first problems to present itself was the selection of the computing language which was best suited to the special requirements of a CAI system such as ACATS.

It was envisaged that the language used would have to be efficient in the following areas :-

- i) Input / Output.
- ii) Character handling.
- iii) Arithmetic operations.
- iv) Handling of direct access devices.

4.3.1) contd.

A number of computing languages were taken into consideration. The language finally chosen was PLAN (ICL, 1970) as it excelld in all these areas. It further had the advantages of producing fast and efficient machine code and of being economical on core storage.

4.3.2) Student Consoles.

The major drawback of the MOP (Multiple On-line Programming) system, under which ACATS operates, is that the only terminals that may be used with the MOP system at present are the 7071 and 7072 console typewriters (ICL, 1972).

As one might expect, an interactive teaching system would place very high demands on the student consoles being used. The 7071 terminals at the University of Aston were found to be inadequate for this purpose.

The major disadvantages are :-

i) They are exceedingly slow. This causes the student a great deal of frustration whenever any appreciable amount of information has to be displayed.

ii) They have a restricted character set. They are limited to the transfer of the standard ICL 1900 64-character set.

iii) No diagrams may be presented.

4.3.2) contd.

iv) They are very noisy. This noise can lead to tiredness and irritation on the part of the student.

v) Paper movement interferes with reading.

A solution to this problem was found by storing only certain sections of the data of each instructional programme on the computer. The items which were not stored on the computer, being by far the bulkiest part of the data, comprised the instruction and question associated with each frame. It was decided that these items, termed the "external data", should be prepared by the author on some external medium (usually on printed handouts) and presented to the student prior to learning sessions.

This not only helped solve the problem of the student consoles, but also benefited the system quite considerably in various other ways :-

i) The author was now free to use any medium of presentation at his disposal. This could be printed handouts, slides, film, tapes, microfilm or any combination of these.

ii) The overall speed of the system was vastly improved. This was brought about not only by the fact that it was no longer the function of the teletype terminals to present the instruction and question associated with each frame. The time spent by the student

4.3.2) contd.

in reading each item of instruction and solving each problem could now be put to good use by the computer. This resulted in a double saving of time.

It was also fortunate that this free time occured at the most critical points in the lesson. At these points the heaviest demands are being placed on the computer, both in the completion of student and frame performance records and in the retrieval and transfer of the following frame's data.

iii) The problem of the restricted character set was largely removed.

iv) Diagrams and other pictorial information could now be used freely.

v) The student would now be almost assured of having a good hard copy of notes after completion of each lesson.

4.3.3) Indexing.

The instructional information of each lesson was stored on direct access devices, the contents of each record comprising the data for one frame in the lesson.

Full use was made of the storage device (SD) macros of the ICL Housekeeping system (ICL, 1971b).

The problem which arose at this point was that the software for the SDIND macro was not

4.3.3) contd.

available at Aston. The function of the SDIND macro is to examine the index tables to find the bucket containing the record key specified by the user.

Some alternative method of obtaining the logical bucket number (LBN) of each record to be retrieved had to be found. This problem was solved by calculating the logical bucket number for each record and forming an index when the file was created.

Since both linear and branching programmes were to be handled, these records would obviously have to be accessed randomly. The items or pointers required to access any record in this fashion are :-

i) The record key.

Here, the frame number itself was used as the record key.

ii) The LBN of the bucket containing the record.

The LBN for each frame could be calculated by measuring the size of each record, accumulating these sizes, and incrementing the LBN each time the accumulated size became greater than the bucket size.

An index was then formed by storing the LBN in the key'th location of the index. This index was then stored in bucket number 1 of the direct access file.

4.3.3) contd.

At the start of each lesson, this index would be read into a special buffer in core store. Thus, during each lesson, each time control passed on to a new frame, the two necessary pointers were immediately available. Therefore, the appropriate record, containing the data for this new frame, could be readily transfered into core store.

4.3.4) Multiple Access.

At the implementation stage of the research project a number of alterations were found to be necessary to allow access to a number of users to each direct access file simultaneously.

Firstly, all direct access files had to be created with integrity code 2 (ICL, 1971b).

Secondly, when the direct access files were being assigned to the ACATS control program, the CLEAN qualifier had to be given (ICL, 1971a).

With these two alterations, access was then permitted to one writer and any number of readers simultaneously. However, since these files were being opened in OVERLAY mode, it was still possible for only one user to access each file at any one time.

During each lesson, the only time records have to be written is at the very end when the updated frame performance records are transfered

4.3.4) contd.

back to buckets 3 and 4 of the direct access file. Thus, it was possible to initially open the direct access file in READ mode, close it at the end of the lesson and then re-open it in WRITE mode to transfer the frame performance records. Once the records have been written, the file was then closed for the second time.

Since, in general, lessons are approximately 30 minutes in length, of which only about one or two minutes are spent writing the frame performance records, it was now possible for a fairly large number of users to access a single lesson simultaneously without encountering any appreciable delays.

CHAPTER 5.

ASTON COMPUTER-ASSISTED TEACHING SYSTEM. (ACATS)

5.1) Environment.

This section describes the particular computing environment in which ACATS was implemented.

5.1.1) Equipment.

The central processor currently in use at the University of Aston's Computer Centre is an ICL 1905E, with 96K 24-bit words of 1.8 microsecond store.

The peripherals used by ACATS include :-

Seven	7008 Telegraph Data terminals. 72 characters/line, and with a maximum transfer rate of 600 chars/minute.
Four	7 track magnetic tape decks, with a maximum transfer rate of 20,800 chars/sec.
Two	9 track magnetic tape decks, with a maximum transfer rate of 160,000 chars/sec.
Four	Exchangeable disc drives, each with a storage capacity of 8,192,000 characters, and a transfer rate of 208,000 chars/sec.
One	Magnetic drum, with a storage capacity of 512K words, and with a maximum transfer rate of 100,000 chars/sec.
One	2101 card reader, 2000 cards/minute.
One	1933 line printer, 1350 lines/minute, 120 chars/line.

5.1.2) The MOP System.

An essential requirement of any interactive computer-controlled teaching system is a multi-access system in which a large number of users can converse simultaneously with the same computer via terminals linked to it.

The multi-access system, under which ACATS operates, is known as the MOP (Multiple On-line Programming) system (ICL, 1972). MOP is part of the general operating system GEORGE 3 (ICL, 1971a) used to control the operation of the larger ICL 1900 series computers such as the 1905E.

There are many excellent facilities offered by MOP without which the implementation of a useful CAI system would have been impossible.

The most advantageous facility is that of being allowed to copy and save compiled programs for future use. Once the user has compiled his source program and has a binary program in core store, he may then save a copy of it in a file so that he can run it again on future occasions without having to recompile. This has resulted in huge savings in both time and money. For example, the ACATS main control program, which controls the interaction between the student and the lesson material, takes approximately 30 minutes and costs over 30 pence to compile.

5.1.2) contd.

Another extremely useful facility is that the user may store a set of frequently used commands that can be implemented by a single command (i.e:- a user macro command.) with run-time values given as parameters. Nine such commands have been defined, one for the use of the students and the remainder for the authors. For simplicity, all commands were formed requiring one parameter only, namely the name of the lesson to be operated on.

These commands are extremely simple to use, and with them, the author can control all aspects of the teaching situation such as the preparation and editing of instructional material, the listing of files and the printing and resetting of frame performance records. These nine simple commands, together with a knowledge of how to operate a teletype terminal, represent almost the sum total of computing knowledge required by authors using ACATS.

Whenever a MOP job is started, the system sets up a monitoring file to contain all the information generated by the system in the course of a job. This information comprises a number of messages which have been assigned various categories. To prevent the system outputting large quantities of information, which would be unintelligible and confusing to the average ACATS user, it is possible

5.1.2) contd.

to suppress the reporting level and to choose which categories of information to receive at the terminal.

Throughout all author and student sessions run under ACATS, the reporting levels are being continually changed so that only essential information is received at the terminal.

In certain instances, the standard ICL messages have been suppressed and replaced by messages which are more appropriate for the ACATS users.

5.2) Entry of Instructional Material.

This section contains a description of the method of entering instructional material onto the computer. This method was developed for ACATS as an alternative to the standard method used by existing CAI systems which requires the use of an author language.

5.2.1) Division of Data.

The instructional data for each programme is divided into three distinct parts. The three data divisions are termed :-

i) External Data.
ii) Response Data.
iii) Subframe Data.

This method of structuring the instructional data was found to be the most convenient for two main reasons. Firstly, one section of the data, the

5.2.1) contd.

external data, is not stored on the computer at all. Secondly, of the data that is stored on the computer, one section, the response data, must be supplied by the author while the other, the subframe data, is optional.

5.2.1.1) External Data.

The external data consists of the instruction and question associated with each frame. This data is not stored on the computer which has the advantage that the author is now free to use any medium of presentation at his disposal.

The only restrictions on the presentation of the external data is that each lesson must consist of a logical sequence of numbered frames, frame numbers being integers in the range 1 - 200, and with the initial frame being Frame 1.

This material should be prepared by the author and presented to the student prior to the student session with ACATS.

By far the most popular means of presentation is by printed handouts. An example of a printed handout can be seen in Appendix 1.

5.2.1.2) Response Data.

The response data consists of all the information considered to be essential for interactive computer-controlled teaching. This data is stored on the computer, and for each frame consists of a number of "Answer Sets", each answer set consisting of :-

i) A list of expected answers to be checked against the student's responses.

ii) Some relevant comment which will be presented to the student if his answer matches one of the items in the list of expected answers. This comment provides the feedback or reinforcement considered so important in education.

iii) Two scores, one of which will be given to the student depending on how many attempts he has made at the question and on what scoring mode (see section 5.4.7) is being used.

iv) A pointer to the next frame to be studied should a match occur between the student's response and one of the expected answers in the answer set.

This material should be prepared by the author on special forms and submitted to the Computer Centre for preparation and submission to ACATS.

5.2.1.3) Subframe Data.

The purpose of subframe data is to give the student some control over the teaching process, and to allow him to seek help should he find difficulty in answering any of the questions. The student has the option of whether or not to use this subframe data, and can do so by issuing the *AID and *GIVE commands described in section 5.4.4.

Associated with each frame in the lesson, the ACATS author may provide any number (including zero) of small units of instruction called "subframes". Each subframe serves a specific purpose such as :-

i) Giving a hint or reminder.
ii) Giving a worked example.
iii) Defining a word.
iv) Giving an explanation of the problem.
v) Giving the solution to the problem.

Each subframe consists of :-

i) A name or identifier. The student should be able to tell from the subframe name what kind of help the subframe offers.
ii) The text of the subframe.
iii) A penalty to be deducted from the marks the student gains in the current frame should he decide to use the

5.2.1.3) contd.

subframe.

iv) A decimal digit (usually 0,1 or 2) giving the number of attempts the student must make at the question before being allowed to use the subframe. This decimal digit is called the "open permit". The purpose of the open permit is to restrict the amount of control the student has over the teaching process and to prevent misuse of the subframes.

As in the case of the response data, this material should also be prepared by the authors on special forms and submitted to theComputer Centre for preparation and submission to ACATS.

Since this data is not essential to interactive teaching sessions, and since simplicity of usage has been the major aim in the design of ACATS, the provision of this data has been made optional.

This has been achieved by storing the response data and subframe data for each lesson on two separate files. Therefore, it is possible for ACATS authors to enter and use instructional programmes which include response data only, and to supply subframe

5.2.1.3) contd.

data at a later stage should they deem it necessary.

At the start of each lesson, the existence or non-existence of this subframe file is determined, and a switch is set or reset depending on the result. During the student session, the state of this switch controls many aspects of the teaching process.

5.2.2) ACATS Entry Forms.

The method employed by ACATS authors to enter instructional information onto the computer is to complete, within certain limits, a set of special forms. These forms are subsequently translated onto punched cards and loaded onto the computer.

Unlike the methods employed by existing systems, this process of filling in forms requires no computing knowledge whatsoever. One of the services offered by the Computer Centre is to punch the contents of these forms onto cards. Thus the author is not required even to operate card punch machines.

One of the functions of this section is to explain the particular rules and conventions which must be observed by ACATS authors when completing these special forms. The reasons for imposing these restrictions, especially when they are the result of

5.2.2) contd.

the limits of ACATS itself, are also discussed.

There are a total of five forms used in the storage of instructional programme data on the computer :-

i) Response Title Form.
ii) Response Data Form.
iii) Subframe Title Form.
iv) Subframe Data Form.
v) Terminators Form.

The response data and subframe data for each lesson are stored on separate files. The two sections thus comprise :-

Response Title Form	Subframe Title Form
Response Data Forms	Subframe Data Forms
Terminators Form	Terminators Form

Once the appropriate forms have been completed, they should be submitted to the Computer Centre for preparation and submission to ACATS. This preparation stage, undertaken by the data preparation service, involves no more than the translation of these forms onto punched cards. There is a one-to-one correspondence between lines on the forms and punched cards, and the resulting deck of cards can be loaded onto the computer without any further preparation.

The files containing this data, which has been input from cards, have been termed "input-files".

5.2.2) contd.

At a later stage this data is checked and stored on direct access files before the student sessions may begin. To distinguish these files from the original input-files, they have been termed "lesson-files".

5.2.2.1) Response Title Form.

This form is used to supply all the information required by the lesson as a whole. i.e:- the information which cannot be attributed to any particular frame. This consists of :-

i) Lesson name.

Lessons are identified by their "lesson names", consisting of up to 10 alphanumeric or hyphen characters.

A system of prefixing (see section 5.3) is used to distinguish between the input-files and lesson-files and between the response and subframe files of each lesson. The above restriction is due to the fact that under GEORGE 3, file names are limited to 12 alphanumeric or hyphen characters and that these prefixes can be either one or two characters in length.

ACATS authors need not be aware of this system of prefixing. At all

ASTON COMPUTER-ASSISTED TEACHING SYSTEM.

RESPONSE TITLE FORM.

Lesson name

1 10 20 27

INPUT :SSP0707,CR

Lesson Name

1 10

☐☐☐☐☐☐☐☐☐☐

Author

1 10 16

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

BLOCK*
CAPITALS
ONLY
PLEASE

Parameters (yes = 1, no = 0)

1 ☐ Strict mode scoring?

1 ☐ Frame performance data to be recorded?

1 ☐ Trace of lesson to be listed?

1 ☐ Instructional programme stored on computer?

Introductory Message

1 10 20 30 40

(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)
(10)
(11)
(12)

////

5.2.2.1) contd.

times, they need only refer to the lesson name itself.

The lesson name is entered twice, once for the use of GEORGE, and once for ACATS.

The GEORGE command, "INPUT", together with the ACATS user number and the prefix for the response input-file has already been inserted for the author.

ii) Author.

16 characters are allowed to identify the instructional programme writer.

iii) Parameters.

There are four parameters governing various aspects of the teaching process, each parameter having two states. These parameters are used by the authors to state :-

(a) Whether strict or non-strict mode scoring (see section 5.4.7) is to be used.

(b) Whether frame performance records (see section 5.5.2) are to be kept. These records will normally be kept during the developmental stages of the instructional programme. Once the

5.2.2.1) contd.

author is completely satisfied with the performance of the instructional programme the state of this parameter can be easily changed.

(c) Whether a trace (see section 5.5.1) of the student's path, responses, scores and penalties are to be listed.

(d) Whether or not the external data has been stored on the computer (see section 5.2.3). In practice, this data will very seldom be stored on the computer.

iv) <u>Introductory Message</u>.

This introductory message is presented to the student at the start of the lesson. The author may use this facility to supply any necessary information about the lesson to be studied.

5.2.2.2) <u>Response Data Form</u>.

Theorder in which frames are entered is unimportant. The data for two answer sets only may be entered on each form but any number of forms can be used for each frame.

The items of information included in the response data form are :-

i) <u>Frame Number</u>.

The first character must be an

ASTON COMPUTER-ASSISTED TEACHING SYSTEM.

RESPONSE DATA FORM

Author's Name	Frame Number
Lesson/File	Page Number

BLOCK\
CAPITALS\
ONLY\
PLEASE

Frame Number 1 * 4

No. of Answer Sets 5 6

Maxscore 7 8

A.S. Identifier 1 8

Next Frame 9 * 12

Score 1 13 14

Score 2 15 16

Lines 17

Expected Answers 1 10 20 30 40

Comment 1 10 20 30 40

(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)

A.S. Identifier 1 8

Next Frame 9 * 12

Score 1 13 14

Score 2 15 16

Lines 17

Expected Answers 1 10 20 30 40

Comment 1 10 20 30 40

(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)

5.2.2.2) contd.

asterisk. This is a marker for the start of each frame and is very valuable during the error checking stage.

Frame numbers must be integers in the range 1 - 200, with the proviso that the first frame in the lesson must be Frame 1.

To minimise the amount of core store required by each student during learning sessions, only 1K of core store was allotted for the recording of frame performance data. Since each frame's record is five words in length, this has resulted in a limit of 200 frames per lesson. However, as the average lesson should contain from 20 to 40 frames, this is not a serious limitation.

ii) <u>No. of Answer Sets</u>.

The number of answer sets in each frame must be given. Each list of expected answers together with its associated data constitutes one answer set. The number of answer sets is limited to 99, again, not a serious limitation.

5.2.2.2) contd.

iii) Maxscore.

The maximum possible marks the student can gain in this frame. This is required for the calculation of the student's percentage score.

iv) A.S. Identifier.

Each answer set (A.S.) is given a name or identifier of up to eight characters. This identifier should give some indication of the correctness of the expected answers included in the answer set.

v) Next Frame.

The frame number of the next frame to be studied if the student's answer should match one of the predicted answers in the answer set.

If the author wishes the student to make another attempt at the question the current frame number is entered. On the other hand, if the end of the lesson has been reached, a zero frame number is entered.

vi) Score 1.

The marks to be added to the student's score if the first answer he gives matches one of the predicted

5.2.2.2) contd.

answers in the answer set.

vii) Score 2.

The marks to be added to the student's score if the answer (not his first attempt) matches one of the predicted answers in the answer set. In strict mode scoring this mark will be given only on his second attempt.

viii) Lines.

The number of lines of text in the author's comment.

ix) Expected Answers.

This is a line of forty characters for entering a list of expected answers with which the student's answers will be compared. There is no real limit to the number of expected answers the author may give, as they can be spread over two or more answer sets.

Each predicted answer must be immediately followed by the symbol @.

Obviously, not all the student's responses can be predicted by the authors. To allow for all unpredicted answers, a special answer set, known as the "catch" answer set, should be supplied for each frame. Placing the @ symbol in the first character position of the expected answers

5.2.2.2) contd.

line will result in a match with all possible student responses. Since during response evaluation each answer set is taken in turn, the catch answer set should be the last answer set of each frame.

x) Comment.

This comment is used to provide the reinforcement or feedback to the student. Should a match occur between the student's response and one of the items in the list of expected answers, this message will be presented to the student.

5.2.2.3) Subframe Title Form.

In the subframe title form the author need only supply the lesson name.

The GEORGE command "INPUT" together with the ACATS user number and the prefix for the subframe input-file has already been inserted for the author.

.ASTON COMPUTER-ASSISTED TEACHING SYSTEM.

.SUBFRAME TITLE FORM. .

BLOCK*
CAPITALS
ONLY
PLEASE

Lesson name

1 10 20 27

INPUT :SSP0707,CS

5.2.2.4) Subframe Data Form.

Again, the order in which the frames are entered is unimportant. When subframe data is supplied by the author, he is not required to supply it for all the frames in the lesson. The data for two subframes only may be entered in each form but any number of forms may be used for each frame.

The items of data included in the subframe data form are :-

i) Frame Number.

The first character must be an asterisk. Frame numbers must be integers in the range 1 - 200.

ii) No. of Subframes.

The number of subframes included in the frame.

iii) Subframe Identifier.

Each subframe is given a name or identifier of up to eight characters. This identifier should give the student some indication of the type of help the subframe offers.

iv) Open Permit.

This single decimal digit gives the number of attempts at the question the student must make before he is allowed to call the subframe.

ASTON COMPUTER-ASSISTED TEACHING SYSTEM.

SUBFRAME DATA FORM

Name	Frame Number
Lesson/File	Page Number

Frame Number 1 4 ☒ ☐ ☐ ☐

No. of Subframes 5 6 ☐ ☐

BLOCK*
CAPITALS
ONLY
PLEASE

Subframe Identifier 1 8 ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

Open Permit 9 ☐

Penalty 10 11 ☐ ☐

Lines 12 ☐

Subframe Text

	1									10										20										30										40
(1)																																								
(2)																																								
(3)																																								
(4)																																								
(5)																																								
(6)																																								
(7)																																								
(8)																																								
(9)																																								

Subframe Identifier 1 8 ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

Open Permit 9 ☐

Penalty 10 11 ☐ ☐

Lines 12 ☐

Subframe Text

	1									10										20										30										40
(1)																																								
(2)																																								
(3)																																								
(4)																																								
(5)																																								
(6)																																								
(7)																																								
(8)																																								
(9)																																								

5.2.2.4) contd.

v) Penalty.

The number of marks to be deducted from the student's score should he call the subframe.

If he receives more penalties than marks, the student is given a score of zero for that frame.

vi) Lines.

The number of lines of text in the subframe.

vii) Subframe Text.

The text which will be presented to the student should he call the subframe.

5.2.2.5) Terminators Form.

The terminators form is used to signify the end of the response data and the subframe data of each lesson. As these terminators are standard, nothing need be added to this form by the authors.

ASTON COMPUTER-ASSISTED TEACHING SYSTEM.

Terminators.

1			4
/	/	/	/

1			4
*	*	*	*

5.2.3) Stored Programmes.

There may be situations where it is desirable to store the external data on the computer. Programmes of this sort are called "stored programmes". ACATS authors are strongly recommended not to write programmes of this sort unless they find it absolutely essential.

Despite the fact that the teletype terminals are unsuited to the task of presenting large amounts of data, it may be that the author wishes to store the external data on the machine. This is quite easily done by storing the external data of the next frame to be studied in the author's comment area of the response data form. The external data of the next frame will therefore be presented to the student immediately before control passes on to the next frame.

The presentation of the text of the initial frame in the lesson creates a problem. The best method is to include the external data of the first frame in the introductory message of the response title form.

With stored programmes, the option of restarting at any frame in the lesson (see section 5.4.2) is not given to the student as the data associated with any one particular frame is in fact spread over two frames in the machine.

5.3) Storage.

The response and subframe data read in from cards make up what have been termed the "response input-file" and the "subframe input-file" respectively. As the random selection of items of data within these card files is virtually impossible, a further stage of preparation, in which the data is stored in direct access files, is carried out. The resulting direct access files, termed lesson-files, are then ready to be used in learning sessions.

A maximum of four files are therefore associated with each lesson. These files must be given a unique file-name, and so a system of prefixing is used to distinguish between them. Each file-name consists of the lesson name itself together with a one or two character prefix. The prefixes used are :-

Type	Prefix	Meaning
Response Input-file	CR	Card Response
Subframe Input-file	CS	Card Subframe
Response Lesson-file	R	Response
Subframe Lesson-file	S	Subframe

The ACATS authors need not be aware of the existence of these prefixes, and at all times need only refer to the lesson name itself.

Storing the instructional information in direct access files is most convenient not only because data can be accessed randomly but also because the access times for all items of information held on file are fast and uniform.

5.3) contd.

During learning sessions, two buffers are used to hold the response and subframe data of the current frame. When control passes on to the next frame in the lesson, the contents of these buffers are immediately replaced by the data for this new frame. Therefore, the most convenient storage format is for each record of the direct access files to contain the response data or the subframe data of one frame.

A bucket size of 512 words was chosen for both the response lesson-file and the subframe lesson-file. To minimise the wastage of storage space and to maximise the possible size of each frame, a bucket size of 1K words could have been chosen. However, this would have increased the core storage requirement of each student by about 3K words, 40% of the present requirement. In general, the response and subframe data for one frame varies in size from around 100 to 300 words. Therefore a limit of 512 words is not expected to be a very serious limitation.

For the reading and writing of this data, the storage device macros of the ICL Housekeeping system are used. At this logical level of control one is allowed to read and write one record for each read or write instruction. Thus, one need not be concerned with the transfer of physical buckets in which the records are batched.

5.3.1) Response Lesson-files.

The first four buckets of each response lesson-file are set aside for the storage of all the

5.3.1) contd.

data which is required by the lesson as a whole.

.BUCKET.	.CONTENTS.
1	Index of logical bucket numbers
2	Response Title Form data
3 & 4	Frame performance records
5 onwards	Response data.

Bucket 1 contains the index of the logical bucket numbers for each frame in the lesson. A logical bucket number of zero indicates that there is no frame in the lesson with this frame number.

Bucket 2 contains all the information entered on the response title form. This data includes the lesson name, the author's name, the state of the four parameters and the text of the introductory message.

Buckets 3 & 4 contain the performance records for each frame in the lesson. During the learning session, these records will be updated and the new records replaced for future use.

At the start of each lesson all the information contained in buckets 1 to 4 is read into special buffers in core store.

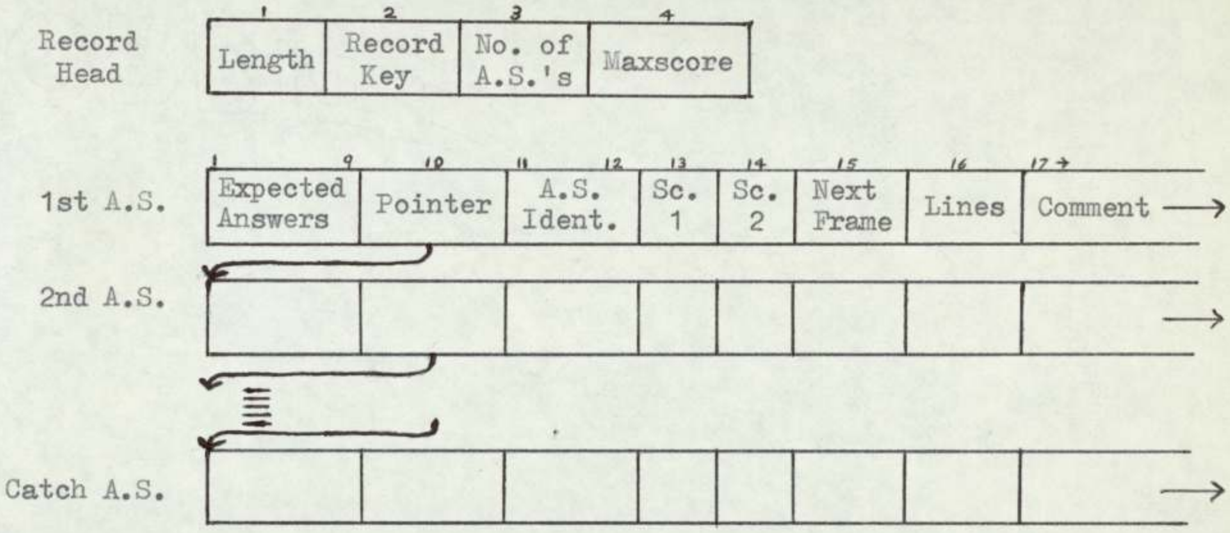
5.3.1) contd.

Buckets 5 and onwards contain the instructional information of the lesson.

As these latter buckets are required only one at a time, one buffer in core store is sufficient.

5.3.2) Response Record Structure.

The record structure employed in the response lesson-files is necessarily quite complex. This results from the fact that we have to deal with variable numbers of answer sets of variable length.



where:- Record Key = Frame Number

A.S. = Answer Set

Ident. = Identifier

Sc. = Score

5.3.3) Subframe Lesson-files.

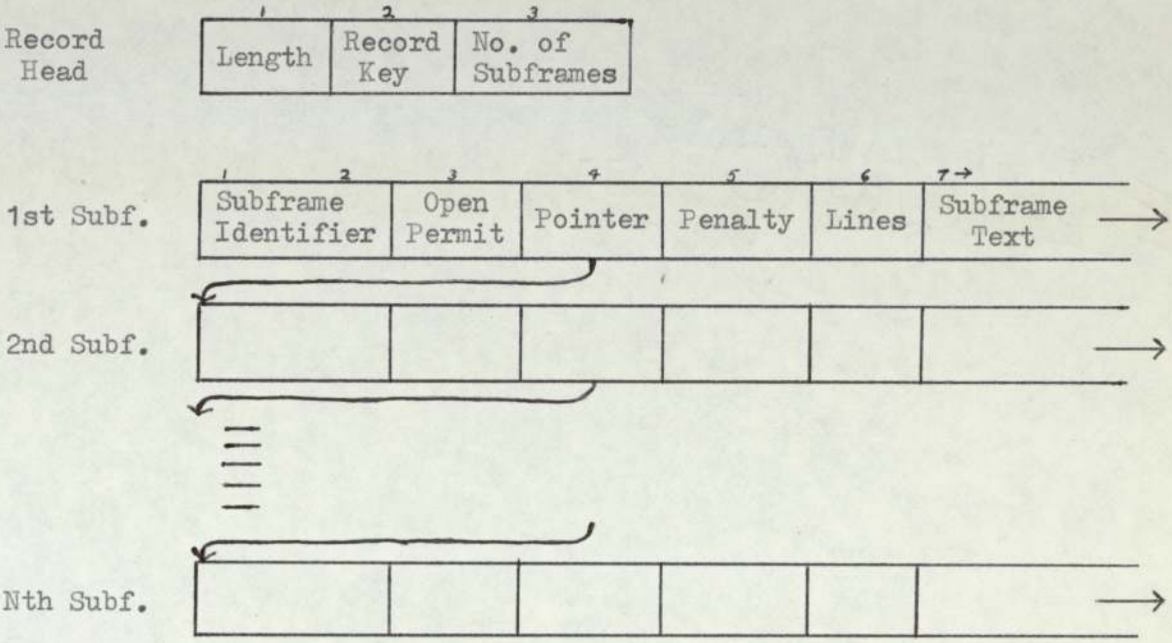
.BUCKET.	.CONTENTS.
1	Index of logical bucket numbers
2 onwards	Subframe data

Bucket 1 contains the index of the logical bucket numbers for each frame in the lesson which has been supplied with subframe data. A logical bucket number of zero indicates that the frame either does not exist or has no subframe data. Again, at the start of each lesson this index is read into a special buffer in core store.

Buckets 2 and onwards contain the subframe data for the lesson.

5.3.4) Subframe Record Structure.

As with the response records, the record structure is quite complex. Here we have to deal with a variable number of subframes of variable length.



where:- Subf. = Subframe

5.4) Student Sessions.

This section discusses the various facilities offered to the student during the learning session. The underlying teaching strategy and the method of response evaluation employed by ACATS are also outlined.

5.4.1) Control Program.

The interaction between the student and the lesson material to be learned is controlled by the main ACATS control program, TEACHER (see Appendix 5). This program consists of 1200 PLAN source statements and requires only 7½K words of core store.
The program controls :-

i) The acceptance and classification of student responses.
ii) The evaluation and marking of answers.
iii) The presentation of feedback to the student.
iv) The recording of student and frame performance data.
v) The recognition and servicing of student commands.
vi) The routeing of students through the instructional programmes.

5.4.2) Starting a Lesson.

The student may initiate any lesson simply by issuing the TEACH macro command (see Appendix 4), followed by the lesson name of the lesson he wishes

5.4.2) contd.

to study.

There is then a short delay before the learning session begins during which time the compiled version of the control program is loaded into core, the appropriate lesson-files are assigned and all necessary data is transfered into buffers in core store. This delay is normally in the region of about one to six minutes.

Once the lesson is prepared and ready for use, various headings are printed out and the student is invited to type his name.

It is possible that the student may not wish to start at Frame 1 in the lesson. At this point, therefore, the student is asked whether he wishes to start at Frame 1 or to restart at some other frame. If he opts to restart, he will then be asked to give the number of the frame at which he wishes to begin. Once the starting frame has been determined, further headings are printed.

Finally, the introductory message provided by the author is presented to the student giving any necessary information about the lesson to be studied.

Once this message has been printed the learning session has begun. He is then given an invitation to type his response to the question in the starting frame.

EXAMPLE.

This is an example of a typical introduction to an ACATS lesson, up to the start of the actual learning stage itself.

N.B. The symbol → is an invitation for the student to type.
All lines beginning with this symbol have been typed by the student.

```
→TEACH  ENGLISH-2
DISPLAY:  LESSON ENGLISH-2 WILL START IN APPROX. 5 MINUTES.

***  **  **  **  **  **  **  **  **  **  **  **  **  **  ***
THIS IS A.C.A.T.S. (ASTON COMPUTER-ASSISTED TEACHING SYSTEM)

WHAT IS YOUR NAME ?
→JOHN SMITH

DO YOU WISH TO START AT FRAME 1 ?   (TYPE "YES" OR "NO")
→NO
WHICH FRAME DO YOU WISH TO START AT ?
→AT FRAME 21.

STARTED LESSON ENGLISH-2 BY JOHN SMITH ON 14/08/73 AT 12/20/15.
            (RESTARTED AT FRAME 21)

THIS IS THE SECOND LESSON OF THE ENGLISH COURSE.
YOU SHOULD NOT ATTEMPT THIS LESSON UNTIL YOU HAVE
SUCCESSFULLY COMPLETED LESSON ENGLISH-1.
START AT THE FIRST FRAME NOW.

→
```

5.4.3) Time Saving.

If it is envisaged that more than one lesson will be studied in a single learning session (this will be particularly so when the restart facility is being used) the student may save a certain amount of time by issuing the TEACH command followed by two lesson names separated by a comma.

Although there will still be a short delay between lessons, the computer will automatically proceed to the second lesson without requiring the student to issue a second TEACH command and without having to reload the control program. Thus, small but appreciable savings in both time and money can be made.

5.4.4) Student Commands.

During the learning sessions the student is allowed a certain degree of control. There are five commands that he may give. All commands begin with an asterisk to distinguish them from answers to questions.

These five commands are :-

i) *AID

The student can obtain information about the subframes available and the penalties associated with each by issuing the *AID command. Only those subframes which the student is allowed to use at the time of issuing the *AID command will be listed.

5.4.4) contd.

ii) *GIVE.

The student may list the contents of any subframe by issuing the *GIVE command followed by the subframe identifier. Before printing the text of the subframe, the control program checks whether the student is in fact allowed to call the subframe. Each subframe may only be called once.

iii) *SCORE.

The *SCORE command will cause the student's percentage score up to the current point in the lesson to be printed.

iv) *FRAME.

The *FRAME command will cause the frame number of the current frame to be printed. This command is sometimes useful if the student is unsure of his current position or if the author fails to inform the student which frame to study next.

v) *STOP

The student may end the learning session at any stage in the lesson by issuing the *STOP command.

By using the restart facility, the student may start at this point in the lesson in his next learning session.

It should be noticed that these commands have been supplied for the convenience of the student. All learning sessions can be conducted without any of

5.4.4) contd.

these commands being issued.

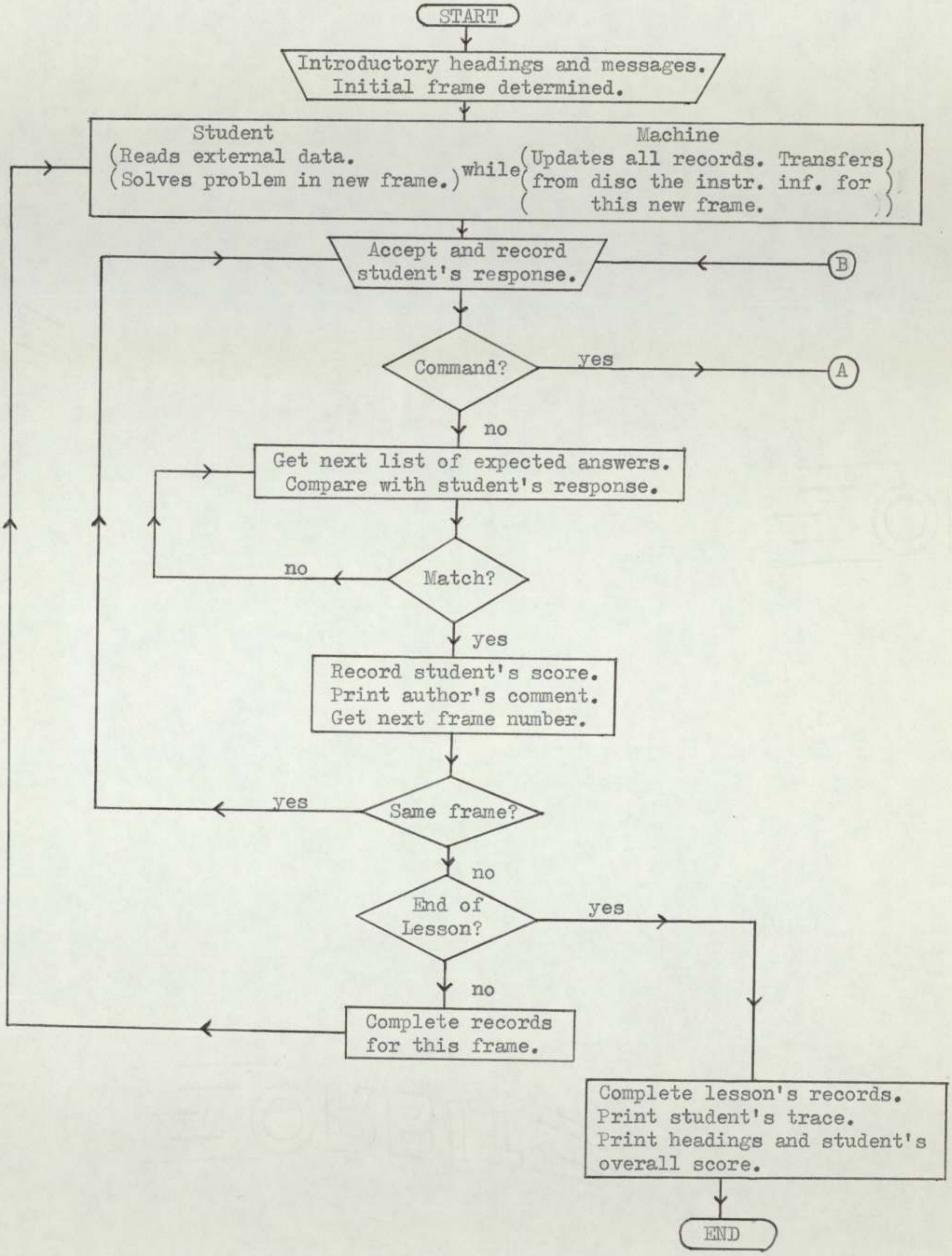
In effect, the only computing knowledge required of the student, apart from the use of the TEACH command, is the operation of a teletype terminal.

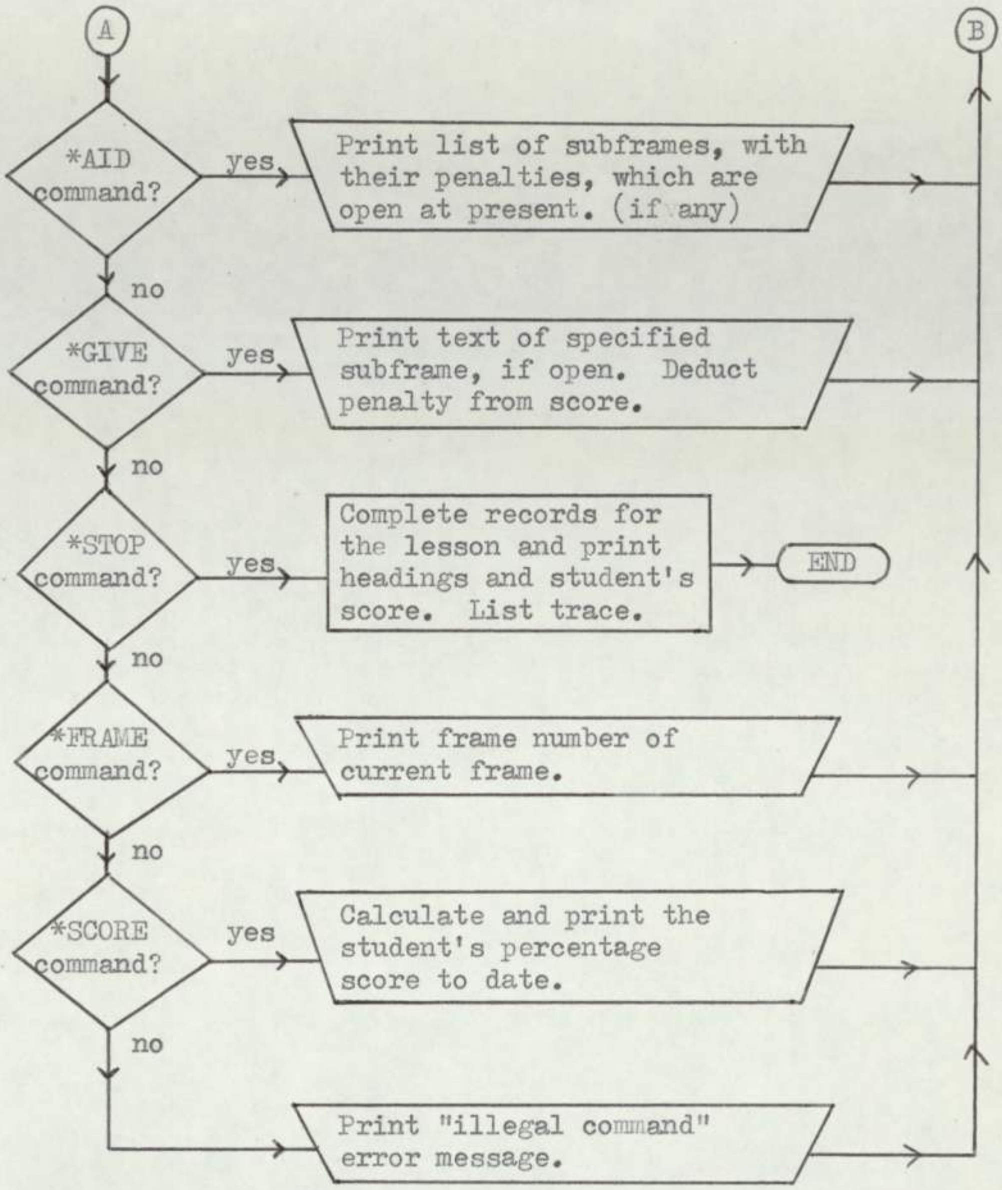
5.4.5) Teaching Strategy.

The teaching strategy employed by ACATS is very flexible in nature. An almost infinite variety of tutorial logics can be used. As a result, ACATS can be used in almost all teaching situations.

This in-built teaching strategy, governed by the control program, can best be illustrated with the use of a flow diagram.

.ACATS TEACHING STRATEGY.




A
B
*AID command?
yes
Print list of subframes, with their penalties, which are open at present. (if any)
no
*GIVE command?
yes
Print text of specified subframe, if open. Deduct penalty from score.
no
*STOP command?
yes
Complete records for the lesson and print headings and student's score. List trace.
END
no
*FRAME command?
yes
Print frame number of current frame.
no
*SCORE command?
yes
Calculate and print the student's percentage score to date.
no
Print "illegal command" error message.

5.4.6) Response Evaluation.

Student responses are evaluated by means of a keyword search. These keywords or key phrases are supplied by the author and entered in the expected answers area of the response data form. Each of these areas may contain any number of expected answers, however, each item in the list must be terminated by the @ symbol.

Student responses are examined for the existence or non-existence of each expected answer in turn. For a match to occur between an expected answer (keyword) and the student's response, the following conditions must be satisfied:-

i) An exact copy of the keyword must be contained in the first forty characters of the student's response.

ii) Unless the first character of the keyword is in the first character position of the student's response, the character immediately prior to the keyword in the response must be a space.

iii) The character immediately following the last character of the keyword in the response must be either:

   (a) a space.

   (b) a comma.

   (c) a period.

   (d) a question mark.

These conditions rule out the possibility of incorrect responses being accepted while at the

5.4.6) contd.

same time allowing the student a great deal of freedom in forming his responses.

5.4.7) Scoring Modes.

There are two scoring modes, strict and non-strict. Part of the data for each answer set consists of two scores, termed Score 1 and Score 2. Which score the student is awarded is dependent on the scoring mode being used and the number of attempts he has made at the question.

Scores are awarded in the following fashion:-

ATTEMPT	STRICT MODE	NON-STRICT MODE
1st	Score 1	Score 1
2nd	Score 2	Score 2
All others	No Score	Score 2

By varying the scoring mode and the values of Score 1 and Score 2, a very large variety of scoring methods may be used by the authors.

5.4.8) User's Guide.

A brief non-technical user's guide was written for the students, the main aim of the guide being to inform the student how to operate a teletype terminal and to use ACATS to best advantage. It was decided that one of the most

5.4.8) <u>contd.</u>

essential features of ACATS should be the provision of concise and easily comprehensible user manuals. This manual also contains a simple introduction to the principles and techniques of programmed instruction.

5.5) Performance Records.

ACATS provides facilities for the recording of both student and frame performance records. Both these facilities are optional and can be switched off, by the authors, independently.

5.5.1) Student Performance Records.

In the case of student performance records, it was decided that all recorded data should be printed out on the line printer in the form of a trace immediately after each lesson was completed. During learning sessions this data is stored temporarily on a workfile.

The reasons for not storing these records on the computer were as follows :-

i) It would become essential for all students to identify themselves with an exact name or code.

ii) With a large number of students, a proliferation of MOP files would result.

iii) Since the most important items of this data are the student's responses themselves, these records would tend to be quite bulky.

After each student session, a trace of the student's progress is output on the line printer. This trace contains the information :-

i) The order of frames studied.

ii) List of student's answers and commands given in each frame.

5.5.1) <u>contd</u>.

iii) Details of the student's performance in each frame.

iv) A summary of the lesson.

The trace consists of a number of units of information of the form :-

```
FRAME X

          ......R1......
          ......R2......
                "
          ......Rn......
A  B  C         D%
```

where:- X = Frame Number.

R1,R2..Rn = Student's answers and commands in the order they were given.

A = Number of attempts made at the question.

B = Marks gained.

C = Penalties incurred through use of subframes.

D = Percentage score for frame.

At the end of each trace a summary of the lesson is given. Normally, this summary consists of the name of the student, the lesson name and the student's overall percentage score.

However, in certain cases, one or two other items or code letters may be included. These two code letters, R and Q, indicate the particular way in which the lesson was started and finished.

5.5.1) <u>contd</u>.

The existence of the code letter "R" indicates that the lesson was in restart mode.

The existence of the code letter "Q" indicates that the lesson was quitted, or ended prematurely, by the student issuing the *STOP command.

Student responses can then be examined by the authors for, amongst other points:-

i) Correct answers that have not been predicted.

ii) Popular incorrect answers that have not been predicted.

iii) Unusual answers indicating misunderstandings on the part of the student.

iv) Mis-spelling or badly formed responses.

v) Mis-use, non-use or excessive use of student commands.

Once the trace has been examined by the author the whole trace or the summary of the lesson can be filed for future reference.

In conjunction with the frame performance records, this information can be used in the development of the lesson material.

On its own, it can be used to judge the performance and progress of any particular student.

.EXAMPLE OF TRACE.

```
TRACE OF LESSON GEOGRAPH-2 BY JOHN SMITH ON 14/08/73.


FRAME      1

                LONDON

   1  10    0      100%


FRAME      2

                GLASGOW

                *AID

                *GIVE HINT

                EDINBURGH

   2   6    1      50%


FRAME      3

                *SCORE

                THE CAPITAL OF PORTUGAL IS LISBON.

   1  10    0      100%


FRAME      4

                 *STOP


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

STUDENT     LESSON        SCORE

JOHN SMITH  GEOGRAPH-2    83%       (Q)

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

5.5.2) Frame Performance Records.

In contrast to the student performance records, these records are stored on the computer, and are continually updated as more performance data becomes available.

Five items are recorded for each frame. These items take the form of accumulated totals. In this way, the updating of the frame performance records can be acheived efficiently and simply by incrementing the appropriate totals.
These five items are :-

i) The total number of times the frame has been studied.

ii) The total number of attempts that have been made at the question.

iii) The total number of subframes used.

iv) The total maximum possible score.

v) The total of the scores gained by the students.

The frame performance records for each lesson are conveniently stored in buckets 3 and 4 of the response lesson-file. ACATS authors may examine the contents of these records by issuing the PRINTRECS command (see section 5.6.5). This command causes a table of frame performance records to be output on the line printer.

5.5.2) contd.

The table includes the following four items for each frame in the lesson :-

i) The number of times the frame has been used. This gives, among other things, an indication of the accuracy of the records in the table.

ii) The average number of attempts made at the question.

iii) The average number of subframes used.

iv) The average percentage score.

This information can be used to accurately pinpoint errors and ambiguities in the lesson material. Should these faults not be immediately obvious to the author, this data can be combined with the more detailed information derived from the "trace" facility.

Normally, during the developmental stages of the programme, this data will be recorded and examined at regular intervals by the authors. Once the author is satisfied that his programme is error free, he can easily stop the recording of this data by changing the state of one of the parameters in the response title form.

.EXAMPLE.

```
                 PRINT OF FRAME PERFORMANCE DATA.

                 LESSON ENGLISH-01  ON 15/10/73.

                           ***********



             TIMES     AV. NO. OF      AV. NO. OF          AVERAGE

   FRAME.    USED.      ATTEMPTS.    SUBFRAMES USED.      %SCORE.

     1         8            1               0                100

     2         8            1               0                 92

     3         8            1               1                 63

     4         3            1               0                100

     5         3            1               0                100

     6         8            2               1                 60

     7         8            1               1                 81

     8         8            1               0                100

     9         8            1               1                 88

    10         8            1               0                100

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

5.6) Author Sessions.

This section discusses the facilities offered to the authors during terminal sessions.

5.6.1) Author Commands.

There are eight macro commands at the author's disposal. They are written in the GEORGE Command Language (ICL, 1971a). For simplicity, all these commands require one parameter only, namely the lesson name of the lesson to be operated on.

The commands are :-

COMMAND	ACTION
CRF	Creates responce lesson-file.
CSF	Creates subframe lesson-file.
MAKEREADY	Retrieves files which are not on-line.
LIST	Lists responce and subframe input files on the line printer.
PRINTRECS	Prints frame performance records.
RESETRECS	Resets frame performance records.
ERF	Edits responce input-file.
ESF	Edits subframe input-file.

A listing of all these macro commands can be found in Appendix 4.

5.6.2) Creation of Lesson-files.

Once the response and subframe data have been input to the computer, a further stage of preparation

5.6.2) contd.

must be carried out before the lesson is ready for use.

The response and subframe data have been read in from cards into the input-files. Before this data can be used in a teaching situation it must first be checked for errors and then organised and stored in direct access lesson-files.

Response and subframe lesson-files are created separately and require the use of the commands CRF (Create Response File) and CSF (Create Subframe File) respectively.

Listings of the source programs (LOADRPF and LOADSFF) which carry out this stage of preparation can be found in appendix 6. and 7 respectively.

After successful completion of each of these commands, a numbered listing of the appropriate input-file is automatically printed on the line printer.

5.6.3) Retrieving Files.

Because of the very large number of files stored on the computer, only those files which are regularly used can be held on-line. An attempt to use a file which is not on-line has been found to result in a delay of around 15 to 30 minutes.

ACATS authors may overcome, or at least alleviate, this problem by issuing the MAKEREADY command at the start of, or a short time before

5.6.3) contd.

each ACATS session (student or author) at the terminal.

The MAKEREADY command differs from all other commands in that it can be followed by up to three lesson names, each separated by a comma.

The issue of the MAKEREADY command will result in the retrieval of all ACATS macro commands and compiled programs, plus all input-files and lesson-files of the lessons specified as parameters of the command.

5.6.4) Listing Files.

A numbered listing of a lesson's input-files can be obtained by issuing the LIST command. An up-to-date numbered listing is essential for the editing of any input-file.

5.6.5) Printing of Frame Performance Records.

During each student session, various items of information are recorded for each frame. These are termed the frame performance records.

The frame performance records of any lesson can be obtained by the author by issuing the PRINTRECS command. A listing of the source program (RECS) which is used in the printing of these records can be seen in Appendix 8.

This recorded information can be used to accurately pinpoint errors and ambiguities in the

5.6.5) contd.

lesson material, especially when combined with the information derived from the "trace" facility (see section 5.5.1).

Each lesson's frame performance records should be inspected at regular intervals during the developmental stage, and recorded until the author is fully satisfied with the lesson material.

The method of inspecting these records depends on the type of the instructional programme. With linear programmes, student errors should be minimised, therefore frames with a low average percentage score should be simplified or stated more clearly. With intrinsic programmes, branching frames with a very high average percentage score should be made more difficult, otherwise the remedial frames will not be of use.

An example of a printout of a typical set of frame performance records can be seen on page 72.

5.6.6) Resetting Frame Performance Records.

The frame performance records of any lesson can be reset (set to zero) by issuing the RESETRECS command. These records are automatically reset when the response lesson-file, in which the records are stored, is created or re-created.

A listing of the source program (RESET) which is used to reset these records can be seen in App. 9.

5.6.7) Editing of Input-files.

As a result of errors found through usage, it may be necessary to amend the input-files. This can be done via the teletype terminals and does not require the re-punching or re-entry of the lesson material.

As the ICL editor is of a very complex nature, an alternative, simplified editor has been written solely for the use of instructional programmers in which editing can be carried out with the use of a few very simple editing instructions.

A listing of this editor (NEWED) can be found in Appendix 10.

The editing of response and subframe input-files is carried out separately, and requires the use of the commands ERF (Edit Response File) and ESF (Edit Subframe File) respectively. An up-to-date numbered listing of the appropriate input-file is essential during the editing process.

The editing instructions are as follows :-

INSTRUCTION	ACTION
#COPY line-number	Copy lines up to but not including the line line-num.
#DELETE line-number	Delete the lines up to but not including the line line-number.
#END	Copy the remainder of the file.

5.6.7) <u>contd</u>.

#ABANDON	Abandon the edit, leaving the file in its original condition.

Insertions are made simply by typing the line to be inserted rather than an editing instruction. The inserted line is placed immediately before the current line.

The current line number is printed immediately before each editing instruction. When an incorrect editing instruction is given, a self-explanatory error message is output on the typewriter log and no action is taken.

After the successful completion of each edit, a numbered listing of the new input-file is automatically printed on the line printer, and the old input-file is erased. It is now necessary to re-create the appropriate lesson-file from this new input-file. All the author need do is issue the CRF or CSF command. This will cause the old version of the lesson-file to be erased, and the new lesson-file to be created in its place. The frame performance records for this new version of the lesson-file will be automatically reset.

EXAMPLE.

Editing of the response input-file for the lesson NUMBERS-01.

Existing response input-file. Desired response input-file.

```
    0   ONE                       0   ONE
    1   TWO                       1   TWO
    2   FIVE                      2   THREE
    3   FIVE                      3   FOUR
    4   FIVE                      4   FIVE
    5   SIX                       5   SIX
    6   NINE                      6   SEVEN
    7   TEN                       7   EIGHT
    8   ****                      8   NINE
                                  9   TEN
                                 10   ****
```

```
10.05.40←ERF  NUMBERS-01
EDITOR IS READY
                                      EFFECT
   0
←#COPY  2                            Copy lines 0 & 1.
   2
←THREE                               Insert "THREE".
   2
←FOUR                                Insert "FOUR".
   2
←#COPY  3                            Copy line 2.
   3
←#DELETE  5                          Delete lines 3 & 4.
   5
←#COPY  6                            Copy line 5.
   6
←SEVEN                               Insert "SEVEN".
   6
←EIGHT                               Insert "EIGHT".
   6
←#END                                Copy lines 6,7 & 8.
EDIT COMPLETED
10.08.23←
```

5.6.8) Author Manual.

A comprehensive, non-technical manual was written for ACATS authors. As previously stated, the provision of concise and easily comprehensible user manuals was considered to be one of the most essential features of ACATS.

This manual contains :-

i) A description of the main principles and techniques of programmed instruction.
ii) An outline of the aims of ACATS and the advantages it has over existing CAI systems.
iii) A detailed description of the method employed by ACATS to enter instructional material onto the computer.
iv) Instructions on the operation of a teletype terminal.
v) A detailed description of the many facilities offered by ACATS, and instructions on how best to use them.

CHAPTER 6.

TRIALS AND CONCLUSIONS.

6.1) Experiments.

To obtain some indication of the effectiveness of ACATS as an instructional aid, certain experiments were carried out. These trials were not meant to compare the relative efficiency of ACATS and conventional teaching methods, as ACATS was intended as a supplement to lectures rather than as a replacement. The main objectives were to judge the performance of various aspects of the teaching system and to note the reactions and opinions of the students involved in the trials.

For this purpose a set of eight linear programmes on statistics were prepared, each programme being of approximately thirty minutes duration. The printed handout for one of these programmes can be seen in Appendix 1.

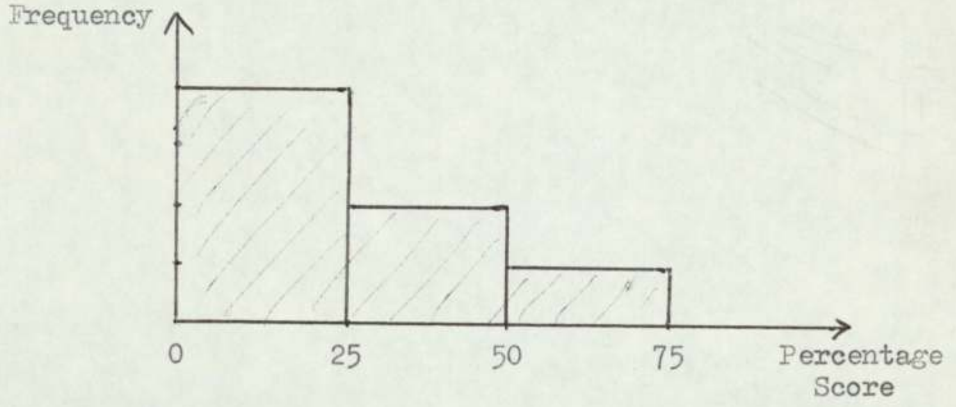
The test procedure adopted was pre-test, practice, post-test, the first and last being identical. Seven students of different backgrounds and experience were involved in the trials. Although with such a small number of students one cannot draw any definite conclusions about the performance of ACATS, the results which were obtained do suggest that it would be a very useful and effective aid to learning.

The test paper which was used in this experiment is shown in Appendix 11.

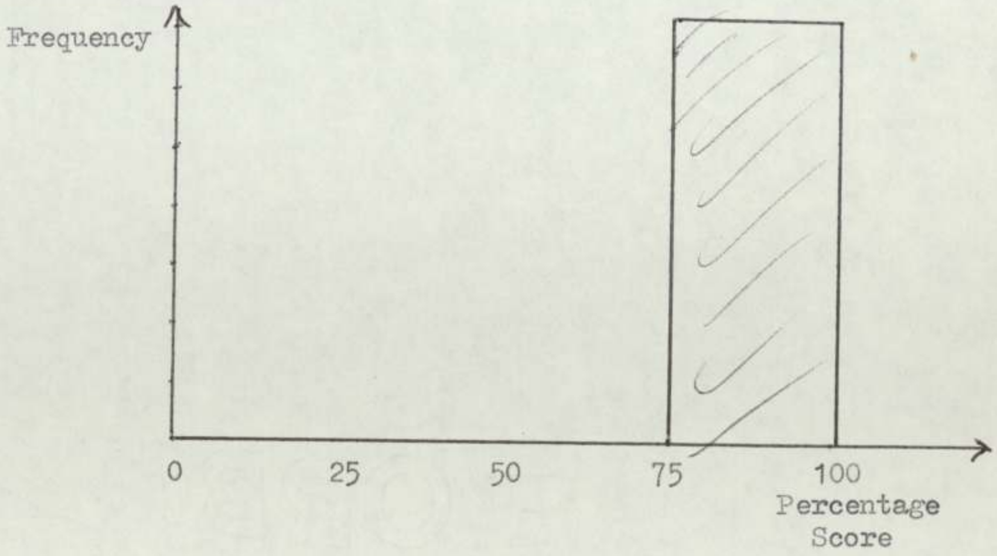
RESULTS.

SUBJECT	EDUCATIONAL STANDARD	EXPERIENCE IN COMP. SC.	EXP. IN STATS.	PRE-TEST SCORE	POST-TEST SCORE
1	Post-graduate	NO	NO	15	98
2	Post-graduate	NO	YES	53	93
3	Post-graduate	YES	YES	48	90
4	Undergraduate	YES	YES	45	85
5	A-level	NO	NO	0	85
6	Post-graduate	NO	YES	20	85
7	Undergraduate	NO	NO	0	80

PRE-TEST DISTRIBUTION.



POST-TEST DISTRIBUTION.



6.1) contd.

Two points of interest can be seen from these results. Firstly, as is normally the case with programmed instruction, the lessons were equally effective with all types of students.

	Pre-test	Post-test
Average score of students with statistics experience.	42	88
Average score of students without statistics experience.	5	88

Secondly, the students who had no experience of working with computer terminals very quickly adapted themselves to this form of education.

	Post-test
Average score of students with computing experience.	88
Average score of students without computing experience.	88

6.2) Attitudes of Students.

As expected, the novelty of this method of education had a great effect on the students, and throughout the trials they showed a great deal of keenness and enthusiasm. Each student, on completion of the test procedure, was asked to complete a questionnaire.

Despite the very slow speed of the teletype terminals and the fact that the lessons held little interest for the students, their reaction to this method of education was very favourable indeed. Three of the students actually prefered this method of teaching to lectures. All agreed

6.2) contd.

that ACATS would be useful as a supplement to lectures, particularly for the purposes of revision, elementary work and supplementary courses.

The students' replies to the questionnaire were as follows :-

	QUESTION	TRUE	FALSE
1	The lessons were too short	1	6
2	The lessons were too long	0	7
3	The questions were too easy	2	5
4	The questions were too difficult	0	7
5	There was too little instruction in each frame	2	5
6	Ther was too much instruction in each frame	1	6
7	The lesson material was too repetitive and boring	3	4
8	The typewriter was too slow	1	6
9	My typing was too slow	2	5
10	The noise of the typewriter was very irritating	0	7
11	I was not able to concentrate at the typewriter	1	6
12	The response from the system was too slow	2	5
13	The lesson material should be stored on the computer despite the slow speed of the teletypes	1	6

6.2) contd.

	QUESTION	TRUE	FALSE
14	Having printed handouts is a good idea	7	0
15	Lessons would be a lot more interesting if the lesson material was stored on slides, film, microfilm or tapes	5	2
16	This method of teaching is preferable to lectures	3	4
17	This method of teaching would be very useful as a supplement to lectures	7	0
18	This method of teaching is too impersonal	5	2
19	If combined with tutorial sessions, this would not be the case	7	0
20	Being continually questioned on the lesson material is a good idea	7	0
21	Feedback messages are beneficial	7	0
22	I prefer learning at my own pace rather than at the lecturer's pace	7	0
23	This method of teaching is suited to both original work and revision	7	0

6.3) Performance of ACATS.

6.3.1) Response Times.

As one would expect, the response times varied considerably depending on the load on the machine at the time. However, the average response time was about two to three seconds, more than adequate for a system using teletype terminals.

6.3.2) Mill Usage.

ACATS was found to place an extremely small demand on the central processor. One student terminal hour required from three to five seconds of mill time. As machine time at the University of Aston was charged at the rate of 1p per second, this worked out at an average of 4p per student terminal hour for machine time.

It was later found that about two seconds of this time is used in loading the program. Therefore, with a few alterations to the TEACH macro it would be possible to further reduce this cost.

6.3.3) Cost.

The costs for a five year project were estimated as follows :-

(i)	Twelve teletype terminals at £500 (5% educational discount)	£6,000
(ii)	Maintenance of teletypes at 5% of cost per year	£1,500
(iii)	Machine time at 4p per student terminal hour	£4,000
(iv)	Stationary and filestore rental	£500
	TOTAL COST =	£12,000

6.3.3) contd.

Total student terminal hours, approx. 80,000

Estimated cost per student terminal hour = 15p

To be added to this figure is the cost of the development of the instructional programmes used. No accurate figure can be given for this, but it must be taken into account that the time spent in the production of the instructional material will be counteracted by the smaller demands on the teacher's time.

6.4) Further Work.

A very powerful computer assisted teaching system could be set up using ACATS adapted for visual display units and employing 1900 Driver and 7903 remote housekeeping. The system would be capable of handling up to 100 remote student terminals simultaneously with a core store requirement of just 32K words.

The programming effort involved in developing such a system I would estimate at one to two man years. Terminal costs would be high, but the resulting system could handle the needs of a fairly large number of universities, colleges and schools.

.APPENDIX 1.

A1) EXAMPLE OF PRINTED HANDOUT.

The following pages contain the handout for one of the lessons of the statistics course used in the ACATS trials. The programme is linear and 'missing word' questions have been used.

The first page contains some general information about the lesson. The second page is used by the student for reference purposes. The remaining pages consist of the instructional material (external data) itself.

.ASTON COMPUTER - ASSISTED TEACHING SYSTEM.

.HANDOUT S2.

LESSON NAME:	STATS-02
SUBJECT:	STATISTICS
DESCRIPTION:	MATHEMATICAL MODELS
AUTHOR(S):	A D FLOCKHART
TYPE:	LINEAR
EXPECTED DURATION:	20 - 25 MINUTES

SUMMARY OF ENTRY PROCEDURE

1) LOGIN MOPINITIALS,:SSP0707

2) ACATS

3) TEACH LESSONNAME

SUMMARY OF STUDENT COMMANDS

COMMAND	ACTION
*STOP	ENDS LESSON
*SCORE	GIVES YOUR PERCENTAGE SCORE
*FRAME	GIVES YOU THE CURRENT FRAME

THE "*AID" AND "*GIVE" COMMANDS WILL BE INEFFECTUAL IN THIS LESSON AS NO AID HAS BEEN PROVIDED.

1 Once the pertinent aspects of a situation have been identified and designated as data they may be treated statistically by a set of ------------ techniques.

7 A mathematical model thus consists of a set of --------, the --------- among them, and the ---------- that may be performed on them.

! Separate answers with commas only, please !

13 When applying statistics to data, care should be taken to ascertain to what extent the operations provided in the ------------ ----- are meaningful for the data.

2	The application of mathematical ---------- involves the use of "mathematical models".
8	Mathematical models used in statistics generally use numbers as elements, but vary in the extent to which they use all the possible relations among numbers and all possible ----------- that may be performed on numbers.
14	For convenience, numbers are sometimes assigned to data for which the mathematical model of arithmetic is not applicable. If punched cards are being used, the number 20 might designate a person who lives in Moseley; the number 40 someone who lives in Aston. There are many r-------- between the two numbers 20 and 40 which have no meaning in describing residents of the two districts.

3	A mathematical model is based, first of all, on a set of elements. Thus the numbers in the number system we use might be the elements of a ------------ -----.
9	Using a mathematical model that is not appropriate for one's data can lead to absurd results. If a psychologist asserted that a team of three boys, each with 50 I.Q., could solve a complicated problem as well as one boy with 150 I.Q., because 50+50+50=150, he would be using a model that was not suitable for his ----.
15	For example, the fact that 40 is twice 20 has no corresponding -------- in the data. It makes no sense to say that someone who lives in Aston is twice someone who lives in Moseley.

4	A mathematical model also specifies the relations among the elements. For example, the facts that 8 is more than 6 and less than 30, and that 50 is 5 times as much as 10, are ---------- among numbers, which, as we have just seen, could form the elements of a model.
10	The arithmetical --------- of addition that the psychologist performed was not appropriate for his data because I.Q.'s of different people cannot be added in this way.
16	Likewise, many other arithmetical operations on these numbers would have no meaning for studying the two districts. The complete ------------ ----- of arithmetic is too comprehensive for these data.

5	Finally, a mathematical model describes the <u>operations</u> that may be perfomed on the -------- of the model.
11	On the other hand, if one had three blocks of wood, each 50 centimetres high, and piled them one on top of the other, the structure would be 150 centimetres high. In this case the arithmetical operation of addition does have a meaningful interpretation in the ----.

6	Adding, subtracting, taking square root, etc., are ---------- that may be performed on numbers, and hence would belong in the mathematical model.
12	Some mathematical models used in statistics are very comprehensive; others need to be more restrictive because certain relations and operations in the broader ------ do not have interpretations in some kinds of data.

.APPENDIX 2.

A2) EXAMPLE OF COMPLETED DATA ENTRY FORMS.

This appendix contains the completed forms for the entry of the response data associated with the lesson in Appendix 1.

N.B. The Response Data Forms for the first three frames only have been included.

RESPONSE TITLE FORM.

Lesson name

INPUT :SSP07C7,CRSTATS-02

Lesson Name

STATS-02

Author

A.D. FLOCKHART.

BLOCK*
CAPITALS
ONLY
PLEASE

Parameters (yes = 1, no = 0)

1 Strict rote scoring?

1 Frame performance data to be recorded?

1 Trace of lesson to be listed?

0 Instructional programme stored on computer?

Introductory Message

(1) THIS IS THE SECØND LESSØN ØF THE CØURSE
(2) ØN STATISTICS. YØU SHØULD NØT ATTEMPT
(3) THIS LESSØN UNTIL YØU HAVE SCØRED AT
(4) LEAST 80% IN LESSØN STATS-01.
(5) CØMMENCE THE LESSØN NØW.
(6)
(7)
(8)
(9)
(10)
(11)
(12)

////

.ASTRE COMPUTER-ASSISTED TEACHING SYSTEM.

.RESPONSE DATA FORM.

Author's Name A. D. F.	Frame Number 1
Lesson/File STATS-02	Page Number 1

Frame Number: *001

No. of Answer Sets: 2

Maxscore: 10

BLOCK*
CAPITALS
ONLY
PLEASE

A.S. Identifier: CORRECT

Next Frame: *002

Score 1: 10

Score 2:

Lines: 4

Expected Answers: MATHEMATICAL@

Comment

(1) GOOD. YOU SEEM TO HAVE REMEMBERED OUR
(2) DEFINITION OF STATISTICS:-
(3) "A SET OF MATHEMATICAL TECHNIQUES
(4) FOR MAKING DATA MORE USEFUL".
(5)
(6)
(7)
(8)
(9)

A.S. Identifier: WRONG

Next Frame: *002

Score 1:

Score 2:

Lines: 4

Expected Answers: @

Comment

(1) NO. REMEMBER WE DEFINED STATISTICS IN
(2) THE LAST LESSON AS BEING A SET OF
(3) "MATHEMATICAL" TECHNIQUES FOR MAKING
(4) DATA MORE USEFUL.
(5)
(6)
(7)
(8)
(9)

ASTON COMPUTER-ASSISTED TEACHING SYSTEM.

RESPONSE DATA FORM.

Author's Name	Frame Number 2
Lesson/File	Page Number 1

Frame Number (1–4): X002

No. of Answer Sets (5–6): 2

Maxscore (7–8): 10

BLOCK*
CAPITALS
ONLY
PLEASE

A.S. Identifier (1–8): CØRRECT

Next Frame (9–12): X003

Score 1 (13–14): 10

Score 2 (15–16):

Lines (17): 1

Expected Answers (1–40): TECHNIQUES∂METHØDS∂

Comment (1–40)

(1) YES.
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)

A.S. Identifier (1–8): WRØNG

Next Frame (9–12): X003

Score 1 (13–14):

Score 2 (15–16):

Lines (17): 1

Expected Answers (1–40): ∂

Comment (1–40)

(1) NØ. THE CØRRECT ANSWER IS "TECHNIQUES".
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)

ASTON COMPUTER-ASSISTED TEACHING SYSTEM.

RESPONSE DATA FORM 'A'.

Author's Name	Frame Number 3
Lesson/File	Page Number 1

BLOCK*
CAPITALS
ONLY
PLEASE

Frame Number (1–4): *003
No. of Answer Sets (5–6): 2
Maxscore (7–8): 10

A.S. Identifier (1–8): CORRECT
Next Frame (9–12): *004
Score 1 (13–14): 10
Score 2 (15–16):
Lines (17): 1

Expected Answers (1–40): MATHEMATICAL MODEL@

Comment (1–40)
(1) RIGHT.
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)

A.S. Identifier (1–8): WRONG
Next Frame (9–12): *004
Score 1 (13–14):
Score 2 (15–16):
Lines (17): 2

Expected Answers (1–40): @

Comment (1–40)
(1) NO. THE CORRECT ANSWER IS:-
(2)   "MATHEMATICAL MODEL".
(3)
(4)
(5)
(6)
(7)
(8)
(9)

.ASTON COMPUTER-ASSISTED TEACHING SYSTEM.

.Terminators.

1			4
/	/	/	/

1			4
*	*	*	*

.APPENDIX 3.

A3) EXAMPLE OF STUDENT TERMINAL SESSION.

This appendix contains an example of a typical student learning session as it appeared on the typewriter log.

N.B. The symbol ← is the invitation for the student to type. All lines beginning with this symbol have been typed by the student.

```
10.30.54← TEACH STATS-02

DISPLAY: STATS-02 WILL START IN APPROX 5 MINS.


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

THIS IS A.C.A.T.S. (ASTON COMPUTER ASSISTED TEACHING SYSTEM)


WHAT IS YOUR NAME?

←MFIDGETT


DO YOU WISH TO START AT FRAME 1?  (TYPE "YES" OR "NO")

←YES


STARTED LESSON STATS-02   BY MFIDGETT   ON 06/09/73 AT 10/33/43.

       (NO AID WILL BE GIVEN IN THIS LESSON)


THIS IS LESSON NUMBER 2 OF THE

STATISTICS COURSE.  YOU SHOULD NOT

ATTEMPT THIS LESSON UNTIL YOU HAVE

SCORED AT LEAST 80% IN 'STATS-01'.

START NOW AT THE FIRST FRAME.


←MATHEMATICAL

GOOD.  YOU SEEM TO HAVE REMEMBERED OUR

DEFINITION OF STATISTICS:-

  'A SET OF MATHEMATICAL TECHNIQUES

     FOR MAKING DATA MORE USEFUL'.
```

```
←TECHNIQUES
YES.


←MATHEMATICAL MODEL
RIGHT.


←RELATIONS
CORRECT.


←ELEMENTS
VERY GOOD.


←OPERATIONS
YES.


←ELEMENTS,RELATIONS,OPERS←ATIONS
EXCELLENT.  KEEP IT UP.


←OPERATIONS
GOOD.


←DATA
O.K.


←OPERATION
CORRECT.
```

```
←DATA
RIGHT.


←A
NO.  THE CORRECT ANSWER IS 'MODELS'.


←*SCORE
SCORE TO DATE  =  92%


←MATHEMATICAL MODEL
RIGHT.


←ELATIONS
YOUR ANSWER IS CORRECT, BUT YOU SHOULD
TYPE THE ENTIRE WORD.  IN FUTURE YOU MAY
BE MARKED WRONG.


←RELATION
NO.  IT HAS NO CORRESPONDING 'MEANING'
OR INTERPRETATION IN THE DATA.


←MATHEMATICAL MODEL
VERY GOOD.  YOU HAVE NOW COMPLETED
LESSON STATS-02.  IF YOU DID NOT SCORE
80% OR MORE, YOU ARE AGAIN ADVISED TO
REPEAT THE LESSON BEFORE GOING ON TO:-
  STATS-03
```

```
DISPLAY : END OF LESSON.

DISPLAY : YOUR SCORE WAS  88%

0.08 :HALTED : OK

0.09 :DELETED

END OF MACRO

10.47.29←
```

.APPENDIX 4.

A4) This appendix contains a description of the nine user macros employed by ACATS. For simplification, all the macros require the same parameter, that is the lesson name.

Each macro has its own error tests and self-explanatory error messages.

A4.1) Macro Name : TEACH

Program entered : TEACHER

Function : To initiate a student terminal session.

```
IF STRING(%A)=(),GO 9F1
IF NOT EXISTS(R%A),GO 9F2
IF STRING(%B)=(),GO 9A
IF NOT EXISTS(R%B),GO 9F3
9A RP DP,OL,OJ
DP 0,%A WILL START IN APPROX 5 MINS.
MZ 8000
LO TEACHERBIN
RP DP,OL,OJ,CT
OL *CR0
OL *LP0
CE !
AS *LP1,!
LF !,*LP
ER !
AS *DA0,R%A(READ,CLEAN)
IF NOT EXISTS(S%A),GO 9B
AS *DA1,S%A(READ)
RP DP,OL,OJ
ON 5
9B RP DP,OL,OJ
EN
RP DP,OL,OJ
IF NOT HAL(WR),GO 9AB
AS *DA0,R%A(WRITE,CLEAN)
RM
IF ABS(PM),GO 9Z1
PT Z1,ALL
LF Z1,*LP
ER Z1
9Z1 IF NOT HAL(OK),GO 9AB
IF STRING(%B)=(),GO 9D
DP 0,LESSON %B WILL COMMENCE SHORTLY.
AL 30,0
RL *CR0
RL *LP0
RL *LP1
RL *DA0
RL *DA1
OL *CR0
OL *LP0
CE !
AS *LP1,!
LF !,*LP
ER !
RP DP,OL,OJ,CT
AS *DA0,R%B(READ,CLEAN)
IF NOT EXISTS(S%B),GO 9C
AS *DA1,S%B(READ)
RP DP,OL,OJ
ON 5
9C RP DP,OL,OJ
EN
```

```
IF NOT HAL(WR),GO 9AB
RP DP,OL,OJ
AS *DA0,R%A(WRITE,CLEAN)
RM
IF ABS(PM),GO 9Z2
PT Z2,ALL
LF Z2,*LP
ER Z2
9Z2 IF NOT HAL(OK),GO 9AB
9D IF CORE,DL
GO 9END
9F1 DP 0,PARAMETER(S) MISSING.
GO 9AB
9F2 DP 0,LESSON %A DOES NOT EXIST.
GO 9AB
9F3 DP 0,LESSON %B DOES NOT EXIST.
9AB DP 0,TEACHING SESSION ABANDONED.
9END UR M
RP AB,CM
EX
****
```

A4.2)	Macro Name :	CRF
	Program entered :	LOADRPF
	Function :	To test a response input-file for errors and create the response lesson-file.

```
IF ABS(%A),GO 9F1
IF NOT EXISTS(CR%A),GO 9F2
9A IF NOT EXISTS(R%A),GO 9B
ER R%A
GO 9A
9B RP DP,OL,OJ
MZ 8000
LO LOADRPBIN
RP DP,OL,OJ,CT
CE R%A(*DA,KWOR 15,RANDOM,KEYPLACE 4,KEYLENGTH 4,INTE 2)
AS *CR0,CR%A
AS *DA0,R%A(WRITE)
RP DP,OL,OJ
DP 0,RESPONSE FILE IS NOW BEING CREATED.
EN
IF CORE,DL
GO 9END
9F1 DP 0,LESSON NAME MISSING.
GO 9END
9F2 DP 0,RESPONSE INPUT-FILE DOES NOT EXIST.
9END UR M
RP AB,CM
EX
****
```

A4.3)	Macro Name :	CSF
	Program entered :	LOADSFF
	Function :	To test a subframe input-file for errors and create the subframe lesson-file.

```
IF ABS(%A),GO 9F1
IF NOT EXISTS(CS%A),GO 9F2
9A IF NOT EXISTS(S%A),GO 9B
ER S%A
GO 9A
9B RP DP,OL,OJ
MZ 8000
LO LOADSFBIN
RP DP,OL,OJ,CT
CE S%A(*DA,KWOR 15,RANDOM,KEYPLACE 4,KEYLENGTH 4)
AS *CR0,CS%A
AS *DA0,S%A(WRITE)
RP DP,OL,OJ
DP 0,SUBFRAME FILE IS NOW BEING CREATED.
EN
IF CORE,DL
GU 9END
9F1 DP 0,LESSON NAME MISSING.
GO 9END
9F2 DP 0,SUBFRAME INPUT-FILE DOES NOT EXIST.
9END UR M
RP AB,CM
EX
****
```

A4.4) Macro Name : PRINTRECS

Program entered : RECS

Function : To print a table of the frame performance records for the specified lesson.

```
IF STRING(%A)=(),GO 9F1
IF NOT EXISTS(R%A),GO 9F2
RP DP,OL,OJ
MZ 4000
LO RECSBIN
RP DP,OL,OJ,CT
AS *DA0,R%A(READ)
CE !
AS *LP0,!
LF !,*LP
ER !
RP DP,OL,OJ
DP 0,PRINTING HAS NOW STARTED.
EN
IF CORE,DL
GO 9END
9F1 DP 0,LESSON NAME MISSING.
GO 9END
9F2 DP 0,RESPONSE FILE DOES NOT EXIST.
9END UR M
RP AB,CM
EX
****
```

A4.5) Macro Name : RESETRECS

Program entered : RESET

Function : To initialise the frame performance records of the specified lesson.

```
IF STRING(%A)=(),GO 9F1
IF NOT EXISTS(R%A),GO 9F2
RP DP,OL,OJ
MZ 4000
LO RESETBIN
RP DP,OL,OJ,CT
AS *DA0,R%A(OVERLAY)
RP DP,OL,OJ
DP 0,RECORDS ARE NOW BEING RESET.
EN
IF CORE,DL
GO 9END
9F1 DP 0,LESSON NAME MISSING.
GO 9END
9F2 DP 0,RESPONSE FILE DOES NOT EXIST.
9END UR M
RP AB,CM
EM
****
```

A4.6) Macro Name : ERF

Program entered : NEWED

Function : To edit the response input-file of the specified lesson. If the edit is completed successfully, the old version of the response input-file will be erased and the new version listed on the line printer.

```
IF STRING(%A)=(),GO 9F1
IF NOT EXISTS(CR%A),GO 9F2
RP DP,OL,OJ
MZ 8000
LO NEWEDBIN
RP DP,OL,OJ,CT
OL *LP0
OL *CR1
AS *CR0,CR%A
AS *CP0,CR%A(+1)
RP DP,OL,OJ
EN
IF CORE,DL
IF NOT DELETED(AB),GO 9A
ER CR%A
GO 9END
9A ER CR%A(-1)
LF CR%A,NU,*LP
GO 9END
9F1 DP 0,LESSON NAME MISSING.
GO 9END
9F2 DP 0,RESPONSE INPUT FILE DOES NOT EXIST.
9END UR M
RP AB,CM
EX
****
```

A4.7) Macro Name : ESF

Program entered : NEWED

Function : To edit the subframe input-file of the specified lesson. If the edit is completed successfully, the old version of the subframe input-file is erased and the new version listed on the line printer.

```
IF STRING(%A)=(),GO 9F1
IF NOT EXISTS(CS%A),GO 9F2
RP DP,OL,OJ
MZ 8000
LO NEWEDBIN
RP DP,OL,OJ,CT
OL *LP0
OL *CR1
AS *CR0,CSZA
AS *CP0,CS%A(+1)
RP DP,OL,OJ
EN
IF CORE,DL
IF NOT DELETED(AB),GO 9A
ER CS%A
GO 9END
9A ER CSZA(-1)
LF CS%A,NO,*LP
GO 9END
9F1 DP 0,LESSON NAME MISSING.
GO 9END
9F2 DP 0,SUBFRAME INPUT FILE DOES NOT EXIST.
9END UR M
RP AB,CM
EX
****
```

A4.8) Macro Name : MAKEREADY

Program entered : none

Function : To retrieve all macros, programs and data files which may be required in the next student or author ACATS session.

```
IF STRING(%A)=(),GO 9F1
RP NONE
RV TEACH,CRF,CSF,ERF,ESF,LIST,PRINTRECS,RESETRECS
RV TEACHERBIN,LOADRPBIN,LOADSFBIN,NEWEDBIN,RECSBIN,RESETBIN
IF EXISTS(CR%A),RV CR%A
IF EXISTS(CS%A),RV CS%A
IF EXISTS(R%A),RV R%A
IF EXISTS(S%A),RV S%A
IF ABS(%B),GO 9END
IF EXISTS(CR%B),RV CR%B
IF EXISTS(CS%B),RV CS%B
IF EXISTS(R%B),RV R%B
IF EXISTS(S%B),RV S%B
IF ABS(%C),GO 9END
IF EXISTS(CR%C),RV CR%C
IF EXISTS(CS%C),RV CS%C
IF EXISTS(R%C),RV R%C
IF EXISTS(S%C),RV S%C
GO 9END
9F1 DP 0,LESSON NAME(S) MISSING.
9END UR M
RP AB,CM
EX
****
```

A4.9) Macro Name : LIST

Program entered : none

Function : To print a numbered listing of the specified lesson's input-files on the line printer.

```
IF STRING(%A)=(),GO 9F1
IF NOT EXISTS(CR%A),GO 9F2
LF CR%A,NU,*LP
IF NOT EXISTS(CS%A),EX
LF CS%A,NU,*LP
EX
9F1 DP 0,LESSON NAME MISSING
EX
9F2 DP 0,RESPONSE INPUT-FILE DOES NOT EXIST
EX
****
```

.APPENDIX 5.

A5) Program Name : TEACHER

Language : PLAN

Core Size : 7½K words

Calling Macro : TEACH

Function : The function of the main control program 'TEACHER' is to control the interaction between the student and the lesson material to be learned. This involves a number of different tasks:-

(i) The acceptance and classification of student responses.

(ii) The evaluation and marking of answers.

(iii) The presentation of feedback to the student.

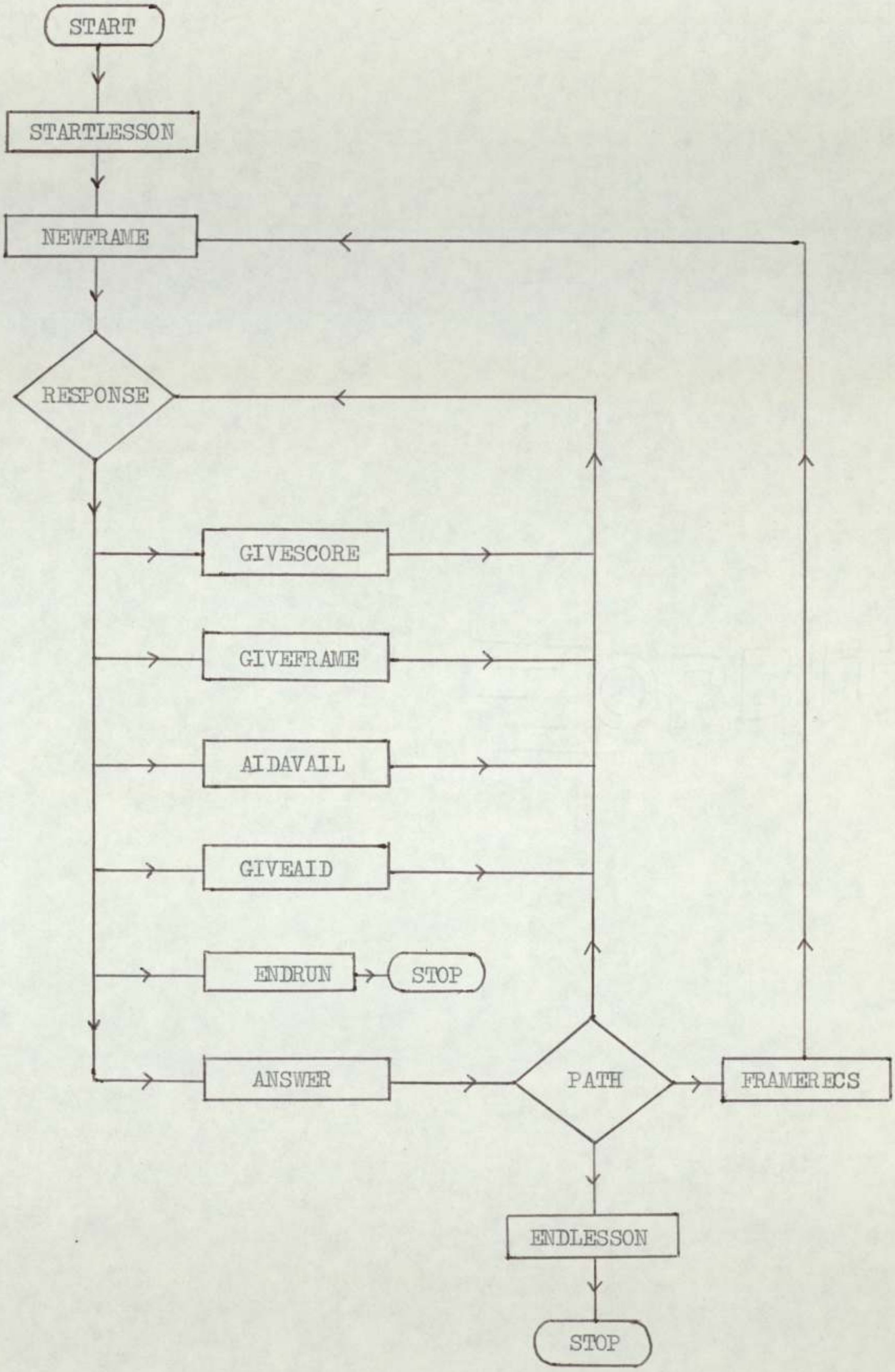
(iv) The recording of student and frame performance data.

(v) The recognition and servicing of student commands.

(vi) The routeing of students through intrinsic programmes.

A5) contd.

Structure : Because of the relatively large size of this program (it comprises 1200 PLAN source statements) it was written in sectional subroutine format. This allowed for easy editing and testing of its component parts. The role of each subroutine can best be illustrated with the use of a flow diagram.


START
STARTLESSON
NEWFRAME
RESPONSE
GIVESCORE
GIVEFRAME
AIDAVAIL
GIVEAID
ENDRUN
STOP
ANSWER
PATH
FRAMERECS
ENDLESSON
STOP

A5) contd.

Switches : The PLAN switch facility employing word 30 of the program was used as follows :-

BIT	SET IF:-
1	Frame performance data to be recorded
3	Trace of lesson to be listed
4	Non-strict mode scoring to be used
5	Subframe Data has been supplied for this lesson
6	Programme is of the "Stored" type
10	Subframe Data has been supplied for this frame
11	Any subframes are found "open"
12	Any unused subframe still exists
13	Lesson is in restart mode
14	After the first attempt at a question
15	After the second attempt at a question
16	Match occurs

A5) contd.

Error Halts :

Event	Cause
DELETED : EXCEPTION CONDITION "R" xxxx	A storage device macro exception condition has occured when reading or writing to the response lesson-file. xxxx is the contents of word 11 which contains the exception condition code.
DELETED : EXCEPTION CONDITION "S" xxxx	As above, when reading or writing to the subframe lesson-file.
DISPLAY : AUTHOR ERROR: DELETED : NO RESPONSE RECORD FOUND IN FRAME *xxx	An attempt has been made to enter a non-existent frame. Author error, probably caused by incorrect 'next frame' pointer.
DISPLAY : AUTHOR ERROR: DELETED : NO MATCH, OR NO ANSWER SETS IN FRAME *xxx	Failure by author to provide a "catch" answer set or any answer sets in the specified frame.

A5) contd.

Event	Cause
DISPLAY : AUTHOR ERROR: DELETED : RUN OFF END OF ANSWER SET IN FRAME *xxx	The predicted answer terminating symbol '@' has been omitted in the specified frame.
DISPLAY : NON-NUMERIC CHARACTER DELETED : IN FRAME *xxx	Non-numeric character found in numeric field of specified frame.

A5.1) Subroutine STARTLESSON.

This subroutine initialises the teaching session as follows:-

- (i) Sets initial conditions.
- (ii) Opens the appropriate lesson-files and reads the file indices, storing them in index buffers in core store.
- (iii) Examines the four lesson parameters, setting switches accordingly.
- (iv) If frame performance data is to be kept, the current frame performance records are transfered to core store buffers.
- (v) Prints introductory headings on the student's console.
- (vi) Converses with the student to determine his name and the starting frame number.

A5.1) contd.

(vii) Determines the acceptibility of this starting frame.

(viii) Prints the author's introductory message.

(ix) Starts student trace if required.

(x) Starts the learning session by calling subroutine NEWFRAME.

A5.2) Subroutine NEWFRAME.

This subroutine is called each time the student enters a new frame. It stores the appropriate bucket and record pointers in each of the file definition areas and reads in the next frame's response and subframe records. If no subframe data has been supplied for this new frame, switch 10 is reset. It also resets a number of counts and switches in preparation for the new frame.

Once all these preparations have been made, subroutine RESPONSE is called to await the student's response.

A5.3) Subroutine RESPONSE.

This routine determines whether the student's response is:-

(i) An answer to the question

(ii) A command

(iii) An illegal command

and passes control to the appropriate subroutine as follows, A5.3) contd.

Response	Subroutine called.
Answer to question	ANSWER
*AID	AIDAVAIL
*GIVE	GIVEAID
*STOP	ENDRUN
*FRAME	GIVEFRAME
*SCORE	GIVESCORE

The response is also added to the student trace if required.

A5.4) Subroutine ANSWER.

Calls subroutine "COMPARE", giving the start addresses of the student's answer and each of the answer sets in turn.

When control returns to this subroutine, switch 16 is tested to see whether a match has occurred between the student's answer and one of the items in the answer set. When a match occurs the author's comment is presented to the student and the student's score is updated.

Finally the key for the next frame is determined and subroutine PATH is called.

A5.5) Subroutine PATH.

This subroutine examines the next frame number and chooses one of three paths as follows:-

(i) Calls subroutine ENDLESSON if the next frame number is zero.

(ii) Calls subroutine RESPONSE if the next frame number equals the current frame number.
i.e:- the student has to make another attempt at this question.

(iii) Otherwise subroutine NEWFRAME is called.

In the first and third cases, subroutine ENDFRAME will also be called to complete the frame's scoring.

A5.6) Subroutine AIDAVAIL.

This routine is called in response to the *AID command. It prints a list of all the subframes which are open, together with the penalty for each.

Subroutine RESPONSE is then called to await the student's next response.

A5.7) Subroutine GIVEAID.

Subroutine GIVEAID is called in response to the *GIVE command. It presents the text of the specified subframe to the student if the subframe is open. The number of penalties incurred by the student will be updated.

Finally control is passed to subroutine RESPONSE.

A5.8) Subroutine GIVESCORE.

This subroutine is called in response to the *SCORE command. It calculates and presents to the student his percentage score to date in the lesson.

Control is then passed to subroutine RESPONSE to await the student's next response.

A5.9) Subroutine GIVEFRAME.

This subroutine is called in response to the *FRAME command. Its sole task is to inform the student of the current frame number.

A5.10) Subroutine ENDRUN.

This subroutine is called in response to the *STOP command, and brings the lesson to an orderly conclusion as follows:-

(i) Calculates the student's final score.

(ii) Prints out various termination messages on the student's console.

(iii) Completes the trace of the lesson, adding the lesson summary.

(iv) Calls subroutine STOREFPDATA to write the updated frame performance records to the response lesson-file.

(v) Closes all files and deletes.

A5.11) Subroutine ENDLESSON.

Subroutine ENDLESSON has a similar function to that of subroutine ENDRUN. It is called when a student completes a lesson without issuing the *STOP command.

A5.12) Subroutine STOREFPDATA.

This subroutine writes the updated frame performance records back to the response lesson-file.

A5.13) Subroutine COMPARE.

Called by subroutine ANSWER.

It accepts as parameters the start addresses of the student's response and of a set of expected answers. It searches this response for each expected answer in turn and sets switch 16 if a match occurs.

Control is then returned to subroutine ANSWER.

A5.14) Subroutine ENDFRAME.

Completes all records for this frame. It also updates the trace and frame performance records if required.

A5.15) Subroutine CDECBIN.

CDECBIN converts a decimal integer to binary, ignoring leading spaces.

A5.16) Subroutine CBINDEC.

CBINDEC converts a binary integer to a four digit decimal integer.

```
#STEER              LIST,OBJECT
#PROGRAM            TEAC
#CMODE              EDS(0,1)
#LOWER
                    RBUFF(512),SBUFF(512)
                    RIBUFF(210),SIBUFF(210)
                    CARD(20),AS(10),A(10)
                    ACC0,ACC1,ACC2,ACC3,ACC4,ACC5,ACC6,ACC7
                    FRAMENUM
                    DEC,BIN,PERCENT
                    PARAM(2)
                    NATTS,MARKS,PENS,SCORE,SUBFSUSED
                    STARTITEM,POINTER
                    LESSON(3),AUTHOR(4),STUDENT(3),PASS(10)
                    FRAMEDATA(1002)
#LOWER
CAIN                3/0,0,80,0/CARD.0
LNOUT               #42,44H
CONT                2/0,0,44,0/LNOUT.3
TRACE               #41,40H
                    20H
TROUT               2/0,0,60,0/TRACE.3
BLANKLN             #41,4H
EMPTY               2/0,0,4,0/BLANKLN.3
BLANKLN2            #42,4H
EMPTY2              2/0,0,4,0/BLANKLN2.3
FDR                 0,0,12HRESPFILENAME,,,,,,,,,,,,,,,,,,,,,,,,,,
FDS                 0,0,12HSUBFFILENAME,,,,,,,,,,,,,,,,,,,,,,,,,,
QUES                4H000?
DASH                4H   -
ZEROFRAME           4H   0
ASTER               4H000*
DOT                 4H000.
ASTERMIN            4H0009
SPAC                4H000 
SEMI                4H000;
COMMA               4H000,
COMAID              4H*AID
COMGIVE             4H*GIV
COMSTOP             4H*STO
COMFRAME            4H*FRA
COMSCORE            4H*SCO
LETN                4H000N
LETY                4H000Y
TERMIN              4H////
SPACS               4H
TOTSCORE            0
TOTMAXSCORE         0
TOTSCLOST           0
BIGPERMIT           999
TEN                 10
NEXTFRAME           4H   1
KEY                 1
```

```
INDKEY          1
HEADKEY         2
FPD1KEY         3
FPD2KEY         4
CHARS           10000
MESS1           #42,28HERROR:  ILLEGAL COMMAND.
CON1            2/0,0,28,0/MESS1.3
MESS2           #42,28HSUBFRAMES AVAILABLE ARE :-
CON2            2/0,0,28,0/MESS2.3
MESS3           #42,44H    NAME          PENALTY (% OF SCORE LOST)
CON3            2/0,0,44,0/MESS3.3
MESS4           #41.40H
CON4            2/0,0,40,0/MESS4.3
MESS5           #42,48HSORRY, NO AID AVAILABLE AT ALL IN THIS FRAME.
CON5            2/0,0,48,0/MESS5.3
MESS6           #42,32HSORRY, ALL AID HAS BEEN GIVEN.
CON6            2/0,0,32,0/MESS6.3
MESS7           #42,44HSORRY, NO AID AVAILABLE YET.  TRY LATER.
CON7            2/0,0,44,0/MESS7.3
MESS8           #42,40HSUBFRAME IDENTIFIER NOT GIVEN.
CON8            2/0,0,40,0/MESS8.3
MESS9           #42,40HTHIS SUBFRAME MAY NOT BE CALLED YET.
CON9            2/0,0,40,0/MESS9.3
MESS10          #42,44HTHIS SUBFRAME HAS ALREADY BEEN USED BY YOU.
CON10           2/0,0,44,0/MESS10.3
MESS11          #42,40HNO SUBFRAME OF THIS NAME CAN BE FOUND.
CON11           2/0,0,41,0/MESS11.3
MESS12          #42,28HSUBFRAME NAME IS MISSING.
CON12           2/0,0,28,0/MESS12.3
MESS13          #42,32HYOU SHOULD BE AT FRAME
CON13           2/0,0,32,0/MESS13.3
MESS14          #42,32HLESSON QUITTED BEFORE END.
CON14           2/0,0,32,0/MESS14.3
MESS15          #42,24HLAST FRAME WAS
CON15           2/0,0,24,0/MESS15.3
MESS16          #42,40HYOU MAY RESTART AT THIS FRAME NEXT TIME.
CON16           2/0,0,40,0/MESS16.3
MESS17          #42,36HPERCENTAGE SCORE TO DATE =      %
CON17           2/0,0,37,0/MESS17.3
MESS18          #42,40HFRAME NUMBER GIVEN IS TOO LARGE.
CON18           2/0,0,41,0/MESS18.3
MESS19          24HLESSON NOT COMPLETED.
CON19           24/MESS19.0
MESS20          16HAUTHOR ERROR:
CON20           16/MESS20.0
MESS21          40HNO MATCH, OR NO ANSWER SETS IN FRAME
CON21           40/MESS21.0
MESS22          40HRUN OFF END OF ANSWER SET IN FRAME
CON22           40/MESS22.0
MESS23          16HEND OF LESSON.
CON23           16/MESS23.0
MESS24          28HLESSON WAS IN RESTART MODE.
CON24           28/MESS24.0
MESS25          24HYOUR SCORE WAS        %
CON25           24/MESS25.0
```

```
MESS26          40HNO RESPONSE RECORD FOUND IN FRAME
CON26           40/MESS26.0
MESS27          #42,40HTHIS IS A.C.A.T.S. (ASTON COMPUTER ASSIS
                24HTED TEACHING SYSTEM)
CON27           2/0,0,64,0/MESS27.3
MESS28          #42,20HWHAT IS YOUR NAME?
CON28           2/0,0,20,0/MESS28.3
MESS29          #41,40H*** **  **  **  **  **  **  **  **  **
                24H **  **  **  **  ***
CON29           2/0,0,64,0/MESS29.3
MESS30          #42,32HDO YOU WISH TO START AT FRAME 1?
                24H  [TYPE "YES" OR "NO"]
CON30           2/0,0,56,0/MESS30.3
MESS31          #42,24HTYPE  YES  OR  NO
CON31           2/0,0,24,0/MESS31.3
MESS32          #42,40HWHICH FRAME DO YOU WISH TO START AT?
CON32           2/0,0,40,0/MESS32.3
MESS33          #42,40HTHERE IS NO SUCH FRAME.
CON33           2/0,0,41,0/MESS33.3
MESS35          #42,36HNO FRAME NUMBER (NUMERIC) FOUND.
CON35           2/0,0,36,0/MESS35.3
MESS36          #42,36H      (RESTARTED AT FRAME     )
CON36           2/0,0,37,0/MESS36.3
MESS39          28HEXCEPTION CONDITION "R"
CON39           28/MESS39.0
MESS40          28HEXCEPTION CONDITION "S"
CON40           28/MESS40.0
MESS41          40HNON-NUMERIC CHARACTER IN FRAME
CON41           40/MESS41.0
MESS42          #42,28HNO SCORE POSSIBLE YET.
CON42           2/0,0,29,0/MESS42.3
MESS43          16H  IN FRAME
CON43           16/MESS43.0
MESS44          #42,44H      (NO AID WILL BE GIVEN IN THIS LESSON.)
CON44           2/0,0,45,0/MESS44.3
MESS45          #42,40HSTARTED LESSON            BY
                36H     ON          AT          .
CON45           2/0,0,76,0/MESS45.3
MESS46          #42,36HTRACE OF LESSON              BY
                28H             ON          .
CON46           2/0,0,64,0/MESS46.3
MESS47          #42,20HFRAME
CON47           2/0,0,20,0/MESS47.3
MESS48          #42,48HSTUDENT         LESSON          SCORE
CON48           2/0,0,48,0/MESS48.3
MESS49          #42,48H                                    %.
CON49           2/0,0,48,0/MESS49.3
MESS50          #42,28H                     %.
CON50           2/0,0,28,0/MESS50.3
MESS51          #42,44H*************************************
CON51           2/0,0,44,0/MESS51.3
#DEFINE         NSUBFS=SBUFF+2
#DEFINE         SUBFAREA=SBUFF+3
#DEFINE         NASS=RBUFF+2
#DEFINE         MAXSCORE=RBUFF+3
#DEFINE         ASAREA=RBUFF+4
#DEFINE         RINDEX=RIBUFF+2
#DEFINE         SINDEX=SIBUFF+2
#DEFINE         FPDATA=FRAMEDATA+2
#DEFINE         FRAMEDATA2=FRAMEDATA+500
```

```
#PROGRAM
#ENTRY           0
      CALL   1   STARTLESSON
EXR   LDX    1   11
      STO    1   MESS39+6
      DELTY      CON39
EXS   LDX    1   11
      STO    1   MESS40+6
      DELTY      CON40
#
#                 SUBROUTINE 'STARTLESSON'
#                 PRINTS INTRODUCTORY MESSAGES.
#                 DETERMINES START/RESTART MODE.
#                 SETS INITIAL CONDITIONS.
#                 OPENS FILES AND READS INDEXES.
#                 BRINGS DOWN FRAME RECORDS.
#                 INITIATES TEACHING SESSION.
#
#CUE             STARTLESSON
#
#                 SET OR RESET INITIAL CONDITIONS.
#
      STOZ       TOTSCORE
      STOZ       TOTMAXSCORE
      STOZ       TOTSCLOST
      LDX    4   '4H   1'
      STO    4   NEXTFRAME
      LDN    4   1
      STO    4   KEY
#
#                 OPEN FILE(S).
#                 READ INDEXES.
#
      SDDEF  0   1,512,FDR,EXR,4,Q
      LDN    4   1
      STO    4   FDR+25
      STO    4   FDS+25
      LDN    4   INDKEY
      STO    4   FDR+26
      STO    4   FDS+26
      SDRD   0   RIBUFF
      TEST   7   5
      BZE    7   HEADS          [ NO SUBFRAME FILE ??
      SDDEF  1   1,512,FDS,EXS,4,Q
      SDRD   1   SIBUFF
#
#                 READ HEADING RECORD FROM BUCKET 2.
#                 PREPARE BOOLEANS FROM PARAMETERS.
#
HEADS LDN    4   2
      STO    4   FDR+25
      LDN    4   HEADKEY
      STO    4   FDR+26
      SDRD   0   RBUFF
      LDX    45  RBUFF+2
      LDX    6   RBUFF+4
      STO    45  LESSON
      STO    6   LESSON+2       [ STORE LESSON NAME.
      STO    45  MESS49+5
```

```
         STO    6  MESS49+7
         LDN    2  RBUFF+12
         LDN    3  AUTHOR
         MVCH   2  16              [ STORE AUTHOR'S NAME.
#
#                  SET BOOLEANS.
#
         LDX    1  RBUFF+22        [ SCORING.
         BNZ    1  *+4
         ON     7  4
         LDX    1  RBUFF+23        [ FRAME PERFORMANCE DATA.
         BZE    1  *+4
         ON     7  1
         LDX    1  RBUFF+24        [ TRACING.
         BZE    1  *+4
         ON     7  3
         LDX    1  RBUFF+25        [ STORED PROGRAMME.
         BZE    1  *+4
         ON     7  6
#
#                  IF REQUIRED, BRING DOWN FRAME RECORDS
#                  FROM BUCKETS 3 & 4.
         TEST   7  1
         BZE    7  HDS             [ NOT REQUIRED ??
         LDN    4  4
         STO    4  FDR+25
         LDN    4  FPD2KEY
         STO    4  FDR+26
         SDRD   0  FRAMEDATA2
         LDN    4  3
         STO    4  FDR+25
         LDN    4  FPD1KEY
         STO    4  FDR+26
         SDRD   0  FRAMEDATA
#
#                  PRINT HEADINGS. (ACATS) (NAME?) (RESTART?)
#
HDS      PERI   0  EMPTY
         SUSBY     LP0
         PERI   0  EMPTY
         SUSBY     LP0
         PERI   0  CON29
         SUSBY     LP0
         PERI   0  CON27           [ THIS IS ACATS.
         SUSBY     LP0
         PERI   0  EMPTY
         SUSBY     LP0
         PERI   0  CON28           [ NAME ??
         SUSBY     LP0
         PERI   0  CAIN            [ READ NAME.
         SUSBY     CR0
         LDX    45 CARD
         LDX    6  CARD+2
         STO    45 STUDENT
         STO    6  STUDENT+2
         STO    45 MESS49+1
         STO    6  MESS49+3
         TEST   7  6               [ STORED PROGRAMME ??
         BNZ    7  START
```

```
       PERI  0  EMPTY
       SUSBY    LP0
       PERI  0  CON30           [ START AT BEGINNING ??
       SUSBY    LP0
#
#               START OR RESTART MODE?
#
MODE   PERI  0  CAIN
       LDN   4  0
       LDCT  1  40
       SUSBY    CR0
CHSL   LDCH  4  CARD(1)
       BXE   4  LETN,RMODE      [ RESTART ??
       BXE   4  LETY,START      [ START ??
       BCHX  1  CHSL
       PERI  0  CON31           [ TYPE YES OR NO.
       SUSBY    LP0
       BRN      MODE
#
#               RESTART MODE.
#
RMODE  PERI  0  CON32           [ WHICH FRAME ??
       SUSBY    LP0
#
#               READ STARTING FRAME.
#
       PERI  0  CAIN
       LDN   4  0
       LDN   5  0
       LDCT  1  40
       SUSBY    CR0
FNUM1  LDCH  4  CARD(1)
       BXL   4  TEN,FNUM2       [ FIRST NUMERIC ??
       BCHX  1  FNUM1
       PERI  0  CON35           [ NO FRAME NUMBER FOUND.
       SUSBY    LP0
       BRN      RMODE
FNUM2  MPY   5  TEN
       LDX   5  6
       ADX   5  4
       BCHX  1  *+2
       BRN      S13SL
       LDCH  4  CARD(1)
       BXL   4  TEN,FNUM2       [ ANOTHER NUMERIC??
#
#               ACC. 5. CONTAINS STARTING FRAME.
#               SET 13, AND CHANGE NEXTFRAME AND KEY.
#
S13SL  ON    7  13
       LDX   2  5
       LDX   4  RINDEX(2)       [ GET LBN.
       BNZ   4  *+4
       PERI  0  CON33           [ NO SUCH FRAME.
       SUSBY    LP0
       BRN      RMODE
       LDN   4  201
       BXL   5  4,*+4
       PERI  0  CON18
       SUSBY    LP0
       BRN      RMODE
```

```
      STO   5  KEY              [ CHANGE KEY.
      CALL  1  CBINDEC
      LDX   5  DEC
      STO   5  NEXTFRAME        [ CHANGE NEXTFRAME.
#
#               START HEADING.
#
START LDN   2  LESSON
      LDN   3  MESS45+5
      MVCH  2  12
      LDN   2  STUDENT
      LDN   3  MESS45+9
      MVCH  2  12
      GIVE  4  1                [ GET DATE.
      STO   45 MESS45+13
      GIVE  4  2
      STO   45 MESS45+16        [ GET TIME.
      PERI  0  EMPTY
      SUSBY    LP0
      PERI  0  CON45            [ "STARTED...
      SUSBY    LP0
#
#
#               PRINT OTHER POSSIBLE HEADINGS.
#               E.G:-  "NO AID"  &  "RESTARTED".
#
      TEST  7  13
      BZE   7  STRT
      LDX   5  NEXTFRAME
      STO   5  MESS36+8
      PERI  0  CON36            [ RESTARTED AT FRAME...
      SUSBY    LP0
STRT  TEST  7  5
      BNZ   7  *+3
      PERI  0  CON44            [NO AID IN LESSON.
      SUSBY    LP0
#
#               PRINT INTRODUCTORY MESSAGE.
#
      PERI  0  EMPTY
      LDN   1  0
PLINE LDX   4  RBUFF+32(1)
      BXE   4  TERMIN,TRAC      [ END OF INTR. MESS.
      LDN   2  RBUFF+32
      ADX   2  1
      LDN   3  LNOUT+1
      MVCH  2  40
      SUSBY    LP0
      PERI  0  CONT
      SUSBY    LP0
      ADN   1  10
      BRN      PLINE
#
#               START TRACING IF REQUIRED.
#
TRAC  TEST  7  3
      BZE   7  SUBS             [ NO TRACING REQUIRED ??
      LDX   45 LESSON
      STO   45 MESS46+5
```

```
        LDX   4  LESSON+2
        STO   4  MESS46+7
        LDX   45 STUDENT
        STO   45 MESS46+9
        LDX   4  STUDENT+2
        STO   4  MESS46+11
        GIVE  4  1                 [ GET DATE.
        STO   45 MESS46+14
        PERI  1  CON46
        SUSBY    LP1
        PERI  1  EMPTY
        SUSBY    LP1
#
#                  START AT FIRST FRAME.
#
SUBS    CALL  1  NEWFRAME
#
#                  SUBROUTINE 'NEWFRAME'
#                  STORES THE APPROPRIATE BUCKET AND RECORD POINTERS
#                  IN EACH OF THE FILE DEFINITION AREAS.
#                  READS IN NEXT RESPONSE AND SUBFRAME RECORDS.
#                  RESETS SWITCH 10 IF NO SUBFRAME RECORD.
#                  RESETS MARKS,PENALTIES, AND NO. OF ATTEMPTS.
#
#CUE               NEWFRAME
        LDX   4  NEXTFRAME
        STO   4  FRAMENUM
        PERI  0  EMPTY2
        SUSBY    LP0
        OFF   7  14               [ FIRST ATTEMPT SWITCH, OFF.
        OFF   7  15               [ SECOND ATTEMPT SWITCH, OFF.
        STOZ     NATTS
        STOZ     PENS
        STOZ     MARKS
        STOZ     SUBFSUSED
        LDX   2  KEY
        LDX   4  RINDEX(2)
        STO   4  FDR+25
        LDX   5  SINDEX(2)
        STO   5  FDS+25
        LDN   6  KEY
        STO   6  FDR+26
        STO   6  FDS+26
        BZE   4  F1NF                        [ NO RESPONSE RECORD ??
        SDRD  0  RBUFF
        TEST  7  5
        BZE   7  R10              [ NO SUBFRAME FILE ??
        BZE   5  R10                         [ NO SUBFRAME RECORD ??
        ON    7  10
        SDRD  1  SBUFF
        BRN      L1NF
R10     OFF   7  10
#
#                  TRACING, IF REQUIRED.
#
L1NF    TEST  7  3
        BZE   7  L2NF             [ NO TRACING ??
        PERI  1  EMPTY
        LDX   4  FRAMENUM
        STO   4  MESS47+3
```

```
       SUSBY   LP1
       PERI  1 CON47
       SUSBY   LP1
L2NF   CALL  1 RESPONSE
#               DELETE -- NO RESPONSE RECORD --
F1NF   DISTY   CON20                   [ DISP 'AUTHOR ERROR'
       LDX   4 FRAMENUM
       STO   4 MESS26+9
       DELTY   CON26            [ DISP 'NO RESPONSE RECORD IN FRAME *XXX'
#
#               SUBROUTINE 'RESPONSE'
#               DECIDES WHETHER A STUDENT'S RESPONSE IS :-
#               A) AN ANSWER TO THE QUESTION.
#               B) A COMMAND.
#               C) AN ILLEGAL COMMAND.
#.              TAKES ACTION ACCORDINGLY.
#
#CUE           RESPONSE
RESP   PERI  0 CAIN                    [ READ STUDENT'S RESPONSE.
       SUSBY   CR0
       TEST  7 3
       BZE   7 T7
       LDX   2 CARD
       LDX   3 TRACE+5
       MVCH  2 40
       PERI  1 TROUT
T7     LDX   6 CARD
       LDN   5 0
       SLL   56 6
       LDX   6 CARD
T1     BXE   5 ASTER,T2                [ COMMAND ??
       CALL  1 ANSWER
T2     BXU   6 COMAID,T3               [ NOT *AID COMMAND ??
       CALL  1 AIDAVAIL
T3     BXU   6 COMGIVE,T4              [ NOT *GIVE COMMAND ??
       CALL  1 GIVEAID
T4     BXU   6 COMSTOP,T5              [ NOT *STOP COMMAND ??
       CALL  1 ENDRUN
T5     BXU   6 COMFRAME,T6            [ NOT *FRAME COMMAND ??
       CALL  1 GIVEFRAME
T6     BXU   6 COMSCORE,ILL            [ NOT *SCORE COMMAND ??
       CALL  1 GIVESCORE
       LDX   4 TOTMAXSCORE
       BZE   4 *+3
       PERI  0 CON17
       SUSBY   LP0
       PERI  0 EMPTY
       SUSBY   LP0
       BRN     RESP
ILL    PERI  0 CON1                    [ DISP 'ERROR: ILLEGAL COMMAND'
       SUSBY   LP0
       PERI  0 EMPTY
       SUSBY   LP0
       BRN     RESP
```

```
#
#                 SUBROUTINE 'ANSWER'
#               . CALLS "COMPARE", GIVING THE STARTING ADDRESSES
#       .         OF THE STUDENT'S RESPONSE, AND EACH ANSWER
#                 SET IN TURN.
#                 TESTS FOR A MATCH.
#               . PRINTS THE AUTHOR'S COMMENT WHEN A MATCH OCCURS.
#                 DOES THE SCORING.
#
#CUE              ANSWER
        LDN   4   1
        ADS   4   NATTS                 [ INCREMENT NO. OF ATTEMPTS.
        LDN   3   ASAREA                [ ACC. 3. IS POINTER.
        LDX   6   NASS
#                  COMPARE RESPONSE WITH ITEMS OF NEXT A.S.
NXTA    BZE   6   F1A
        STO   3   ACC3
        LDX   2   3
        LDN   3   AS
        MVCH  2   40
        LDX   2   CARD
        LDN   3   R
        MVCH  2   40
        LDX   3   ACC3
        CALL  1   COMPARE
T1A     TEST  7   16
        BNZ   7   T2A                    [ MATCH ??
DECA    SBN   6   1                      [ NASS=NASS-1
        LDX   3   10(3)               [ GET POINTER.
        ADX   3   ASAREA
        BRN       NXTA
#                  DO THE MARKING.
T2A     TEST  7   14
        BNZ   7   T3A                 [ NOT FIAST ATTEMPT ??
        ON    7   14
        LDX   4   13(3)
        ADS   4   MARKS               [ ADD SCORE 1 TO MARKS.
        BRN       COMM
T3A     TEST  7   15
        BNZ   7   T4A                 [ NOT SECOND ATTEMPT ??
        ON    7   15
        LDX   4   14(3)               [ ADD SCORE 2 TO MARKS.
        ADS   4   MARKS
        BRN       COMM
#                  TEST MODE OF MARKING.
T4A     TEST  7   4
        BZE   7   COMM                [ STRICT MODE ??
        LDX   4   14(3)
        ADS   4   MARKS               [ ADD SCORE 2 TO MARKS.
#                  PRINT AUTHOR'S COMMENT.
COMM    LDX   4   16(3)               [ LOAD NLINES.
        STO   3   POINTER             [ SAVE OLD POINTER.
T5A     BZE   4   FINA                [ NO MORE COMMENT TEXT ??
        STO   1   ACC1
        LDN   1   17
        ADX   1   3
        LDN   2   LNOUT+1
        MVCH  1   40                   [ MOVE LINE OF TEXT.
        PERI  0   CONT                 [ PRINT LINE.
        LDX   1   ACC1
```

```
      ADN   3  10                          [ INCREMENT POINTER.
      SBN   4  1                           [ DECREMENT NLINES.
      SUSBY    LP0
      BRN      T5A
#               GET KEY OF NEXT FRAME.
FINA  LDX   3  POINTER                     [ LOAD OLD POINTER.
      LDX   5  15(3)
      STO   5  KEY                         [ KEY=BINARY
      CALL  1  CBINDEC
      LDX   4  DEC
      STO   4  NEXTFRAME                   [ NEXTFRAME=DECIMAL
      CALL  1  PATH
#               NO MATCH SUCCESSFUL ?
F1A   LDX   4  FRAMENUM
      STO   4  MESS21+9
      DISTY    CON20                       [ DISP ' AUTHOR ERROR '
      DELTY    CON21                       [ DISP 'NO MATCH OR NO A.S.'
#
#               SUBROUTINE 'PATH'
#               EXAMINES "NEXTFRAME", AND CHOOSES ONE OF THREE PATHS
#               1) CALLS "ENDLESSON"  IF NEXTFRAME = ZERO.
#               2) CALLS "RESPONSE"  IF NEXTFRAME = FRAMENUM.
#               3) CALLS "NEWFRAME"  OTHERWISE.
#               SUBROUTINE "ENDFRAME" WILL BE CALLED, IF NECESSARY
#               TO DO THE FRAME'S SCORING.
#
#CUE           PATH
      LDX   4  NEXTFRAME
      BXU   4  FRAMENUM,DIFF               [ DIFFERENT FRAME?
      CALL  1  RESPONSE
DIFF  CALL  1  ENDFRAME
      LDX   4  NEXTFRAME
      BXE   4  ZEROFRAME,ENDL                     [ END OF LESSON ??
      CALL  1  NEWFRAME
ENDL  CALL  1  ENDLESSON
#
#               SUBROUTINE 'AIDAVAIL'
#               FINDS ALL THOSE SUBFRAMES WHICH ARE OPEN,
#               PRINTING THE IDENTIFIER AND PENALTY FOR EACH.
#
#CUE           AIDAVAIL
      TEST  7  10
      BZE   7  M1AA                        [ NO SUBFRAME RECORD ??
      OFF   7  11
      OFF   7  12
      LDX   5  NSUBFS                      [ ACC 5. CONTAINS NSQBFS.
      BZE   5  M1AA                        [ NO SUBFRAMES ??
      LDN   3  SUBFAREA                    [ ACC 3. CONTAINS POINTER.
NEXTS BZE   5  T11                         [ NO MORE SUBFRAMES ??
      LDX   6  2(3)                        [ LOAD OPEN PERMIT.
      BXE   6  BIGPERMIT,DECR              [ ALREADY USED ??
      ON    7  12
      LDX   4  NATTS                       [ LOAD NO. OF ATTEMPTS.
      BXL   4  6,DECR                      [ SUBFRAME NOT OPEN ??
      TEST  7  11
      BNZ   7  IDPEN                       [ NOT FIRST OPEN SUBFRAME ??
      PERI  0  CON2
      SUSBY    LP0
      PERI  0  CON3                        [ DISP HEADING.
      SUSBY    LP0
```

```
#                PRINT IDENTIFIER AND PENALTY.
IDPEN LDX    2  0(3)
      STO    2  MESS4+2
      LDX    2  1(3)
      STO    2  MESS4+3
      STO    5  NSUBFS
      LDX    5  4(3)
      LDN    7  100
      MPY    5  7
      DVR    5  MAXSCORE
      LDX    5  6
      CALL   1  CBINDEC
      LDX    5  NSUBFS
      LDX    2  DEC
      STO    2  MESS4+5
      PERI   0  CON4                    [ DISP 'ID. AND PEN.'
      SUSBY     LP0
      ON     7  11
DECR  SBN    5  1                       [ DECREMENT NSUBFS.
      LDX    3  3(3)                   [ UPDATE POINTER.
      ADN    3  SUBFAREA
      BRN       NEXTS
M1AA  PERI   0  CON5                    [ DISP 'NO AID AT ALL'
      SUSBY     LP0
      BRN       EMAA
T11   TEST   7  11
      BNZ    7  EMAA                    [ SOME WERE FOUND ??
T12   TEST   7  12
      BNZ    7  M2AA                    [ SOME FOUND, BUT NOT OPEN YET ??
M3AA  PERI   0  CON6                    [ DISP 'ALL AID GIVEN'
      SUSBY     LP0
      BRN       EMAA
M2AA  PERI   0  CON7                    [ DISP 'NO AID YET'
      SUSBY     LP0
EMAA  PERI   0  EMPTY
      SUSBY     LP0
      CALL   1  RESPONSE                [ OVER TO THE STUDENT.
#
#               SUBROUTINE 'GIVEAID'
#               SEARCHES FOR A SUBFRAME IDENTIFIER WHICH MATCHES
#               THE *GIVE PARAMETER.
#               PRINTS OUT THE SUBFRAME'S TEXT, IF IT IS OPEN.
#
#CUE            GIVEAID
      TEST   7  10
      BZE    7  M1GA                    [ NO SUBFRAME RECORD ??
      LDX    6  NSUBFS                  [ ACC 6. CONTAINS NSUBFS.
      BZE    6  M1GA                    [ NO SUBFRAMES ??
      LDCT   2  35
      LDN    4  0
#               LOOK FOR FIRST SPACE CHARACTER.
CHGA  LDCH   4  CARD+1(2)
      BXE    4  SPAC,SPGA               [ SPACE ??
      BCHX   2  CHGA
      BRN       DISGA
#               LOOK FOR FIRST NON-SPACE CHARACTER.
SPGA  LDCH   4  CARD+1(2)
      BXU    4  SPAC,PARGA                  [ NOT SPACE ??
      BCHX   2  SPGA
      BRN       DISGA
```

```
#                PUT THE NEXT 8 CHARS. IN PARAM & PARAM+1.
PARGA LDCT  3  8
PAR   LDCH  4  CARD+1(2)
      DCH   4  PARAM(3)
      BCHX  2  *+2
      BRN      DISGA
      BCHX  3  PAR
      LDN   3  SUBFAREA                  [ ACC 3. IS POINTER.
#                EXAMINE THE NEXT SUBFRAME.
NXTGA BZE   6  M2GA                      [ NO MORE SUBFRAMES ??
      LDX   45 PARAM
      BXU   45 0(3),DECGA                [ NO MATCH ??
      LDX   4  NATTS
      BXL   4  2(3),TBIGP                [ NOT OPEN ??
      LDN   4  1
      ADS   4  SUBFSUSED
      LDX   4  BIGPERMIT
      STO   4  2(3)                      [ CLOSE SUBFRAME.
      LDX   4  4(3)
      ADS   4  PENS                      [ INCREMENT PENALTIES.
      LDX   4  5(3)                      [ ACC 4. CONTAINS NLINES.
#                PRINT TEXT OF SUBFRAME.
LIST  BZE   4  EXGA                      [ NO MORE LINES OF TEXT ??
      LDN   1  6
      ADX   1  3
      LDN   2  LNOUT+1
      MVCH  1  40                        [ MOVE LINE OF TEXT.
      PERI  0  CONT                      [ PRINT LINE.
      ADN   3  10                        [ INCREMENT POINTER.
      SBN   4  1                         [ DECREMENT NLINES.
      SUSBY    LP0
      BRN      LIST
#                DECREMENT NSUBFS & LOAD NEW POINTER.
DECGA SBN   6  1
      LDX   3  3(3)
      ADN   3  SUBFAREA
      BRN      NXTGA
#                HAS IT BEEN USED BEFORE?
TBIGP LDX   4  2(3)                      [ LOAD OPEN PERMIT.
      BXE   4  BIGPERMIT,M3GA            [ ALREADY CALLED ??
#                NOT OPEN YET?
M4GA  PERI  0  CON9                      [ DISP 'NOT AV. YET'
      BRN      EXGA
#                ALREADY USED?
M3GA  PERI  0  CON10                     [ DISP 'ALREADY USED'
      BRN      EXGA
#                DOES NOT EXIST
M2GA  PERI  0  CON11                      [ DISP 'NO SUCH SUBFRAME'
      BRN      EXGA
#                NO AID.
M1GA  PERI  0. CON5                       [ DISP 'NO AID AT ALL'
      BRN      EXGA
#                NO PARAMETER.
DISGA PERI  0  CON12                     [DISP 'PARAMETER OF *GIVE NOT FOUND'
#                OVER TO THE STUDENT.
EXGA  SUSBY    LP0
      CALL  1  RESPONSE
```

```
#
#                 SUBROUTINE 'GIVESCORE'
#                 CALCULATES THE LATEST PERCENTAGE SCORE.
#                 CONVERTS IT TO DECIMAL.
#                 STORES IT IN LOCATION "PERCENT".
#CUE             GIVESCORE
      STO   1  ACC1
      STO   4  ACC4
      STO   5  ACC5
      STO   6  ACC6
      LDX   4  TOTMAXSCORE
      BZE   4  MIGS                    [ NO SCORE ??
      LDX   4  TOTSCORE
      LDN   6  100
      MPY   4  6
      DVR   4  TOTMAXSCORE
      CALL  1  CBINDEC
      LDX   5  DEC
      STO   5  PERCENT
      STO   5  MESS17+8
      BRN      EXGS
MIGS  PERI  0  CON48                        [ DISP 'NO SCORE YET'
      SUSBY    LP0
EXGS  LDX   4  ACC4
      LDX   5  ACC5
      LDX   6  ACC6
      LDX   1  ACC1
      PERI  0  EMPTY
      SUSBY    LP0
      EXIT  1  0
#
#                 SUBROUTINE 'GIVEFRAME'
#                 PRINTS OUT A MESSAGE TELLING THE STUDENT
#                 WHICH FRAME IS BEING USED AT THAT MOMENT.
#
#CUE             GIVEFRAME
      LDX   4  FRAMENUM
      STO   4  MESS13+7
      PERI  0  CON13
      SUSBY    LP0
      PERI  0  EMPTY
      SUSBY    LP0
      CALL  1  RESPONSE
#
#                 SUBROUTINE 'ENDRUN'
#                 PRINTS OUT VARIOUS MESSAGES.
#                 CLOSES THE DIRECT ACCESS FILES.
#                 SUSPENDS THE PROGRAM.
#
#CUE             ENDRUN
      PERI  0  EMPTY
      SUSBY    LP0
      LDX   4  FRAMENUM
      STO   4  MESS15+5
      PERI  0  CON14                      [ DISP 'LESSON QUITTED BEFORE END'
      CALL  1  GIVESCORE
      LDX   5  TOTMAXSCORE
      BZE   5  *+5
      LDX   4  PERCENT
      STO   4  MESS25+4
```

```
        SUSBY    LP0
        DISTY    CON25                          [ DISP '% SCORE =  '
        TEST  7  6
        BNZ   7  *+4                  [ STORED PROGRAMME ??
        PERI  0  CON15                          [DISP 'LAST FRAME WAS *XXX'
        SUSBY    LP0
        PERI  0  CON16                        [ DISP 'MAY RESTART AT THIS FRAME'
        SUSBY    LP0
#
#                FINISH TRACE OF LESSON.
#
        TEST  7  3
        BZE   7  L3ER
        PERI  1  EMPTY2
        SUSBY    LP1
        PERI  1  CON51
        SUSBY    LP1
        PERI  1  CON48
        LDX   4  PERCENT
        BNZ   4  *+2
        LDX   4  DASH
        STO   4  MESS49+9
        TEST  7  13
        BZE   7  *+5                  [ START MODE ??
        LDX   5  SPACS
        STO   5  MESS49+11
        LDX   4  '4H(RS)'
        BRN      *+2
        LDX   4  '4H (S)'
        STO   4  MESS49+11
        SUSBY    LP1
        PERI  1  CON49
        SUSBY    LP1
        TEST  7  13
        BZE   7  *+2                  [ NOT RESTART MODE ??
        DISTY    CON24                [ DISP 'LESSON WAS IN RESTART MODE'
        DISTY    CON19                [ DISP 'LESSON NOT COMPLETED'
 L3ER   PERI  0  EMPTY
        SUSBY    LP0
        SDEND 0  CLOSE
        SUSWT    2HER
        SDDEF 0  1,512,FDR,EXA,5,0
        TEST  7  1
        BZE   7  L2ER
        CALL  2  STOREFPDATA
 L2ER   SDEND 0  CLOSE
        TEST  7  5
        BZE   7  L1ER
        SDEND 1  CLOSE
 L1ER   SUSWT    2HOK
#
#                SUBROUTINE 'ENDLESSON'.
#                PRINTS OUT VARIOUS MESSAGES.
#                CLOSES THE DIRECT ACCESS FILES.
#                SUSPENDS THE PROGRAM.
#
#CUE             ENDLESSON
        PERI  0  EMPTY
        SUSBY    LP0
        DISTY    CON23                          [ DISP 'END OF LESSON'
```

```
      TEST  7  13
      BZE   7  L1EL              [ NOT RESTART MODE ??
      DISTY    CON24              [ DISP 'LESSON WAS IN RESTART MODE'
L1EL  CALL  1  GIVESCORE
      LDX   4  TOTMAXSCORE
      BZE   4  *+4
      LDX   4  PERCENT
      STO   4  MESS25+4
      DISTY    CON25              [ DISP 'YOUR SCORE WAS      %'
#
#               FINISH TRACE OF LESSON, IF REQUIRED.
      TEST  7  3
      BZE   7  L3EL
      PERI  1  EMPTY2
      SUSBY    LP1
      PERI  1  CON51
      SUSBY    LP1
      PERI  1  CON48
      LDX   4  PERCENT
      BNZ   4  *+2
      LDX   4  DASH
      STO   4  MESS49+9
      TEST  7  13
      BZE   7  *+5         [ START ??
      LDX   5  SPACS
      STO   5  MESS49+11
      LDX   4  '4H (R)'
      STO   4  MESS49+11
      SUSBY    LP1
      PERI  1  CON49
      SUSBY    LP1
L3EL  SDEND 0  CLOSE
      SUSWT    2HWR
      SDDEF 0  1,512,FDR,EXR,5,Q
      TEST  7  1
      BZE   7  L2EL
      CALL  2  STOREFPDATA .
L2EL  SDEND 0  CLOSE
      PERI  0  EMPTY
      SUSBY    LP0
      TEST  7  5
      BZE   7  FIN
      SDEND 1  CLOSE
FIN   SUSWT    2HOK
#
#               SUBROUTINE 'STOREFPDATA'
#               REPLACES FPD IN BUCKETS 3 & 4.
#
#CUE            STOREFPDATA
#
#               DELETE EXISTING RECORDS.
#
      LDN   4  3
      STO   4  FDR+25
      LDN   4  FPD1KEY
      STO   4  FDR+26
      SDDEL 0
      LDN   4  4
      STO   4  FDR+25
      LDN   4  FPD2KEY
      STO   4  FDR+26
      SDDEL 0
```

```
#
#               WRITE UPDATED RECORDS.
#
      LDN   5  502
      STO   5  FRAMEDATA
      LDN   4  3
      STO   4  FRAMEDATA+1
      STO   4  FDR+25
      LDN   4  FPD1KEY
      STO   4  FDR+26
      SDWRI 0  FRAMEDATA
      STO   5  FRAMEDATA2
      LDN   4  4
      STO   4  FRAMEDATA2+1
      STO   4  FDR+25
      LDN   4  FPD2KEY
      STO   4  FDR+26
      SDWRI 0  FRAMEDATA2
      EXIT  2  0
#
#               SUBROUTINE 'COMPARE'
#               SEARCHES RESPONSE FOR EXPECTED ANSWERS.
#               SWITCH 16 IS SET IF MATCH, RESET OTHERWISE.
#
#CUE           COMPARE
      STO   1  ACC1
      LDCT  1  40                [ MODIFIER OF A.S.
      STO   1  STARTITEM          [ START OF ITEM OF A.S.
#               TEST FOR A "CATCH" ANSWER SET.
      LDN   4  0
      LDCH  4  AS(1)
      BXE   4  ASTERMIN,MATCH    [ CATCH ANSWER SET ??
#               START AT FIRST CHAR. OF RESPONSE.
SERCH LDCT  2  40                [ MODIFIER OF RESPONSE.
      LDCH  4  AS(1)
      LDCH  5  R(2)
#               COMPARE PAIR OF CHARACTERS.
COMP  BXE   4  5,TEND              [ EQUAL ??
      LDX   1  STARTITEM           [ BACK TO START OF ITEM OF A.S.
      LDCH  4  AS(1)
      BRN      SPACE
#               TEST IF FINISHED. ·
TEND  BCHX  1  *+2
      BRN      DISPC                [ END OF A.S. ??
      BCHX  2  *+2
      BRN      FIEND                [ LAST CHAR. OF RESPONSE ??
      LDCH  4  AS(1)
      LDCH  5  R(2)
      BXU   4  ASTERMIN,COMP        [ NOT END ??
      BXE   5  SPAC,MATCH           [
      BXE   5  DOT,MATCH            [ END ??
      BXE   5  QUES,MATCH
      BXE . 5  COMMA,MATCH          [
      BRN      NEWIT                [ GET NEXT ITEM OF A.S.
#               LOOK FOR NEXT SPACE IN RESPONSE.
SPACE LDCH  5  R(2)
      BCHX  2  *+2
      BRN      FIEND                [ LAST CHAR. OF RESPONSE ??
      BXU   5  SPAC,SPACE           [ NOT SPACE ??
      LDCH  5  R(2)                 [ LOAD FIRST CHAR. AFTER SPACE.
      BRN      COMP
```

```
#               FIND END OF THIS ITEM OF A.S.
FIEND LDCH  4  AS(1)
      BXE   4  ASTERMIN,NEWIT          [ END OF ITEM ??
      BCHX  1  FIEND
      BRN      DISPC                   [ END OF A.S. ??
#               ANY MORE ITEMS IN A.S. ?
NEWIT BCHX  1  *+2
      BRN      DISPC
      STO   1  STARTITEM
      LDCH  4  AS(1)
      BXE   4  SPAC,NOMAT                [ NO MORE ITEMS ??
      BRN      SERCH                     [ START AGAIN WITH NEXT ITEM.
DISPC DISTY    CON20                     [ DISP 'AUTHOR ERROR'
      LDX   4  FRAMENUM
      STO   4  MESS22+9
      DELTY    CON22                     [ DISP 'RUN OFF END OF A.S.
MATCH LDX   1  ACC1
      ON    7  16
      EXIT  1  0
NOMAT LDX   1  ACC1
      OFF   7  16
      EXIT  1  0
#
#               SUBROUTINE 'ENDFRAME'
#               CALCULATES THE OVERALL SCORE FOR THE LAST FRAME.
#               INCREMENTS THE TOTAL SCORE & TOTSCLOST.
#
#CUE            ENDFRAME
      STO   1  ACC1
      STO   2  ACC2
      STO   3  ACC3
      STO   4  ACC4
      STO   5  ACC5
      STO   6  ACC6
      STO   7  ACC7
      LDX   4  MAXSCORE
      ADS   4  TOTMAXSCORE  [ INCREMENT TOTAL MAXIMUM SCORE.
      LDX   5  NATTS
      CALL  1  CBINDEC
      LDX   5  DEC
      STO   5  MESS50+1
      LDX   5  MARKS
      CALL  1  CBINDEC
      LDX   5  DEC
      STO   5  MESS50+2
      LDX   5  PENS
      CALL  1  CBINDEC
      LDX   5  DEC
      STO   5  MESS50+3
      LDX   4  MARKS
      SBX   4  PENS
      STO   4  SCORE
      BNG   4  NVSC
      BZE   4  NVSC          [ OVERALL SCORE -VE OR ZERO ??
#
#               CALCULATE % SCORE FOR FRAME.
#
      LDX   4  MAXSCORE
      BNZ   4  *+4
      LDX   4  DASH
```

```
      STO   4  MESS50+5
      BRN      PVSC
      LDX   4  SCORE
      LDN   6  100
      MPY   4  6
      DVR   4  MAXSCORE
      CALL  1  CBINDEC
      LDX   4  DEC
      STO   4  MESS50+5         [ GET % SCORE FOR TRACE.
PVSC  LDX   4  SCORE
      ADS   4  TOTSCORE         [ INCREMENT TOTAL SCORE.
      LDX   4  PENS
      ADS   4  TOTSCLOST        [ INCREMENT TOTAL SCORE LOST.
      TEST  7  3
      BZE   7  EXEF
      PERI  1  CON50
      BRN      EXEF
NVSC  STOZ     SCORE
      LDX   4  MARKS
      ADS   4  TOTSCLOST
      LDX   4  MAXSCORE
      BNZ   4  *+4
      LDX   4  DASH
      STO   4  MESS50+5
      BRN      *+3
      LDX   4  '4H    0'
      STO   4  MESS50+5
      TEST  7  3
      BZE   7  EXEF
      PERI  1  CON50
#
#              RECORD FRAME PERFORMANCE DATA
#              IF REQUIRED.
#
EXEF  TEST  7  1
      BZE   7  ENDEF
      LDX   2  '4/FRAMENUM.0'
      CALL  1  CDECBIN
      LDX   2  BIN
      LDN   4  5
      MPY   2  4                [ ACC 3. CONTAINS 5*KEY
      LDN   6  1
      SBX   3  6
      LDX   4  SUBFSUSED
      ADS   4  FPDATA(3)
      SBX   3  6
      LDX   4  SCORE
      ADS   4  FPDATA(3)
      SBX   3  6
      LDX   4  MAXSCORE
      ADS   4  FPDATA(3)
      SBX   3  6
      LDX   4  NATTS
      ADS   4  FPDATA(3)
      SBX   3  6
      ADS   6  FPDATA(3)
```

```
#
#                 FRAME RECORD COMPLETE.
#
ENDEF LDX    1   ACC1
      LDX    2   ACC2
      LDX    3   ACC3
      LDX    4   ACC4
      LDX    5   ACC5
      LDX    6   ACC6
      LDX    7   ACC7
      EXIT   1   0
#
#                 SUBROUTINE 'CDECBIN'
#                 EXPECTS ACC 2 = NO. OF CHARS./START ADDRESS
#                 CONVERTS DECIMAL TO BINARY.
#                 STORES THE ANSWER IN LOC. "BIN".
#                 LEADING SPACES ARE IGNORED.
#
#CUE             CDECBIN
      STO    4   ACC4
      STO    6   ACC6
      STO    7   ACC7
      LDN    4   0
      LDN    6   0
      LDN    7   0
SPDB  LDCH   4   0(2)
      BXU    4   SPAC,CONDB
      BCHX   2   SPDB
      BRN        EXDB
CONDB CDB    6   0(2)
      BCC        *+5
      LDX    4   FRAMENUM
      STO    4   MESS43+3
      DISTY      CON41                    [ DISP 'NON-NUMERIC'
      DELTY      CON43                    [ DISP 'IN FRAME *XXX'
      BCHX   2   CONDB
EXDB  STO    7   BIN
      LDX    4   ACC4
      LDX    6   ACC6
      LDX    7   ACC7
      EXIT   1   0
```

```
#
#                    SUBROUTINE 'CBINDEC'
#                    EXPECTS ACC 5 = BINARY NUMBER
#                    CONVERTS THIS BINARY NUMBER INTO
#                    A 4 DIGIT DECIMAL NUMBER.
#                    STORES IT IN LOCATION "DEC".
#
#CUE               CBINDEC
        STO    2   ACC2
        STO    5   ACC5
        STO    6   ACC6
        STO    7   ACC7
        LDCT   2   3
        LDN    6   0
        DVD    5   CHARS
        ADN    6   1
        LDN    7   0
        MODE       1
        CBD    6   DEC(2)
        BCHX   2   *-1
        MODE       0
        CBD    6   DEC(2)
        LDX    2   ACC2
        LDX    5   ACC5
        LDX    6   ACC6
        LDX    7   ACC7
        EXIT   1   0
#END
#FINISH
ENDPROG
****
```

.APPENDIX 6.

A6) Program Name : LOADRPF

Language : PLAN

Core Size : 2K words.

Calling Macro : CRF

Function : To test a response input-file for errors. If no errors are found it will convert this card file to direct access format, creating the response lesson-file. Each record on the response lesson-file consists of the response data for one frame in the lesson.

This program also creates a file index in bucket 1, stores the Response Title form data in bucket 2 and initialises the frame performance records in buckets 3 and 4 of the response lesson-file.

A6) contd.

Error Halts :

Event	Cause
DISPLAY : FIRST WORD IN FRAME IS NOT A FRAME NUMBER DISPLAY : LAST FRAME WAS *xxx DELETED : JOB ABANDONED	As stated. Probable error in or omission of number of answer sets or number of lines fields in specified frame.
DISPLAY : NON-NUMERIC CHARACTER IN FRAME *xxx DELETED : JOB ABANDONED	Non-numeric character found in numeric field of specified frame.
DISPLAY : FRAME NUMBER TOO LARGE IN FRAME *xxx DELETED : JOB ABANDONED	As stated.
DISPLAY : FRAME *xxx TOO LARGE DELETED : JOB ABANDONED	Specified frame too large to be held in one bucket.
DISPLAY : EXCEPTION CONDITION DELETED : JOB ABANDONED	Program error. Storage device macro exception condition.

A6.1) Subroutine CDECBIN.

CDECBIN converts a decimal integer to binary, ignoring leading spaces.

A6.2) Subroutine CBINDEC.

CBINDEC converts a binary integer to a four digit decimal integer.

```
#STEER          LIST,OBJECT
#PROGRAM        LDRP
#CMODE          EDS(0)
#LOWER
                BUFF(510),RINDEX(510),CARD(20)
                LENGTH,NLINES
                NASS,POINT
                LASTFRNO,DEC
                ACC4,ACC6,ACC7
#LOWER
CAIN            3/0,0,80,0/CARD.0
FD0             0,0,12HRESPFILENAME,,,,,,,,,,,,,,,,,,,,,,,,,,,,
MESS1           40HFIRST WORD IN FRAME IS NOT A FRAME NO.
COUNT1          40/MESS1.0
MESS2           36HNON-NUMERIC CHARACTER IN FRAME
COUNT2          36/MESS2.0
MESS3           40HFRAME NUMBER TOO LARGE IN FRAME
COUNT3          40/MESS3.0
MESS4           24HFRAME         TOO LARGE.
COUNT4          24/MESS4.0
MESS5           20HEXCEPTION CONDITION.
COUNT5          20/MESS5.0
MESS7           20HLAST FRAME WAS
COUNT7          20/MESS7.0
MESS8           40HLESSON-FILE SUCCESSFULLY CREATED.
COUNT8          40/MESS8.0
MESS9           20HJOB ABANDONED.
COUNT9          20/MESS9.0
CHARS           10000
LBN             5
BLOCKL          511
LIMIT           200
TOTLENGTH       0
TERMIN          4H////
ASTER           4H000*
SPAC            4H000
#DEFINE         KEY=BUFF+1
#DEFINE         MAXSCORE=BUFF+3
#DEFINE         ASAREA=BUFF+4
#DEFINE         INDEX=RINDEX+2
#PROGRAM
#ENTRY          0
      SDDEF 0   1,512,FD0,EX0,6,Q    [ OPEN DA FILES
#
#               LOAD HEADINGS AND PARAMETERS INTO
#               BUCKET 2, USING INDEX AREA.
#
      LDN    1  300
      STO    1  RINDEX        [ STORE LENGTH IN BUFFER
      LDN    1  2
      STO    1  RINDEX+1      [ STORE KEY IN BUFFER
```

```
       STO    1  FD0+25
       STO    1  FD0+26         [ SET BUCKET AND RECORD POINTERS
       PERI   0  CAIN
       SUSBY     CR0
       LDN    1  CARD
       LDN    2  INDEX
       MVCH   1  40             [ PUT LESSON NAME IN BUFFER
       PERI   0  CAIN
       SUSBY     CR0
       LDN    1  CARD
       LDN    2  INDEX+10
       MVCH   1  40             [ PUT AUTHOR'S NAME IN BUFFER
#
#                START LOADING PARAMETERS
#
       LDCT   1  10
       STOZ      INDEX+20(1)
       BUX    1  *-1            [ ZEROISE PARAMETERS
       LDCT   1  4
PARS   PERI   0  CAIN
       SUSBY     CR0
       LDN    4  0
       LDX    5  CARD
       SLL    45 6
       STO    4  INDEX+20(1)
       BUX    1  PARS           [ LOAD PARAMETER CHARACTERS
#
#                LOAD INTRODUCTORY MESSAGE
#
       LDCT   1  20             [ LIMIT OF 20 LINES
       LDN    4  0
INTR   PERI   0  CAIN
       SUSBY     CR0
       LDX    5  CARD
       LDN    2  CARD
       LDN    3  INDEX+30
       ADX    3  4
       MVCH.  2  40
       ADN    4  10
       BXE    5  TERMIN,FINT    [ END OF INTR. MESS. ??
       BUX    1  INTR
       DEL       2HIM
#
#                STORE RECORD IN D.A. FILE, BUCKET 2
#
FINT   SDWRI 0   RINDEX
#
#                ZEROISE INDEX BUFFER AND PERFORMANCE
#                DATA RECORD, BUCKETS 3 & 4.
#
       LDCT   1  500
       STOZ      INDEX(1)
       BUX    1  *-1
       LDN    1  502
```

```
      STO   1  RINDEX        [ LENGTH
      LDN   1  3
      STO   1  RINDEX+1      [ KEY
      STO   1  FD0+25
      STO   1  FD0+26
      SDWRI 0  RINDEX
      LDN   1  4
      STO   1  RINDEX+1
      STO   1  FD0+25
      STO   1  FD0+26
      SDWRI 0  RINDEX
      LDX   1  LBN
      STO   1  FD0+25
#
#                NEW RECORD
#                READ RECORD HEADER
#
START PERI  0  CAIN
      SUSBY    CR0
      LDX   4  CARD
      BXE   4  TERMIN,ENDE             [ END OF FILE/LESSON ??
      STOZ     3
      SLL   34 6
      BXU   3  ASTER,FAIL1              [ CHECK IS FRAME NUMBER ??
      LDX   7  CARD
      STO   7  LASTFRNO
      CALL  3  CDECBIN
      LDX   2  '3/CARD+0.1'
      STO   7  KEY                     [ STORE FRAME KEY
      BXGE  7  LIMIT,FAIL3              [ FRAME NUMBER TOO LARGE ??
      CALL  3  CDECBIN
      LDX   2  '2/CARD+1.0'
      STO   7  NASS
      STO   7  BUFF+2                  [ STORE NO. OF ANSWER SETS
      CALL  3  CDECBIN
      LDX   2  '2/CARD+1.2'
      STO   7  MAXSCORE               [STORE MAXIMUM SCORE.
      LDN   6  17
      LDN   7  4
      MPA   6  NASS                    [ LENGTH=3+17*NASS
      STO   7  LENGTH
      BXGE  7  BLOCKL,FAIL4          [RECORD TOO LARGE ??
      STOZ     1                       [ POINTER=0
#
#                NEW ANSWER SET
#                READ ANSWER SET HEADER
#
TEST1 LDX   4  NASS
      BZE   4  RCEND                   [ END OF RECORD/FRAME ??
      STO   1  POINT
      PERI  0  CAIN
      SUSBY    CR0
```

```
        LDX   4  CARD
        STO   4  ASAREA+11(1)           [ STORE AS IDENTIFIER WORD 1
        LDX   4  CARD+1
        STO   4  ASAREA+12(1)           [ STORE AS IDENTIFIER WORD 2
        CALL  3  CDECBIN
        LDX   2  '3/CARD+2.1'
        STO   7  ASAREA+15(1)           [ STORE NEXT FRAME'S KEY
        CALL  3  CDECBIN
        LDX   2  '2/CARD+3.0'
        STO   7  ASAREA+13(1)           [ STORE SCORE 1
        CALL  3  CDECBIN
        LDX   2  '2/CARD+3.2'
        STO   7  ASAREA+14(1)           [ STORE SCORE 2
        CALL  3  CDECBIN
        LDX   2  '1/CARD+4.0'
        STO   7  ASAREA+16(1)           [ STORE NO. OF LINES IN COMMENT
        STO   7  NLINES
        LDN   6  10
        MPY   6  NLINES                 [ LENGTH=LENGTH+10*NLINES
        ADS   7  LENGTH
        LDX   7  LENGTH
        BXGE  7  BLOCKL,FAIL4       [RECORD TOO LARGE ??
        PERI  0  CAIN
        SUSBY    CR0
        LDN   2  CARD
        LDN   3  ASAREA                 [ STORE ANSWER SET
        ADX   3  1
        MVCH  2  40
#
#                NEW LINE OF TEXT
#                READ LINE OF COMMENT
#
TEST2   LDX   4  NLINES
        BZE   4  DECR                    [ END OF COMMENT ??
        PERI  0  CAIN
        SBN   4  1
        STO   4  NLINES
        SUSBY    CR0
        LDN   2  CARD
        LDN   3  ASAREA+17              [ STORE LINE IN BUFFER
        ADX   3  1
        MVCH  2  40
        ADN   1  10                     [ INCREMENT POINTER
        BRN      TEST2
#
#                END OF ANSWER SET
#
DECR    LDN   4  1
        SBS   4  NASS                   [ DECREASE NASS COUNT
        LDX   4  1
        ADN   4  17
```

```
      LDX   2 POINT
      STO   4 ASAREA+10(2)          [ STORE POINTER
      STO   4 1
      BRN     TEST1
#
#             END OF RECORD/FRAME
#
RCEND LDX   4 LENGTH
      STO   4 BUFF                    [ STORE RECORD LENGTH
      BXGE  4 BLOCKL,FAIL4         [RECORD TOO LARGE ??
      BRN     LODE
#
#             FAILURE....FIRST WORD IN FRAME IS NOT A FRAME NUMBER
#
FAIL1 DISTY   COUNT1
      LDX   1 LASTFRNO
      STO   1 MESS7+4
      DISTY   COUNT7.
      DELTY   COUNT9
#
#             FAILURE....NON-NUMERIC CHARACTER BEING CONVERTED
#
FAIL  LDX   1 LASTFRNO
      STO   1 MESS2+8
      DISTY   COUNT2
      DELTY   COUNT9
#
#            FAILURE....FRAME NUMBER TOO LARGE
#
FAIL3 LDX   1 LASTFRNO
      STO   1 MESS3+9
      DISTY   COUNT3
      DELTY   COUNT9
#
#            FAILURE....RECORD TOO LARGE
#
FAIL4 LDX   1 LASTFRNO
      STO   1 MESS4+2
      DISTY   COUNT4
      DELTY   COUNT9
#
#             WRITE RECORD TO DA FILE
#
LODE  LDX   1 BUFF                  [ INCREASE TOTAL LENGTH
      ADS   1 TOTLENGTH
      LDX   1 TOTLENGTH
      BXL   1 BLOCKL,IND            [ NEED NEW BUCKET ??
      LDX   1 BUFF
      STO   1 TOTLENGTH
      LDX   1 LBN
      ADN   1 1                     [INCREMENT LOGICAL BUCKET NUMBER
      STO   1 LBN
```

```
       STO    1  FD0+25
IND    LDX    1  '0/BUFF+1.0'
       STO    1  FD0+26
       SDWRI  0  BUFF
       LDX    1  LBN
       LDX    2  BUFF+1
       STO    1  INDEX(2)
       BRN       START
#
#                  EXCEPTION CONDITION
#
EX0    DISTY     COUNT5
       DELTY     COUNT9
#
#                  CLOSE DA FILE
#                  PRINT INDEX AREA
#                  DELETE PROGRAM
#
ENDE   LDN    4  210
       STO    4  RINDEX
       LDN    5  1
       STO    5  RINDEX+1
       STO    5  FD0+25
       STO    5  FD0+26
       SDWRI  0  RINDEX
       SDEND  0  CLOSE
       DISTY     COUNT8
       DEL       2HOK
#
#                  SUBROUTINE 'CDECBIN'
#                  CONVERTS A GIVEN NUMBER OF CHARACTERS FROM
#                  A GIVEN STARTING ADDRESS TO BINARY, STORING
#                  THE ANSWER IN ACC. 7.
#
#CUE              CDECBIN
       STO    4  ACC4
       STO    6  ACC6
       STO    7  ACC7
       LDN    4  0
       LDN    6  0
       LDN    7  0
       OBEY      0(3)
SPACE  LDCH   4  0(2)
       BXU    4  SPAC,CONV
       BCHX   2  SPACE
       BRN       EX
CONV   CDB    6  0(2)
       BCS       FAIL
       BCHX   2  *-2
EX     LDX    4  ACC4
       LDX    6  ACC6
       EXIT   3  1
```

```
#
#                 SUBROUTINE 'CBINDEC'
#                 CONVERTS A BINARY INTEGER TO
#                 A 4 DIGIT DECIMAL NUMBER,
#                 STORING THE ANSWER IN 'DEC'.
#
#CUE              CBINDEC
        LDCT   2  3
        OBEY      0(1)
        LDN    6  0
        DVD    5  CHARS
        ADN    6  1
        LDN    7  0
        MODE      1
        CBD    6  DEC(2)
        BCHX   2  *-1
        MODE      0
        CBD    6  DEC(2)
        EXIT   1  1
#END
#FINISH
ENDPROG
****
```

.APPENDIX 7.

A7) Program Name : LOADSFF

Language : PLAN

Core Size : 2K words

Calling Macro : CSF

Function : To test a subframe input-file for errors. If no errors are found it will convert this card file to direct access format, creating the subframe lesson-file. Each record on the subframe lesson-file consists of the subframe data for one frame in the lesson.

This program also creates a file index in bucket 1.

Error Halts : As for program LOADRPF (see appendix 6)

A7.1) Subroutine CDECBIN.

CDECBIN converts a decimal integer to binary, ignoring leading spaces.

A7.2) Subroutine CBINDEC.

CBINDEC converts a binary integer to a four digit decimal integer.

```
#STEER          LIST,OBJECT
#PROGRAM        LDSF
#CMODE          EDS(0)
#LOWER
                BUFF(510),SINDEX(210),CARD(20)
                LENGTH,NLINES,NSUBFS,POINT
                LASTFRNO,DEC
                ACC4,ACC6,ACC7
#LOWER
CAIN            3/0,0,80,0/CARD.0
FD0             0,0,12HSUBFFILENAME,,,,,,,,,,,,,,,,,,,,,,,,,,,,
MESS1           40HFIRST WORD IN FRAME IS NOT A FRAME NO.
COUNT1          40/MESS1.0
MESS2           36HNON-NUMERIC CHARACTER IN FRAME
COUNT2          36/MESS2.0
MESS3           40HFRAME NUMBER TOO LARGE IN FRAME
COUNT3          40/MESS3.0
MESS4           24HFRAME          TOO LARGE.
COUNT4          24/MESS4.0
MESS5           24HEXCEPTION CONDITION
COUNT5          24/MESS5.0
MESS7           20HLAST FRAME WAS
COUNT7          20/MESS7.0
MESS8           40HLESSON-FILE CREATED SUCCESSFULLY.
COUNT8          40/MESS8.0
MESS9           20HJOB ABANDONED.
COUNT9          20/MESS9.0
CHARS           10000
LBN             2
LIMIT           200
BLOCKL          511
TOTLENGTH       0
TERMIN          4H////
SPAC            4H000
ASTER           4H000*
#DEFINE         KEY=BUFF+1
#DEFINE         SUBFAREA=BUFF+3
#DEFINE         INDEX=SINDEX+2
#PROGRAM
#ENTRY          0
      SDDEF 0   1,512,FD0,EX1,6,Q    [ OPEN DA FILE
      LDX   1   LBN
      STO   1   FD0+25
      LDCT  1   201
      STOZ      INDEX(1)             [ ZEROISE INDEX AREA
      BUX   1   *-1
#
#                NEW RECORD
#                READ RECORD HEADER
#
START PERI  0   CAIN
      SUSBY     CR0
```

```
       LDX    4  CARD
       BXE    4  THRMIN,ENDE                [END OF FILE/LESSON ??
       STOZ      3
       SLL    34 6
       BXU    3  ASTER,FAIL1                 [ CHECK IT IS FRAME NO. ??
       LDX    7  CARD
        STO    7  LASTFRNO
       CALL   3  CDECBIN
       LDX    2  '3/CARD+0.1'
       STO    7  KEY                        [ STORE FRAME KEY
       BXGE   7  LIMIT,FAIL3                [ FRAME NUMBER TOO LARGE ??
       CALL   3  CDECBIN
       LDX    2  '2/CARD+1.0'
       STO    7  NSUBFS
       STO    7  BUFF+2                     [ STORE NO. OF SUBFRAMES
       LDN    6  6
       LDN    7  3
       MPA    6  NSUBFS                     [ LENGTH=3+6*NSUBFS
       STO    7  LENGTH
       BXGE   7  BLOCKL,FAIL4          [RECORD TOO LARGE ??
       STOZ      1                          [ POINTER=0
#
#                 NEW SUBFRAME
#                 READ SUBFRAME HEADER
#
TEST1  LDX    4  NSUBFS
       BXE    4  RCEND                      [ END OF RECORD/FRAME ??
       STO    1  POINT
       PERI   0  CAIN
       SUSBY     CR0
       LDX    4  CARD
       STO    4  SUBFAREA(1)                [ STORE SF IDENTIFIER WORD 1
       LDX    4  CARD+1
       STO    4  SUBFAREA+1(1)              [ STORE SF IDENTIFIER WORD 2
       CALL   3  CDECBIN
       LDX    2  '1/CARD+2.0'
       STO    7  SUBFAREA+2(1)              [ STORE OPEN PERMIT
       CALL   3  CDECBIN
       LDX    2  '2/CARD+2.1'
       STO    7  SUBFAREA+4(1)               [ STORE PENALTY
       CALL   3  CDECBIN
       LDX    2  '1/CARD+2.3'
       STO    7  SUBFAREA+5(1)              [ STORE NO. OF LINES IN SUBFRAME
       STO    7  NLINES
       LDN    6  10
       MPY    6  NLINES                   . [ LENGTH=LENGTH+10*NLINES
       ADS    7  LENGTH
       LDX    7  LENGTH
       BXGE   7  BLOCKL,FAIL4          [RECORD TOO LARGE ??
```

```
#
#                    NEW LINE OF TEXT
#                    READ LINE OF SUBFRAME TEXT
#
TEST2 LDX   4  NLINES
      BZE   4  DECR                        [ END OF SUBFRAME TEXT ??
      PERI  0  CAIN
      SBN   4  1
      STO   4  NLINES
      SUSBY    CR0
      LDN   2  CARD
      LDN   3  SUBFAREA+6                  [ STORE LINE IN BUFFER
      ADX   3  1
      MVCH  2  40
      ADN   1  10                          [ INCREMENT POINTER
      BRN      TEST2
#
#                    END OF SUBFRAME
#
DECR  LDX   4  1
      SBS   4  NSUBFS                      [ DECREASE NSUBFS COUNT
      LDX   4  1
      ADN   4  6
      LDX   2  POINT
      STO   4  SUBFAREA+3(2)               [ STORE POINTER
      STO   4  1
      BRN      TEST1
#
#                    END OF RECORD/FRAME
#
RCEND LDX   4  LENGTH
      STO   4  BUFF                        [ STORE LENGTH OF RECORD
      BXGE  4  BLOCKL,FAIL4         [ RECORD TOO LARGE ??
      BRN      LODE
#
#                    FAILURE....FIRST WORD IN FRAME IS NOT A FRAME NUMBER
#
FAIL1 DISTY    COUNT1
      LDX   1  LASTFRNO
      STO   1  MESS7+4
      DISTY    COUNT7
      DELTY    COUNT9
#
#                    FAILURE....NON-NUMERIC CHARACTER BEING CONVERTED
#
FAIL  LDX   1  LASTFRNO
      STO   1  MESS2+3
      DISTY    COUNT2
      DELTY    COUNT9
```

```
#
#                 FAILURE....FRAME NUMBER TOO LARGE
#
FAIL3 LDX   1  LASTFRNO
      STO   1  MESS3+9
      DISTY    COUNT3
      DELTY    COUNT9
#
#                 FAILURE....RECORD TOO LARGE.
#
FAIL4 LDX   1  LASTFRNO
      STO   1  MESS4+2
      DISTY    COUNT4
      DELTY    COUNT9
#
#                 WRITE RECORD TO DA FILE
#
LODE  LDX   1  BUFF                       [ INCREASE TOTAL LENGTH
      ADS   1  TOTLENGTH
      LDX   1  TOTLENGTH
      BXL   1  BLOCKL,IND                 [ NEED NEW BUCKET ??
      LDX   1  BUFF
      STO   1  TOTLENGTH
      LDX   1  LBN
      ADN   1  1                          [ INCREMENT LOGICAL BUCKET NO.
      STO   1  LBN
      STO   1  FD0+25
IND   LDX   1  '0/BUFF+1.0'
      STO   1  FD0+26
      SDWRI 0  BUFF
      LDN   2  BUFF+1
      LDX   1  LBN
      STO   1  INDEX(2)
      BRN      START
#
#                 EXCEPTION CONDITION
#
EX1   DISTY    COUNT5
      DELTY    COUNT9
#
#                 CLOSE DA FILE
#                 PRINT INDEX AREA
#                 DELETE PROGRAM
#
ENDE  LDN   4  210
      STO   4  SINDEX
      LDN   5  1
      STO   5  SINDEX+1
      STO   5  FD0+25
      STO   5  FD0+26
      SDWRI 0  SINDEX
      SDEND 0  CLOSE
      DISTY    COUNT8
      DEL      2HOK
```

```
#
#                     SUBROUTINE 'CDECBIN'
#                     CONVERTS A GIVEN NUMBER OF CHARACTERS FROM
#                     A GIVEN STARTING ADDRESS TO BINARY, STORING
#                     THE ANSWER IN ACC. 7.
#
#CUE              CDECBIN
      STO    4   ACC4
      STO    6   ACC6
      LDN    4   0
      LDN    6   0
      LDN    7   0
      OBEY       0(3)
SPACE LDCH   4   0(2)
      BXU    4   SPAC,CONV
      BCHX   2   SPACE
      BRN        EX
CONV  CDB    6   0(2)
      BCS        FAIL
      BCHX   2   *-2
EX    LDX    4   ACC4
      LDX    6   ACC6
      EXIT   3   1
#
#                    SUBROUTINE 'CBINDEC'
#                    CONVERTS A BINARY INTEGER TO
#                    A 4 DIGIT DECIMAL NUMBER,
#                    STORING THE ANSWER IN 'DEC'.
#
#CUE              CBINDEC
      LDCT   2   3
      OBEY       0(1)
      LDN    6   0
      DVD    5   CHARS
      ADN    6   1
      LDN    7   0
      MODE       1
      CBD    6   DEC(2)
      BCHX   2   *-1
      MODE       0
      CBD    6   DEC(2)
      EXIT   1   1
#END
#FINISH
ENDPROG
****
```

.APPENDIX 8.

A8) Program Name : RECS

Language :	PLAN
Core Size :	2K words
Calling Macro :	PRINTRECS
Function :	Prints a table of the frame performance records for a particular lesson using the accumulated frame performance data stored in the lesson's response lesson-file.

Error Halts :

Event	Cause
DISPLAY : EXCEPTION CONDITION xxxx DELETED : EX	Storage device macro exception condition has occurred. xxxx is the contents of word 11 which contains the exception condition code.

A8.1) <u>Subroutine CBINDEC.</u>

CBINDEC converts a binary integer to a four digit decimal integer.

```
#STEER          LIST,OBJECT
#PROGRAM        RECS
#CMODE          EDS(0)
#LOWER
                IBUFF(210),RECBUFF(1002)
                DEC,TIMES,TOTMAXSCORE
                ACC0,ACC1,ACC2,ACC3,ACC4,ACC5,ACC6,ACC7

#LOWER
INDKEY          1
HEADKEY         2
FPD1KEY         3
FPD2KEY         4
DASH            4H   -
SPACS           4H
LIMIT           201
CHARS           10000
FD0             0,0,12HLESSONNAME  ,,,,,,,,,,,,,,,,,,,,,,,,,,,
LNOUT           #41,44H
                20H
CONT            2/0,0,15,0/LNOUT.3
BLANK           #42,40H
EMPTY           2/0,0,40,0/BLANK.3
MESS1           #42,40H                          PRINT OF FRAME PERFO
                16HRMANCE DATA.
CON1            2/0,0,56,0/MESS1.3
MESS2           #41,44H                          LESSON          ON
                12H
CON2            2/0,0,54,0/MESS2.3
MESS3           #42,40H                   TIMES       AV. NO. OF
                23H  AV. NO. OF           AVERAGE
CON3            2/0,0,59,0/MESS3.3
MESS4           #41,40H    FRAME.         USED.          ATTEMPTS.
                23H SUBFRAMES USED.   % SCORE.
CON4            2/0,0,60,0/MESS4.3
MESS5           32HEXCEPTION CONDITION
CON5            32/MESS5.0                               ***********
MESS6           #41,44H
CON6            2/0,0,44,0/MESS6.3
#DEFINE         INDHDR=IBUFF+2
#DEFINE         FPDATA=RECBUFF+2
#PROGRAM
#ENTRY          0
      SDDEF 0   1,512,FD0,EM0,4,0
#
#                PRINT HEADINGS AND READ INDEX
#                AND RECORD BUCKETS.
#
      PERI   0  EMPTY
      SUSBY     LP0
      PERI   0  CON1
      LDN    1  1
```

```
      STO    1  FD0+25
      LDN    1  INDKEY
      STO    1  FD0+26
      SDRD   0  IBUFF          [ READ INDEX BUFFER.
      LDN    2  2
      STO    2  FD0+25
      LDN    2  HEADKEY
      STO    2  FD0+26
      SDRD   0  RECBUFF
      LDN    2  RECBUFF+2
      LDN    3  MESS2+8
      MVCH   2  12            [ GET "LESSON".
      GIVE   4  1
      STO    45 MESS2+12       [ GET DATE.
      SUSBY     LP0
      PERI   0  CON2           [ 'LESSON AND DATE'
      LDN    4  4
      STO    4  FD0+25
      LDN    4  FPD2KEY
      STO    4  FD0+26
      SDRD   0  RECBUFF+500   [ READ REC BUCKET 4.
      SUSBY     LP0
      PERI   0  CON6
      SUSBY     LP0
      PERI   0  EMPTY
      LDN    3  3
      STO    3  FD0+25
      LDN    3  FPD1KEY
      STO    3  FD0+26
      SDRD   0  RECBUFF       [ READ REC. BUCKET 3.
      SUSBY     LP0
      PERI   0  CON3
      SUSBY     LP0           [ PRINT HEADINGS.
      PERI   0  CON4
      SUSBY     LP0
      LDN    1  1             [ ACC 1=KEY
NKEY  LDX    4  INDEX(1)
      BZE    4  SKIP          [ NO SUCH FRAME.
      LDN    2  BLANK+1
      LDN    3  LNOUT+8
      MVCH   2  36            [ CLEAR OUTPUT AREA.
      STO    1  5
      CALL   2  CBINDEC
      STO    5  LNOUT+2       [ STORE FRAME NUMBER.
      LDN    2  5
      MPY    2  1             [ ACC 3=5*KEY.
      LDX    5  FPDATA-5(3)
      BZE    5  NEVER         [ NEVER BEEN USED ??
      CALL   2  CBINDEC
      STO    5  LNOUT+5       [ STORE TIMES USED.
      STO    5  TIMES
```

```
      LDX   5  FPDATA-4(3)
      STOZ     4
      DVR   4  FPDATA-5(3)
      CALL  2  CBINDEC
      STO   5  LNOUT+8        [ STORE AV. ATTEMPTS.
      LDX   5  FPDATA-1(3)
      STOZ     4
      DVR   4  FPDATA-5(3)
      CALL  2  CBINDEC
      STO   5  LNOUT+12       [ STORE AV. SUBFS. USED.
      LDX   4  FPDATA-3(3)
      BZE   4  NOSCO           [ TOTMAXSCORE=ZERO ??
      STO   4  TOTMAXSCORE
      LDX   4  FPDATA-2(3)
      LDN   6  100
      MPY   4  6
      DVR   4  TOTMAXSCORE
      CALL  2  CBINDEC
      STO   5  LNOUT+16       [ STORE AV. % SCORE.
      PERI  0  CONT
      SUSBY    LP0
SKIP  ADN   1  1
      BXL   1  LIMIT,NKEY
      DEL      2HOK
NEVER LDX   4  DASH
      STO   4  LNOUT+5
      PERI  0  CONT
      SUSBY    LP0
      ADN   1  1
      BXL   1  LIMIT,NKEY
      DEL      2HOK
NOSCO LDX   4  DASH
      STO   4  LNOUT+16
      PERI  0  CONT
      SUSBY    LP0
      ADN   1  1
      BXL   1  LIMIT,NKEY
      DEL      2HOK
#
#               EXCEPTION CONDITION.
#
EX0   LDX   4  11
      STO   4  MESS5+5
      DISTY    CON5
      DEL      2HEX
```

```
#
#                  SUBROUTINE 'CBINDEC'
#                  EXPECTS ACC 5=BINARY INTEGER.
#                  LEAVES FOUR DIGIT DECIMAL NUMBER IN ACC 5
#                  USES ACC 2 AS LINK ADDRESS.
#
#CUE              CBINDEC
        STO   2   ACC2
        STO   6   ACC6
        STO   7   ACC7
        LDCT  2   3
        LDN   6   0
        DVD   5   CHARS
        ADN   6   1
        LDN   7   0
        MODE      1
        CBD   6   DEC(2)
        BCHX  2   *-1
        MODE      0
        CBD   6   DEC(2)
        LDX   5   DEC
        LDX   2   ACC2
        LDX   6   ACC6
        LDX   7   ACC7
        EXIT  2   0
#END
#FINISH
ENDPROG
****
```

.APPENDIX 9.

A9) Program Name : RESET

Language : PLAN

Core Size : $1\frac{1}{2}$ K words

Calling Macro : RESETRECS

Function : To initialise a lesson's frame performance records by zeroising buckets 3 and 4 of the lesson's response lesson-file.

Error Halts :

Event	Cause
DELETED : EXCEPTION CONDITION xxxx	Storage device macro exception condition has occurred. xxxx is the contents of word 11 which contains the exception condition code.

```
#STEER          LIST,OBJECT
#PROGRAM        RESE
#CMODE          EDS(0)
#LOWER
                BUFF(512)
#LOWER
FD0             0,0,12HLESSONNAME   ,,,,,,,,,,,,,,,,,,,,,,,,,
FPD1KEY         3
FPD2KEY         4
LNOUT           32HEXCEPTION CONDITION:-
CONT            34/LNOUT.0
#DEFINE         BUFFER=BUFF+2
#PROGRAM
#ENTRY          0
#
#                OPEN FILE.
#
      SDDEF 0   1,512,FD0,EM0,1,0
#
#                DELETE OLD RECORDS.
#
      LDN   1   3
      STO   1   FD0+25
      LDN   1   FPD1KEY
      STO   1   FD0+26
      SDDEL 0
      LDN   1   4
      STO   1   FD0+25
      LDN   1   FPD2KEY
      STO   1   FD0+26
      SDDEL 0
#
#                ZEROISE BUFFER AREA.
#
      LDCT  1   500
      STOZ      BUFFER(1)
      BUX   1   *-1
```

```
#
#                   RESET RECORD BUCKETS 3 & 4.
#
       LDN    1   502
       STO    1   BUFF
       LDN    1   4
       STO    1   BUFF+1
       STO    1   FD0+25
       LDN    1   FPD2KEY
       STO    1   FD0+26
       SDWRI  0   BUFF
       LDN    1   3
       STO    1   BUFF+1
       STO    1   FD0+25
       LDN    1   FPD1KEY
       STO    1   FD0+26
       SDWRI  0   BUFF
#
#                   CLOSE FILE.
#
       SDEND  0   CLOSE
       DEL        2HOK
#
#                   EXCEPTION CONDITION.
#
EX0    LDX    1   11
       STO    1   LNOUT+6
       DELTY      CONT
#END
#FINISH
ENDPROG
****
```

.APPENDIX 10.

A10) Program Name : NEWED

Language : PLAN

Core Size : ½ K words

Calling Macros : ERF & ESF

Function : To edit a lesson's input-files.

A10.1) Subroutine CBINDEC.

CBINDEC converts a binary integer to a four digit decimal integer.

```
#STEER              LIST,OBJECT
#PROGRAM            EDIT
#LOWER
                    LINE(20),INST(20)
                    ACC1,ACC2,ACC3,ACC4,ACC5,ACC6,ACC7,ACC0
                    DEC
#LOWER
CRINST              3/0,0,80,0/INST.0
CPINST              4/0,0,80,0/INST.0
CRLINE              3/0,0,80,0/LINE.0
CPLINE              4/0,0,80,0/LINE.0
TEN                 10
HASH                4H000#
ABAN                4H#ABA
COPY                4H#COP
DEL                 4H#DEL
END                 4H#END
SPAC                4H000
DOT                 4H000.
STARS               4H****
POINTER             0
LINENUM             0
CHARS               10000
MESSAB              16HEDIT ABANDONED.
CONAB               16/MESSAB.0
MESSHEAD            #41,40HEDITOR IS READY
CONHEAD             2/0,0,41,0/MESSHEAD.3
MESS1               #41,24HLINE NUMBER NOT FOUND.
CON1                2/0,0,25,0/MESS1.3
MESS2               #41,28HLINE NUMBER TOO LARGE.
CON2                2/0,0,29,0/MESS2.3
MESS3               #41,28HILLEGAL EDITING INSTRUCTION.
CON3                2/0,0,29,0/MESS3.3
MESS4               #41,28HLINE NUMBER TOO SMALL.
CON4                2/0,0,29,0/MESS4.3
MESS5               #41,28HILLEGAL LINE NUMBER.
CON5                2/0,0,29,0/MESS5.3
MESS6               #41,28HEDIT COMPLETED.
CON6                2/0,0,29,0/MESS6.3
MESSPOINT           #41,8H
CONPOINT            2/0,0,9,0/MESSPOINT.3
#PROGRAM
#ENTRY              0
        PERI     0  CONHEAD
        SUSBY       LP0
RINST   LDX      5  POINTER
        CALL     1  CBINDEC
        LDX      4  DEC
        STO      4  MESSPOINT+1
```

```
          PERI   0  CONPOINT       [ PRINT LINE POINTER.
          SUSBY     LP0
          PERI   1  CRINST         [ READ EDITING INSTRUCTION.
          SUSBY     CR1
          LDN    4  0
          LDX    5  INST
          SLL    45 6
          BXU    4  HASH,INS       [ INSERT LINE ??
          LDX    4  INST
          BXE    4  COPY,COP       [ COPY INSTRUCTION.
          BXE    4  DEL,DLT        [ DELETE INSTRUCTION.
          BXE    4  END,FIN        [ COPY TO END OF FILE.
          BXE    4  ABAN,ABA       [ ABANDON EDIT.
          PERI   0  CON3           [ ILLEGAL EDITING INSTRUCTION.
          SUSBY     LP0
          BRN       RINST
#
#                    COPY INSTRUCTION.
#
COP       LDCT   2  40
          LDN    4  0
GAP       LDCH   4  INST+1(2)
          BXE    4  SPAC,SPACS     [ GAP ??
          BCHX   2  GAP
          BRN       ERR1           [ LINE NUMBER NOT FOUND ??
SPACS     LDCH   4  INST+1(2)
          BXU    4  SPAC,LNUM
          BCHX   2  SPACS
          BRN       ERR1           [ LINE NUMBER NOT FOUND ??
LNUM      LDCT   1  6
          STOZ      4
          STOZ      6
          STOZ      7
NXT       LDCH   4  INST+1(2)
          BXE    4  SPAC,END1
          BXE    4  DOT,END1
          BXGE   4  TEN,NONNM  [ NON  NUMERIC ??
          MPY    6  TEN
          LDX    6  7
          ADX    6  4
          BCHX   2  *+2
          BRN       ERR1
          BUX    1  NXT
          PERI   0  CON2           [ LINE NUMBER TOO LARGE.
          SUSBY     LP0
          BRN       RINST
END1      STO    6  LINENUM
          BXGE   6  POINTER,TRAN
          PERI   0  CON4           [ LINE NUMBER TOO SMALL.
          SUSBY     LP0
          BRN       RINST
```

```
TRAN  BXE    6  POINTER,RINST  [ TRANSFER FINISHED ??
      PERI   0  CRLINE
      SUSBY     CR0
      PERI   0  CPLINE
      SUSBY     CP0
      LDX    4  LINE
      BXE    4  STARS,ENDE    [ EDIT FINISHED ??
      LDN    5  1
      ADS    5  POINTER
      BRN       TRAN
#
#                 DELETE INSTRUCTION.
#
DLT   LDCT   2  40
      LDN    4  0
GAPD  LDCH   4  INST+1(2)
      BXE    4  SPAC,SPCS     [ GAP ??
      BCHX   2  GAPD
      BRN       ERR1          [ LINE NUMBER NOT FOUND ??
SPCS  LDCH   4  INST+1(2)
      BXU    4  SPAC,LINUM
      BCHX   2  SPCS
      BRN       ERR1
LINUM LDCT   1  6
      STOZ      4
      STOZ      6
      STOZ      7
NXTD  LDCH   4  INST+1(2)
      BXE    4  SPAC,END2
      BXE    4  DOT,END2
      BXGE   4  TEN,NONNM  [ NON  NUMERIC ??
      MPY    6  TEN
      LDX    6  7
      ADX    6  4
      BCHX   2  *+2
      BRN       ERR1
      BUX    1  NXTD
      PERI   0  CON2           [ LINE NUMBER TOO LARGE.
      SUSBY     LP0
      BRN       RINST
END2  STO    6  LINENUM
      BXGE   6  POINTER,DELT
      PERI   0  CON4           [ LINE NUMBER TOO SMALL.
      SUSBY     LP0
      BRN       RINST
DELT  BXE    6  POINTER,RINST  [ DELETIONS FINISHED ??
      PERI   0  CRLINE
      SUSBY     CR0
      LDX    4  LINE
      BXU    4  STARS,*+4
```

```
      PERI  0  CPLINE
      SUSBY    CP0
      BRN      ENDE          [ EDIT FINISHED ??
      LDN   5  1
      ADS   5  POINTER
      BRN      DELT
#
#               INSERT LINE.
#
INS   PERI  0  CPINST
      SUSBY    CP0
      BRN      RINST
#
#               END INSTRUCTION.
#
FIN   PERI  0  CRLINE
      SUSBY    CR0
      PERI  0  CPLINE
      SUSBY    CP0
      LDX   4  LINE
      BXU   4  STARS,FIN
ENDE  PERI  0  CON6          [ EDIT COMPLETED.
      SUSBY    LP0
      DEL      2HOK
#
#               ABANDON INSTRUCTION.
#
ABA   DISTY    CONAB
      DEL      2HAB
ERR1  PERI  0  CON1
      SUSBY    LP0
      BRN      RINST
NONNM PERI  0  CON5          [ NON NUMERIC.
      SUSBY    LP0
      BRN      RINST
```

```
#
#                    SUBROUTINE 'CBINDEC'
#                    EXPECTS ACC 5 = BINARY INTEGER.
#                    CONVERTS IT TO DECIMAL, STORING
#                    RESULT IN 'DEC'.
#
#CUE               CBINDEC
        STO    2   ACC2
        STO    6   ACC6
        STO    7   ACC7
        LDCT   2   3
        LDN    6   0
        LDX    7   CHARS
        DVD    5   7
        ADN    6   1
        LDN    7   0
        MODE       1
        CBD    6   DEC(2)
        BCHX   2   *-1
        MODE       0
        CBD    6   DEC(2)
        LDX    2   ACC2
        LDX    6   ACC6
        LDX    7   ACC7
        EXIT   1   0
#END
#FINISH
ENDPROG
****
```

.APPENDIX 11.

A11) This appendix contains the test paper used in the ACATS trials.

The marks allotted to each question were as follows:-

QUESTION	1	2	3	4	5	6
PERCENTAGE SCORE	10	15	25	10	20	20

1) Define statistics.

2) Name the three basic constituents of a mathematical model.

3) Name the four levels of scaling described by S.S. Stevens..
Associate each level with one of the following:-
   - a) identification.
   - b) difference.
   - c) fixed zero.
   - d) order.

Name the highest and lowest levels of scaling.

4) State the main purpose of:-
   - a) Descriptive statistics.
   - b) Statistical inference.

5) In a test given to a class of 20 students, the following scores were made:-

6,7,5,8,8,9,8,10,8,8,8,9,5,7,6,8,9,7,6,7.

Complete a frequency distribution for the test scores.

6) Complete a <u>grouped</u> frequency distribution for the test scores given in question 5, using the following intervals,

5 - 6 , 7 - 8 , 9 - 10 .

REFERENCES.

AVNER A. & TENCZAR P. (1969) "The Tutor Manual", CERL Report X-4, 1969.

COMPUTER WEEKLY (1973) "Computer Weekly (International)", July 12, 1973.

CROWDER N. A. (1960) "Teaching Machines and Programmed Learning", eds. Lumsdaine & Glaser, National Education Association, 1960.

DOWSEY M. W. (1973) "Future Directions for Computer Based Learning", IBM UKSC - 0032, August 1973.

HUNT E. & ZOSEL M. (1968) "WRITEACOURSE: An Educational Programming Language", AFIPS - Conference Proceedings, Vol. 33.

IBM (1967) "COURSEWRITER II, Author's Guide", Technical Information Centre, IBM United Kingdom Limited.

ICL (1970) "PLAN Reference Manual", Technical Publication 4004.

ICL (1971a) "GEORGE 3 & 4 Operating Systems" Technical Publication 4267.

ICL (1971b) "Direct Access Manual" Technical Publication 4107.

ICL (1972) "Introduction to MOP" Technical Publication 4959.

LIPPERT H. T. (1971) "Computer Support of Instruction and Student Services in a College or University", Educational Technology, 11, 5, May 1971.

NCET (1969) "A Programme for Action", National Council for Educational Technology, August 1969.

PRESSEY S. L. (1926) "Teaching Machines and Programmed Learning", eds. Lumsdaine & Glaser, National Education Association, 1960.

RATH G. J., ANDERSON N. S. & BRAINERD R. C. (1960) "The IBM Research Centre Teaching Machine Project" in "Automatic Teaching, The State of the Art", ed. Galanter.

SCHRAMM W. (1964) "The Research on Programmed Instruction, an Annotated Bibliography", U.S. Office of Education Bulletin No. 35.

SILVERN G. M. & SILVERN L. C. (1966) "Computer Assisted Instruction: Specification of attributes for CAI programs and programmers", Proceedings of ACM National Conference, 1966, pp 57-62.

SKINNER B. F. (1954) "The Science of Learning and the Art of Teaching", Harvard Educ. Review, 24, Spring, 1954, pp86-97.

UTTAL W. R. (1962) "My Teacher has Three Arms", IBM Watson Research Centre, RC788, September, 1962.

ZINN K. L. (1969) "Comparative Study of Languages for Programming Interactive Use of Computers in Education", EDUCOM (microfilm), Boston, Mass., Feb. 1969.

ZINN K. L. (1971) "Requirements for Programming Languages in CAI Systems", Educational Yearbook 1971/72, British Computer Society.